Using Controlled Numbers of Real Faults and Mutants to Empirically Evaluate Coverage-Based Test Case Prioritization

David Paterson
University of Sheffield

Gregory Kapfhammer
Allegheny College

Gordon Fraser
University of Passau

Phil McMinn
University of Sheffield

Workshop on Automation of Software Test
29th May 2018

dpaterson1@sheffield.ac.uk
Test Case Prioritization

- Testing is required to ensure the correct functionality of software
- Larger software → more tests → longer running test suites
Testing is required to ensure the correct functionality of software

- Larger software -> more tests -> longer running test suites

How can we reduce the time taken to identify new faults whilst still ensuring that all faults are found?

Find an *ordering* of test cases such that faults are detected as early as possible.

**Test Case Prioritization**
Types of Fault

Real

Artificial
Test Case Prioritization

Strategy A
- 100 subjects
- Evaluated on mutants
- Score = 0.75

Strategy B
- 100 subjects
- Evaluated on real faults
- Score = 0.72

Which strategy performs the best?
Research Objectives

1. Compare prioritization strategies across fault types

2. Investigate the impact of multiple faults
Evaluating Test Prioritization

Average Percentage of Faults Detected (APFD)

- % Faults Found vs % Test Suite executed

\[
APFD = 1 - \frac{\sum_{i=1}^{m} TF_i}{mn} + \frac{1}{2n}
\]

- TCP aims to **maximize** APFD by **minimizing** \(TF_i\)
Evaluating Test Prioritization

1 fault detected after 7 test cases (n=10)

\[ APFD = 1 - \frac{7}{10} + \frac{1}{20} = 0.35 \]
Evaluating Test Prioritization

1 fault detected after 1 test cases (n=20)

\[
APFD = 1 - \frac{1}{20} + \frac{1}{40} = 0.975
\]
Evaluating Test Prioritization

1 fault detected after 2 test cases
2nd fault detected after 8 test cases (n=10)

$$APFD = 1 - \frac{2 + 8}{20} + \frac{1}{20} = 0.55$$
## Test Case Prioritization

<table>
<thead>
<tr>
<th></th>
<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>
<th>APFD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Version 2</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.55</td>
</tr>
<tr>
<td>Version 3</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>$t_1$</td>
<td>$t_8$</td>
<td>$t_4$</td>
<td>$t_5$</td>
<td>$t_7$</td>
<td>$t_9$</td>
<td>$t_2$</td>
<td>$t_{10}$</td>
<td>$t_6$</td>
<td>$t_3$</td>
<td>APFD</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>Version 1</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>-</td>
</tr>
<tr>
<td>Version 2</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>0.85</td>
</tr>
<tr>
<td>Version 3</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>0.8</td>
</tr>
</tbody>
</table>
public int abs(int x) {
    if (x >= 0) {
        return x;
    } else {
        return -x;
    }
}
RQ1: How does the effectiveness of test case prioritization compare between a single real fault and a single mutant?

RQ2: How does the effectiveness of test case prioritization compare between single faults and multiple faults?
Subjects

- **Defects4J**: Large repository containing 357 real faults from 5 open-source repositories [1]

<table>
<thead>
<tr>
<th>Project</th>
<th>GitHub</th>
<th>Number of Bugs</th>
<th>KLOC</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFreeChart</td>
<td><a href="https://github.com/jfree/jfreechart">https://github.com/jfree/jfreechart</a></td>
<td>26</td>
<td>96</td>
<td>2,205</td>
</tr>
<tr>
<td>Closure Compiler</td>
<td><a href="https://github.com/google/closure-compiler">https://github.com/google/closure-compiler</a></td>
<td>133</td>
<td>90</td>
<td>7,927</td>
</tr>
<tr>
<td>Apache Commons Lang</td>
<td><a href="https://github.com/apache/commons-lang">https://github.com/apache/commons-lang</a></td>
<td>65</td>
<td>85</td>
<td>3,602</td>
</tr>
<tr>
<td>Apache Commons Math</td>
<td><a href="https://github.com/apache/commons-math">https://github.com/apache/commons-math</a></td>
<td>106</td>
<td>28</td>
<td>4,130</td>
</tr>
<tr>
<td>Joda Time</td>
<td><a href="https://github.com/JodaOrg/joda-time">https://github.com/JodaOrg/joda-time</a></td>
<td>27</td>
<td>22</td>
<td>2,245</td>
</tr>
</tbody>
</table>

- Contains developer written test suites

- Provides 2 versions of every subject – one buggy and one fixed

[1] https://github.com/rjust/defects4
Experimental Process

Defects4J → Fixed Version → Major

Apply Patch

Buggy Version

Apply Patch

Program

Program

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>testOne</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>testTwo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>testN</td>
<td></td>
</tr>
</tbody>
</table>

Kanonizo

Test Prioritization

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>test42</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>test378</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>test201</td>
<td></td>
</tr>
</tbody>
</table>
- **Wilcoxon U-Test** measures likelihood that 2 samples originate from the same distribution $p$
  - Significant differences occur often when samples are large

- **Vargha-Delaney** effect size calculates the *magnitude* of differences $\hat{A}_{12}$ – the practical difference between two samples
### Metrics

- **Wilcoxon U Test** measures the likelihood that two samples originate from the same distribution. Significant differences occur often when samples are large.

