

Regression Testing: Theoretical Underpinnings, Practical Techniques, and Empirical Insights

Gregory M. Kapfhammer* and Jonathan Miller Kauffman†

Department of Computer Science
Allegheny College

*<http://www.cs.allegheny.edu/~gkapfham/>

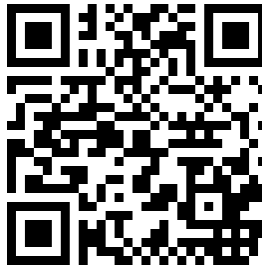
†<https://sites.google.com/a/allegheny.edu/jonathan-kauffman/>

15th International Conference on Software Engineering and Applications
December 14 - 16, 2011



ALLEGHENY COLLEGE

Accessing the Presentation



Scan this QR Code with your smartphone!

... or, visit this Web site:

<http://www.cs.allegheeny.edu/~gkapfham/sea2011.pdf>

... or, ask us for a USB drive!



Presenter Introduction: Jonathan Miller Kauffman



Software Challenges

① Introduction

Important Points

Software Challenges

② Software Testing

Basic Concepts

Testing Opportunities

③ Regression Testing

Supporting Methods

Key Algorithms

④ Empirical Evaluation

Model for Experimentation

Concrete Examples

⑤ Conclusion

Software is Everywhere

Program

Computer
Server

Software is Everywhere

Program

Program

Desktop
Computer

Computer
Server

Software is Everywhere

Program

Desktop
Computer

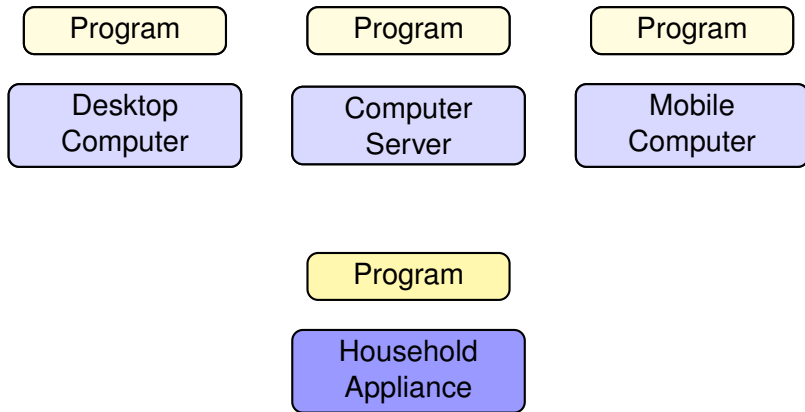
Program

Computer
Server

Program

Mobile
Computer

Software is Everywhere



Software is Everywhere

Program

Desktop
Computer

Program

Computer
Server

Program

Mobile
Computer

Program

Scientific
Device

Program

Household
Appliance

Software is Everywhere

Program

Desktop
Computer

Program

Computer
Server

Program

Mobile
Computer

Program

Scientific
Device

Program

Household
Appliance

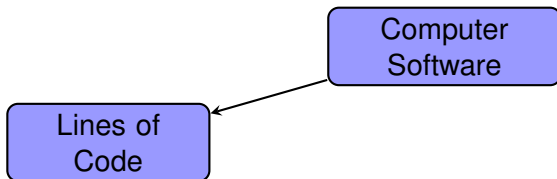
Program

Network
Router

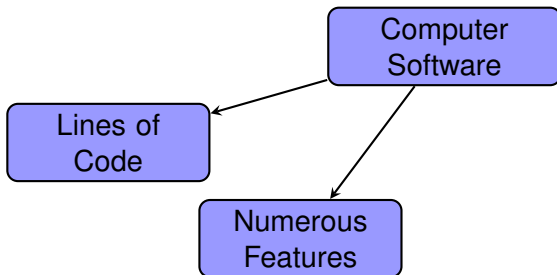
Software is Complex

Computer
Software

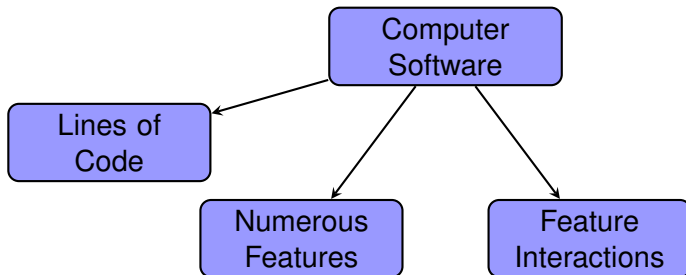
Software is Complex



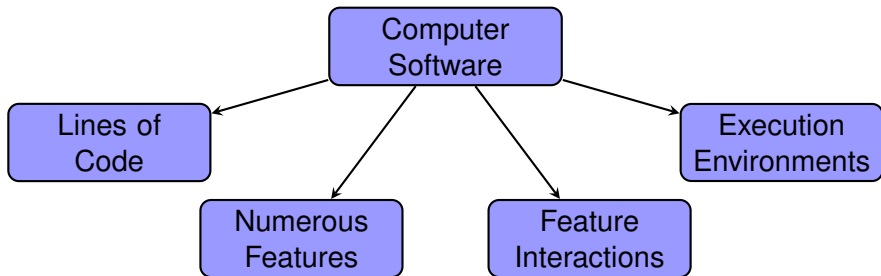
Software is Complex



Software is Complex

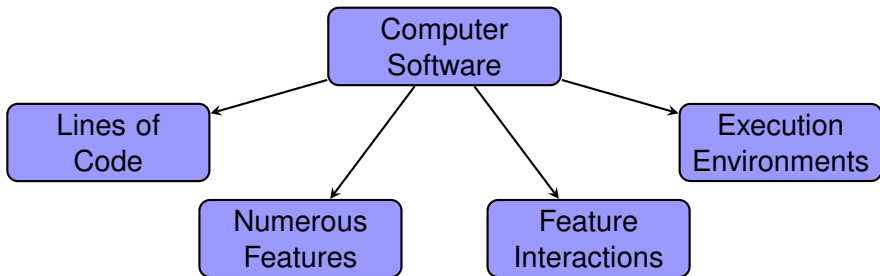


Software is Complex



Software is Complex

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

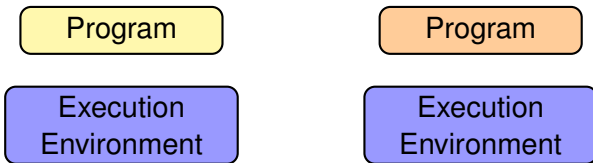


Software is Evolving

Program

Execution
Environment

Software is Evolving



Software is Evolving

Program

Program

Execution
Environment

Execution
Environment

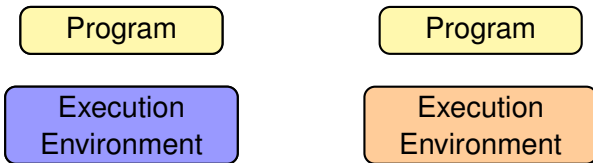
Program Changed because of the addition
of a new feature or the correction of a defect

Software is Evolving

Program

Execution
Environment

Software is Evolving



Software is Evolving

Program

Execution
Environment

Program

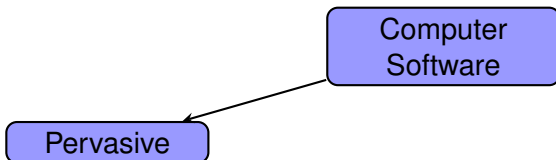
Execution
Environment

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

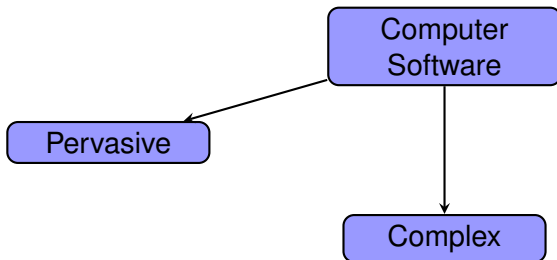
Regression Testing to the Rescue

Computer
Software

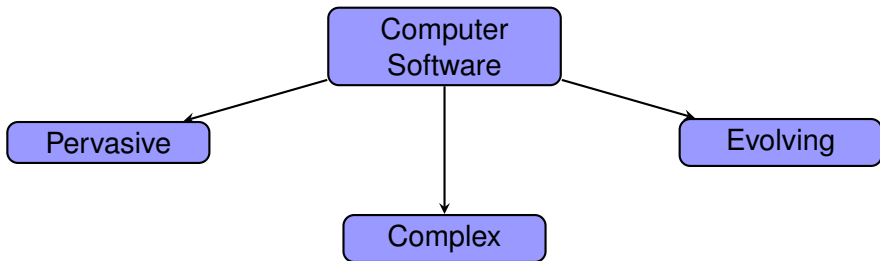
Regression Testing to the Rescue



Regression Testing to the Rescue

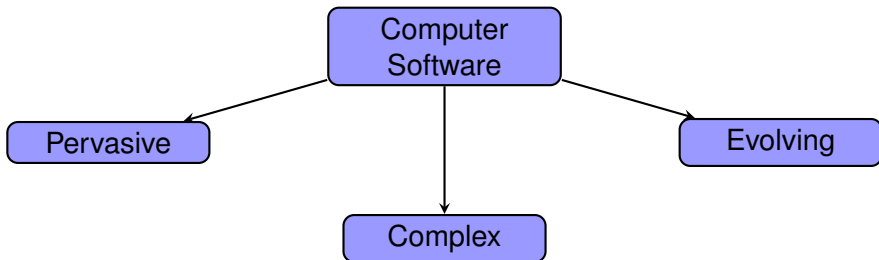


Regression Testing to the Rescue



Regression Testing to the Rescue

Regression Testing supports the efficient construction of pervasive software that is complex and rapidly evolving



Basic Concepts

1 Introduction

Important Points

Software Challenges

2 Software Testing

Basic Concepts

Testing Opportunities

3 Regression Testing

Supporting Methods

Key Algorithms

4 Empirical Evaluation

Model for Experimentation

Concrete Examples

5 Conclusion

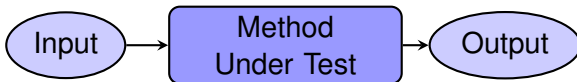
What is a Test Case?

Method
Under Test

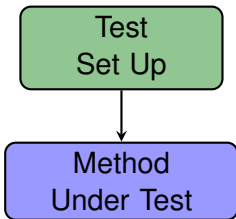
What is a Test Case?



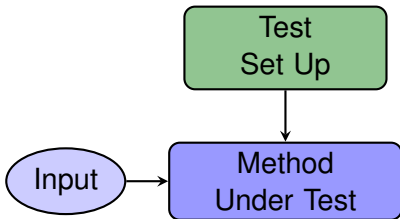
What is a Test Case?



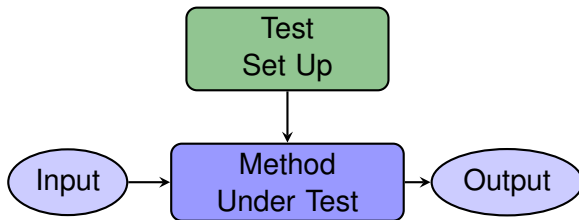
What is a Test Case?



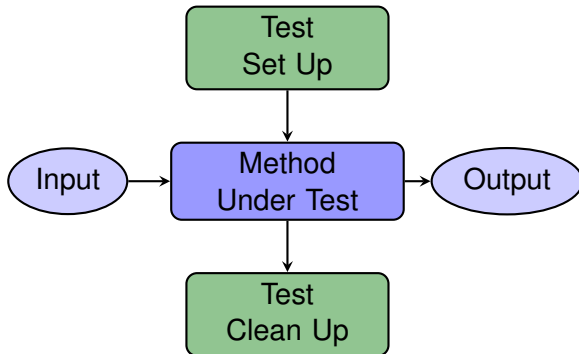
What is a Test Case?



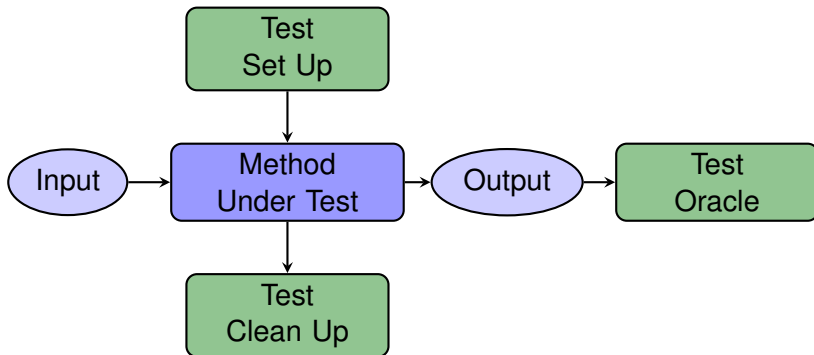
What is a Test Case?



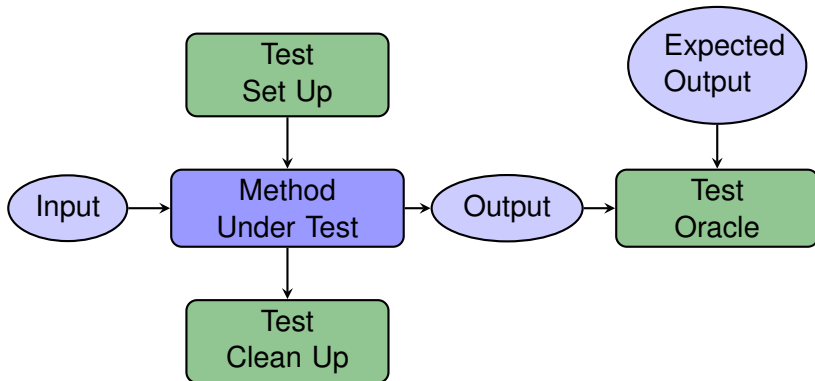
What is a Test Case?



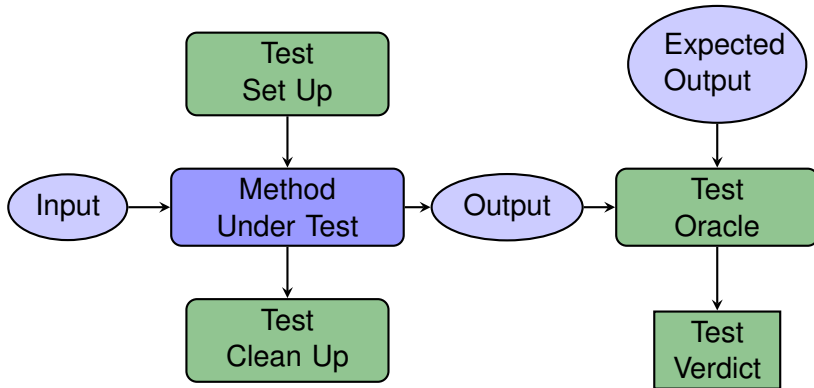
What is a Test Case?



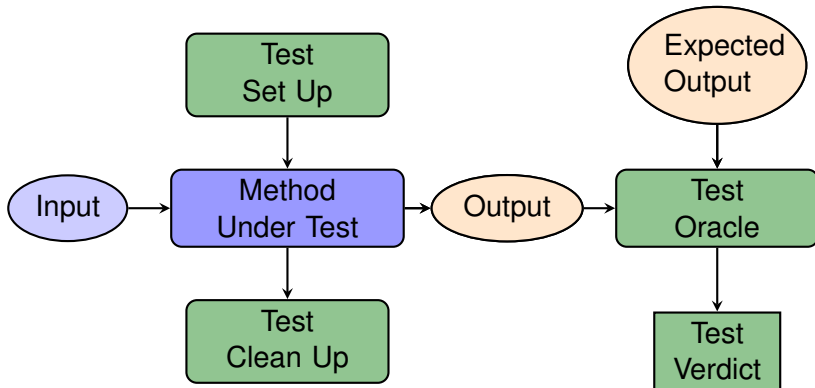
What is a Test Case?



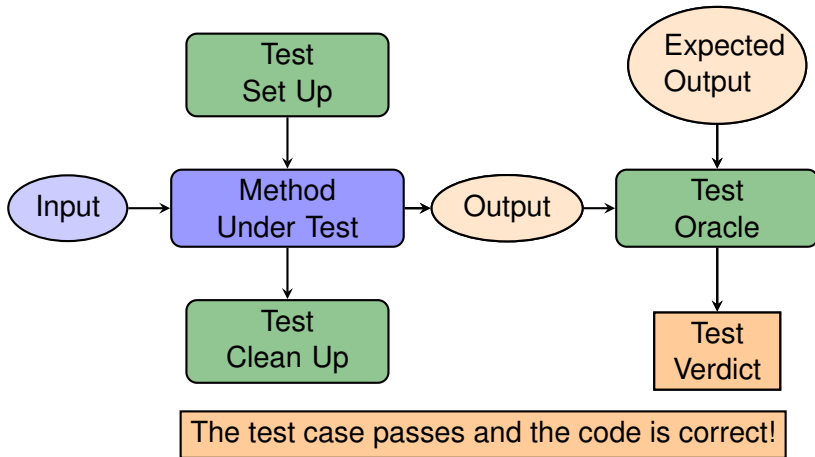
What is a Test Case?



