Diversity Maximization Speedup for Fault Localization

Liang Gong¹, David Lo², Lingxiao Jiang², and Hongyu Zhang¹

Authored by

¹ Tsinghua University
² Singapore Management University
Outline

Motivation
- Fault Localization
- Test Case Prioritization

Diversity Maximization Speedup
- Technical Motivation
- Detailed Approach

Experiments
- Settings & Results

Conclusion & Future work
• Software errors cost the US economy 59.5 billion dollars (0.6% of 2002's GDP) [1]

• Testing and debugging activities are labor-intensive (30% to 90% of a Project) [2]


Spectrum-based Fault Localization (abbr. SBFL)

- Automatically recommend a list of suspicious program elements for inspection.

- Program Spectra consists of coverage information and execution labels.

**Program Spectra**

**Profile of an execution trace**

Coverage information of one element \(s_i\) in all executions

Correct or incorrect?
For a given statement $S$

The formula calculates the suspiciousness of $S$.\[ \frac{a_{ef}}{\sqrt{(a_{ef} + a_{nf}) (a_{ef} + a_{ep})}} \]

Intuition: If $S$ is covered more in failed traces and less in passed traces, it is more likely to contain faults.
Process

In Experiments

Diversity Maximization Speedup for Fault Localization

Correct program + Test script + program (contains faults)

Correct Output (Oracle)

Test Program Output

Compare

Label for the execution trace (pass or fail)

Collect by tools

Execution trace

Program Spectra

Fault localization

Fault list

Presented by Liang Gong
School of Software
Tsinghua University
Diversity Maximization Speedup for Fault Localization

**Process In Practice**

1. **Test script** + **program (contains faults)** → **Test Program Output**
   - Collect by tools

2. **developer** + **Test Program Output** → **Label for the execution trace (pass or fail)**
   - Manually Label

3. **Execution trace** → **Label for the execution trace (pass or fail)**

4. **Program Spectra** → **Fault localization**
   - **Fault list**

---

**Presented by**

Liang Gong  
School of Software  
Tsinghua University
Research Goal:

• Minimize No. of executions to label
• Preserve fault localization effectiveness
In [3], Rothermel et al. define the problem of test case prioritization as follows:

Definition 2.1 (Test Case Prioritization). Given
(1) $T$, a set of test cases,
(2) $PT$, the set of permutations of $T$
(3) $f$, a function mapping $PT$ to real numbers,
the problem is to find a permutation $p \in PT$ such that:
for all $p' \in PT$: $f(p) \geq f(p')$.

\[ \text{arg max} \{ f(p) \} \]

In [3], Rothermel et al. define the problem of test case prioritization as follows:

**Definition 2.1 (Test Case Prioritization).** Given

- $T$, a set of test cases,
- $PT$, the set of permutations of $T$,
- $f$, a function mapping $PT$ to real numbers,

the problem is to find a permutation $p \in PT$ such that:

$$f(p) \geq f(p') \quad \text{for all } p' \in PT.$$ 

$f(p)$ is larger, when permutation $p$ allows the faulty program elements to be ranked higher meanwhile a shorter prefix are considered.

Diversity Maximization Speedup (abbr. DMS)

- Greedy algorithm
- Use diversity of suspiciousness as the selecting criterion.
- Speedup suspiciousness rank changing process of promising program elements to further save labeling effort.
**Diversity Maximization Speedup (abbr. DMS)**

- $t_0$ is the initial failed trace that reveals the fault.
- $t_1$ and $t_2$ are candidates to be selected for labeling.

### Initial Diagnostic Distribution

<table>
<thead>
<tr>
<th>...</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
<th>$s_5$</th>
<th>$s_6$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>$\bullet$</td>
<td>$\bullet$</td>
<td>$\bullet$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>$\circ$</td>
<td>...</td>
</tr>
</tbody>
</table>

### Initial Inspection List

- $s_1, s_2, s_3$
- $s_4, s_5, s_6$

---

### Diversity as Criterion
- $t_1$ is preferred by approaches aiming to detect faults

- $t_2$ is selected according to Diversity Maximization Criterion

- $t_1$ is preferred by approaches aiming to detect faults

- $t_2$ is selected according to Diversity Maximization Criterion

- More Diverse, Better Result
## Diversity Maximization Speedup (abbr. DMS)

