International Journal of Control and Automation
Vol.6, No.5 (2013), pp.117-126
http://dx.doi.org/10.14257/ijca.2013.6.5.11

# Noise Reduction for Spectrum-based Fault Localization

Patrick Daniel and Kwan Yong Sim

*Faculty of Engineering, Computing and Science*
*Swinburne University of Technology, Kuching, Sarawak, Malaysia*

*pdaniel_koe@yahoo.com, ksim@swinburne.edu.my*

***Abstract***

*Spectrum-based Fault Localization (SBFL) has been proven to be an effective technique to locate faulty statement in program code. SBFL metrics exploit the records of statement execution (spectra) by pass and fail test cases for each statement to rank its likeliness to be faulty. However, in some cases, these spectra contain duplicated and ambiguous information or noise which may deteriorate the performance of SBFL metrics. We propose six noise reduction schemes to eliminate test cases which provide duplicated and ambiguous information and evaluate the resulting performance improvements in SBFL metrics. Based on our findings, we further provide a guide for SBFL practitioners to select the best performing noise reduction scheme for the SBFL metrics that they use.*

***Keywords:*** *Software Engineering, Software Testing, Software Debugging, Spectrum-based Fault Localization, Noise Reduction*

## 1. Introduction

Software testing and debugging is one of the most crucial phases but resource consuming activity in software development life cycle. Software testing and debugging consume 50% to 80% of the software development cost, making it the most expensive phase of software development life cycle [1]. Many fault localization techniques have been proposed by researchers in order to reduce the cost and time spent in the debugging process to find the faulty statement in software code.

Mistakes inadvertently committed by software developer in program statement may lead to software failures such as software crash and unexpected software behaviors or incorrect outputs. The software testing process is conducted by executing a range of test cases (test inputs) on software developed with the aim to detect software failures. For a given test case, software tester is able to determine the output of the software as pass or fail by comparing the it with the software specification. However, the software testing process can only show presence of failures but not locating the faulty statement that causes the failure. Locating the faulty statement in the software code is a time consuming task, especially for software that contain thousands of lines of codes. Hence, various fault localization techniques have been proposed to reduce the time required to locate the faulty statement.

Spectrum-based Fault Localization (*SBFL*) is one of the widely studied approaches for fault localization. In SBFL, the spectra (statement execution records) of pass and fail test cases are used to rank the program statements according to their likeliness to the faulty. Intuitively, the faulty statement is likely to be executed by more fail test cases than pass test cases. Based on this intuition, many SBFL metrics have been developed to calculate the likeliness that a statement is faulty. A limitation of the SBFL approach is that it requires testing oracle to judge the correctness of the test outputs and categorize the test cases as pass or fail test cases.