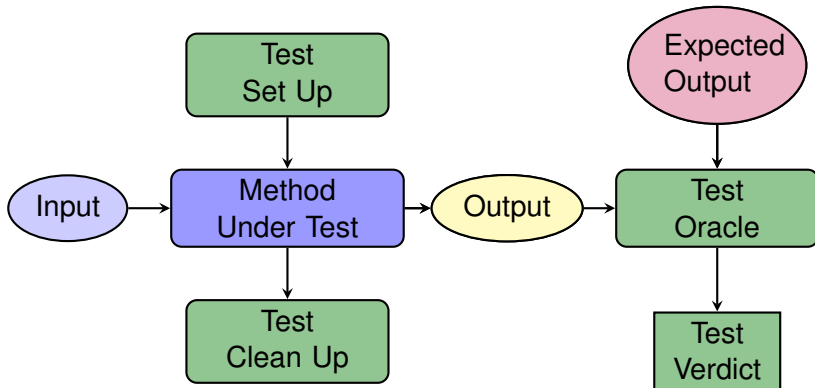
What is a Test Case?



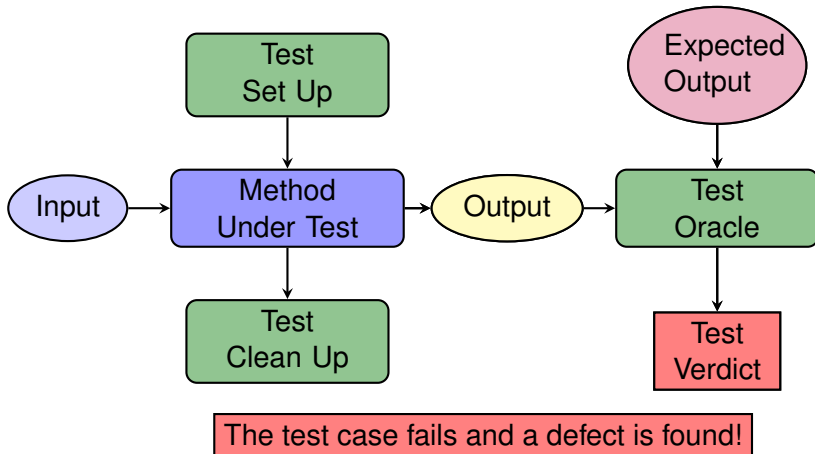
What is a Test Case?



What is a Test Case?



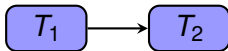
What is a Test Case?



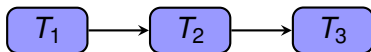
What is a Test Suite?

 T_1

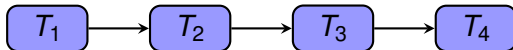
What is a Test Suite?



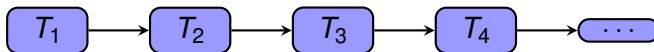
What is a Test Suite?



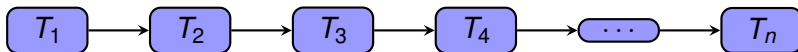
What is a Test Suite?



What is a Test Suite?

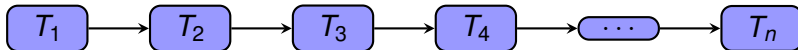


What is a Test Suite?



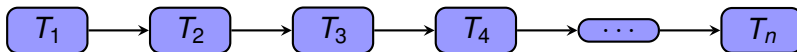
What is a Test Suite?

Organize the Test Cases into a Test Suite



What is a Test Suite?

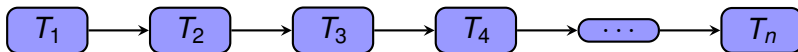
Organize the Test Cases into a Test Suite



Tool Support for Software Testing?

What is a Test Suite?

Organize the Test Cases into a Test Suite

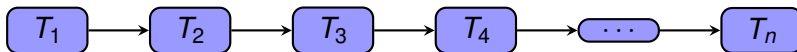


Tool Support for Software Testing?

JUnit

What is a Test Suite?

Organize the Test Cases into a Test Suite



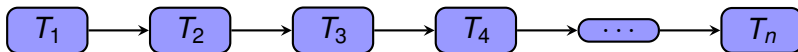
Tool Support for Software Testing?

JUnit

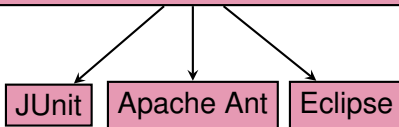
Apache Ant

What is a Test Suite?

Organize the Test Cases into a Test Suite



Tool Support for Software Testing?



Testing Opportunities

1 Introduction

Important Points

Software Challenges

2 Software Testing

Basic Concepts

Testing Opportunities

3 Regression Testing

Supporting Methods

Key Algorithms

4 Empirical Evaluation

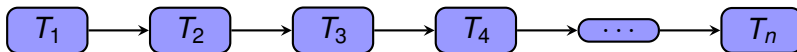
Model for Experimentation

Concrete Examples

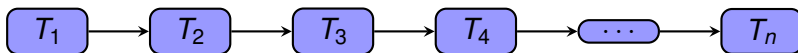
5 Conclusion

Test Suite Management

Organize the Test Cases into a Test Suite



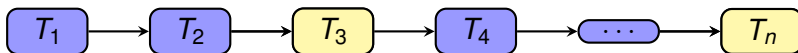
Test Suite Management



Regression Testing Technique

Test Suite Management

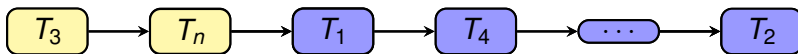
What if Some Test Cases are More Effective?



Regression Testing Technique

Test Suite Management

What if Some Test Cases are More Effective?

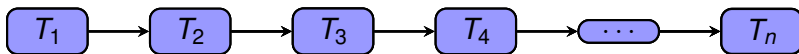


Regression Testing Technique

Prioritization

Test Suite Management

What if Some Test Cases are More Effective?

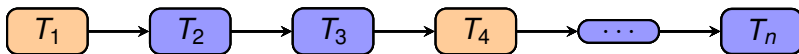


Regression Testing Technique

Prioritization

Test Suite Management

What if Some Test Cases are Redundant?

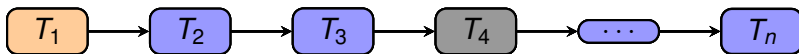


Regression Testing Technique

Prioritization

Test Suite Management

What if Some Test Cases are Redundant?



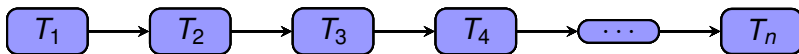
Regression Testing Technique

Prioritization

Reduction

Test Suite Management

What if Some Test Cases are Redundant?



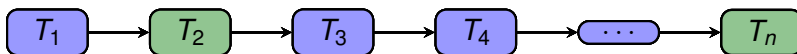
Regression Testing Technique

Prioritization

Reduction

Test Suite Management

What if Only Certain Tests are Needed?



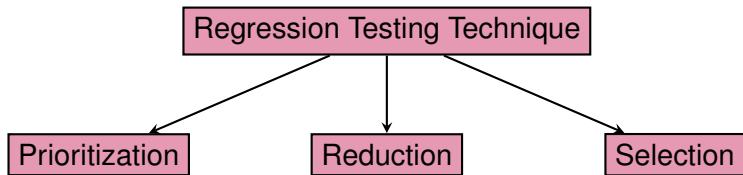
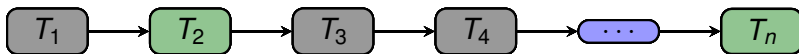
Regression Testing Technique

Prioritization

Reduction

Test Suite Management

What if Only Certain Tests are Needed?

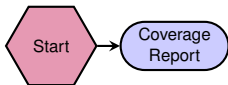


3 Regression Testing

Supporting Methods

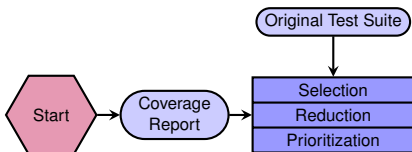


Model of Regression Testing

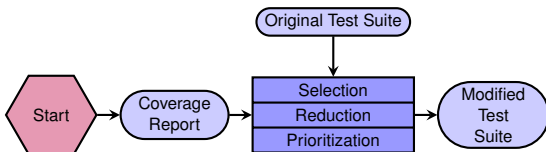


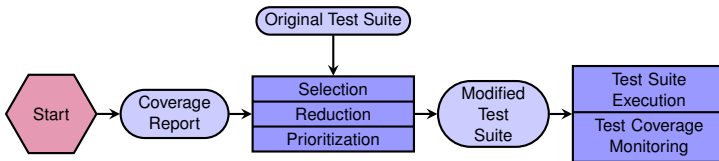
Model of Regression Testing

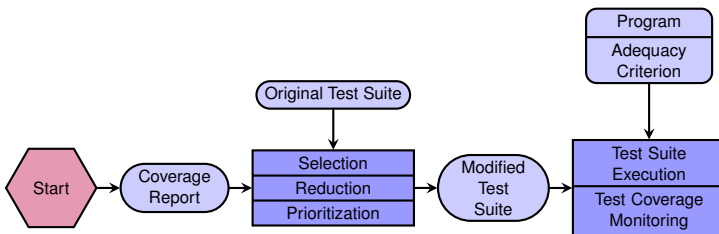


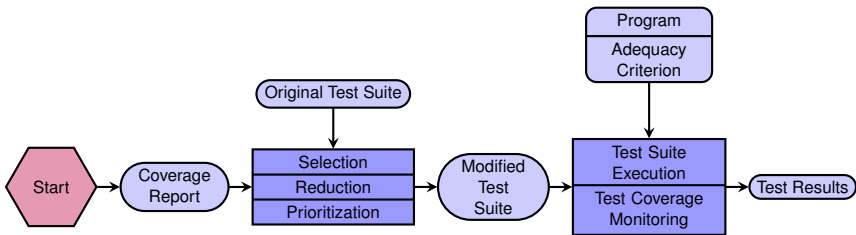


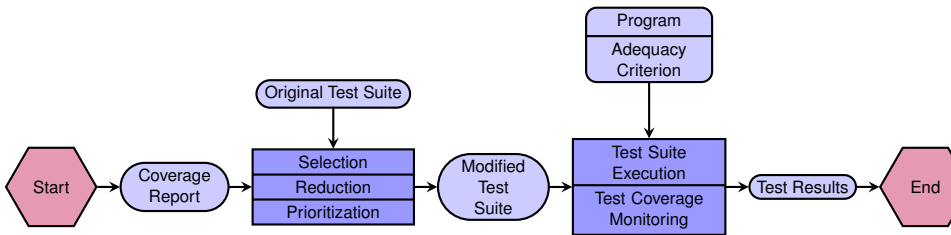
Model of Regression Testing

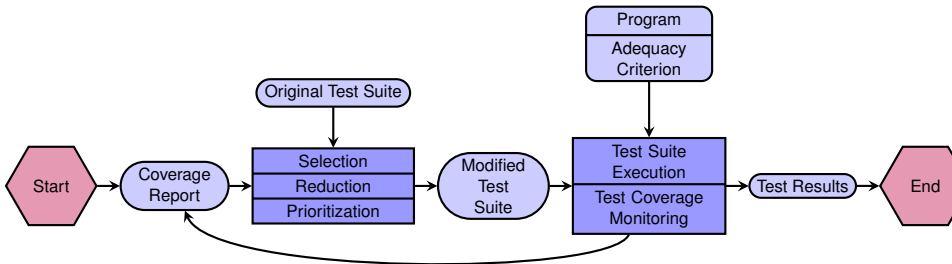


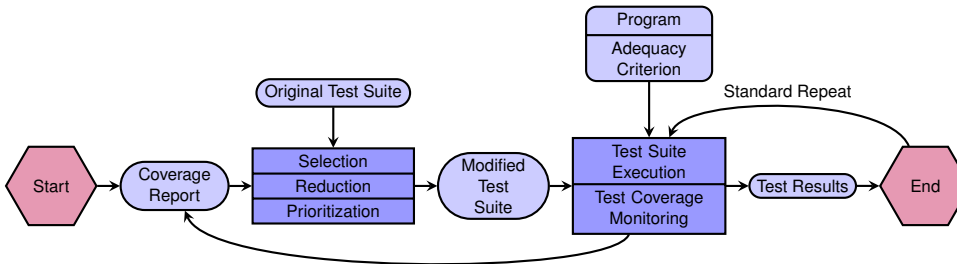


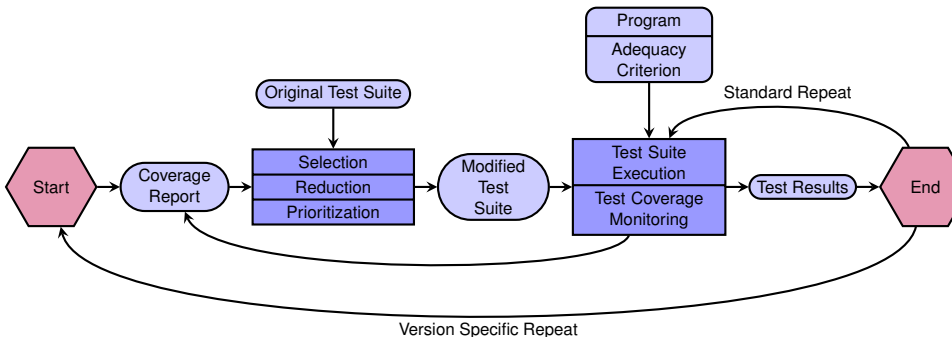


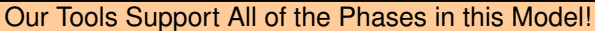




[illegible]





[illegible]

Test Suite Adequacy

 T_1 T_2

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

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

Test Suite Adequacy

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

 T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10} R_1 R_2

Test Suite Adequacy

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

 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

Test Suite Adequacy

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

 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 $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$

 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 $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$

 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}

Test Suite Adequacy

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

 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}

Test Suite Adequacy

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

 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}

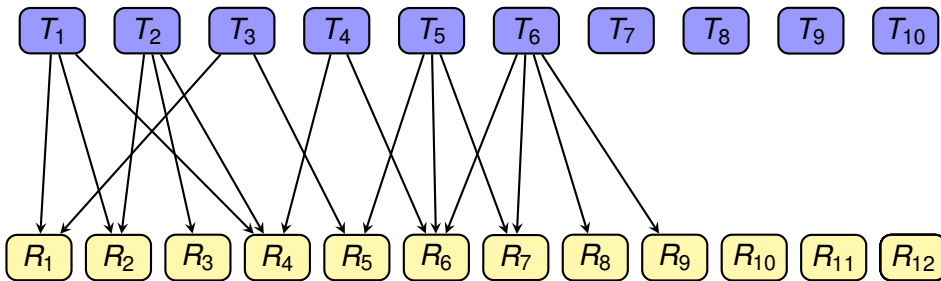
Requirements Set $R = \{R_1, R_2, \dots, R_{11}, R_{12}\}$

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


Regression Testing: Theoretical Underpinnings, Practical Techniques, and Empirical Insights

Test Suite Adequacy

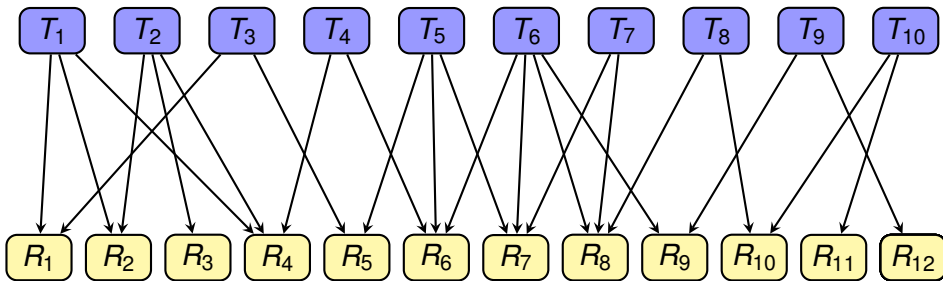
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$



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

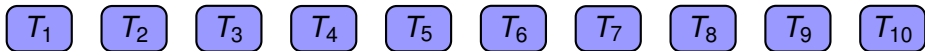
Test Suite Adequacy

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



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

Test Suite Execution



Test Suite Execution

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

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

Test Suite Execution

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

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

JUnit Test Automation Framework

Test Suite Execution

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

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

JUnit Test Automation Framework

Run Test Case

Test Suite Execution

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

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

JUnit Test Automation Framework

Passing Test Case: $O_A = O_E$

Test Suite Execution

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

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

JUnit Test Automation Framework

Test Suite Execution

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

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

JUnit Test Automation Framework

Test Suite Execution

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



JUnit Test Automation Framework

Run Test Case

Test Suite Execution

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



JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

Test Suite Execution

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



JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

Stop Running T

Test Suite Execution

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



JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

Stop Running T

Test Suite Execution

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



JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

Stop Running T

Test Suite Execution

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



JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

Stop Running T

Continue Running T

Test Suite Execution

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



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 $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$