- **Vargha-Delaney effect size** calculates the magnitude of differences—the practical difference between two samples. 

  - $p = 0.5544$
  - Significant = ❌
  - $\hat{A}_{12} = 0.5007$
  - Effect Size = None
Metrics

- Wilcoxon U Test measures the likelihood that two samples originate from the same distribution. Significant differences occur often when samples are large.

- Vargha-Delaney effect size calculates the magnitude of differences - the practical difference between two samples.

\[ p = 2.2 \times 10^{-16} \]

Significant = ✅

\[ \hat{A}_{12} = 0.4075059 \]

Effect Size = Small
Metrics

- **Wilcoxon U-Test** measures likelihood that 2 samples originate from the same distribution - Significant differences occur often when samples are large.

- **Vargha-Delaney effect size** calculates the magnitude of differences - the practical difference between two samples.

\[ p = 2.2e-16 \]

Significant = ✅

\[ \hat{A}_{12} = 0.3250598 \]

Effect Size = Medium
• Wilcoxon U-Test measures likelihood that 2 samples originate from the same distribution. Significant differences occur often when samples are large.

• Vargha-Delaney effect size calculates the magnitude of differences—the practical difference between two samples.

\[ p = 2.2 \times 10^{-16} \]

Significant = ✅

\[ \hat{A}_{12} = 0.005826003 \]

Effect Size = Large
### Comparisons

#### RQ1

<table>
<thead>
<tr>
<th>Strategy 1</th>
<th>Strategy 2</th>
<th>Fault Type 1</th>
<th>Fault Type 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>Real</td>
<td>Mutant</td>
</tr>
</tbody>
</table>

#### RQ2

<table>
<thead>
<tr>
<th>Strategy 1</th>
<th>Strategy 2</th>
<th>Faults 1</th>
<th>Faults 2</th>
<th>Faults 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategy 1</th>
<th>Strategy 2</th>
<th>Faults 1</th>
<th>Faults 2</th>
<th>Faults 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>1 real</td>
<td>5 real</td>
<td>10 real</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategy 1</th>
<th>Strategy 2</th>
<th>Faults 1</th>
<th>Faults 2</th>
<th>Faults 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>1 mutant</td>
<td>5 mutant</td>
<td>10 mutant</td>
</tr>
</tbody>
</table>
Results

RQ1: Real Faults vs Mutants

- APFD is significantly higher for mutants than real faults in all but one case.
- On average, over 10% additional test cases were required to find the real faults.
- For real faults, 3 out of 16 project/strategy combinations significantly improve over the baseline, compared to 10 out of 16 improvements for mutants.

<table>
<thead>
<tr>
<th>Project</th>
<th>Real</th>
<th>Mutant</th>
<th>Test Cases</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chart</td>
<td>703.4</td>
<td>498.5</td>
<td>1826.0</td>
<td>11.2%</td>
</tr>
<tr>
<td>Lang</td>
<td>818.9</td>
<td>611.4</td>
<td>1960.8</td>
<td>10.6%</td>
</tr>
<tr>
<td>Math</td>
<td>1461.7</td>
<td>815.8</td>
<td>3566.9</td>
<td>18.1%</td>
</tr>
<tr>
<td>Time</td>
<td>1341.9</td>
<td>683.4</td>
<td>3929.1</td>
<td>16.8%</td>
</tr>
</tbody>
</table>
Results

RQ1: Real Faults vs Mutants

- APFD is **significantly higher** for **mutants** than **real faults** in all but one case.

- On average, over **10% additional** test cases were required to find the **real faults**.

- For **real faults**, 3 out of 16 project/technique combinations significantly improve over the baseline, compared to 10 out of 16 improvements for **mutants**.