Diversity Maximization Speedup for Fault Localization

**Introduction**

Presented by Liang Gong

School of Software
Tsinghua University

---

### Test Case Prioritization

<table>
<thead>
<tr>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>main() {</td>
</tr>
<tr>
<td>int let, dig, c;</td>
</tr>
<tr>
<td>let = dig = 0;</td>
</tr>
<tr>
<td>while(c=getchar()) {</td>
</tr>
<tr>
<td>if (&quot;A&quot; &lt;= c &amp;&amp; 'Z'&gt;=c)</td>
</tr>
<tr>
<td>let += 1;</td>
</tr>
<tr>
<td>else if ('a' &lt;= c &amp;&amp; 'z'&gt;=c) /<em>FAULT</em>/</td>
</tr>
<tr>
<td>let += 1;</td>
</tr>
<tr>
<td>else if (&quot;0&quot; &lt;= c &amp;&amp; '9'&gt;=c)</td>
</tr>
<tr>
<td>dig += 1;</td>
</tr>
<tr>
<td>printf(&quot;%d \n&quot;, let, dig);</td>
</tr>
<tr>
<td>pass/fail</td>
</tr>
</tbody>
</table>

### Suspiciousness Metrics

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Ne</th>
<th>Np</th>
<th>N_top</th>
<th>N_np</th>
<th>Ochiai</th>
<th>Tarantula</th>
<th>Jaccard</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0.500</td>
<td>0.500</td>
<td>0.250</td>
</tr>
<tr>
<td>s2</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0.322</td>
<td>0.329</td>
<td>0.273</td>
</tr>
<tr>
<td>s3</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>0.408</td>
<td>0.500</td>
<td>0.222</td>
</tr>
<tr>
<td>s4</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>0.655</td>
<td>0.692</td>
<td>0.429</td>
</tr>
<tr>
<td>s5</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>0.516</td>
<td>0.607</td>
<td>0.333</td>
</tr>
<tr>
<td>s6</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0.471</td>
<td>0.600</td>
<td>0.286</td>
</tr>
<tr>
<td>s7</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0.516</td>
<td>0.607</td>
<td>0.333</td>
</tr>
<tr>
<td>s8</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0.500</td>
<td>0.500</td>
<td>0.250</td>
</tr>
</tbody>
</table>

### (a) Fault Localization with All Test Cases

<table>
<thead>
<tr>
<th>Ambiguity Group</th>
<th>Selected Test Case</th>
<th>Program Spectra</th>
<th>Normalized Ochiai Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>(the groups are ordered according to their suspiciousness)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11}</td>
<td>t2</td>
<td>1 1 1 1 1 1 1 1 1</td>
<td>0.0909 0.0909 0.0909 0.0909 0.0909 0.0909 0.0909 0.0909 0.0909 0.0909</td>
</tr>
<tr>
<td>{s5,s6,s7,s8,s9,s10},{s1,s2,s3,s4,s11}</td>
<td>t8</td>
<td>1 1 1 1 1 0 0 0 0 0 0 0</td>
<td>0.0742 0.0742 0.0742 0.0742 0.1049 0.1049 0.1049 0.1049 0.1049 0.1049 0.1049</td>
</tr>
<tr>
<td>{s7,s8,s9,s10},{s6,s5},{s1,s2,s3,s4,s11}</td>
<td>t6</td>
<td>1 1 1 1 1 1 0 0 0 0 1</td>
<td>0.0696 0.0696 0.0696 0.0696 0.0852 0.0852 0.0852 0.1205 0.1205 0.1205 0.1205 0.0696</td>
</tr>
<tr>
<td>{s7,s8},{s5,s6},{s1,s2,s3,s4,s11},{s9,s10}</td>
<td>t4</td>
<td>1 1 1 1 1 1 1 1 1 1 0 1</td>
<td>0.0824 0.0824 0.0824 0.0824 0.0824 0.0824 0.0824 0.1165 0.1165 0.1165 0.1165 0.0824</td>
</tr>
<tr>
<td>{s7,s8},{s6},{s5},{s10},{s1,s2,s3,s4,s11},{s9}</td>
<td>t7</td>
<td>1 1 1 1 1 0 1 1 1 0 1</td>
<td>0.0840 0.0840 0.0840 0.0840 0.0940 0.1085 0.1085 0.1085 0.1085 0.0940 0.0840</td>
</tr>
<tr>
<td>{s7},{s10},{s5},{s1,s2,s3,s4,s11},{s6},{s8},{s9}</td>
<td>t9</td>
<td>1 1 1 1 1 0 1 0 0 1 1</td>
<td>0.0885 0.0885 0.0885 0.0885 0.0969 0.0834 0.1084 0.0834 0.0834 0.1022 0.0885</td>
</tr>
</tbody>
</table>

---

**Figure 1: Running Example**
Diversity Maximization Speedup for Fault Localization

Evolution Trend Opportunities

Looking for test cases that could offer more changing opportunities to "promising" elements like s7 (with clear trend) instead of s9
Two questions prompt:

- How can we know which statements are “promising”?
- With “promising” statements, how can we speed up their suspiciousness changing process?
Representative Time Series

- When the rank of the program element decreases, its time series increases by 1.