 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 $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$

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

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

Test Coverage Monitoring

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

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

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 $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$

 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}

Test Coverage Monitoring

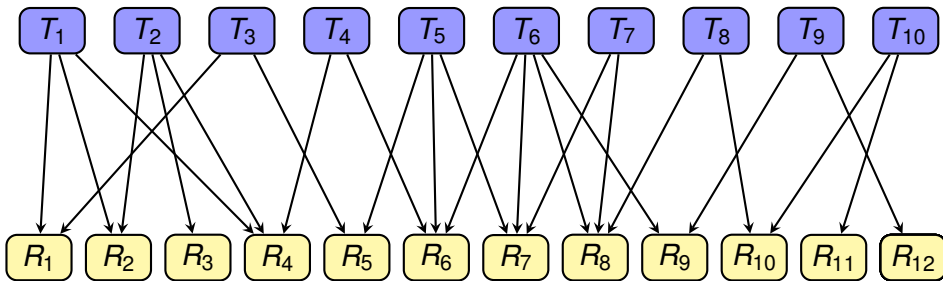
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$

 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}

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

Test Coverage Monitoring

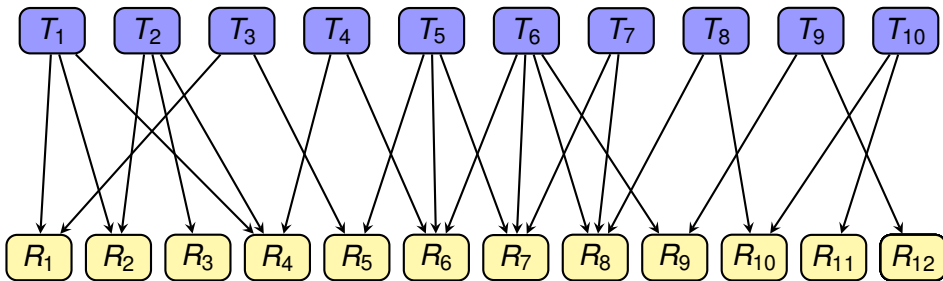
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$



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

Test Coverage Monitoring

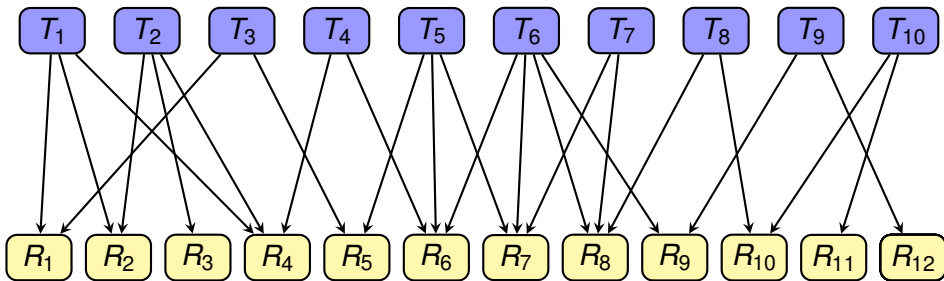
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$



Requirements Set R for ... Statement Coverage

Test Coverage Monitoring

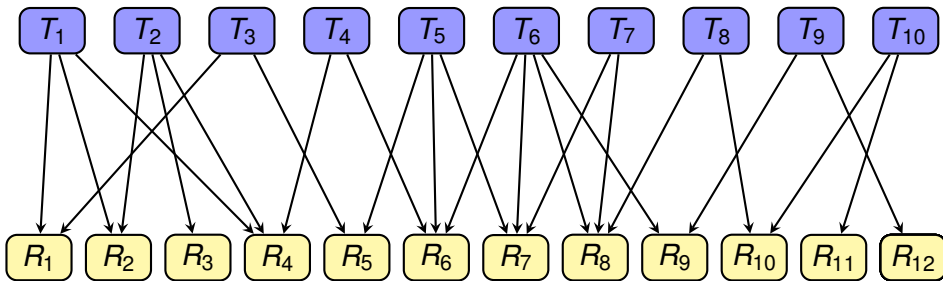
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$



Requirements Set R for ... Mutation Coverage

Test Coverage Monitoring

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



Requirements Set R for ... Definition-Use Coverage

Hands-on: Preparing the Applications and Tools

1

Compile the Sudoku Case Study Application

```
cd proteja-installation-directory  
cd apps/SK  
ant init  
ant compile  
cd ../../
```


Hands-on: Preparing the Applications and Tools

1

Compile the Sudoku Case Study Application

```
cd proteja-installation-directory  
cd apps/SK  
ant init  
ant compile  
cd ../../
```

2

Compile the Proteja Test Coverage Monitor

```
cd ../../  
ant init  
ant build
```

Hands-on: Executing Tests and Monitoring Coverage

1

Run Sudoku's Test Suite Without Coverage Monitoring

```
cp data/settingsFiles/SK/* .  
ant addAnnotations  
ant proteja
```

Hands-on: Executing Tests and Monitoring Coverage

1

Run Sudoku's Test Suite Without Coverage Monitoring

```
cp data/settingsFiles/SK/* .  
ant addAnnotations  
ant proteja
```

2

Run Sudoku's Test Suite With Coverage Monitoring

```
ant proteja -Dcobertura=true  
-DtestTimer=true -DapplicationName=Sudoku
```

Screenshot: Configuring Test Suite Execution

```

^  v  x  build.xml (~/.research/proteja) - gedit
File Edit View Search Tools Documents Help

[Icons: Open, Save, Undo, etc.]

build.xml
<!-- Specifies tasks that may need to be performed when using Proteja.

Jonathan Miller Kauffman -->

<project name="proteja" basedir=".">

  <!-- The locations of the program source code, program bytecode, and test suite bytecode. If you want to
  change these locations, modify these properties in 'proteja.xml' and set them on the command line when
  running the proteja ant task. -->
  <property name="userSource" value="apps/SK/src/sudoku/" />
  <property name="userBytecode" value="apps/SK/bin/" />
  <property name="testSuite" value="apps/SK/bin/" />

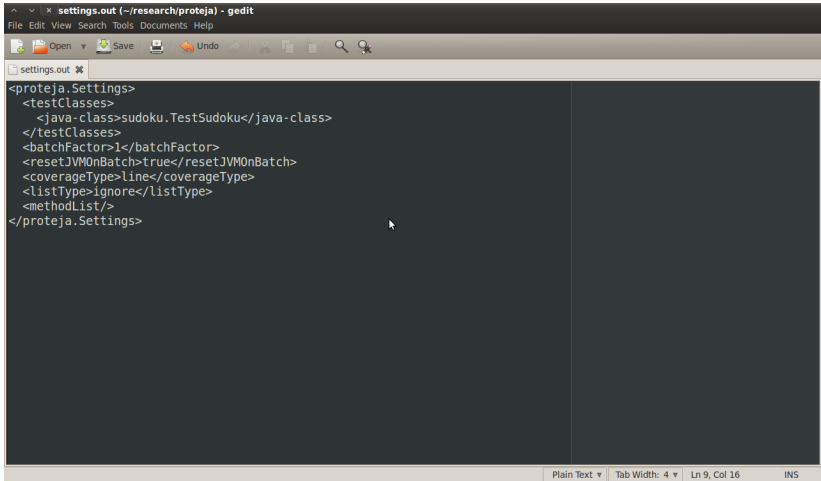
  <property name="src" value="src/" />
  <property name="build" value="build/classes/" />

  <!-- Cobertura will not run unless the argument '-Dcobertura=true' is supplied on the command-line. -->
  <property name="cobertura" value="false" />
  <!-- Per-test case timing information will not be captured unless '-DtestTimer=true' is supplied on the
  command-line. -->
  <property name="testTimer" value="false" />
  <!-- The name of the application being run. -->
  <property name="applicationName" value="default" />
  <!-- The name of the file holding a list of test cases to run. -->
  <property name="modificationFile" value="data.dat" />
  
```

XML ▾ Tab Width: 4 ▾ Ln 20, Col 7 INS

Supporting Methods

Screenshot: Configuring the Test Coverage Monitor



The screenshot shows a gedit editor window titled "settings.out (~/.research/proteja) - gedit". The editor contains XML configuration for the Test Coverage Monitor. The configuration is as follows:

```
<proteja.Settings>
  <testClasses>
    <java-class>sudoku.TestSudoku</java-class>
  </testClasses>
  <batchFactor>1</batchFactor>
  <resetJVMOnBatch>true</resetJVMOnBatch>
  <coverageType>line</coverageType>
  <listType>ignore</listType>
  <methodList/>
</proteja.Settings>
```

The status bar at the bottom indicates "Plain Text", "Tab Width: 4", "Ln 9, Col 16", and "INS".

Screenshot: Running the Test Suite

```
jonathan@jonathan-laptop: ~/research/roteja
File Edit View Terminal Help
jonathan@jonathan-laptop:~/research/roteja$ ant roteja
Buildfile: build.xml

roteja:
[java] Reading file
[java] File read
[java] Settings in.
[java] include list found!
[java] Starting new JVM
[java]
[java]
[java] run:
[java] [java] testInitialize_Null passed.
[java] Starting new JVM
[java]
[java]
[java] run:
[java] [java] testGetPeersInSameSubSquare passed.
[java] Starting new JVM
[java]
[java]
[java] run:
[java] [java] testGetUnsolvedBoxes passed.
[java] Starting new JVM
[java]
```

Supporting Methods

Screenshot: Performing Test Coverage Monitoring

```
jonathan@jonathan-laptop: ~/research/proteja
File Edit View Terminal Help
jonathan@jonathan-laptop:~/research/proteja$ ant proteja -Dcobertura=true -DtestTimer=true
-DdapApplicationName=Sudoku
Buildfile: build.xml

proteja:
[java] Reading file
[java] File read
[java] Settings in.
[java] include list found!
[java] inst:
[java] [mkdir] Created dir: /home/jonathan/research/proteja/instrumentedClasses
[java] [cobertura-instrument] Cobertura 1.9.1 - GNU GPL License (NO WARRANTY) - See C
OPYRIGHT file
[java] [cobertura-instrument] Instrumenting 4 files to /home/jonathan/research/protej
a/instrumentedClasses
[java] [cobertura-instrument] Cobertura: Saved information on 4 classes.
[java] [cobertura-instrument] Instrument time: 192ms
[java]
[java] createCopy:
[java] [mkdir] Created dir: /home/jonathan/research/proteja/temp
[java] [copy] Copying 1 file to /home/jonathan/research/proteja/temp
[java] Starting new JVM
[java]
```


Supporting Methods

Screenshot: Viewing a Test Case Timings Report

```

Sudoku_Timing.dat (~/.research/modificare/reqMatrices/TimingsFiles) - gedit
File Edit View Search Tools Documents Help

Sudoku_Timing.dat

sudoku.TestSudoku.testSolveByPropagatingKnownValues 137746198 1 line true Sudoku false
sudoku.TestSudoku.testSolvedValues 17867635 1 line true Sudoku false
sudoku.TestSudoku.testSolvedValues_SomeValuesMissing 17653082 1 line true Sudoku false
sudoku.TestSudoku.testInitialize_Empty 17088901 1 line true Sudoku false
sudoku.TestSudoku.testPossibleValues_Tiny 17739966 1 line true Sudoku false
sudoku.TestSudoku.testGetPeers_Unit6 17987762 1 line true Sudoku false
sudoku.TestSudoku.testGetPeersInSameSubSquare 17968833 1 line true Sudoku false
sudoku.TestSudoku.testGetUnsolvedBoxes 17769788 1 line true Sudoku false
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_NoUnitsHave9AsPossibleValue 32366684 1
line true Sudoku false
sudoku.TestSudoku.testInitialize_NotSquare 17090510 1 line true Sudoku false
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_Row 31559389 1 line true Sudoku false
sudoku.TestSudoku.testRemoveSolvedValuesFromPeers 25016796 1 line true Sudoku false
sudoku.TestSudoku.testInitialize_NoSeededValues 17790041 1 line true Sudoku false
sudoku.TestSudoku.testInitialize_NoSeededValues_Zeros 17631850 1 line true Sudoku false
sudoku.TestSudoku.testSolvedValue_Error 17651334 1 line true Sudoku false
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_Column 32211987 1 line true Sudoku false
sudoku.TestSudoku.testRemoveSolvedValuesFromPeers_ReachContradiction 19033426 1 line true Sudoku
false
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_SubSquare 32107781 1 line true Sudoku
false
sudoku.TestSudoku.testRemoveSolvedValuesFromPeers_NewSolvedValue 47971324 1 line true Sudoku false
sudoku.TestSudoku.testGetPeers_Unit1 18416308 1 line true Sudoku false
sudoku.TestSudoku.testGetPeersInSameColumn 20217304 1 line true Sudoku false
sudoku.TestSudoku.testInitialize_NoSeededValues_Dashes 17590924 1 line true Sudoku false
sudoku.TestSudoku.testPossibleValues_InitiateBoxState 18697206 1 line true Sudoku false