ISSN: 2005-4297 IJCA

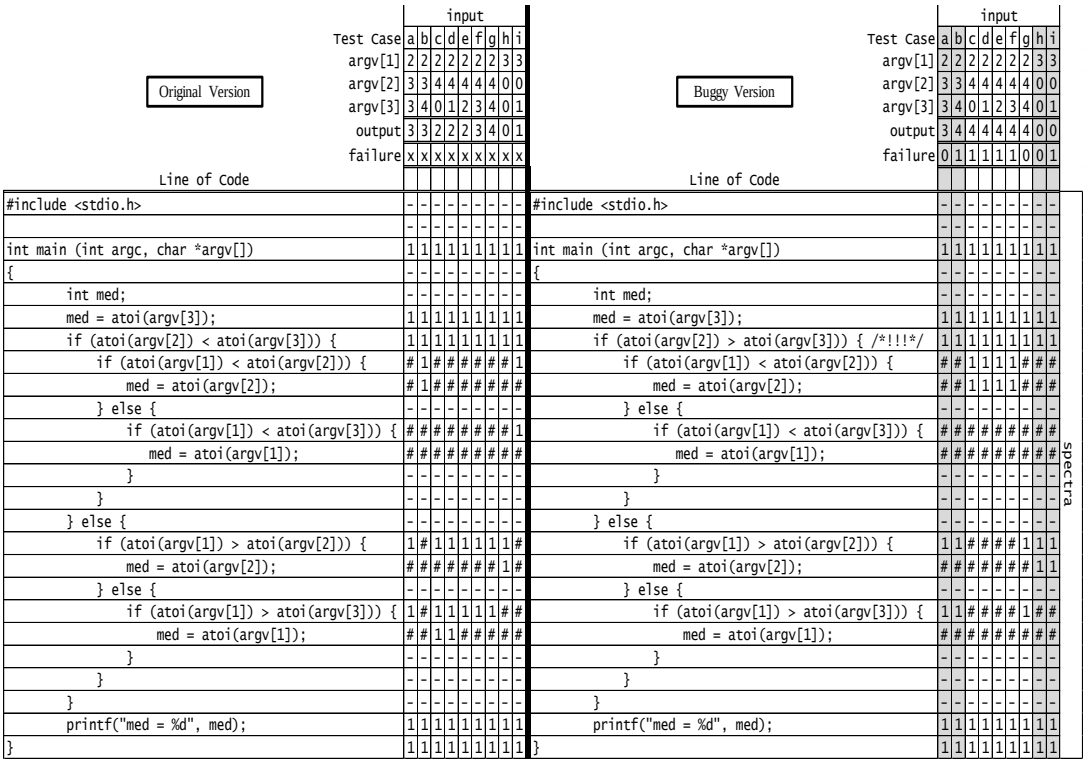


**Figure 1. Simple Median program for *noisy* test case example**

The spectra provide the essential information on the number of times a statement is executed or not executed by pass test cases and fail test cases respectively. This information is usually presented in four coefficients for each statement, namely, *aef*, *anf*, *aep*, and *anp*. *aef* represents the number of times the statement has been executed in all fail test cases while *aep* represents the number of times the statement has been executed in all pass test case. On the other hand, *anf* represents the number of times the statement has not been executed in all fail test cases while *anp* represents the number of times the statement has not been executed in all pass test case. Many SBFL metrics have been proposed to calculate the likeliness of a statement to be faulty based on one or more of these four coefficients. The statements are then ranked according their likeliness to be faulty based on the calculated SBFL metric values. This will allow software developer to examine the statements that are more likely to be buggy first, rather than examining statement by statement from beginning to end of the program, which is very time consuming. The performance of a SBFL metric is measured by EXAM score, which is the percentage of the ranked statements to be examined before the faulty statement is successfully located.

Nine test cases {*a, b, …, i*}have been executed on the buggy version of the program. From the spectra of test cases executed on the buggy version, there are two pairs of test cases that produce identical spectra. The first pair are test cases {*a, b*} and the second pair are test cases {*h, i*}. Test case *a* is a pass test case as it has the same output with correct version. However, test case *b* is a fail test case because its output from the faulty version is different from the correct version. Since both test case *a* (a pass test case) and *b* (a fail test case) share the same spectra, they provide ambiguous information to the SBFL metric. In other words, the spectra from this pair of test cases are essentially *noise* for SBFL metrics which in turn will deteriorate the performance of SBFL.

 

From this example, it is evident that the performance of SBFL is not only affected by the effectiveness of SBFL metric, but also the noisiness of the spectra collected from the executed test cases. In this paper, we propose a suite of noise reduction schemes as pre-processor filter out noisy test cases before the spectra are used to calculate for calculation by the SBFL metrics. With the removal of noisy test cases through these noise reduction schemes, it is envisaged that the performance of SBFL metrics will improve further to reduce the cost and time spent to locate the faulty statement.

This paper makes the following contributions:

1. We propose six noise reduction schemes to improve the performance of over 30 SBFL metrics.
2. We empirically evaluate the performance improvement in over 30 SBFL metrics as a result of applying the proposed noise reduction schemes.
3. We generated a guide for practitioners to select the best performing noise reduction scheme for the SBFL metrics that they are using.

## 2. Preliminaries

### 2.1. Software Artifacts Used for Empirical Study

In our study, we use the well-known Siemens Test Suite as our subject programs for testing and debugging. Siemens Test Suite is widely used as the software artifacts studies related to fault localization in the research community [2, 3, 4, 5, 6, 7]. It can be downloaded from Software Information Repository [8]. There are seven programs in Siemen Test Suite. For each program, there is one correct version and multiple faulty versions of the program. Table 1 shows the programs in the Siemens Test Suite, the total number of faulty versions for each program, line of code, total number of test cases and the description of the program.