- When the rank of the program element increases, its time series decreases by 1.

- If the element's rank stays the same, its time series stays the same.

<table>
<thead>
<tr>
<th>Evolution trend and time series($y_i$) of $S_8$</th>
</tr>
</thead>
</table>
| Iteration ($x_i$) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ...
| Rank | 11 | 6 | 4 | 2 | 3 | 11 | 5 | ...
| Trend ($\mathcal{T}$) | [+ | + | + | - | - | [+ | ...
| $y_i$ | 0 | 1 | 2 | 3 | 2 | 1 | 2 | ...
Promising

How to evaluate?

| Iteration ($x_i$) | 1   | 2   | 3   | 4   | 5   | 6   | 7   | ...
|-------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| Rank              | 11  | 6   | 4   | 2   | 3   | 11  | 5   | ...
| Trend ($T$)       | [+ ]| [+ ]| [+ ]| [- ]| [- ]| [+ ]|     |     |
| $y_i$             | 0   | 1   | 2   | 3   | 2   | 1   | 2   | ...

• Linear Regression Analysis:

$$y_i = \beta_1 \cdot x_i + \beta_0 + \epsilon_i$$

• Change-potential Score:

$$W_T = \hat{\beta}_1 \cdot \frac{1}{\hat{\sigma}_{\beta_1} + 1}$$

Example trends and their potentials

<table>
<thead>
<tr>
<th>$T$</th>
<th>$\hat{\beta}_1$</th>
<th>$\hat{\sigma}_{\beta_1}$</th>
<th>$W_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[+ ]</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>[+ ]</td>
<td>0</td>
<td>0.577</td>
<td>0</td>
</tr>
<tr>
<td>[+ ]</td>
<td>0.5</td>
<td>0.289</td>
<td>0.388</td>
</tr>
<tr>
<td>[0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Two questions prompt:

- How can we know which statements are “promising”?

- With “promising” statements, how can we speed up their suspiciousness changing process?
• Speed up the suspiciousness ranking changing process by competing in **Suspicious Group**.

**Newly Added Execution Trace**

<table>
<thead>
<tr>
<th>s1</th>
<th>...</th>
<th>s7</th>
<th>s8</th>
<th>s9</th>
<th>s10</th>
<th>...</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>?</td>
</tr>
</tbody>
</table>

No matter what the label is, ties are broken anyway. Speedup happens anyway. It could keep “promising” or become less “promising”.

**Trend**

Labeled as **fail**

Labeled as **pass**
Diversity Maximization Speedup for Fault Localization

Presented by Liang Gong
School of Software, Tsinghua University

Speed up

Our Method

- Change-potential Score of Suspicious Group:

\[ \mathcal{W}_g = \sum_{d \in g} \mathcal{W}_{T_d} \]

- Change-potential Score of program element \( d \)

- Sums of Squares of Change-potential Score of all Groups (G)

\[ \mathcal{H}_G = \sum_{g_i \in G} \mathcal{W}_{g_i}^2 \]

- To choose the next trace \( t \) to label, we use the following formula:

\[ \arg \max_{t \in T_U} \left\{ \mathcal{H}_G - \mathcal{H}(G \leftarrow t) \right\} \]

The Sum of Squares of change-potential of all groups

The Sum of Squares of change-potential of all groups when \( t \) is added
Intuition: When \( t \) breaks ties in more promising or larger Suspicious Groups, it is more likely to be selected.

Change-potential Score of program element \( d \):

\[
\mathcal{W}_d = \sum_{d \in g} \mathcal{W}_{T_d}
\]

Change-potential Score of Suspicious Group:

\[
\mathcal{W}_g = \sum_{d \in g} \mathcal{W}_d
\]

Sums of Squares of Change-potential Score of all Groups(\( G \)):

\[
\mathcal{H}_G = \sum_{g_i \in G} \mathcal{W}_{g_i}^2
\]

To choose the next trace \( t \) to label, we use the following formula:

\[
\arg \max_{t \in T_U} \{ \mathcal{H}_G - \mathcal{H}(G \leftarrow t) \}
\]
• Coverage Based Prioritization
  • STMT-TOTAL, STMT-ADDTL, and ART.

• Fault-Exposing Potential
  • FEP-TOTAL and FEP-ADDTL.