```

Key Algorithms

① Introduction

Important Points

Software Challenges

② Software Testing

Basic Concepts

Testing Opportunities

③ Regression Testing

Supporting Methods

Key Algorithms

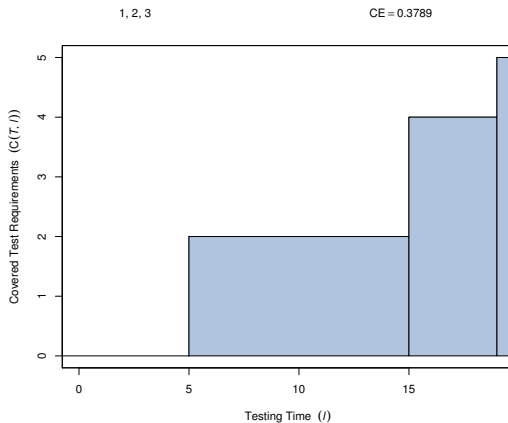
④ Empirical Evaluation

Model for Experimentation

Concrete Examples

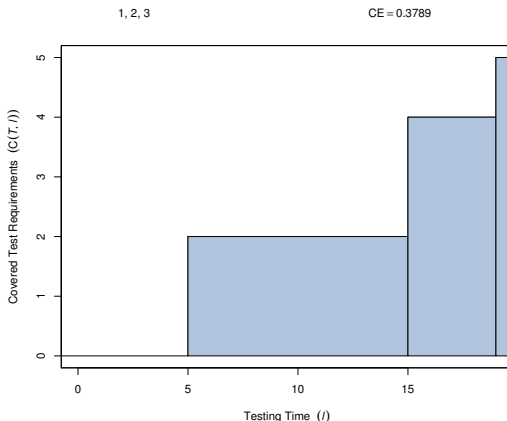
⑤ Conclusion

Importance of Test Suite Prioritization



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

Importance of Test Suite Prioritization

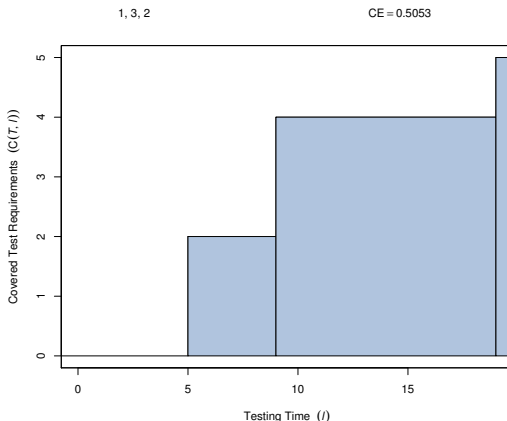


Test Orderings

1,2,3

Original ordering exhibits poor effectiveness score - **CE = 0.3789**

Importance of Test Suite Prioritization



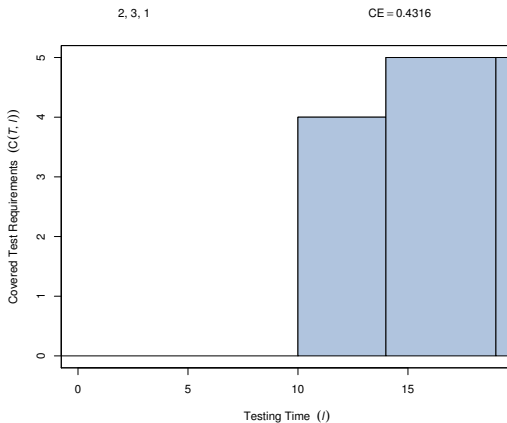
Test Orderings

1,2,3

1,3,2

Different ordering improves the effectiveness score - CE = 0.5053

Importance of Test Suite Prioritization



Test Orderings

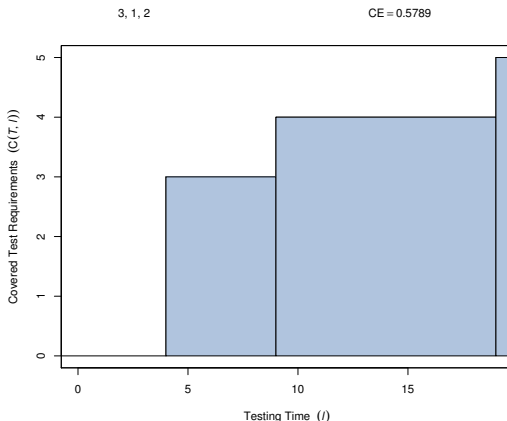
1,2,3

1,3,2

2,3,1

Some orderings have less improved scores - CE = 0.4316

Importance of Test Suite Prioritization



Test Orderings

1,2,3

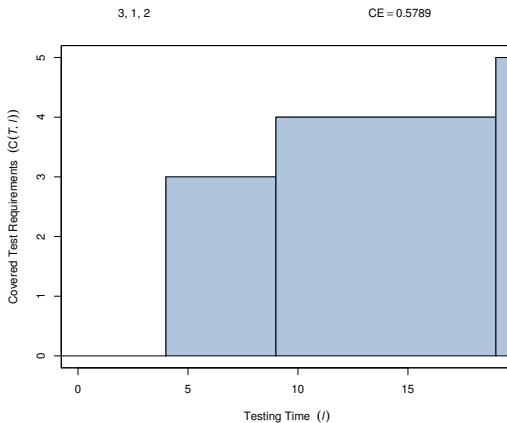
1,3,2

2,3,1

3,1,2

Best ordering shows a higher effectiveness scores - **CE = 0.5789**

Importance of Test Suite Prioritization



Test Orderings

1,2,3

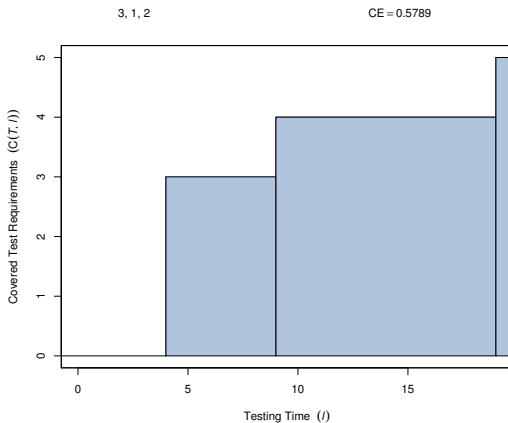
1,3,2

2,3,1

3,1,2

Greedy methods often produce high-effectiveness orderings

Importance of Test Suite Prioritization



Test Orderings

1,2,3

1,3,2

2,3,1

3,1,2

Search-based techniques may have some desirable characteristics

Greedy Algorithms

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

Greedy Algorithms

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

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

Greedy Algorithms

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

 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}

Greedy Algorithms

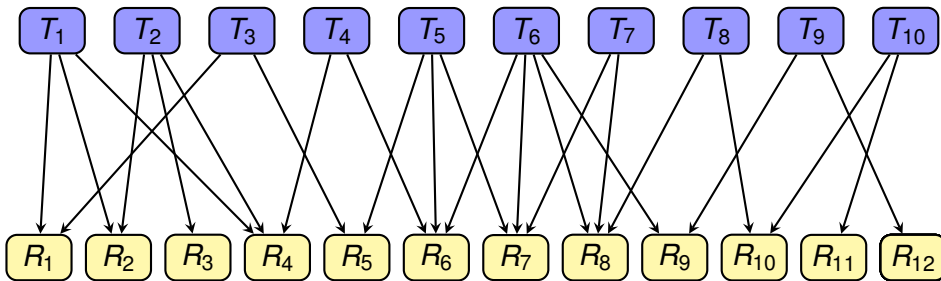
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$

 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}

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

Greedy Algorithms

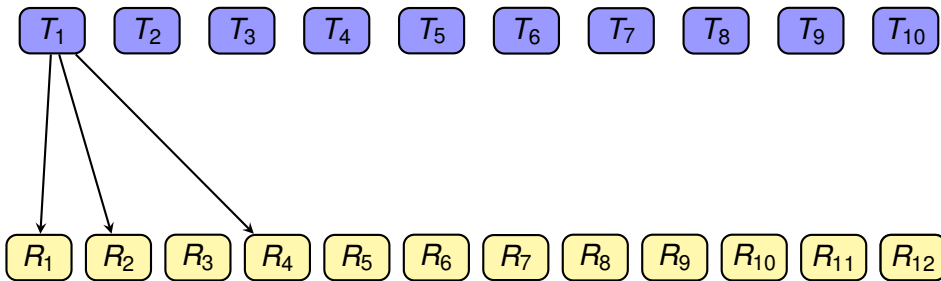
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$



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

Greedy Algorithms

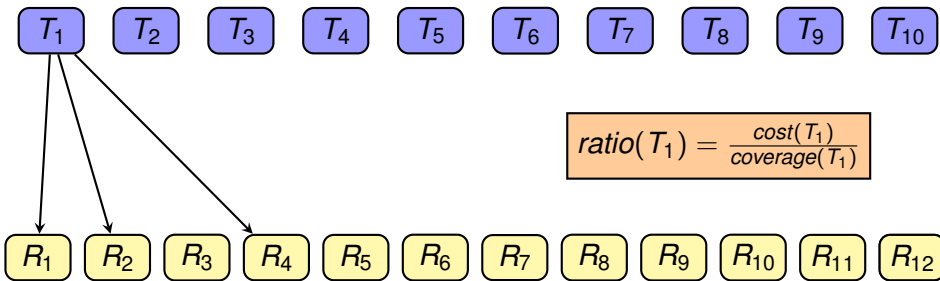
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$



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

Greedy Algorithms

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

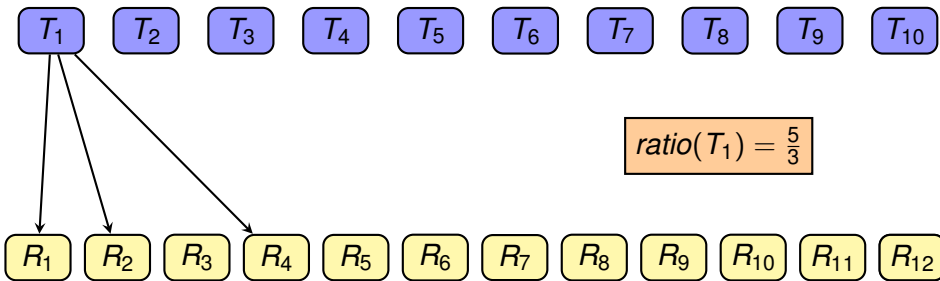


$$\text{ratio}(T_1) = \frac{\text{cost}(T_1)}{\text{coverage}(T_1)}$$

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

Greedy Algorithms

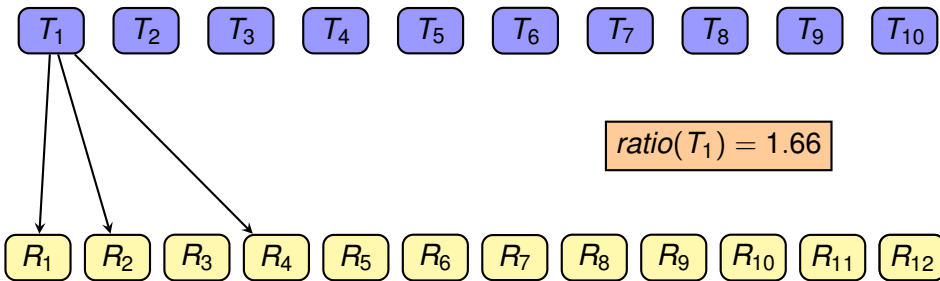
Test Suite $T = \langle T_1, T_2, \dots, T_9, T_{10} \rangle$



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

Greedy Algorithms

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



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

Greedy Algorithms

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

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

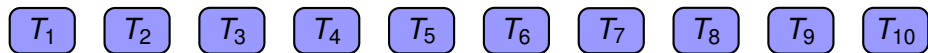
1.66

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

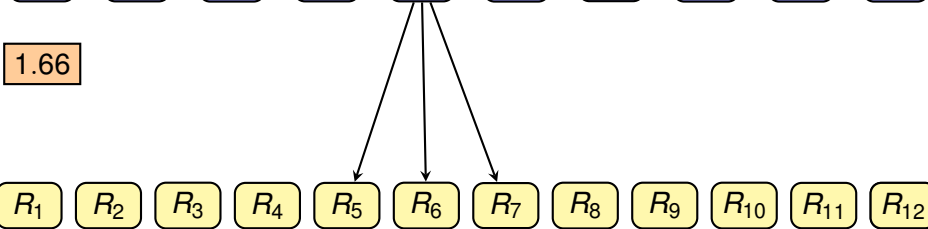
Requirements Set $R = \{R_1, R_2, \dots, R_{11}, R_{12}\}$

Greedy Algorithms

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



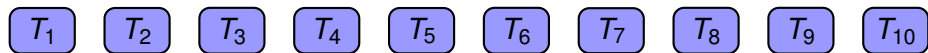
1.66



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

Greedy Algorithms

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



1.66

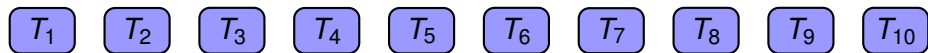
$$\text{ratio}(T_5) = \frac{\text{cost}(T_5)}{\text{coverage}(T_5)}$$



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

Greedy Algorithms

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



1.66

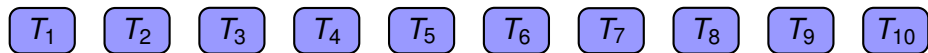
$$\text{ratio}(T_5) = \frac{8}{3}$$



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

Greedy Algorithms

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



1.66

$ratio(T_5) = 2.66$



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

Greedy Algorithms

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

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

1.66

2.66

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

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

Greedy Algorithms

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

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

1.66

2.66

$ratio(T_1) < ratio(T_5)$
Prefer T_1 over T_5

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

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

Greedy Algorithms

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

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

1.66

2.66

Proceed incrementally,
picking the test case with the
lowest *ratio* value for the
uncovered requirements

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

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

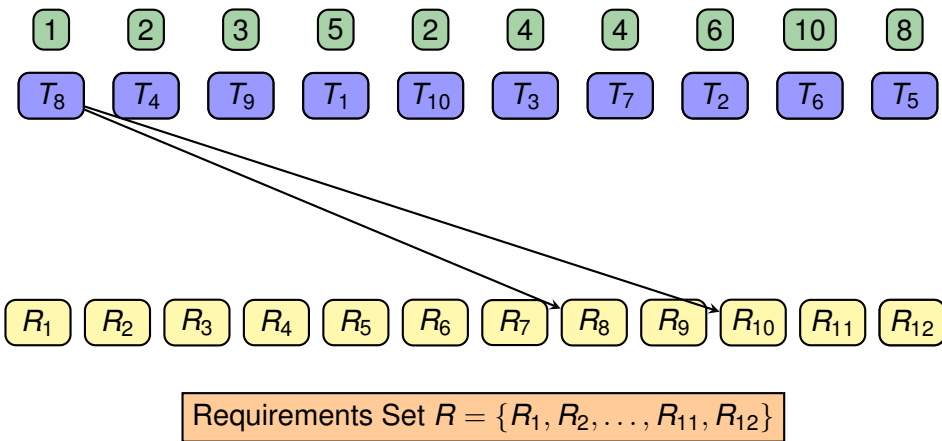
Greedy Algorithms

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

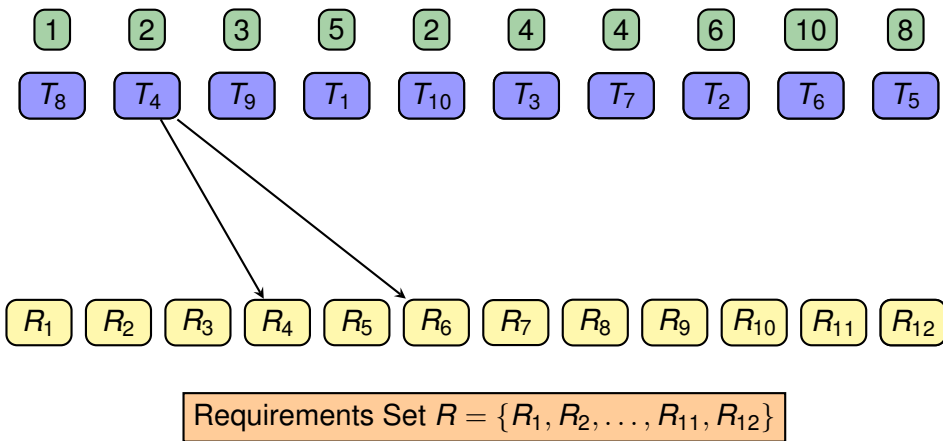
 T_8 T_4 T_9 T_1 T_{10} T_3 T_7 T_2 T_6 T_5 R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12}