For the experiments conducted in this paper, we have executed all test cases for each program. *print_tokens* {v4, v6} have been excluded because there is no faulty statement found in these versions. We focus our study on on single-fault versions. Therefore, *print_tokens* {v1}, *replace* {v21}, *schedule* {v2, v7}, and *tcas* {v10, v11, v15, v31, v32, v33, v40} have been excluded because multiple faulty statements exist in these versions. We have also excluded *print_tokens* {v2}, *replace* {v12}, *tcas* {v13, v14, v36, v38}, *tot_info* {v6, v10, v19, v21} because the faulty statement is an non-executable statement. In addition, *print_tokens2* {v10}, *replace* {v32}, and *schedule2* {v9} have been excluded because there is no fail test case (that is, failure not detected) even though faulty statement exists in program code. We have also excluded the versions which not contain any duplicate or ambiguous test case, which are *print_tokens* {v3, v7}, *print_tokens2* {v2, v5}, *replace* {v15, v17, v19, v20}, *schedule* {v1, v6, v9}, *schedule2* {v4}, *tcas* {v1, v4, v20, v21, v24, v25, v39, v41}, and *tot_info* {v1, v11, v15}. Lastly, *schedule* {v8}, *tcas* {v2, v3, v6, v7, v8, v9, v16, v17, v18, v19, v22, v23, v26, v28, v29, v30, v35, v37} and *tot_info* {v3, v14} have been excluded because all failed test cases are eliminated by the one of the proposed noise reduction scheme. In summary, from total 132 versions of programs, 70 versions have been excluded, leaving 62 versions to be used in the experiment.

We have used the GCC version 4.6.1 and Gcov (GNU-GCC), running on a Ubuntu 11.10 workstation to collect the spectra information from these versions of programs.

**Table 1. Programs in Siemens Test Suite**

| Program | Faulty Versions | LOC | Number of Test Cases | Description | Versions excluded in experiments |
|---|---|---|---|---|---|
| *print_tokens* | 7 | 563 | 4130 | Lexical analyser | 1, 2, 3, 4, 6, 7 |
| *print_tokens2* | 10 | 508 | 4115 | Lexical analyser | 2, 5, 10 |
| *replace* | 32 | 563 | 5542 | Pattern recognition | 12, 15, 17, 19, 20, 21, 32 |
| *schedule* | 9 | 410 | 2650 | Priority scheduler | 1, 2, 6, 7, 8, 9 |
| *schedule2* | 10 | 307 | 2710 | Priority scheduler | 4, 9 |
| *tcas* | 41 | 173 | 1608 | Altitude separation | 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 40, 41 |
| *tot_info* | 23 | 406 | 1052 | Information measure | 1, 3, 6, 10, 11, 15, 16, 19, 21 |

### 2.2. Spectrum-based Fault Localization Metrics

Many SBFL metrics have been proposed for SBFL. Each metric might have varying performance. The main purpose of these metrics is to produce good ranking for faulty statement so that software developer only needs to examine as few statement as possible to locate the faulty statement. Table 2 lists the SBFL metrics that will be studied in this paper.

## 3. Problem

From the analysis conducted on the spectra of faulty program in Siemen Test Suite, we have observed that, in many versions of the programs, there exist test cases have identical spectra (statement execution coverage record) even though the test inputs are different. These observations on test cases with identical spectra can be divided into three categories.

1. A fail test case and a pass test case share the same spectra for a faulty program. In this case, they provide ambiguous information to the SBFL metric on whether or not the statements executed by this pair of test cases are likely to be faulty. Hence, the spectra from this pair of test cases are essentially noise for SBFL metrics which in turn may deteriorate the performance of SBFL.
2. More than one fail test cases share the same spectra for a faulty program. In this case, they provide duplicated information to the SBFL metrics on all the statements executed by this set of fail test cases. This duplicated information is essentially noise to the correct statements which are executed by this set of fail test cases, which in turn may deteriorate the performance of SBFL.
3. More than one pass test cases share the same spectra for a faulty program. In this case, they provide duplicated information to the SBFL metrics on all the statements executed by this set of pass test cases. This duplicated information is essentially noise to the faulty statements which are executed by this set of pass test cases, which in turn may deteriorate the performance of SBFL.

Based on these observations, it is evident that the duplicated and ambiguous information presence in test cases with identical spectra may cause inadvertent deterioration to the performance of SBFL metrics.