  Test Case Prioritization is much more effective for **mutants** than **real faults**.
Results

RQ2: Single faults vs Multiple Faults

- Variance in APFD scores **significantly** reduces as more faults are introduced

- In **37/40** cases, median APFD **decreased** as more faults are introduced
  - APFD punishes test suites that are not able to find **all** faults
Results

RQ2: Single faults vs Multiple Faults

- However, **real faults** and **mutants** still disagree on the effectiveness of TCP techniques

- For **real faults**, there is very rarely any practical difference when including more faults
  - 17 of 40 comparisons are significant, of which 3 are **Medium** or **Large** effect size

- For **mutants**, increasing the number of faults makes the results clearer
  - 35 of 40 comparisons are significant, of which 16 are **Medium** or **Large** effect size
  - Effect size increases in **all but one** case for more faults
Results

RQ2: Single faults vs Multiple Faults

• However, real faults and mutants still disagree on the effectiveness of TCP techniques

• For real faults, there is very rarely any practical difference when including more faults
  - 17 of 40 comparisons are significant, of which 3 are Medium or Large effect size

• For mutants, increasing the number of faults makes the results clearer
  - 35 of 40 comparisons are significant, of which 16 are Medium or Large effect size
  - Effect size increases in all but one case for more faults

Using more faults lessens the effect of randomness, but still does not make mutants and real faults consistent.
Real Faults vs Mutants

- Real faults are much more complex than mutants

```java
for (final EventState state : eventsStates) {
    state.stepAccepted(eventT, eventY);
    isLastStep = isLastStep || state.stop();
}

// handle the first part of the step, up to the event
for (final StepHandler handler : stepHandlers) {
    handler.handleStep(interpolator, isLastStep);
}

if (isLastStep) {
    // the event asked to stop integration
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    return eventT;
}

boolean needReset = false;
for (final EventState state : eventsStates) {
    needReset = needReset || state.reset(eventT, eventY);
}

if (needReset) {
    // some event handler has triggered changes that
    // invalidate the derivatives, we need to recompute them
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    computeDerivatives(eventT, y, yDot);
    resetOccurred = true;
    return eventT;
}
```
Real Faults vs Mutants

- Real faults are much more complex than mutants

```java
currentEvent.stepAccepted(eventT, eventY);
isLastStep = currentEvent.stop();

// handle the first part of the step, up to the event
for (final StepHandler handler : stepHandlers) {
    return eventT;
}

boolean needReset = currentEvent.reset(eventT, eventY);

if (needReset) {
    // some event handler has triggered changes that
    // invalidate the derivatives, we need to recompute them
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    computeDerivatives(eventT, y, yDot);
    resetOccurred = true;

    for (final EventState remaining : occurringEvents) {
        remaining.stepAccepted(eventT, eventY);
    }

    return eventT;
}
```
Real Faults vs Mutants

- Real faults are much more complex than mutants
  - On average, fixing a **real fault** added 1.98 lines and removed 7.2
  - Fixing a **mutant** is always **max +/- 1 line**

```java
boolean needsReset = false;
```

- This results in **more** test cases detecting **mutants**
  - On average, 3.18 test cases detected single **real faults**
  - Meanwhile, 57.38 test cases detected single **mutants**
Summary

**Tool:**
https://github.com/kanonizo/kanonizo

**Data:**
https://bitbucket.org/djpaterson/ast2018_data

---

**Test Case Prioritization**

- Testing is required to ensure the correct functionality of software
- Larger software -> more tests -> longer running test suites

How can we reduce the time taken to identify new faults as they are found?

Find an *ordering* of test cases such that faults are detected as early as possible.

---

**Strategy A**
- 100 subjects
- Evaluated on *mutants*
- Score = 0.75

**Strategy B**
- 100 subjects
- Evaluated on *real faults*
- Score = 0.72

Which strategy performs the best?

---

**Results**

RQ1: Real Faults vs Mutants

- APFD is significantly higher for *mutants* than *real faults* in all but one case.
- On average, over 10% additional test cases were required to find the *real faults*.

<table>
<thead>
<tr>
<th>Project</th>
<th>Real</th>
<th>Mutant</th>
<th>Test Cases</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chart</td>
<td>705.4</td>
<td>498.5</td>
<td>1826.0</td>
<td>11.2%</td>
</tr>
<tr>
<td>Lang</td>
<td>818.9</td>
<td>613.4</td>
<td>1960.8</td>
<td>10.4%</td>
</tr>
<tr>
<td>Math</td>
<td>1461.7</td>
<td>815.8</td>
<td>3566.9</td>
<td>18.1%</td>
</tr>
<tr>
<td>Time</td>
<td>1341.9</td>
<td>683.4</td>
<td>3929.1</td>
<td>16.8%</td>
</tr>
</tbody>
</table>

- For *real faults*, 3 out of 16 project/strategy combinations significantly improve over the baseline, compared to 10 out of 16 improvements for *mutants*.

---

RQ2: Single faults vs Multiple Faults

- However, *real faults* and *mutants* still disagree on the effectiveness of TCP techniques.
  - For *real faults*, there is very rarely any practical difference when including more faults.
  - For *mutants*, increasing the number of faults makes the results clearer.
    - Of 40 comparisons are significant, of which 16 are Medium or Large effect size.
    - Effect size increases in all but one case for more faults.