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

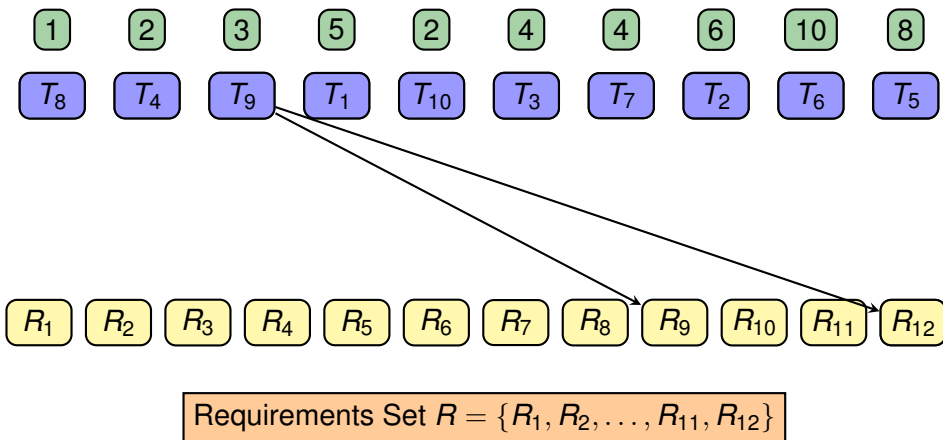
Greedy Algorithms



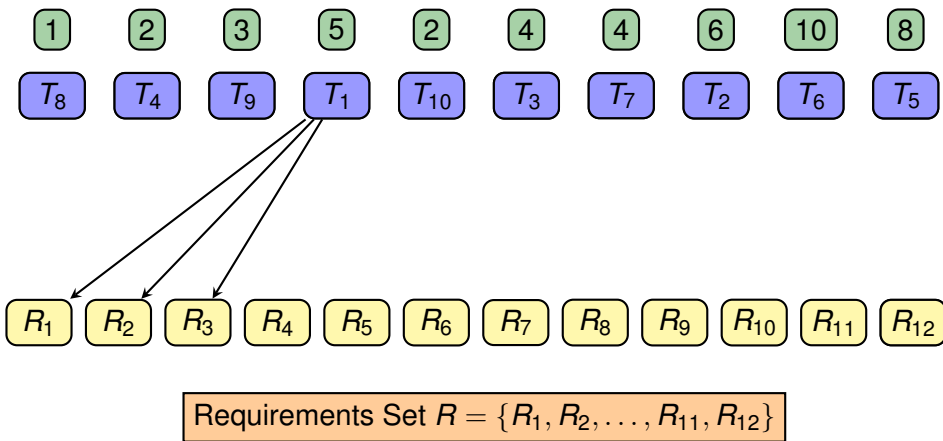
Greedy Algorithms



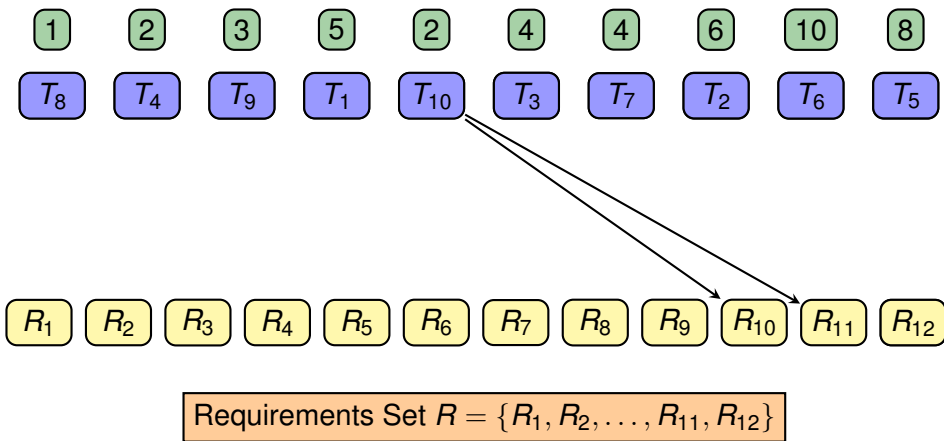
Greedy Algorithms



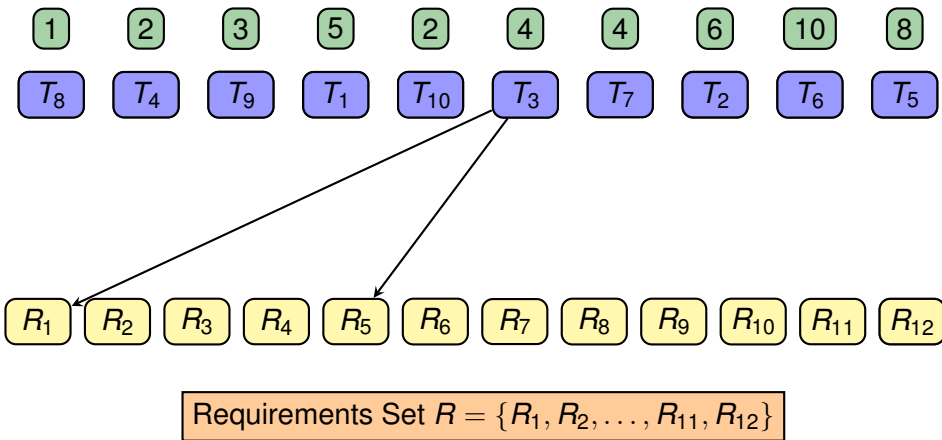
Greedy Algorithms



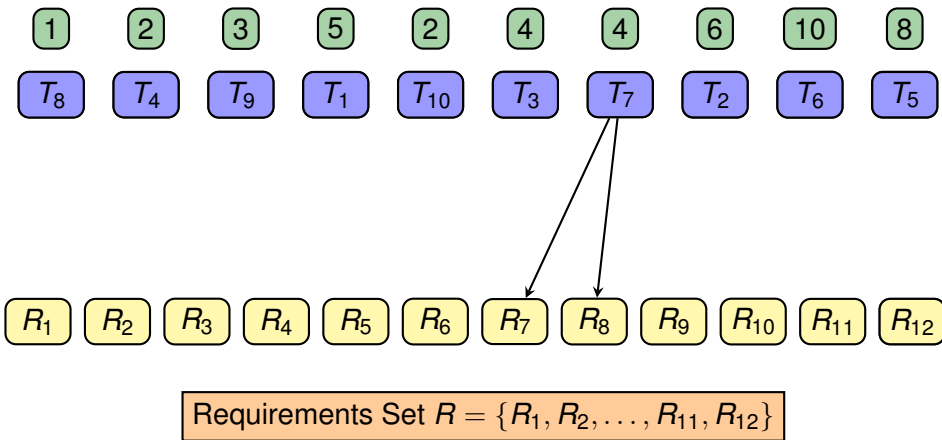
Greedy Algorithms



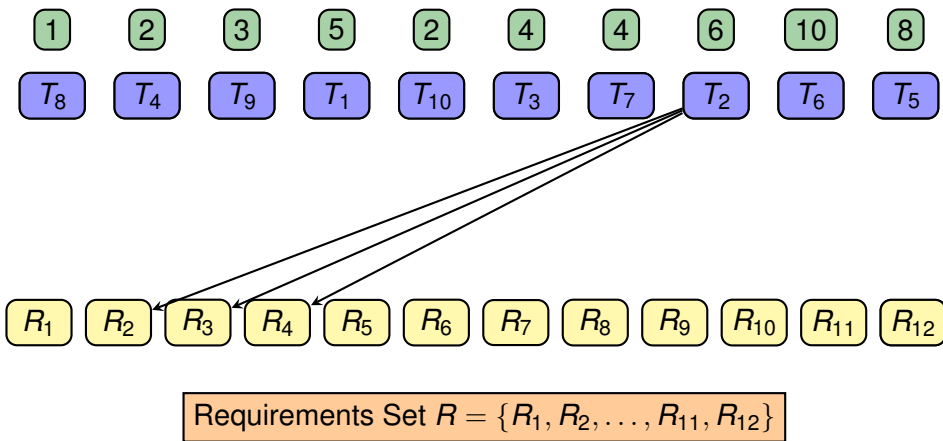
Greedy Algorithms



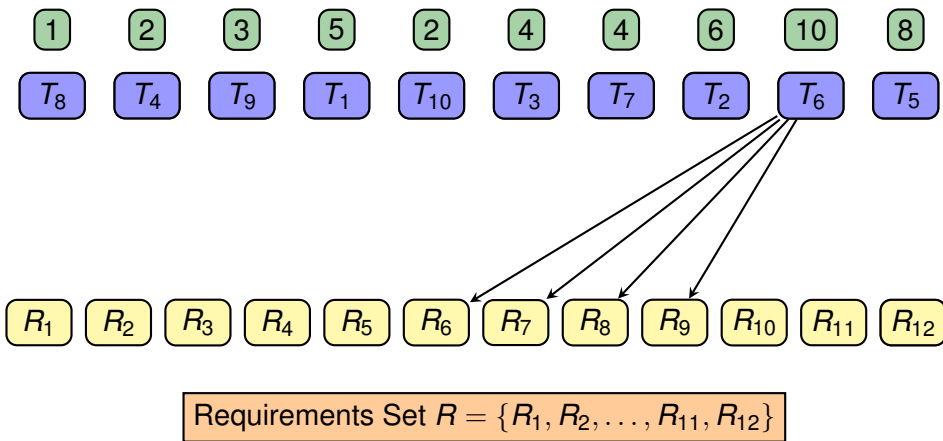
Greedy Algorithms



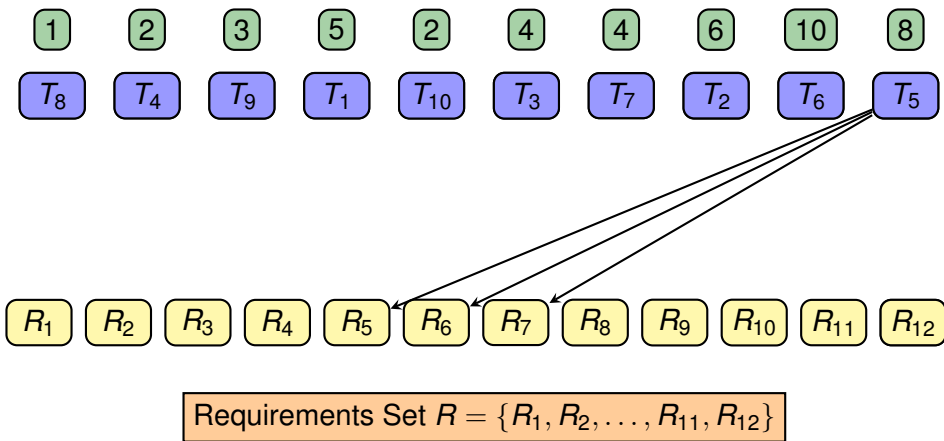
Greedy Algorithms



Greedy Algorithms

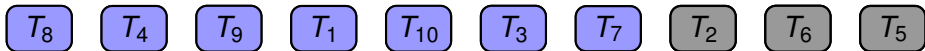


Greedy Algorithms



Greedy Algorithms

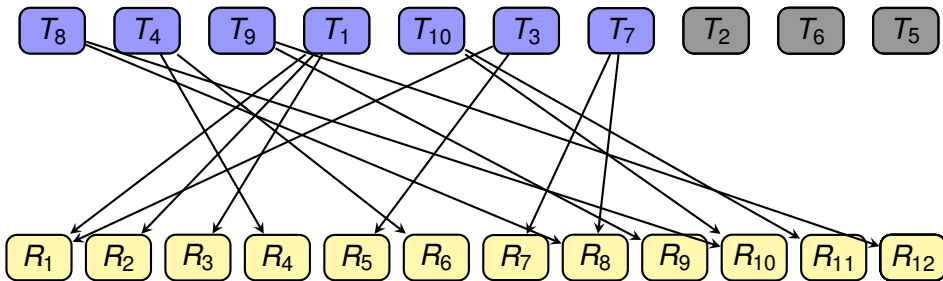
Test Suite $T = \langle T_8, T_4, T_9, T_1, T_{10}, T_3, T_7 \rangle$



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

Greedy Algorithms

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



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

Hands-on: Running the Greedy Algorithms

1

Load the Regression Testing Techniques

```
cd modificare-installation-directory  
R  
source (``FDP_Start.R``)  
zLoadFit()  
zLoadGRD()  
zLoadHC()  
zLoadGA()  
zLoadProduceOrderingReductionDataFile()
```


Hands-on: Running the Greedy Algorithms

2

Run the Greedy Reduction Algorithm

```
x1<-GRD_reduction(coverageMatrixName=
  "reqMatrices/StatementCoverageMatrices/
  Sudoku_1_line_true_Coverage.dat",
  timingsFileName="reqMatrices/
  TimingsFiles/Sudoku_Timing.dat")
```

Hands-on: Running the Greedy Algorithms

3

Run the Greedy Prioritization Algorithm

```
x2<-GRD (coverageMatrixName=
  "reqMatrices/StatementCoverageMatrices/
  Sudoku_1_line_true_Coverage.dat",
  timingsFileName="reqMatrices/
  TimingsFiles/Sudoku_Timing.dat")
```

Hands-on: Running the Greedy Algorithms

4

Map the Test Indices to Test Names

```
y1<-produceOrderingReductionDataFile  
  (testSuite=x1$Ord,  
   timingsFile="reqMatrices/  
   TimingsFiles/Sudoku_Timing.dat")
```

```
y2<-produceOrderingReductionDataFile  
  (testSuite=x2$Ord,  
   timingsFile="reqMatrices/  
   TimingsFiles/Sudoku_Timing.dat")
```

Hands-on: Running the Greedy Algorithms

5

Store the Test Case Timings

```
write.table(y1, file="path-to-proteja/  
SK_ordering1.dat", row.names=FALSE,  
col.names=FALSE, quote=FALSE)
```

```
write.table(y2, file="path-to-proteja/  
SK_ordering2.dat", row.names=FALSE,  
col.names=FALSE, quote=FALSE)
```

Hands-on: Running the Greedy Algorithms

6

Execute the Reduced Test Suite

```
cd proteja-installation-directory  
ant modifyTestSuite -DmodificationFile=  
    "SK_ordering1.dat"  
ant proteja
```

Hands-on: Running the Greedy Algorithms

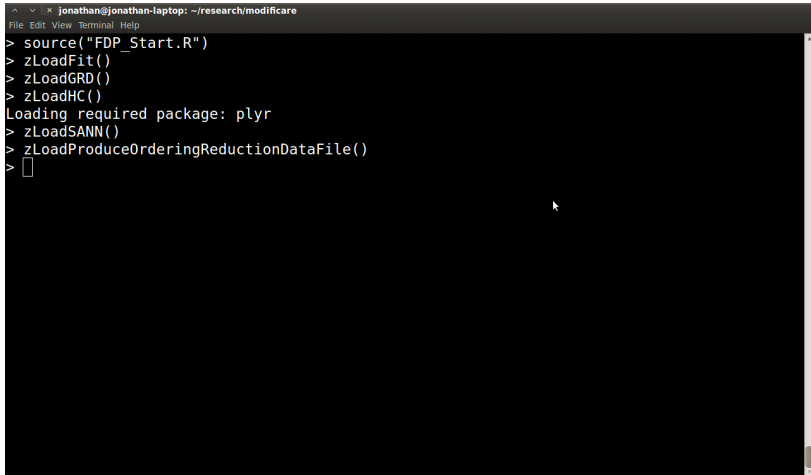
7

Execute the Prioritized Test Suite

```
cd proteja-installation-directory  
ant modifyTestSuite -DmodificationFile=  
    "SK_ordering2.dat"  
ant proteja
```

Key Algorithms

Screenshot: Loading Regression Testing



```
jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help
> source("FDP_Start.R")
> zLoadFit()
> zLoadGRD()
> zLoadHC()
Loading required package: plyr
> zLoadSANN()
> zLoadProduceOrderingReductionDataFile()
> □
```

Key Algorithms

Screenshot: Running the Greedy Reducer

```

^  v  x  jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help
> x1<-GRD_reduction(coverageMatrixName="reqMatrices/StatementCoverageMatrices/Sudoku_1_line_true_Coverage.dat",timingsFileName="reqMatrices/TimingsFiles/Sudoku_Timing.dat",seed=100)
> x1
$Ord
[1] 25 20  8 23 10 14 15  3 13  2 16  9

$Fit
[1] 0.7590638

$Pri
[1] "GRD_reduction"

> 

```


Screenshot: Running the Greedy Prioritizer

```
jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help
> x2<-GRD(coverageMatrixName="reqMatrices/StatementCoverageMatrices/Sudoku_1_line_true_Cov
> x2
$Ord
[1] 25 20 8 23 10 14 15 3 13 2 16 9 22 17 7 12 1 21 18 4 24 19 11 5 6

$Fit
[1] 0.7590638

$Pri
[1] "GRD"

> □
```

Screenshot: Test Index to Test Name Mapping

```
jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help
> y1<-produceOrderingReductionDataFile(testSuite=x1$Ord,timingsFile="reqMatrices/TimingsFiles/Sudoku_Timing.dat")
> y1
[1] "sudoku.TestSudoku.testInitialize_Null"
[2] "sudoku.TestSudoku.testGetPeers_Unit1"
[3] "sudoku.TestSudoku.testGetUnsolvedBoxes"
[4] "sudoku.TestSudoku.testPossibleValues_InitiateBoxState"
[5] "sudoku.TestSudoku.testInitialize_NotSquare"
[6] "sudoku.TestSudoku.testInitialize_NoSeededValues_Zeros"
[7] "sudoku.TestSudoku.testSolvedValue_Error"
[8] "sudoku.TestSudoku.testSolvedValues_SomeValuesMissing"
[9] "sudoku.TestSudoku.testInitialize_NoSeededValues"
[10] "sudoku.TestSudoku.testSolvedValues"
[11] "sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_Column"
```

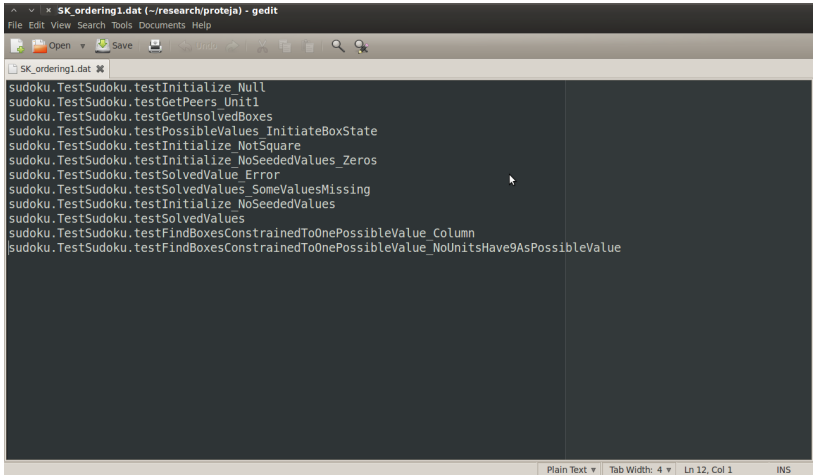
Screenshot: Storing the Test Case Timings



```
jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help
> write.table(y1,file=~ /research/proteja/SK_ordering1.dat",row.names=FALSE,col.names=FALSE,quote=FALSE)
> write.table(y2,file=~ /research/proteja/SK_ordering2.dat",row.names=FALSE,col.names=FALSE,quote=FALSE)
> 
```

Key Algorithms

Screenshot: The Reduced Test Suite

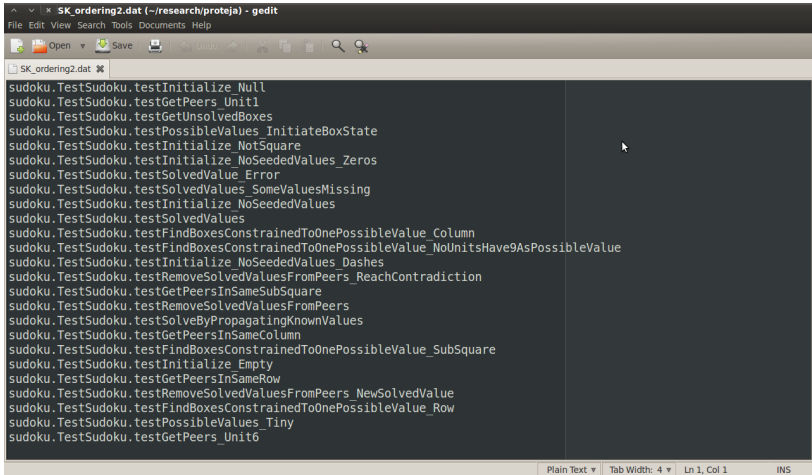


The screenshot shows a gedit editor window titled "SK_ordering1.dat (~/.research/proteja) - gedit". The window contains a list of test cases for the Sudoku solver. The test cases are:

```
sudoku.TestSudoku.testInitialize_Null  
sudoku.TestSudoku.testGetPeers_Unit1  
sudoku.TestSudoku.testGetUnsolvedBoxes  
sudoku.TestSudoku.testPossibleValues_InitiateBoxState  
sudoku.TestSudoku.testInitialize_NotSquare  
sudoku.TestSudoku.testInitialize_NoSeededValues_Zeros  
sudoku.TestSudoku.testSolvedValue_Error  
sudoku.TestSudoku.testSolvedValues_SomeValuesMissing  
sudoku.TestSudoku.testInitialize_NoSeededValues  
sudoku.TestSudoku.testSolvedValues  
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_Column  
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_NoUnitsHave9AsPossibleValue
```

The status bar at the bottom of the window indicates "Plain Text", "Tab Width: 4", "Ln 12, Col 1", and "INS".