 

**Table 2. Spectrum-based Fault Localization metrics**

| Name | Formula | Name | Formula |
|---|---|---|---|
| Naish1 | $\begin{cases}-1 & if\ aef < F \\ P - aep & if\ aef = F\end{cases}$ | Zoltar | $\frac{aef}{aef + anf + aep + \frac{10000anfaep}{aef}}$ |
| Naish2 | $aef - \frac{aep}{aep + anp + 1}$ | Simple Matching | $\frac{aef + anp}{aef + anf + aep + anp}$ |
| Jaccard | $\frac{aef}{aef + anf + aep}$ | Sokal | $\frac{2(aef + anp)}{2(aef + anp) + anf + aep}$ |
| Anderberg | $\frac{aef}{aef + 2(anf + aep)}$ | Rogers & Tanimoto | $\frac{aef + anp}{aef + anp + 2(anf + aep)}$ |
| Sorensen-Dice | $\frac{2aef}{2aef + anf + aep}$ | Russel & Rao | $\frac{aef}{aef + anf + aep + anp}$ |
| Dice | $\frac{2aef}{aef + anf + aep}$ | AMPLE | $\left\lvert\frac{aef}{aef + anf} - \frac{aep}{aep + anp}\right\rvert$ |
| qe | $\frac{aef}{aef + aep}$ | Tarantula | $\frac{aef}{aef + anf} \Big/ \left(\frac{aef}{aef + anf} + \frac{aep}{aep + anp}\right)$ |
| Wong1 | $aef$ | CBI Inc. | $\frac{aef}{aef + aep} - \frac{aef + anf}{aef + anf + aep + anp}$ |
| Hamming etc. | $aef + anp$ | Ochiai | $\frac{aef}{\sqrt{(aef + anf)(aef + aep)}}$ |
| Binary | $\begin{cases}0 & if\ aef < F \\ 1 & if\ aef = F\end{cases}$ | Euclid | $\sqrt{aef + anp}$ |
| Kulczynski1 | $\frac{aef}{anf + aep}$ | AMPLE2 | $\frac{aef}{aef + anf} - \frac{aep}{aep + anp}$ |
| M1 | $\frac{aef + anp}{anf + aep}$ | M2 | $\frac{aef}{aef + anp + 2(anf + aep)}$ |
| Wong3 | $aef - h, \quad where\ h = \begin{cases}aep & if\ aep \le 2 \\ 2 + 0.1(aep - 2) & if\ 2 < aep \le 10 \\ 2.8 + 0.001(aep - 10) & if\ aep > 10\end{cases}$ | | |
| Ochiai2 | $\frac{aefanp}{\sqrt{(aef + aep)(anp + anf)(aef + anf)(aep + anp)}}$ | | |
| Arithmetic Mean | $\frac{2aefanp - 2anfaep}{(aef + aep)(anp + anf) + (aef + anf)(aep + anp)}$ | | |
| Geometric Mean | $\frac{aefanp - anfaep}{\sqrt{(aef + aep)(anp + anf)(aef + anf)(aep + anp)}}$ | | |
| Harmonic Mean | $\frac{(aefanp - anfaep)((aef + aep)(anp + anf) + (aef + anf)(aep + anp))}{(aef + aep)(anp + anf)(aef + anf)(aep + anp)}$ | | |
| Rogot2 | $\frac{1}{4}\left(\frac{aef}{aef + aep} + \frac{aef}{aef + anf} + \frac{anp}{anp + aep} + \frac{anp}{anp + anf}\right)$ | | |
| Cohen | $\frac{2aefanp - 2anfaep}{(aef + aep)(anp + aep) + (aef + anf)(anf + anp)}$ | | |

## 4. Proposed Noise Reduction Schemes

To tackle the problem in Section 3, we propose six noise reduction schemes to eliminate test cases with identical spectra by comparing the test case execution coverage information between test cases. The noise reduction schemes are:

1. Noise Reduction Scheme 1 (NRS1) – for each fail test case, all pass test cases with spectra identical to the fail test case will be removed and eliminated from the calculations of SBFL metrics.
2. Noise Reduction Scheme 2 (NRS2) – for each pass test case, all fail test cases with spectra identical to the pass test case will be removed and eliminated from the calculations of SBFL metrics. Note that this noise reduction scheme may not be used if it results in all fail test cases being removed from the calculation of SBFL metrics because SBFL require at least one fail test case to be presence.
3. Noise Reduction Scheme 3 (NRS3) – for each set of pass and fail test cases with identical spectra, all test cases in the set will be removed and eliminated from the calculations of SBFL metrics. Note that this noise reduction scheme may not be used if it results in all fail test cases being removed from the calculation of SBFL metrics because SBFL require at least one fail test case to be presence.
4. Noise Reduction Scheme 4 (NRS4) – This technique was proposed by Lee, Naish and Ramamohanarao in [9] in a related study on the effect of using non-redundant test cases for SBFL. For each set of pass test cases with identical spectra, all except one test case will be removed and eliminated from the calculations of SBFL metrics. Similarly, for each set of fail test cases with identical spectra, all except one test case will be removed and eliminated from the calculations of SBFL metrics.
5. Noise Reduction Scheme 5 (NRS5) – This noise reduction scheme is a cascading of NRS4 and NRS 1. In other words, NRS4 is applied to the test cases before NRS1
6. Noise Reduction Scheme 6 (NRS6) – This noise reduction scheme is a cascading of NRS4 and NRS 2. In other words, NRS4 is applied to the test cases before NRS2.
7. Noise Reduction Scheme 7 (NRS7) – This noise reduction scheme is a cascading of NRS4 and NRS 3. In other words, NRS3 is applied to the test cases before NRS3.

