mathematics

Article

Effectively Combining Risk Evaluation Metrics for Precise Fault Localization

Adekunle Ajibode [1], Ting Shu [1,*], Laghari Gulsher [2] and Zuohua Ding [1]

[1] School of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China
[2] Institute of Mathematics and Computer Science, University of Sindh, Jamshoro 76080, Sindh, Pakistan
* Correspondence: shuting@zstu.edu.cn

Abstract: Spectrum-based fault localization (SBFL) is an automated fault localization technique that uses risk evaluation metrics to compute the suspiciousness scores from program spectra. Thus, risk evaluation metrics determine the technique's performance. However, the existing experimental studies still show no optimal metric for different program structures and error types. It is possible to further optimize SBFL's performance by combining different metrics. Therefore, this paper effectively explores the combination of risk evaluation metrics for precise fault localization. Based on extensive experiments using 92 faults from SIR and 357 faults from Defects4J repositories, we highlight what and which risk evaluation metrics to combine to maximize the efficiency and accuracy of fault localization. The experimental results show that combining risk evaluation metrics with high negative correlation values can improve fault localization effectiveness. Similarly, even though the combination of positively correlated effective risk evaluation metrics can outperform the majority of negatively correlated non-effective ones, it still cannot improve the fault localization effectiveness. Furthermore, low-correlated risk evaluation metrics should also be considered for fault localization. The study concluded that getting highly negatively correlated risk evaluation metrics is almost impossible. The combination of such risk evaluation metrics would improve fault localization accuracy.

Keywords: Spectrum-based fault localization; Risk evaluation metrics; metric combination; software testing

MSC: 68N30

check for updates

Citation: Ajibode, A.; Shu, T.; Gulsher, L; Zuohua D. Effectively Combining Risk Evaluation Metrics for Precise Fault Localization. *Mathematics* **2022**, *10*, 3924. https://doi.org/10.3390/math10213924

Academic Editor: Ioannis G. Tsoulos

Received: 9 September 2022
Accepted: 19 October 2022
Published: 22 October 2022

1. Introduction

Program fault localization [1–4] plays an indispensable role in software debugging. It can assist developers in efficiently locating the root cause of software failure and greatly improve the development efficiency and quality of software products. Currently, there exist many families of fault localization, such as spectrum-based fault localization (SBFL), mutation-based fault localization, dynamic program slicing, information-retrieval-based fault localization, and mutating the results of conditional expressions [5]. The SBFL method only uses program spectra (test coverage information) to determine which program statements are most suspicious and associated with the fault by computing the suspiciousness scores for each executed program statement [6,7]. Therefore, the SBFL is lightweight, so it has become one of the most effective methods at present [8].

In SBFL, the risk evaluation metric is the key to determining the performance of fault localization, which is a well-designed formula responsible for computing the suspiciousness score for each program statement based on program spectra. Therefore, researchers have extensively researched constructing a risk evaluation metric to maximize fault localization performance. In recent years, although a large number of metrics, such as Tarantula[1], Ample [9], MECO [10], D^{*2} [11], and GP [12,13], have been proposed one after another, the existing experimental studies [12,14–16] have shown that there is no optimal metric for different program structures and error types. Even though, the researchers, through empirical evaluations, have identified that some risk evaluation metrics are more effective (maximal metrics) than others (non-maximal) [13,14]. Some risk evaluation metrics such as Tarantula [1], Ample [9], D^{*2} [11], GP [12,13], OP1 and OP2 [17], SBI [18], Jaccard [19], Ochiai1 and Ochiai2 [20], and Wong 1–3 [21] are regarded as maximal ones because of their superiority in performance against others [14,22]. At the same time, other metrics such as Anderberg, Sørensen-Dice, Dice, Goodman, Hamman, Simple Matching, Rogers-Tanimoto, Hamming, Euclid, Russel&Rao, Rogot1, Scott, M1, M2, Arithmetic-Mean, Fleiss, and Cohen, are regarded as non-maximal due to their effectiveness in locating the faults [14,17,22].