Screenshot: The Prioritized Test Suite

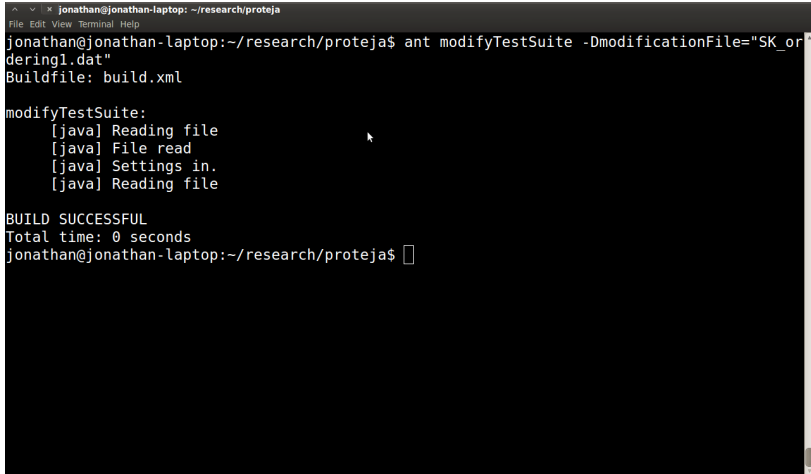


The screenshot shows a gedit editor window titled "SK_ordering2.dat (~/.research/proteja) - gedit". The editor contains a list of 30 test cases for a Sudoku solver, ordered by priority. The test cases are as follows:

```
sudoku.TestSudoku.testInitialize_Null  
sudoku.TestSudoku.testGetPeers_Unit1  
sudoku.TestSudoku.testGetUnsolvedBoxes  
sudoku.TestSudoku.testPossibleValues_InitiateBoxState  
sudoku.TestSudoku.testInitialize_NotSquare  
sudoku.TestSudoku.testInitialize_NoSeededValues_Zeros  
sudoku.TestSudoku.testSolvedValue_Error  
sudoku.TestSudoku.testSolvedValues_SomeValuesMissing  
sudoku.TestSudoku.testInitialize_NoSeededValues  
sudoku.TestSudoku.testSolvedValues  
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_Column  
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_NoUnitsHave9AsPossibleValue  
sudoku.TestSudoku.testInitialize_NoSeededValues_Dashes  
sudoku.TestSudoku.testRemoveSolvedValuesFromPeers_ReachContradiction  
sudoku.TestSudoku.testGetPeersInSameSubSquare  
sudoku.TestSudoku.testRemoveSolvedValuesFromPeers  
sudoku.TestSudoku.testSolveByPropagatingKnownValues  
sudoku.TestSudoku.testGetPeersInSameColumn  
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_SubSquare  
sudoku.TestSudoku.testInitialize_Empty  
sudoku.TestSudoku.testGetPeersInSameRow  
sudoku.TestSudoku.testRemoveSolvedValuesFromPeers_NewSolvedValue  
sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_Row  
sudoku.TestSudoku.testPossibleValues_Tiny  
sudoku.TestSudoku.testGetPeers_Unit6
```

The editor interface includes a menu bar (File, Edit, View, Search, Tools, Documents, Help), a toolbar with icons for Open, Save, Undo, and other actions, and a status bar at the bottom showing "Plain Text", "Tab Width: 4", "Ln 1, Col 1", and "INS".

Screenshot: Preparing the Reduced Test Suite

A terminal window titled 'jonathan@jonathan-laptop: ~/research/proteja' with a menu bar (File, Edit, View, Terminal, Help). The prompt is 'jonathan@jonathan-laptop:~/research/proteja\$'. The command 'ant modifyTestSuite -DmodificationFile="SK_ordering1.dat"' has been entered. The output shows 'Buildfile: build.xml', followed by a 'modifyTestSuite:' section with four lines of Java output: '[java] Reading file', '[java] File read', '[java] Settings in.', and '[java] Reading file'. Below this, it says 'BUILD SUCCESSFUL' and 'Total time: 0 seconds'. The prompt returns to 'jonathan@jonathan-laptop:~/research/proteja\$' with a cursor.

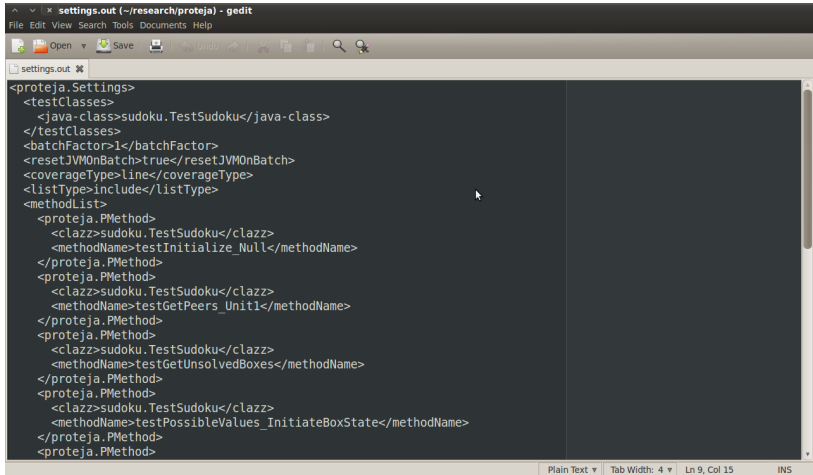
```
jonathan@jonathan-laptop: ~/research/proteja
File Edit View Terminal Help
jonathan@jonathan-laptop:~/research/proteja$ ant modifyTestSuite -DmodificationFile="SK_ordering1.dat"
Buildfile: build.xml

modifyTestSuite:
    [java] Reading file
    [java] File read
    [java] Settings in.
    [java] Reading file

BUILD SUCCESSFUL
Total time: 0 seconds
jonathan@jonathan-laptop:~/research/proteja$
```

Key Algorithms

Screenshot: Viewing the Reduced Test Suite



```
settings.out (~/.research/proteja) - gedit
File Edit View Search Tools Documents Help
[Icons: Open, Save, Undo, Redo, Find, etc.]
settings.out
<proteja.Settings>
  <testClasses>
    <java-class>sudoku.TestSudoku</java-class>
  </testClasses>
  <batchFactor>1</batchFactor>
  <resetJVMOnBatch>true</resetJVMOnBatch>
  <coverageType>line</coverageType>
  <listType>include</listType>
  <methodList>
    <proteja.PMethod>
      <clazz>sudoku.TestSudoku</clazz>
      <methodName>testInitialize_Null</methodName>
    </proteja.PMethod>
    <proteja.PMethod>
      <clazz>sudoku.TestSudoku</clazz>
      <methodName>testGetPeers_Unit1</methodName>
    </proteja.PMethod>
    <proteja.PMethod>
      <clazz>sudoku.TestSudoku</clazz>
      <methodName>testGetUnsolvedBoxes</methodName>
    </proteja.PMethod>
    <proteja.PMethod>
      <clazz>sudoku.TestSudoku</clazz>
      <methodName>testPossibleValues_InitiateBoxState</methodName>
    </proteja.PMethod>
  </methodList>
</proteja.Settings>
```

Plain Text ▾ Tab Width: 4 ▾ Ln 9, Col 15 INS

Screenshot: Running the Reduced Test Suite

```
jonathan@jonathan-laptop: ~/research/proteja
File Edit View Terminal Help
jonathan@jonathan-laptop:~/research/proteja$ ant proteja
Buildfile: build.xml

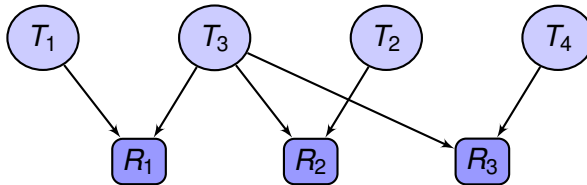
proteja:
[java] Reading file
[java] File read
[java] Settings in.
[java] include list found!
[java] Starting new JVM
[java]
[java]
[java] run:
[java] [java] testInitialize_Null passed.
[java] Starting new JVM
[java]
[java]
[java] run:
[java] [java] testGetPeers_Unit1 passed.
[java] Starting new JVM
[java]
[java]
[java] run:
[java] [java] testGetUnsolvedBoxes passed.
[java] Starting new JVM
[java]
```


Limitations of Greedy Algorithms



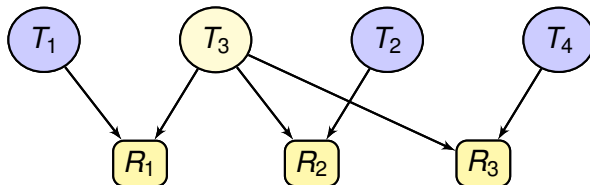
Possible configuration of the **coverage report**

Limitations of Greedy Algorithms



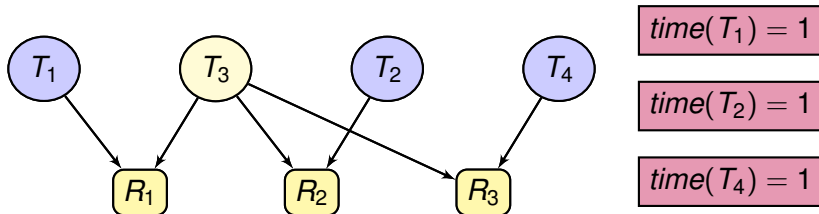
Possible configuration of the **coverage report**

Limitations of Greedy Algorithms



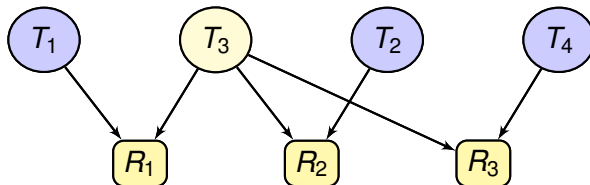
Possible configuration of the **coverage report**

Limitations of Greedy Algorithms



Execution time of the test cases may mislead greedy

Limitations of Greedy Algorithms



$$time(T_1) = 1$$

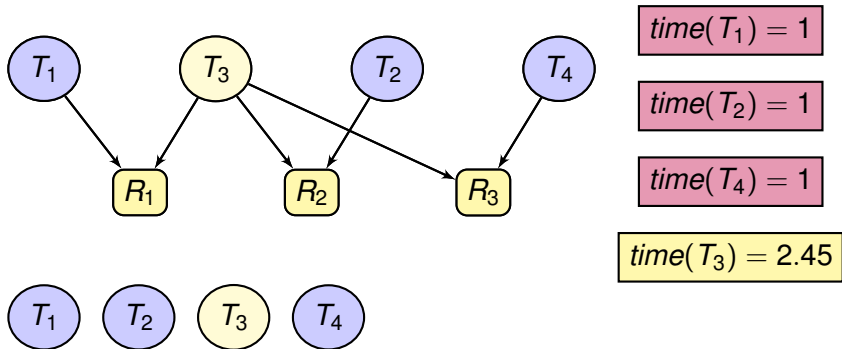
$$time(T_2) = 1$$

$$time(T_4) = 1$$

$$time(T_3) = 2.45$$

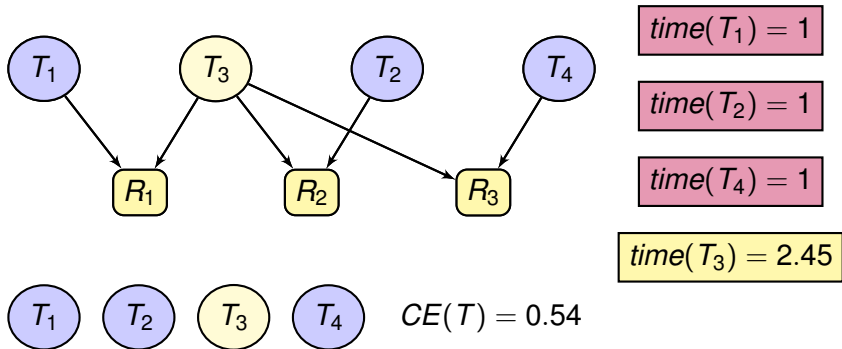
Execution time of the test cases may mislead greedy

Limitations of Greedy Algorithms



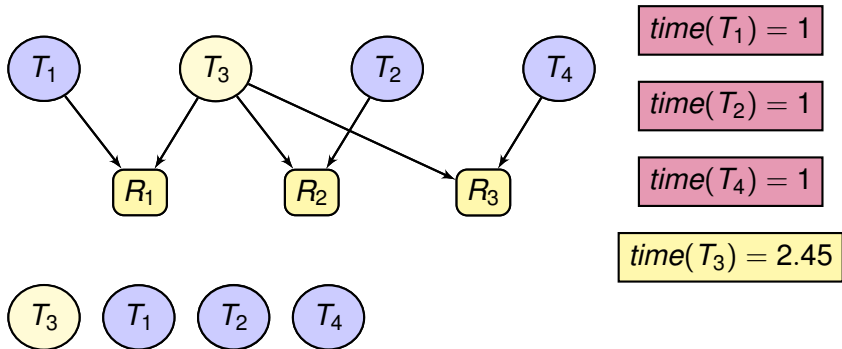
Original ordering has low effectiveness score

Limitations of Greedy Algorithms



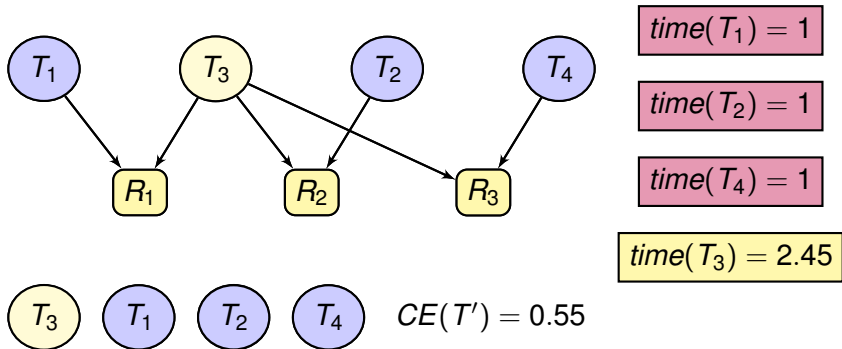
Original ordering has low effectiveness score

Limitations of Greedy Algorithms



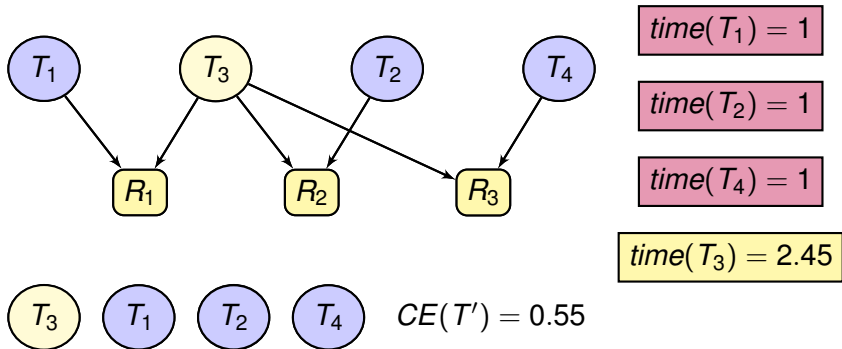
Greedy method constructs suite with marginal improvement

Limitations of Greedy Algorithms



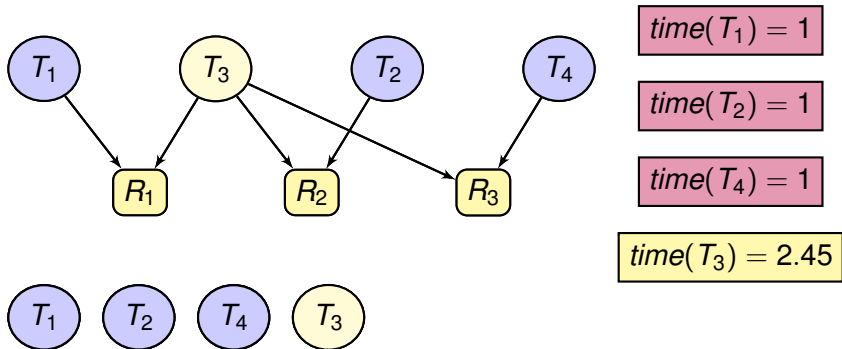
Greedy method constructs suite with marginal improvement