Each of these noise reduction schemes is applied to the executed test suite (the set of test cases executed) of the faulty program to remove and eliminate test cases with identical spectra with the aim improve the performance of SBFL metrics.

## 5. Experiment Results

In order to validate our observation that there exist test cases with identical spectra in faulty program, experiment has been conducted on faulty versions of programs in Siemen Test Suite. Table 3 shows the number of test cases with identical spectra that have been detected and successfully removed by the proposed noise reduction schemes. Due to the space limitation, only the numbers of test cases removed for NRS1 to NRS3 are shown as indication. It can be observed that the percentage of test cases detected to have identical spectra and removed by the noise reduction schemes ranges from 0.45% to 27.48% for the programs in Siemen Test Suites. This implies that, without noise reduction scheme, the levels of noise presence in the Siemen Test Suite can be significantly high, which in turn may deteriorate the performance of SBFL.

 

**Table 3. The number of test cases removed by the proposed noise reduction scheme**

| Program | No. of Version Tested | No. of Test Cases in each Faulty Version | Total No. of Test Cases for all Faulty versions | Total No. of Test Cases Removed by NRS1 | Total No. of Test Cases Removed by NRS2 | Total No. of Test Cases Removed by NRS3 |
|---|---|---|---|---|---|---|
| *print_tokens* | 1 | 4130 | 4130 | 63 (1.53%) | 17 (0.41%) | 80 (1.94%) |
| *print_tokens2* | 7 | 4115 | 28805 | 877 (3.04%) | 200 (0.69%) | 1077 (3.74%) |
| *replace* | 25 | 5542 | 138550 | 1212 (0.87%) | 624 (0.45%) | 1836 (1.33%) |
| *schedule* | 3 | 2650 | 7950 | 1743 (21.92%) | 442 (5.56%) | 2185 (27.48%) |
| *schedule2* | 8 | 2710 | 21680 | 1423 (6.56%) | 195 (0.90%) | 1618 (7.46%) |
| *tcas* | 4 | 1608 | 6432 | 1076 (16.73%) | 149 (2.32%) | 1225 (19.05%) |
| *tot_info* | 14 | 1052 | 14728 | 2100 (14.26%) | 324 (2.20%) | 2424 (16.46%) |

Experiments have been conducted by applying the proposed noise reduction scheme on programs in Siemen Test Suite to evaluate the effects of these schemes on the performance of SBFL metrics. The performance of a SBFL metric can be evaluated with the EXAM score, which is the percentage of statements needs to be examined before the faulty statement is located. The EXAM scores of the SBFL metrics under each noise reduction scheme are presented in Table 4. The EXAM scores shown are average for all 62 faulty versions of programs in Siemen Test Suite. The EXAM scores of SBFL metrics when there is no noise reduction scheme (NRS) is used is presented in the left most column of Table 4 as the benchmark for comparison. A ↑ symbol is used to indicate that the performance of the SBFL metrics has improved (that is, EXAM score becomes lower) under an NRS in comparison to the performance when no NRS is used. Conversely, a ↓ symbol is used to indicate that the performance of the SBFL metrics has worsen (that is, EXAM score becomes higher) under an NRS in comparison to the performance when no NRS is used. Lastly, a – is used to indicate that the performance of the SBFL metrics is unchaged (that is, EXAM score remains unchanged) under an NRS in comparison to the performance when no NRS is used.

Based on the experiment results in Table 4, it can be observed that, under NRS1, NRS3, NRS4, NRS5, NRS6 and NRS7, the EXAM scores for most SBFL metrics have become lower, which means that the performance of these SBFL metrics have improved. On the other hand, NRS2 only bring improvement to a small number of SBFL metrics.