• Diagnostic Prioritization
  • SEQUIA and RAPTOR.
Diversity Maximization Speedup for Fault Localization

Experiment

Dataset & Evaluation Metric

- **Benchmarks for Fault Localization**

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
<th>LOC</th>
<th>Tests</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcas</td>
<td>Aircraft Control</td>
<td>173</td>
<td>1609</td>
<td>41</td>
</tr>
<tr>
<td>schedule2</td>
<td>Priority Scheduler</td>
<td>374</td>
<td>2710</td>
<td>8</td>
</tr>
<tr>
<td>schedule</td>
<td>Priority Scheduler</td>
<td>412</td>
<td>2651</td>
<td>8</td>
</tr>
<tr>
<td>replace</td>
<td>Pattern Matcher</td>
<td>564</td>
<td>5543</td>
<td>31</td>
</tr>
<tr>
<td>tot_info</td>
<td>Info Measure</td>
<td>565</td>
<td>1052</td>
<td>22</td>
</tr>
<tr>
<td>print_tokens2</td>
<td>Lexical Analyzer</td>
<td>570</td>
<td>4055</td>
<td>10</td>
</tr>
<tr>
<td>print_tokens</td>
<td>Lexical Analyzer</td>
<td>726</td>
<td>4070</td>
<td>7</td>
</tr>
<tr>
<td>space</td>
<td>ADL Compiler</td>
<td>9564</td>
<td>1343</td>
<td>30</td>
</tr>
<tr>
<td>flex</td>
<td>Lexical Parser</td>
<td>10124</td>
<td>567</td>
<td>43</td>
</tr>
<tr>
<td>sed</td>
<td>Text Processor</td>
<td>9289</td>
<td>371</td>
<td>22</td>
</tr>
<tr>
<td>grep</td>
<td>Text Processor</td>
<td>9089</td>
<td>809</td>
<td>17</td>
</tr>
<tr>
<td>gzip</td>
<td>Data Compressor</td>
<td>5159</td>
<td>217</td>
<td>15</td>
</tr>
</tbody>
</table>

1. Siemens Suite
2. UNIX Programs

- **Evaluation Metric for Fault Localization**

\[
\text{cost} = \frac{|\{ j \mid f_{TS}(d_j) \geq f_{TS}(d_*) \}|}{|D|}
\]
**Experiments comparing with existing methods:**

- Effectiveness on reducing the number of test cases (i.e., labeling effort) needed for a *target cost*

- Effectiveness on reducing cost for a given number of labeled test cases

- Defining *target cost* $c_x$:

  $$c_x = \frac{x}{100} \times C$$

### Labeling Effort Needed When Setting $c_{101}$ as Target Cost

<table>
<thead>
<tr>
<th>Subject Programs</th>
<th>DMS</th>
<th>RAPTOR</th>
<th>SEQ-uoia</th>
<th>Stmt-Addtl</th>
<th>Stmt-Total</th>
<th>FEP-Addtl</th>
<th>ART-Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siemens</td>
<td>18</td>
<td>20</td>
<td>500+</td>
<td>500+</td>
<td>500+</td>
<td>97</td>
<td>150</td>
</tr>
<tr>
<td>UNIX</td>
<td>16</td>
<td>48</td>
<td>176</td>
<td>150</td>
<td>500+</td>
<td>98</td>
<td>56</td>
</tr>
</tbody>
</table>
Experiments comparing with existing methods:

- Effectiveness on reducing the number of test cases (i.e., labeling effort) needed for a target cost.
- Effectiveness on reducing cost for a given number of labeled test cases.

Average Cost of DMS when Selecting Different Number of Test Cases
• Effectiveness on Reducing Cost for a Given Number of Labeled Test Cases

Pair-wised T-test shows the improvements are statistically significant at 95% interval.
• **Conclusions**
  - We propose a new technique aiming to minimize the amount of effort in manual oracle construction, while still permitting effective fault localization.

  ✓ Given a target fault localization accuracy, our approach can significantly reduce the number of test cases needed to achieve it.

  ✓ Given a maximum number of test cases that a programmer can manually label, DMS can improve the accuracy of fault localization and thus helps reduce the debugging cost.

• **Future Work**
  - Evaluate on more subject programs.
  - We will also explore the possibility of adopting more sophisticated trend analysis methods.
Conclusions

- We propose a new technique aiming to minimize the amount of effort in manual oracle construction, while still permitting effective fault localization.

- Given a target fault localization accuracy, our approach can significantly reduce the number of test cases needed to achieve it.

- Given a maximum number of test cases that a programmer can manually label, DMS can improve the accuracy of fault localization and thus helps reduce the debugging cost.

Future Work

- Evaluate on more subject programs.

- We will also explore the possibility of adopting more sophisticated trend analysis methods.

Any questions?