Limitations of Greedy Algorithms



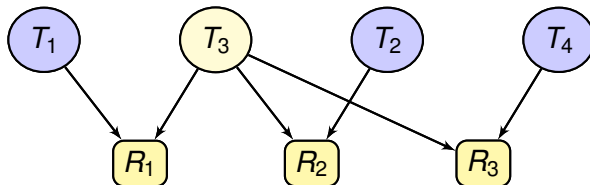
Greedy can exhibit high run-times (Jiang et al. ASE 2009)

Limitations of Greedy Algorithms



Genetic may find **better** orderings (Conrad et al. GECCO 2010)

Limitations of Greedy Algorithms

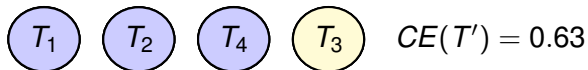


$$\text{time}(T_1) = 1$$

$$\text{time}(T_2) = 1$$

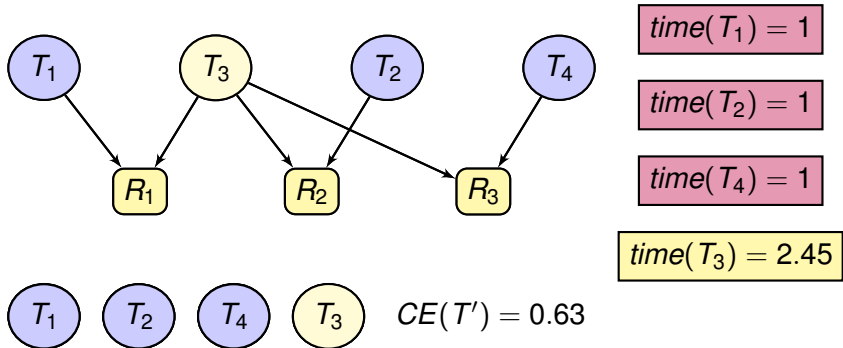
$$\text{time}(T_4) = 1$$

$$\text{time}(T_3) = 2.45$$



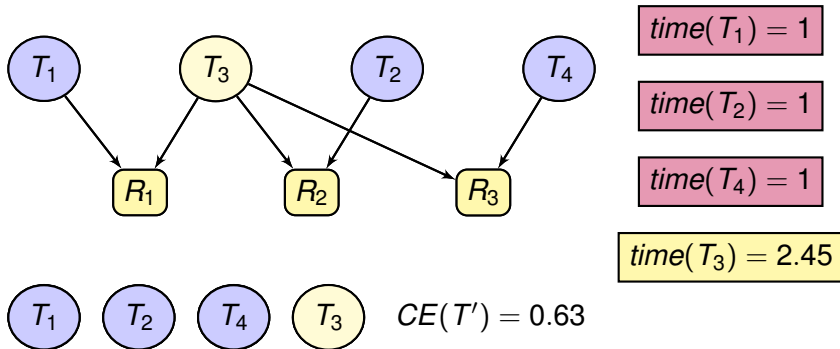
Genetic may find **better** orderings (Conrad et al. GECCO 2010)

Limitations of Greedy Algorithms



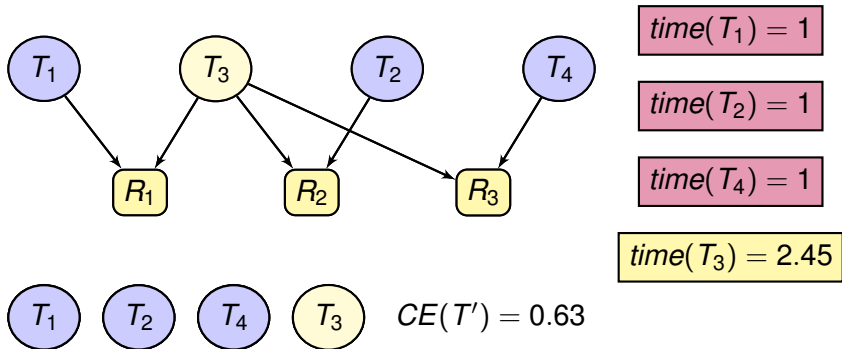
Search-based algorithms are amenable to **parallelization**

Limitations of Greedy Algorithms



Search-based algorithms support “human in the loop”

Limitations of Greedy Algorithms



Search-based algorithms construct **diverse** test orderings

Hill Climbing Algorithm

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

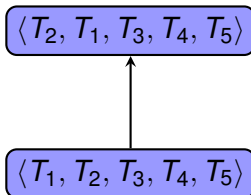
Hill Climbing Algorithm

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

$\langle T_1, T_2, T_3, T_4, T_5 \rangle$

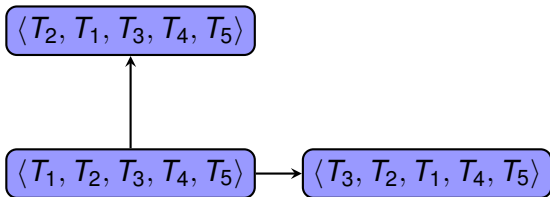
Hill Climbing Algorithm

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



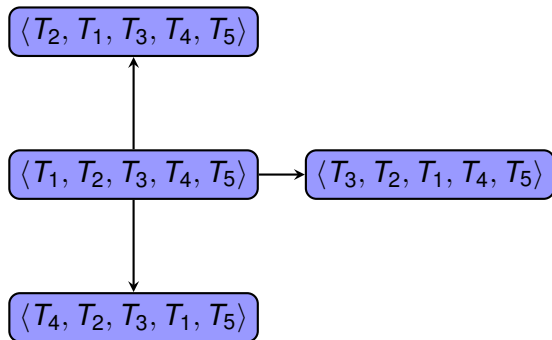
Hill Climbing Algorithm

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



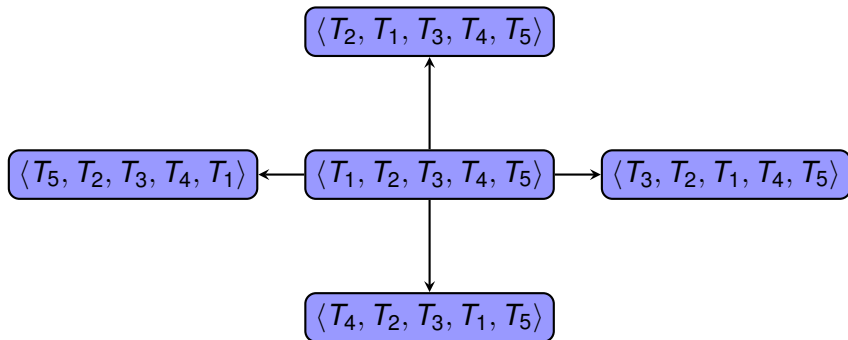
Hill Climbing Algorithm

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

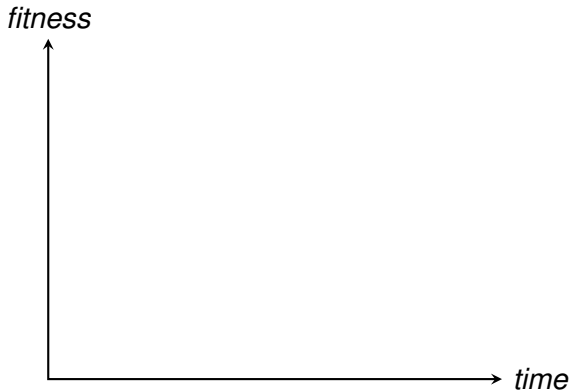


Hill Climbing Algorithm

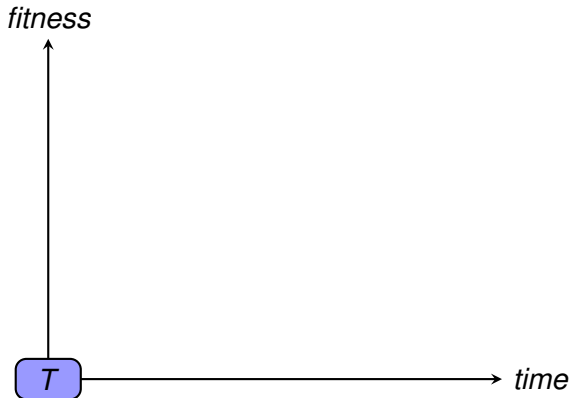
Explore the “neighborhood” of test suites from a starting point



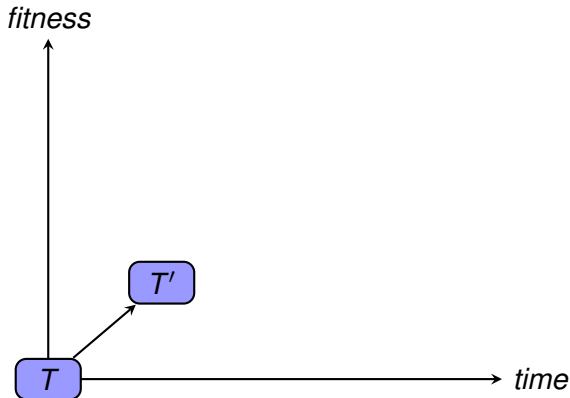
Hill Climbing Algorithm



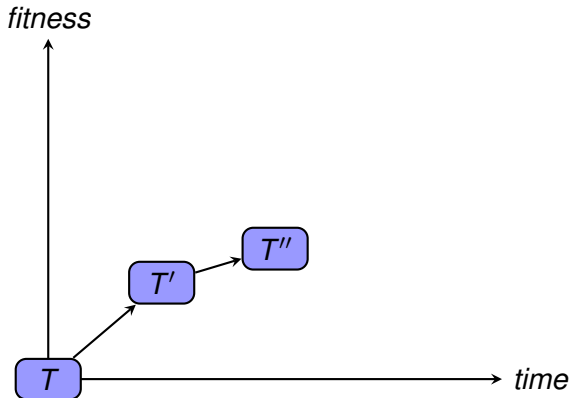
Hill Climbing Algorithm



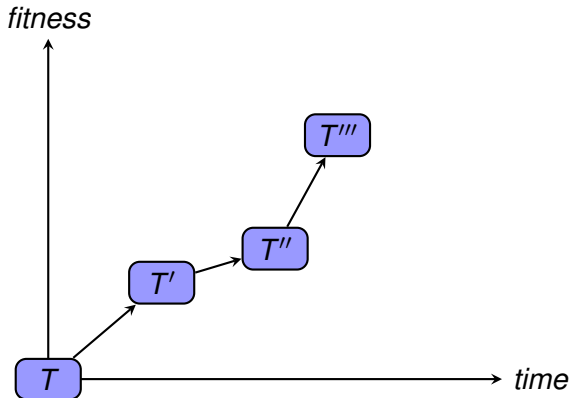
Hill Climbing Algorithm



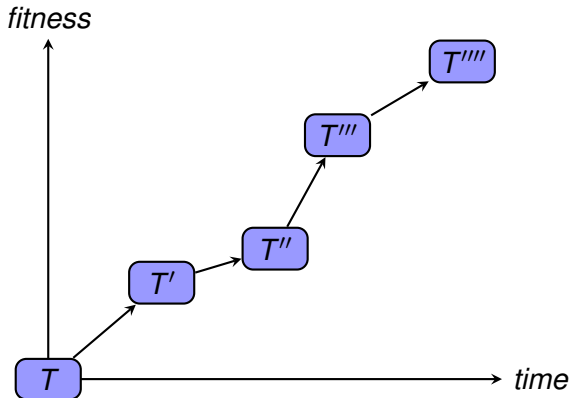
Hill Climbing Algorithm



Hill Climbing Algorithm

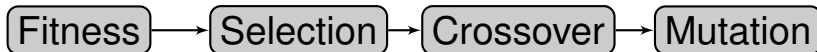


Hill Climbing Algorithm



Genetic Algorithm

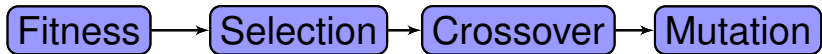
Initialize



Randomly create suites by repeatedly **shuffling** $\langle T_1, \dots, T_n \rangle$

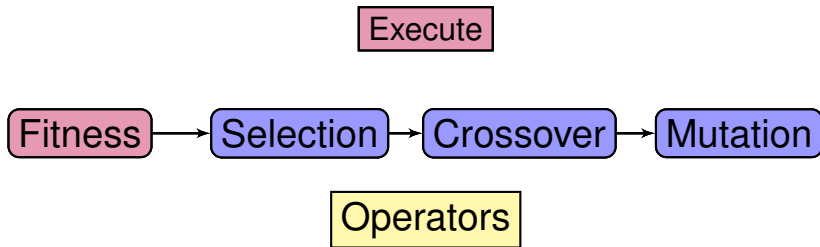
Genetic Algorithm

Execute



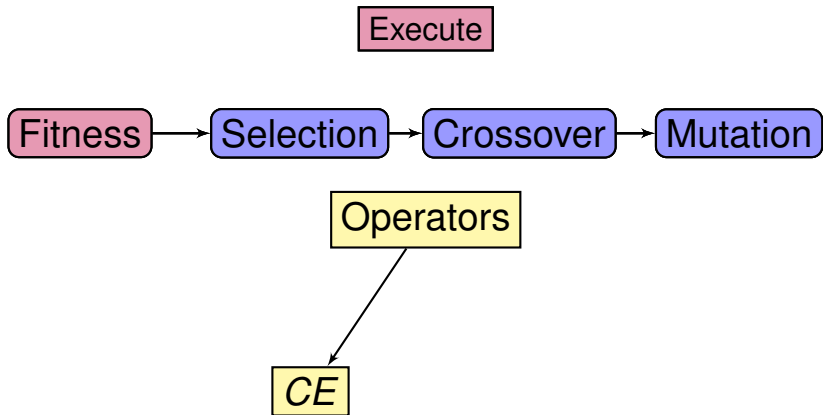
Execute the phases until the genetic algorithm **stagnates**

Genetic Algorithm



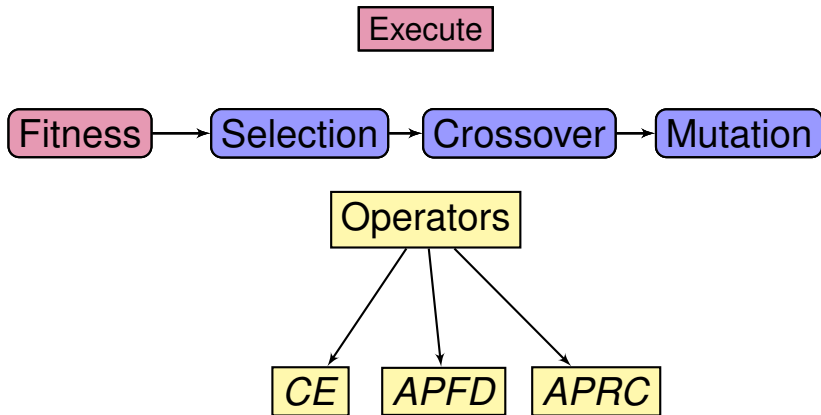
Use a “higher is better” effectiveness metric

Genetic Algorithm



Use a “higher is better” effectiveness metric

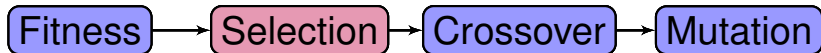
Genetic Algorithm



Use a “higher is better” effectiveness metric

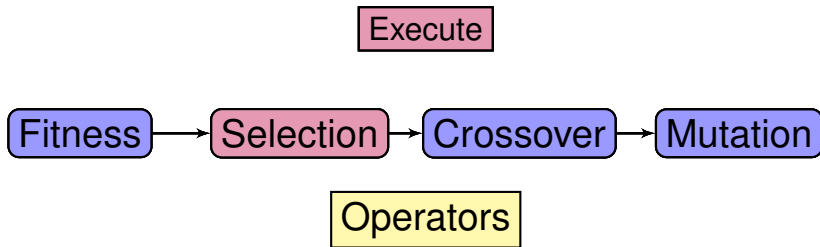
Genetic Algorithm

Execute



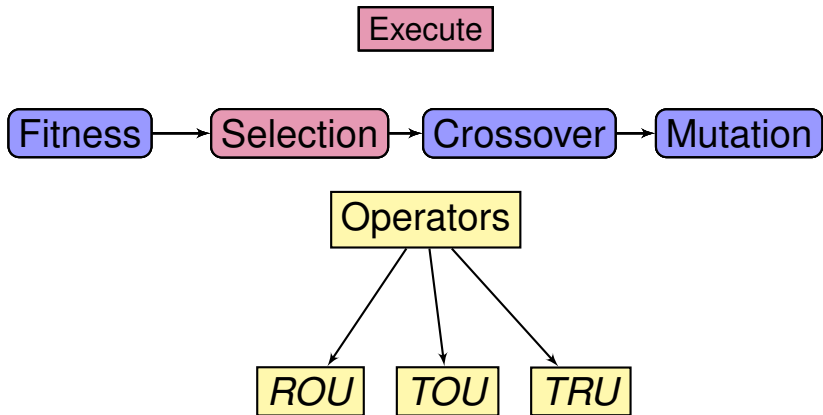
Choose orderings to become **parents** of next generation