**Table 4. The EXAM scores of SBFL metrics under the proposed noise reduction schemes**

| | No NRS | NRS1 | | NRS2 | | NRS3 | | NRS4 | | NRS5 | | NRS6 | | NRS7 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Naish1 | 5.99 | 5.99 | - | 6.70 | ↓ | 6.70 | ↓ | 5.98 | ↑ | 5.98 | ↑ | 6.72 | ↓ | 6.70 | ↓ |
| Naish2 | 5.89 | 5.89 | - | 6.60 | ↓ | 6.60 | ↓ | 5.89 | ↑ | 5.89 | ↑ | 6.62 | ↓ | 6.60 | ↓ |
| Jaccard | 10.16 | 9.90 | ↑ | 10.10 | ↑ | 9.84 | ↑ | 8.70 | ↑ | 8.86 | ↑ | 8.59 | ↑ | 8.57 | ↑ |
| Anderberg | 10.16 | 9.90 | ↑ | 10.10 | ↑ | 9.84 | ↑ | 8.70 | ↑ | 8.86 | ↑ | 8.59 | ↑ | 8.57 | ↑ |
| Sorensen-Dice | 10.16 | 9.90 | ↑ | 10.10 | ↑ | 9.84 | ↑ | 8.70 | ↑ | 8.86 | ↑ | 8.59 | ↑ | 8.57 | ↑ |
| Dice | 10.16 | 9.90 | ↑ | 10.10 | ↑ | 9.84 | ↑ | 8.70 | ↑ | 8.86 | ↑ | 8.59 | ↑ | 8.57 | ↑ |
| Tarantula | 10.80 | 10.66 | ↑ | 10.50 | ↑ | 10.18 | ↑ | 9.58 | ↑ | 9.64 | ↑ | 9.18 | ↑ | 9.14 | ↑ |
| qe | 10.80 | 10.66 | ↑ | 10.50 | ↑ | 10.18 | ↑ | 9.58 | ↑ | 9.64 | ↑ | 9.18 | ↑ | 9.14 | ↑ |
| CBI_Inc. | 10.80 | 10.66 | ↑ | 10.50 | ↑ | 10.18 | ↑ | 9.58 | ↑ | 9.64 | ↑ | 9.18 | ↑ | 9.14 | ↑ |
| Simple_Matching | 17.17 | 16.73 | ↑ | 17.27 | ↓ | 16.87 | ↑ | 16.57 | ↑ | 16.25 | ↑ | 16.79 | ↑ | 16.57 | ↑ |
| Sokal | 17.17 | 16.73 | ↑ | 17.27 | ↓ | 16.87 | ↑ | 16.57 | ↑ | 16.25 | ↑ | 16.79 | ↑ | 16.57 | ↑ |
| Rogers&Tanimoto | 17.17 | 16.73 | ↑ | 17.27 | ↓ | 16.87 | ↑ | 16.57 | ↑ | 16.25 | ↑ | 16.79 | ↑ | 16.57 | ↑ |
| Hamming_etc. | 17.17 | 16.73 | ↑ | 17.27 | ↓ | 16.87 | ↑ | 16.57 | ↑ | 16.25 | ↑ | 16.79 | ↑ | 16.57 | ↑ |
| Euclid | 17.17 | 16.73 | ↑ | 17.27 | ↓ | 16.87 | ↑ | 16.57 | ↑ | 16.25 | ↑ | 16.79 | ↑ | 16.57 | ↑ |
| Wong1 | 10.45 | 10.45 | - | 10.99 | ↓ | 10.99 | ↓ | 10.45 | - | 10.45 | - | 10.99 | ↓ | 10.99 | ↓ |
| Russel & Rao | 10.45 | 10.45 | - | 10.99 | ↓ | 10.99 | ↓ | 10.45 | - | 10.45 | - | 10.99 | ↓ | 10.99 | ↓ |
| Binary | 10.55 | 10.55 | - | 11.09 | ↓ | 11.09 | ↓ | 10.55 | - | 10.55 | - | 11.09 | ↓ | 11.09 | ↓ |
| Ochiai | 7.78 | 7.88 | ↓ | 7.96 | ↓ | 7.91 | ↓ | 6.98 | ↑ | 7.22 | ↑ | 7.45 | ↑ | 7.42 | ↑ |
| M2 | 6.92 | 6.97 | ↓ | 7.24 | ↓ | 7.22 | ↓ | 6.65 | ↑ | 6.68 | ↑ | 7.22 | ↓ | 7.20 | ↓ |
| AMPLE2 | 10.45 | 10.45 | - | 10.99 | ↓ | 10.99 | ↓ | 10.45 | - | 10.45 | - | 10.99 | ↓ | 10.99 | ↓ |
| Wong3 | 6.17 | 6.17 | - | 18.78 | ↓ | 18.84 | ↓ | 6.31 | ↓ | 6.39 | ↓ | 20.61 | ↓ | 17.56 | ↓ |
| Arithmetic_Mean | 8.64 | 8.62 | ↑ | 8.66 | ↓ | 8.49 | ↑ | 8.43 | ↑ | 8.61 | ↑ | 8.21 | ↑ | 8.17 | ↑ |
| Cohen | 10.65 | 10.27 | ↑ | 10.31 | ↑ | 10.09 | ↑ | 9.32 | ↑ | 9.35 | ↑ | 8.88 | ↑ | 8.85 | ↑ |
| Kulczynski1 | 10.16 | 9.90 | ↑ | 10.10 | ↑ | 9.76 | ↑ | 8.70 | ↑ | 8.86 | ↑ | 8.59 | ↑ | 8.49 | ↑ |
| M1 | 17.17 | 16.73 | ↑ | 17.27 | ↓ | 16.79 | ↑ | 16.57 | ↑ | 16.25 | ↑ | 16.79 | ↑ | 16.48 | ↑ |
| Ochiai2 | 10.88 | 10.67 | ↑ | 10.19 | ↑ | 9.99 | ↑ | 10.74 | ↑ | 10.63 | ↑ | 10.36 | ↑ | 10.32 | ↑ |
| Zoltar | 5.90 | 6.06 | ↓ | 6.62 | ↓ | 6.62 | ↓ | 5.89 | ↑ | 5.97 | ↓ | 6.64 | ↓ | 6.61 | ↓ |
| Ample | 10.74 | 10.86 | ↓ | 10.87 | ↓ | 10.75 | ↓ | 10.45 | ↑ | 10.69 | ↑ | 10.81 | ↓ | 10.77 | ↓ |
| Geometric_Mean | 8.97 | 8.92 | ↑ | 8.55 | ↑ | 8.46 | ↑ | 8.39 | ↑ | 8.52 | ↑ | 8.29 | ↑ | 8.27 | ↑ |
| Harmonic_Mean | 9.19 | 9.08 | ↑ | 8.70 | ↑ | 8.57 | ↑ | 8.65 | ↑ | 8.88 | ↑ | 8.47 | ↑ | 8.45 | ↑ |
| Rogot2 | 9.19 | 9.08 | ↑ | 8.70 | ↑ | 8.57 | ↑ | 8.65 | ↑ | 8.88 | ↑ | 8.47 | ↑ | 8.45 | ↑ |