Due to such a predicament, some recent works propose a new way to improve fault localization performance by combining metrics. Some preliminary studies by Wang [23] and Jifeng [24] have tried to combine multiple ranking metrics. The former proposed search-based algorithms to form a combination. This study searched for the optimal candidate combination from 22 risk evaluation metrics but did not consider their correlations, which is one of the most critical factors in combining two techniques. The latter proposed a learning-to-rank approach using machine learning, randomly selecting the existing risk evaluation metrics without considering their suitability for combination. However, these initial attempts have yielded some encouraging results. Unfortunately, how to pick and combine risk evaluation metrics to achieve the best performance has become a critical problem to solve.

Fortunately, Zou et al.'s work [5] on how to combine multiple different families of techniques to achieve efficient fault localization found that (1) when there is less correlation between two techniques of different families, they may provide different information and utilizing that information may enhance fault localization effectiveness (2) when two metrics are good at locating specific faults, they are positively correlated, and the performance of their combination might not optimize the efficiency of SBFL. Nevertheless, a prolific qualitative concern is if different metrics in SBFL have high or low correlation, and can their combination boost fault localization?

Similar to the studies mentioned above, and because of different intuitions in proposing each risk evaluation metric in the literature, it is challenging to determine the practicability of the existing studies. To this effect, Xiao-Yi Zhang and Mingye Jiang [25] proposed a SPectra Illustration for Comprehensive Analysis (SPICA) method for reviewing and analyzing the existing SBFL techniques via visualization of the spectrum data. They found that some metrics' performance curves (MPC) have similar behaviours in SPectra space (SP). In this case, it is the best practice to consider the diversity of metrics by selecting the metrics with different MPCs.

The similarity of this problem inspires us to explore how to choose the optimal combination of metrics to maximize the solution's performance. That is, whether it is possible to propose an optimal strategy for metrics combination selection from the perspective of the correlation between them. Accordingly, these ideas prompted us to use the following rubric for combining the existing risk evaluation metrics to conduct our experiments:

- *Two positively correlated maximal metrics are already good at locating software faults. The effectiveness of their combination might outweigh their individual fault localization effectiveness.*
- *Two negatively correlated maximal metrics are good at locating different sorts of faults. Their combined performances might outweigh their individual fault localization effectiveness.*
- *Two positively correlated non-maximal metrics are not good at locating software faults, yet their combination might increase the overall fault localization effectiveness.*
- *Two negatively correlated non-maximal metrics are not good at locating different sorts of faults, yet combining them might increase overall fault localization effectiveness.*
- *Two positively or negatively correlated maximal and non-maximal metrics, where one is good, and the other is not good at locating software faults, might complement each other, and their combination might outweigh their individual fault localization effectiveness.*

In order to corroborate the above-listed rubric, a risk evaluation metric is considered *good or effective* if it can place the faulty statements in the program under test to the top of the ranking list, thereby helping the developers locate faults without going through many non-faulty statements. Therefore, this paper explores and evaluates all the combinations mentioned above. This empirical evaluation of metrics combination will help further research improving spectrum-based fault localization.

It is important to note that this study only focuses on exploring an effective way of choosing and combining the existing risk evaluation metrics to improve the performance of fault localization further.

This paper makes the following main contributions:

- An empirical study is performed to explore the performance and combination of different risk evaluation metrics based on whether two metrics are maximal or non-maximal, correlated positively or negatively, and the magnitude of the correlation. We then evaluate which combination is effective for fault localization.
- Experimental results demonstrate that negatively correlated risk evaluation metrics are best combined for effective fault localization, especially metrics with negatively low correlation values because they can outperform both the existing risk evaluation metrics and the individual ones that make the combination.
- This is the first study to use the degree of correlation of the same fault localization family, spectrum-based fault localization, to suggest the ideal techniques to be combined.

The rest of this paper is organized as follows. Section 2 provides a background on spectrum-based fault localization. Section 3 provides details on how we combine the risk evaluation metrics. In Section 4, we explain the setup of the experiments for our empirical study, and in Section 5, we provide the results. Section 6 discuses the threats to the validity of this empirical study, and Section 7 concludes the paper.

2. Spectrum-Based Fault Localization

The first step of the debugging process is to precisely locate the fault at any granularity level, e.g., statements, blocks, methods, or classes. Spectrum-based fault localization is an automated technique to pinpoint the location of a fault in code.