Genetic Algorithm



Choose orderings to become **parents** of next generation

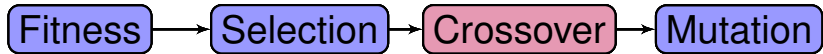
Genetic Algorithm



Choose orderings to become **parents** of next generation

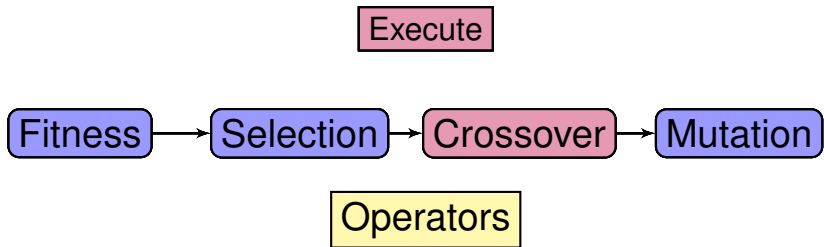
Genetic Algorithm

Execute



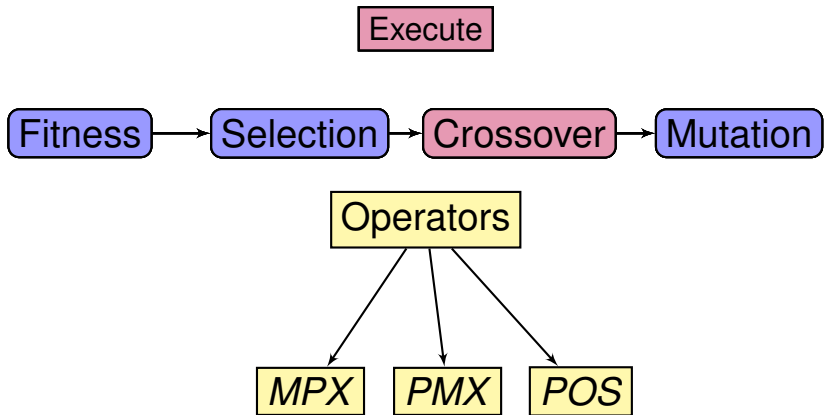
Five possible operators **combine** parents to produce children

Genetic Algorithm



Five possible operators **combine** parents to produce children

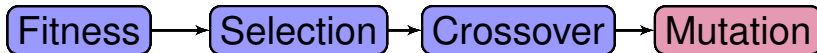
Genetic Algorithm



Five possible operators **combine** parents to produce children

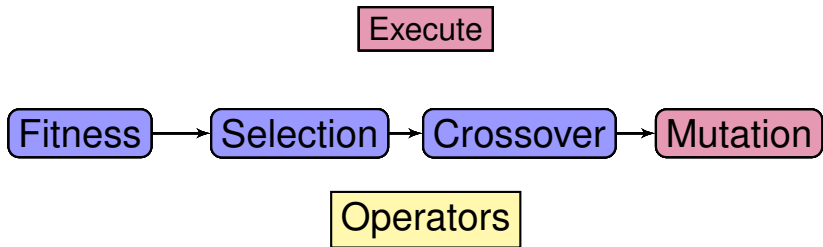
Genetic Algorithm

Execute



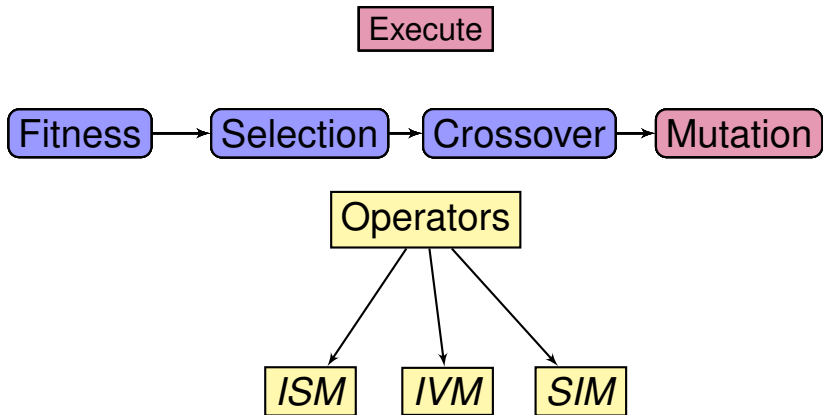
Six possible operators make random **changes** to orderings

Genetic Algorithm



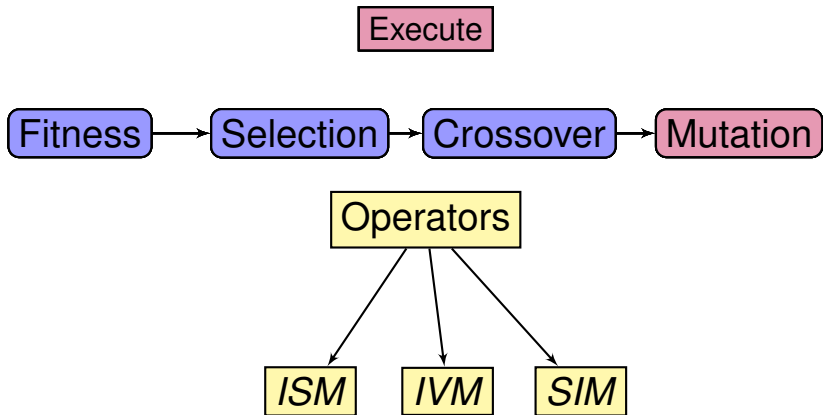
Six possible operators make random **changes** to orderings

Genetic Algorithm



Six possible operators make random **changes** to orderings

Genetic Algorithm



See (Conrad et. al, GECCO 2010) for additional details

Hands-on: Running the Search-Based Algorithms

1

Run the Hill Climbing Prioritization Algorithm

```
HC_FA(lFM=makeLogFM(read.table(  
  "reqMatrices/StatementCoverageMatrices/  
  Sudoku_1_line_true_Coverage.dat")),  
  NG="NG_FS", Seed=100)
```

Hands-on: Running the Search-Based Algorithms

2

Run the Genetic Prioritization Algorithm

```
GA(lFM=makeLogFM(read.table("reqMatrices/
StatementCoverageMatrices/
Sudoku_1_line_true_Coverage.dat")) ,
  cChildDens=.6, cMutRate=.2, cPopSize=20,
  CrossOp="CO_OX1", MutOp="MO_DM",
  SelOp="SO_ROU", TransOp="TO_EXP",
  TermCond="TermTotalIt",
  EndCond=10, Seed=100)
```

Screenshot: Running the Hill Climbing Algorithm

```

^ v x jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help
> HC_FA(lfm=makeLogFM(read.table("reqMatrices/StatementCoverageMatrices/Sudoku_1_line_true
Coverage.dat")),NG="NG_FS",Seed=100)
$Ord
[1] 25 7 8 2 10 13 16 24 20 3 17 23 4 5 9 18 21 19 22 12 15 6 11 14 1

$Fit
[1] 0.7068085

$Pri
[1] "HC_FA"

$Conf
[1] "NG_FS"

$TotalIt
[1] 4

$Seed
[1] 100

> █

```

Screenshot: Running the Genetic Algorithm

```

jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help
> GA(lfM=makeLogFM(read.table("reqMatrices/StatementCoverageMatrices/Sudoku_1_line_true Co
verage.dat")),cChildDens=.6,cMutRate=.2,cPopSize=20,CrossOp="C0_0X1",MutOp="M0_DM",SelOp="
S0_R0U",TransOp="T0_EXP",TermCond="TermTotalIt",EndCond=10,Seed=100)
$Ord
[1]  2  5 24  9 10  1 14 18  6 11 13 23  7 21  8  3 22 16 19  4 17 15 25 20 12

$Fit
[1] 0.752766

$CPopSize
[1] 20

$cChildDens
[1] 0.6

$cMutRate
[1] 0.2

$CrossOp
[1] "C0_0X1"

$MutOp
[1] "M0_DM"

$SelOp

```

Key Algorithms

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
1: read("Enter 3 numbers: ",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓
4: if(x< y)	✓	✓			✓		✓	✓
5: m=y;		✓						
6: else if(x<z)	✓				✓		✓	✓
7: m=y; // *** bug ***	✓						✓	✓
8: else			✓	✓		✓		
9: if(x>y)			✓	✓		✓		
10: m=y;			✓			✓		
11: else if (x>z)				✓				
12: m=z;								
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓
} Pass/Fail Status	P	P	P	P	P	P	F	F

Fault Localization

$$suspiciousness(s) = \frac{failed(s)}{\sqrt{totalfailed * (failed(s) + passed(s))}}$$

Fault Localization

$$suspiciousness(s) = \frac{failed(s)}{\sqrt{totalfailed * (failed(s) + passed(s))}}$$

Fault Localization

$$suspiciousness(s) = \frac{failed(s)}{\sqrt{totalfailed * (failed(s) + passed(s))}}$$

Fault Localization

$$suspiciousness(s) = \frac{failed(s)}{\sqrt{totalfailed * (failed(s) + passed(s))}}$$

Key Algorithms

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Key Algorithms

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Examine the most suspicious statement first

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Assumption: Developer examines source code statements in order

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Assumption: Developer can identify faulty statement

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Our framework supports fault localization with a wide variety of metrics

Model for Experimentation

① Introduction

Important Points

Software Challenges

② Software Testing

Basic Concepts

Testing Opportunities

③ Regression Testing

Supporting Methods

Key Algorithms

④ Empirical Evaluation

Model for Experimentation

Concrete Examples

⑤ Conclusion

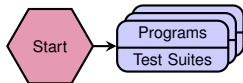
How Do I Evaluate Regression Testing Methods?



Start

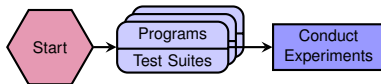
Model for Experimentation

How Do I Evaluate Regression Testing Methods?



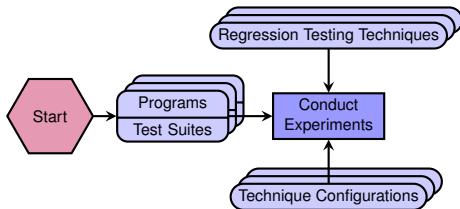
Model for Experimentation

How Do I Evaluate Regression Testing Methods?



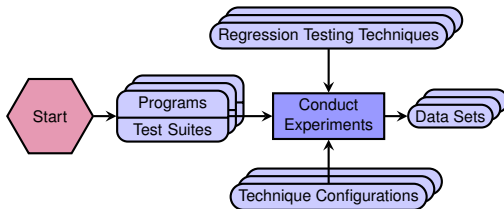
Model for Experimentation

How Do I Evaluate Regression Testing Methods?



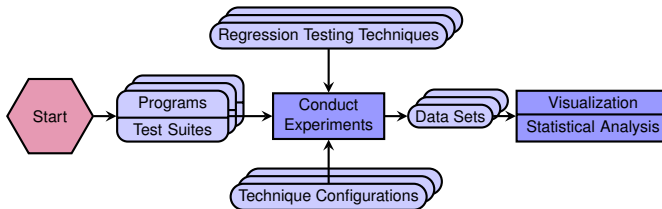
Model for Experimentation

How Do I Evaluate Regression Testing Methods?



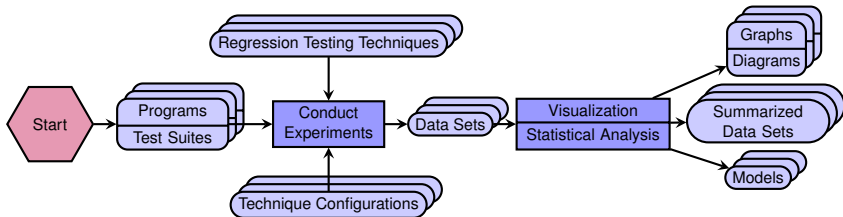
Model for Experimentation

How Do I Evaluate Regression Testing Methods?



Model for Experimentation

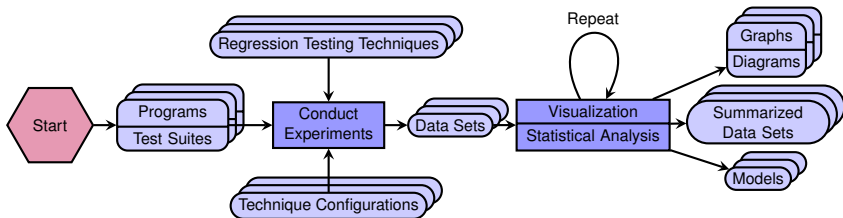
How Do I Evaluate Regression Testing Methods?



Model for Experimentation

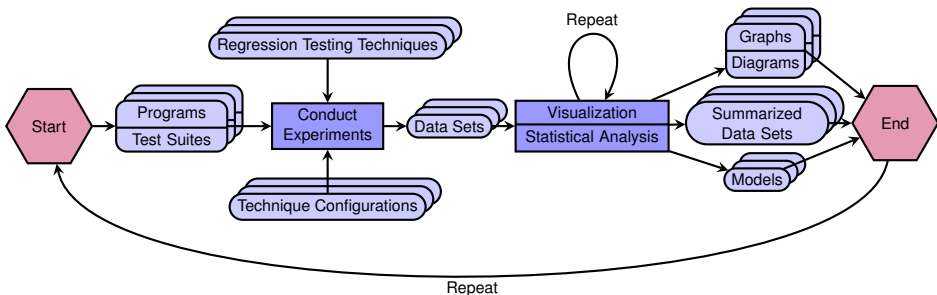
How Do I Evaluate Regression Testing Methods?

Iteratively Perform Visualization and Statistical Analysis



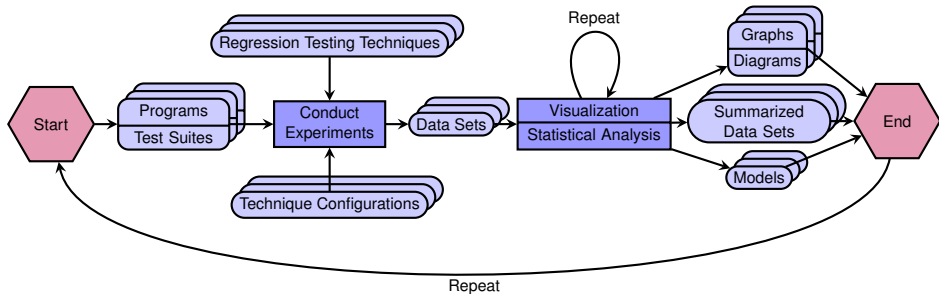
How Do I Evaluate Regression Testing Methods?

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!

Concrete Examples

① Introduction

Important Points

Software Challenges

② Software Testing

Basic Concepts

Testing Opportunities

③ Regression Testing

Supporting Methods

Key Algorithms

④ Empirical Evaluation

Model for Experimentation

Concrete Examples

⑤ Conclusion

Case Study Applications

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

Case Study Applications

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

Case Study Applications

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

Case Study Applications

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

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Greedy and hill climbing produce comparable orderings

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

However, hill climbing is slightly more efficient than greedy

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Greedy produces a slightly better ordering than hill climbing

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

But the hill climbing algorithm executes over four times faster!

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

A small fitness increase may result in a large runtime increase

Empirical Insights

Prioritizer	Fitness	Runtime (sec)	Application
GRD	0.98	0.22	Sudoku
GRD	0.36	0.38	ReductionAndPrioritization
GRD	0.71	33.88	JodaTime
HC_SA_FS	0.98	0.01	Sudoku
HC_SA_FS	0.36	0.04	ReductionAndPrioritization
HC_SA_FS	0.69	7.88	JodaTime

Our tools support efficient experimentation through parallelism

1 Introduction

Important Points

Software Challenges

2 Software Testing

Basic Concepts

Testing Opportunities

3 Regression Testing

Supporting Methods

Key Algorithms

4 Empirical Evaluation

Model for Experimentation

Concrete Examples

5 Conclusion

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
- 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
- Develop statistically meaningful empirical results

Regression Testing: Theoretical Underpinnings, Practical Techniques, and Empirical Insights

Gregory M. Kapfhammer* and Jonathan Miller Kauffman†

* <http://www.cs.allegheny.edu/~gkapfham/>

† <https://sites.google.com/a/allegheny.edu/jonathan-kauffman/>

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



ALLEGHENY COLLEGE