## 6. Discussion

From the experiment results in Section 5, the performance of most SBFL metrics has improved when noise reduction scheme NRS1 is applied to remove pass test cases with identical spectra to the fail test cases. Conversely, less SBFL metrics has improved shown improvements in their performance when NRS2 is applied to remove failed test cases with identical spectra to the pass test cases. These observation suggests that fail test cases carries more information on the location of the faulty statement than pass test cases, which justify why NRS1 perform better than NRS2.

On the other hand, the SBFL demonstrated a mix performance under the NRS3, where both pass and fail test cases with identical spectra are removed. This is because the essential information about the location of faulty statement is lost when the fail test cases are removed together with the pass test cases.

In NRS5, NRS6 and NRS7, when NRS4 is applied to the test suite before the NRS1, NRS2 and NRS3, we could observe the performance of SBFL metrics are better compared to when NRS1, NRS2 and NRS3 is applied alone. In particular, we could observe that NRS5, which cascade NRS4 and NRS1 delivers the best performance for many SBFL metrics.

**Table 5. Noise reduction guide for SBFL practitioners**

| NRS TO USE | SBFL METRICS |
|---|---|
| NRS1 | - |
| NRS2 | - |
| NRS3 | Ochiai2 |
| NRS4 | Naish1, Naish2, Ochiai, M2, Zoltar, Ample |
| NRS5 | Naish1, Naish2, Simple_Matching, Sokal, Rogers&Tanimoto, Hamming_etc, Euclid, M1 |
| NRS6 | - |
| NRS7 | Jaccard, Anderberg, Sorensen-Dice, Tarantula, qe, CBI_Inc, Arithmetic_Mean, Cohen, Kulczynski1, Geometric_Mean, Harmonic_Mean, Rogot2 |
| DO NOT USE NRS | Wong1, Russel&Rao, Binary, Ample2, Wong3 |

On the other hand, while NRS7, which cascades NRS4 and NRS3, may worsen the performance of a few SBFL metrics, it delivers the best performance for most number of SBFL metrics studied.