The primary assumption in spectrum-based fault localization is that the program statements covered in more failing tests and few or no passing tests are assumed to be faulty. Thus, it collects the coverage information of program statements as a tuple of four values (e_f, e_p, n_f, n_p) called program spectrum. Typically, e_f and e_p represent the failing and passing tests that execute the program statement, and n_f and n_p are the numbers of failing and passing tests that do not execute the program statement, respectively. The risk evaluation metric then translates the program spectrum into a suspiciousness score for each program statement, indicating its likelihood of being faulty.

Various risk evaluation metrics have been proposed by different researchers (e.g., Table 1) to assign the highest rank to the faulty program statement, as mentioned in the introduction part of this paper. The spectrum information of seeded and real faults are not similar, making many risk evaluation metrics' performances more superior in one sort of fault location to another.

Thus, combining two different risk evaluation metrics precisely localizing faults in different programs could enhance their performances in localizing unknown faults since both provide information on various aspects of the faults [5]. Therefore, in this paper, our goal is to explore different metrics suitable for combination for effective fault localization.

Many studies have theoretically evaluated the effectiveness of risk evaluation metrics as a standalone method, and some have categorized them according to their efficacy for effective fault localization. This study, therefore, gives a clue to the possible combination of these risk evaluation metrics for effective fault localization.

Theoretically, Xie et al. [26] assess the manually-created risk evaluation metrics used in earlier studies to compute suspiciousness scores and show that the two best SBFL families are ER1 (a, b) and ER5 (a, b, c). The follow-up study by Xie et al. [27] investigated SBFL metrics generated by executing an automated genetic programming technique in a Shin Yoo study [12]. Their research discovered that GP (02, 03, 13, and 19) are the best GP-generated metrics. SAVANT was presented by Le et al. [28], which introduced Daikon [29] invariants to SBFL as an extra feature. They used the learning to rank model to combine SBFL approaches with invariant data. SAVANT outperformed the best four SBFL metrics, including MULTRIC, on real-world faults from the Defects4J dataset [30]. Sohn and Yoo [31] proposed FLUCCS, which combined SBFL approaches with code change measures. They used Genetic Programming and linear rank Support Vector Machines to learn how to rank items. They also evaluated FLUCCS on the Defects4J dataset and discovered that it outperforms existing SBFL approaches. Kim et al. [32] also presented a learn-to-rank fault localization technique named PRINCE. This approach used a genetic algorithm to combine dynamic features, such as spectrum-based fault localization, mutation-based fault localization, and static features, such as dependency information and structural complexity of the program's entity. The approach was evaluated on 459 artificial and real-world faults. The results show that PRINCE is more effective than all the state-of-the-art techniques, spectrum-based fault localization, mutation-based fault localization, and learning-to-rank.

Table 1. The Maximal risk evaluation metrics.

Name	Definition
Tarantula [33]	$\frac{\frac{e_f}{e_f+n_f}}{\frac{e_f}{e_f+n_f}+\frac{e_p}{e_p+n_p}}$
Ample [9]	$\lvert\frac{e_f}{e_f+n_f}-\frac{e_p}{e_p+n_p}\rvert$
Ochiai1 [34]	$\frac{e_f}{\sqrt{(e_f+n_f)(e_f+e_p)}}$
Jaccard [19]	$\frac{e_f}{e_f+e_p+n_f}$
Ochiai2 [34]	$\frac{e_f*n_p}{\sqrt{(e_f+e_p)(n_p+n_f)(e_f+n_f)(e_p+n_p)}}$
Kulczynski1 [35]	$\frac{e_f}{n_f+e_p}$
OP2 [17]	$e_f-\frac{e_p}{e_p+n_p+1}$
D^{*2} [11]	$\frac{{e_f}^2}{e_p+n_f}$
GP02 [12]	$2(e_f+\sqrt{n_p})+\sqrt{e_p}$
GP03 [12]	$\sqrt{\lvert{e_f}^2-\sqrt{e_p}\rvert}$
GP19 [12]	$e_f\sqrt{\lvert e_p-e_f+n_f-n_p\rvert}$
Wong1 [21]	e_f
Wong2 [21]	e_f-e_p
Wong3 [21]	$e_f-h, h=\begin{cases} e_p & \text{if } e_p\leq 2 \\ 2+0.1(e_p-2) & \text{if } 2<e_p\leq 10 \\ 2.8+0.001(e_p-10) & \text{if } e_p>10 \end{cases}$

3. Combining the Risk Evaluation Metrics

Zou et al. demonstrated that when two techniques are effective in different kinds of fault in a different way, they may likely outperform the individual standalone technique [5]. Inspired by this, we study the different performance correlations of existing metrics and then, combine them to determine which combination is the best for effective spectrum-based fault localization.

Therefore, the combinations of the metrics in this study involve direct combination method where suspiciousness scores of two risk evaluation metrics are combined to enhance the efficiency of spectrum-based fault localization.

3.1. Selection of Suspiciousness Metrics for Combination

This study explores and assesses the combinations of risk evaluation metrics based on how the two metrics are correlated and whether or not the metrics are maximal. Thus, according to our rubric, when two risk evaluation metrics are combined, there are three possible combinations to whether or not the metrics are maximal.

(1) Both metrics are maximal.
(2) Both metrics are non-maximal.
(3) One metric is maximal, and the other metric is non-maximal.

To decide whether a risk evaluation metric is effective (maximal) or non-effective (non-maximal), we rely on the studies by Yoo et al. and Wu et al., who evaluated and found some metrics as maximal [14,22]. Consequently, the other risk evaluation metrics are termed as non-maximal in this study.

Similarly, when two risk evaluation metrics are combined, there are five possible combinations with respect to how much the two are correlated in locating the faults. The correlation of two metrics may result in (very) high, moderate, low, negligible, and neutral [36].

(1) Both metrics have *high* (H) correlation ($0.71 \leq r \leq 1.00$).
(2) Both metrics have *moderate* (M) correlation ($0.51 \leq r \leq 0.70$).
(3) Both metrics have *low* (L) correlation ($0.31 \leq r \leq 0.50$).
(4) Both metrics have *negligible* (N) correlation ($0.00 < r \leq 0.30$).
(5) Both metrics have *neutral* (U) correlation ($r = 0.00$).

In this case, we first run a pre-experiment to determine which metrics to combine based on their performance correlation. To quantify the correlation between each pair of techniques, we compute *Pearson correlations* r, which measures the linear correlation between two variables [37]. Recall that the developers will only examine the first few program statements suggested as faulty by any fault localization method. This shows the possibility of using the total number of statements checked by the developers before locating the first fault, *wasted effort*, to assess the effectiveness of each risk evaluation metric. To this effect, we selected 44 existing risk evaluation metrics (see [17] for the details of all the risk evaluation metrics) to compute the wasted effort scores in 30 iterations for 30 selected faulty programs from Defects4J and SIR-repository datasets to determine their performances. We selected five faults from the SIR-repository and five from each project of Defects4J i-e Charts, Closure, Lang, Math, and Time. Thus, effectively selecting both real and seeded faults in this pre-experiment. Table 2 provides the correlation values (r) amongst risk evaluation metrics (see Algorithm 1 for the process of selecting the studied combined-risk evaluation metrics). .

Algorithm 1 Metrics Selection

```
Require: risk evaluation metrics (I), faulty and non-faulty programs (F)
  $I \leftarrow (\alpha_1), (\alpha_2), (\alpha_3), ......(\alpha_n)$        ▷ list of metrics
  $F \leftarrow f_1, f_2, f_3......f_n$        ▷ list of the 30 selected faulty and non-faulty programs
  for each $i \in \mathcal{I}$ do
      for each $f \in \mathcal{F}$ do
          $we \leftarrow$ compute wasted efforts for each statement
      end for
  end for
  for all $we$ do
      compute the Pearson correlation ($r$)
  end for
  for all $r$ do
      compare the correlation of ($i$), and group in twos accordingly
      extract the possible combinations as suggested above
      group the possible combinations
  end for
  return grouped risk evaluation metrics
```

Table 2. The correlation results of the selected combined metrics: p = 0.005.