Based on SBFL metrics performance in Table 4, we have generated a guide for practitioners of SBFL to select the best performing noise reduction scheme for the SBFL metrics that they use in Table 5.

## 7. Conclusion

In this paper, we propose six noise reduction schemes to remove and eliminate test cases which provide duplicated and ambiguous information and evaluate the resulting performance improvements in over 30 SBFL metrics under study. From the experiments conducted on 62 faulty versions of programs in Siemen Test Suite, we found that test cases with identical spectra can be as high as 27% in Siemen Test Suite. This significantly high percentage of test cases with identical spectra is essentially noise to the SBFL metrics which may in turn deteriorate the performance of SBFL metrics.

Experiments were conducted by applying the six proposed noise reduction schemes on programs in Siemen Test Suite to evaluate the effects of these schemes on the performance of SBFL metrics. The experiment results showed that the proposed noise reduction schemes have successfully improved the performance of SBFL metrics under study. In addition, we have further identified the best performing noise reduction scheme for each SBFL metric. Based on our findings, we further provide a guide for SBFL practitioners to select the best performing noise reduction scheme for the SBFL metrics that they use.

As for future work, we plan to extend this study to programs with multiple faulty statements. In addition, we are also conducting theoretical analysis on the effect of noise reduction on the performance of SBFL.

## Acknowledgements

This work is supported via Malaysian Government MOHE Fundamental Research Grant Scheme (FRGS/2/2010/TK/SWIN/02/03).

## References

[1] J. S. Collofello and S. N. Woodfield, “Evaluating the effectiveness of reliability-assurance techniques”, Journal of Systems and Software, vol. 9, no. 3, **(1989)**, pp. 191–195.
[2] R. Abreu, P. Zoeteweij and A. van Gemund, “An Evaluation of Similarity Coefficients for Software Fault Localization”, Proceedings of the 12th PRDC, **(2006)**, pp. 39–46.

[3] R. Abreu, P. Zoeteweij and A. van Gemund, "On the Accuracy of Spectrum-based Fault Localization", TAICPARTMUTATION, **(2007)**, pp. 89–98.
[4] C. Liu, X. Yan, L. Fei, J. Han and S. P. Midkiff, "Sober: statistical model-based bug localization", SIGSOFT Softw. Eng. Notes, **(2005),** pp. 286–295.
[5] B. Pytlik, M. Renieris, S. Krishnamurthi and S. Reiss, "Automated Fault Localization Using Potential Invariants", Arxiv preprint cs.SE/0310040, **(2003)**.
[6] M. Renieres and S. Reiss, "Fault localization with nearest neighbor queries", Proceeding of ASE, (**2003)**, pp. 30–39.
[7] H. J. Lee, L. Naish and K. Ramamohanarao, "A model for spectra-based software diagnosis", ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 20, no. 3, **(2011)**, pp. 11.
[8] H. Do, S. Elbaum and G. Rothermel, "Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact", Empirical Software Engineering, vol.10, no.4, **(2005)**, pp. 405–435.
[9] H. J. Lee, L. Naish and K. Ramamohanarao, "The Effectiveness of Using Non redundant Test Cases with Program Spectra for Bug Localization", Proceedings of the 2nd IEEE international Conference, Computer Science and Information Technology, ICCSIT, **(2009)** August 8–11, pp. 127–134.

## Authors

**Patrick Daniel**

Patrick Daniel received his Bachelor of Science (CSSE) from Swinburne University of Technology in 2011. He is currently a Master of Science candidate at Swinburne University of Technology, Sarawak Campus, Malaysia. His research interest is software testing and debugging.

**Kwan Yong Sim**

Kwan Yong Sim received his BEng (Hons) from the National University of Malaysia in 1999 and Masters of Computer Science from University of Malaya, Malaysia in 2001. He is currently a Senior Lecturer and the Associate Dean for Curriculum Enhancement and Accreditation at the Faculty of Engineering, Computing and Science, Swinburne University of Technology, Sarawak Campus, Malaysia. His research interests include software testing and analysis. Mr. Sim is a member of IEEE and IEEE Computer Society.

 