	OP2	Tarantula	Ochiai1	Ample	Jaccard	D^{*2}	GP02	Dice	Rogers-Tanimoto	SEM3	Barinnel	Hamman	SEM1	SEM2	Rogot1
Ochiai2							0.005								
D^{*2}															
GP02			0.005												
GP03						0.439									
GP19						0.537			−0.002						
Wong1	0.999														
Wong2		−0.119		−0.129											
Wong3		−0.119		−0.129											
Euclid		−0.371								−0.007					
Dice									0.860						
Rogers-Tanimoto		−0.456													
SEM3								0.860							
Russel&Rao	0.999														
Barinnel		0.633													0.468
M1						0.005									
Harmonic-Mean					0.472										
Scott													−0.414		
SEM1									−0.361						
Rogot1											0.468				
M2												0.005			
Cohen														0.597	
Fleiss													−0.985		

Note that this is an experiment to determine which risk evaluation metrics to select for combination after observing their correlations. Another experiment is conducted after the combination, and the result is given in Section 5.

Furthermore, we performed a series of experiments to combine three, four, and even five metrics before considering only two risk evaluation metrics for combination. This is because the combination of three, four, and higher numbers of combination require more in-depth analysis, which is beyond the scope of this study. For example, if we want to combine three different metrics, A, B, and C, we must observe the following scenario; A is correlated or not correlated with B and C, B is correlated or not correlated with A and C, and C is correlated or not correlated with A and B. Therefore, our future work hopes to evaluate the higher number of combinations empirically, and their degree of correlation, such as A is highly correlated with B and lowly correlated with C.

Consequently, due to the size of the correlation table, this study only shows the correlation values of the metrics considered for combination.

3.2. Combination Method

In order to demonstrate how our combination method works, we take spectra information of a program used in Zheng et al. [38]. First, we randomly selected different metrics without considering their correlation to compute suspiciousness scores, Wong1 and Russel&Rao. Second, we purposely selected two metrics after observing their correlation, GP19 and Rogers-Tanimoto, to compute suspiciousness scores for the used spectra. The correlation value of the two metrics, GP19 and Rogers-Tanimoto, is (-0.002). See Section 3.1 and Algorithm 2 for the full details and algorithm of how the correlations and combinations are computed.

Notably, the natural combination of the suspiciousness scores computed by different metrics is not a fair practice for fault localization. This is because different metrics have different ranges [39]. Some metrics have a range of $[0, 1]$, such as Rogers-Tanimoto and Russel&Rao, and others have a range of $[0, infinity]$, such as Wong1 and Wong2. In this case, we normalized the suspiciousness scores computed by each metric to be combined. To achieve this, we employed *MinMaxScaler* function of python *Scikit-learn* [40] library, which can convert the suspiciousness scores to $[0, 1]$. Concretely, the *MinMaxScaler* transforms the minimum suspiciousness score to 0 and the maximum one to 1. All other suspiciousness scores are normalized between 0 and 1. Table 3 demonstrates the initial suspiciousness scores before and after applying *MinMaxScaler* function to the example in Zheng et al. [38].

The columns with the letter *(n)* imply normalized suspiciousness scores, and column *Purposely* and column *Randomly* imply the combined suspiciousness scores. In the given example program spectra, the S_6 is the faulty statement, and both GP19 and Rogers-Tanimoto metrics did not localize the fault individually. However, when combined with their normalized scores, the fault is localized successfully.

Algorithm 2 Metrics Combination

```
Require: paired risk evaluation metrics (I), faulty and non-faulty programs (F)
  $I \leftarrow (\alpha_a, \alpha_b), (\alpha_a, \alpha_b), (\alpha_a, \alpha_b), ......(\alpha_n, \alpha_m)$        ▷ list of paired metrics
  $F \leftarrow f_1, f_2, f_3......f_n$        ▷ list of the faulty programs
  for each $i \in \mathcal{I}$ do
      for each $f \in \mathcal{F}$ do
          $s_a \leftarrow$ compute suspiciousness scores for each statement using metric 'a'
          $s_b \leftarrow$ compute suspiciousness scores for each statement using metric 'b'
      end for
  end for
  for all the $s_a, s_b$ do
      normalize using MinMaxScaler()
      $N_a \leftarrow$ normalized metric 'a'
      $N_b \leftarrow$ normalized metric 'b'
  end for
  for each $j \in \mathcal{N}_a$ do
      for each $k \in \mathcal{N}_b$ do
          $Final_{susp} \leftarrow (j + k)$        ▷ Final Suspiciousness score
          $Rank \leftarrow Final_{susp}$
      end for
  end for
  return Final Rank
```

Note that we first used each metric to compute the suspiciousness scores for each statement. The suspiciousness scores are normalized, but we did not normalize the spectra data.

Protocol: If a metric has a high correlation with more than one metric, we select the highest correlation for that group. This implies that if metric α has a correlation ($r = 0.92$) with metric β and also has a correlation ($r = 0.91$) with metric γ, we only select (α and β) for that group.

It is essential to note in Table 2 that no two maximal metrics have a high-negative or neutral correlation. The highest negative correlation ($r = -0.128$) between two maximal metrics is between (Ample and Wong1) and (Ample and Wong2), which according to [36] falls under negligible correlation. This can be explained that the maximal metrics are efficiently locating software faults. Hence, they are always correlated even though the level of correlation may vary; one metric may outperform the other. Therefore, we use only the available negatively correlated maximal metrics in this study.

Table 3. Example of the MinMaxScaler values from scikitlearn. The (n) signifies the normalized columns.

S_i	e_p	e_f	n_p	n_f	GP19	Rogers-Tanimoto	Wong1	Russel&Rao	GP19 **(n)**	Rogers-Tanimoto **(n)**	Wong1 **(n)**	Russel&Rao **(n)**	**Randomly**	**Purposely**
S_1	8	2	0	0	4.899	0.111	2.000	0.200	1.000	0.000	1.000	1.000	2.000	1.000
S_2	8	2	0	0	4.899	0.111	2.000	0.200	1.000	0.000	1.000	1.000	2.000	1.000
S_3	4	2	4	0	2.828	0.429	2.000	0.200	0.577	0.572	1.000	1.000	2.000	1.149
S_4	1	0	7	2	0.000	0.538	0.000	0.000	0.000	0.768	0.000	0.000	0.000	0.768
S_5	3	2	5	0	4.000	0.538	2.000	0.200	0.816	0.768	1.000	1.000	2.000	1.584
S_6	2	2	6	0	4.899	0.667	2.000	0.200	1.000	1.000	1.000	1.000	2.000	2.000
S_7	4	0	4	2	0.000	0.250	0.000	0.000	0.000	0.250	0.000	0.000	0.000	0.250
S_8	4	0	4	2	0.000	0.250	0.000	0.000	0.000	0.250	0.000	0.000	0.000	0.250
S_9	3	0	5	2	0.000	0.333	0.000	0.000	0.000	0.399	0.000	0.000	0.000	0.399
S_10	1	0	7	2	0.000	0.538	0.000	0.000	0.000	0.768	0.000	0.000	0.000	0.768
S_11	0	0	8	2	0.000	0.667	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000
S_12	8	2	0	0	4.899	0.111	2.000	0.200	1.000	0.000	1.000	1.000	2.000	1.000

Table 4 shows the group of the shortlisted metrics. The combined metrics are labelled after their types and their correlation level. The first two letters represent if the two combined metrics are both maximal (MM), both non-maximal (NN), or one is maximal, and the other is non-maximal (MN). The subscript letter indicates the level of correlation i-e high correlation (H), moderate correlation (M), low correlation (L), or negligible correlation (N). The superscript signs $+$ and $-$ indicate whether the correlation is positive or negative, respectively.

Table 4. Details of selected combined risk evaluation metrics based on correlation.

Group	Metric Combination	Negative Combined Metrics	Correlation	Acronyms	Positive Combined	Correlation
1	MM_{H^-}	Tarantula and Wong2	High	MM_{H^+}	OP2 and Wong1	High
	MM_{M^-}	Tarantula and Wong3	Moderate	MM_{M^+}	GP19 and Dstar	Moderate
	MM_{L^-}	Ample and Wong2	Low	MM_{L^+}	Dstar and GP03	Low
	MM_{N^-}	Ample and Wong3	Negligible	MM_{N^+}	Ochiai2 and GP02	Negligible
2	NM_{H^-}	Wong3 and SEM1	High	NM_{H^+}	OP2 and Rutsel	High
	NM_{M^-}	Rogers and Tarantula	Moderate	NM_{M^+}	Tarantula and Barinel	Moderate
	NM_{L^-}	Tarantula and Euclid	Low	NM_{L^+}	Jaccard and Harmonic	Low
	NM_{N^-}	GP19 and Rogers-Tanimoto	Negligible	NM_{N^+}	Dstar and M1	Negligible
3	NN_{H^-}	Fleiss and SEM1	High	NN_{H^+}	Dice and SEM3	High
	NN_{M^-}	Scott and SEM1	Moderate	NN_{M^+}	SEM2 and Cohen	Moderate
	NN_{L^-}	Rogers and SEM1	Low	NN_{L^+}	Rogot1 and Barinel	Low
	NN_{N^-}	Euclid and SEM3	Negligible	NN_{N^+}	Hamman and M2	Negligible

For example, the first acronym MM_{H^-} means that the two combined risk evaluation metrics are both maximal and highly negatively correlated.

4. Experimental Design

This section provides the experimental details for assessing and evaluating the combinations of risk evaluation metrics for effective fault localization.

4.1. Fault Localization

We use spectrum-based fault localization as the fault localization process. The fault localization is performed at the statement granularity level. Spectrum-based fault localization can use any risk evaluation metrics to compute the suspiciousness of the statements for being faulty.

Thus, we evaluate the effectiveness of the combined risk evaluation metrics and compare the performances against 14 well-known maximal (see Table 1) and 16 non-maximal risk evaluation metrics. We also compare the performance of the combined metrics against 14 well-known maximal (see Table 1) and 16 non-maximal risk evaluation metrics (see [17]).

4.2. Dataset

For this experimental study, we use two datasets: SIR-repository [41] and Defects4J (version 1.2.0) [30]. This study cannot use the latest version of the defects4j dataset, v2.0.0, due to unavoidable circumstances. However, version 1.2.0 still serves our purpose in this study. Furthermore, all the benchmarks used in this study may represent majorities of real faults of Defects4J. SIR-repository is the dataset of seeded faults in programs written in C language that has been used in fault localization research [20,21,26,42]. In contrast, Defects4J is the dataset of 357 real faults from 5 large open-source Java projects recently used in fault localization and repair [43–46]. The details of the datasets can be found in our previous study [10].

To evaluate the effectiveness of the combined and existing risk evaluation metrics in fault localization concerning project size, we partition the projects of SIR-repository and Defects4J into three categories. This shall indicate which metrics are suitable for projects of different sizes.

There is no precisely defined limit to determine if a program is small, medium, or large, as Zhang et al. consider Flex, Grep, Gzip, and Sed programs in SIR-repository measuring between 5.5 and 9.69 KLOC as real-life medium-sized programs [47]. In comparison, others consider these programs as large [2,48]. Keller et al. categorize Defects4J projects measuring in 28–96 KLOC range as a medium size [49]. While de Souza et al. consider Flex, Grep, Gzip, and Sed as small with the assumption that programs with more than 10 KLOC are large programs, programs containing between 2 and 10 KLOC are medium-size, while programs with less than 2 KLOC are small programs [50].

In this study, we categorize the datasets as:

Small:	Projects with $\leq$ 10 KLOC, such as *Flex*, *Sed*, *Grep*, and *Gzip* with an average executed statements of 3037, and have 92 faults.
Medium	Projects $>$10 KLOC $\leq$50 KLOC, such as *Lang* and *Time* with an average executed statements of 5725, and have 92 faults.
Large:	Projects $>$50 KLOC, such as *Math*, *Closure*, and *Chart* with an average executed statements of 14333, and contain 265 faults.

4.3. Research Questions

This study answers the following research questions.

RQ1. *Which combined metric performs the best among the combinations?* To answer this research question, we evaluate the performance of each group of the risk evaluation metrics (see Table 4) in the given datasets in [10].

RQ2. *How do the best performing combined risk evaluation metrics compare against the performance of standalone maximal and non-maximal risk evaluation metrics?* We select the best-performing risk evaluation metrics among the combined (maximal and maximal, non-maximal and maximal, and non-maximal and non-maximal) risk evaluation metrics to answer this research question. This is conducted by comparing all the combined risk evaluation metrics with each other. If a combined risk evaluation metric outperforms all other ones in two or more categories of the partitioned dataset, it is assumed to be better than the other risk evaluation metrics. We, therefore, select such a metric to represent the group of combinations. This resulted in 6 combined risk evaluation metrics compared with 14 maximal and 16 non-maximal risk evaluation metrics. In total, we compared 36 metrics.

RQ3. *Is there any statistical performance difference between the combined and standalone maximal and non-maximal risk evaluation metrics?* This research question statistically analyses the overall performance differences of the combined and existing risk evaluation metrics. We combine all the datasets for this experiment. We set the experiment to iterate fifteen times using each risk evaluation metrics to compute the average wasted effort for each fault (see Section 4.4.3 for the details). The experiment is instrumented to automatically exclude five faults per iteration to obtain a different *wasted effort* value for each risk evaluation metrics in each iteration. We then used Wilcoxon signed-rank test to test the statistical differences and Cliff's delta to test the effect sizes using the scores computed above. This aims to examine the significance of the performance difference between the combined and existing risk evaluation metrics.

4.4. Evaluation Metrics

We use the following four evaluation metrics to assess the effectiveness of the risk evaluation metrics.

4.4.1. Exam Score

The *Exam* score is the percentage of statements a developer needs to go through until the faulty statement is found [16,44]. Thus, the metric with the lowest *Exam* score has the highest effectiveness in locating the faults. The *Exam* metric is defined in Equation (1).

$$Exam = \frac{R(s_f)}{\mathrm{N}} * 100 \tag{1}$$

where the $R(s_f)$ is the rank of the first faulty statement, and N is the total number of executable statements.

4.4.2. acc@n

The acc@n metric counts the number of faults successfully localized at the top n position in the ranking list [28]. In our case, the $n \in \{1, 3, 5, 10\}$. When the fault expands multiple statements, we assume the fault is localized if any of the faulty statements is ranked among the top n positions. This study also used a max tie-breaker for this evaluation metric.

4.4.3. Average Wasted Effort (AWE)

As spectrum-based fault localization produces a ranked list of statements with the aim of ranking the faulty statements at the top, the common assumption is that developers start from the top of the list to identify the fault. The wasted effort metric measures the developer effort wasted in inspecting the non-faulty statements ranked higher than the faulty ones. While some non-faulty statements may be ranked higher than the faulty ones, some non-faulty statements likely share the same rank as the faulty ones—a tie. Sarhan et al. approximated that for 54–56% of the cases in Defects4J, the faulty methods share the same rank with at least one other method.

In the case of ties, there are two cases. The best case is that the faulty statement (s_f) is ranked higher than non-faulty statements (s_n) in the tie. Hence, the best case wasted effort is defined in Equation (2).

$$WE_{best} = |rank(s_f) > rank(s_n)| \tag{2}$$

The worst case is that the faulty statement (s_f) is ranked last than non-faulty statements (s_n) in the tie. Hence, the worst-case wasted effort is defined in Equation (3).

$$WE_{worst} = |rank(s_n) \geq rank(s_f)| \tag{3}$$

Therefore, inspired by Keller et al., we define the *wasted effort* (WE) as an average case in Equation (4) [49].

$$AWE = \frac{WE_{best} + WE_{worst}}{2} \tag{4}$$

Finally, we use Equation (4) to calculate the *Average Wasted Effort* (AWE), which is the mean of the *wasted effort* (WE) in all the rankings.

4.5. Tie Breaking

Ties always occur when ranking each statement in the program under test using the SBFL formulas. Breaking a tie between two or more program statements with the same suspiciousness scores is expedient. To achieve this, we use the *rankdata* function from a Scipy [51] library of python programming language, which is also a module for statistical functions and probability distribution. Tie-breaking is achievable on *rankdata* by calling a function *scipy.stats.rankdata(a, method)*, where *a* is the list of the suspiciousness scores and *method* can be *average, Min, Max, dense, and ordinal*. In our case, this study uses the *Max()* method because we hope to assign the maximum ordinal rank for the corresponding ties. When the returned values are sorted in descending order, the first value assigned with the maximum ordinal rank among the tied values is ranked first. Here, we assume that the sorting function breaks ties arbitrarily as specified by Spencer Pearson et al. [44]. For example, when *rankdata* function is applied to column *purposely* in Table 3, it returns [9, 9, 10, 5, 11, 12, 2, 2, 3, 5, 9, 9] and when applying a sorting function, it returns [12, 11, 10, 9, 9, 9, 9, 5, 5, 3, 2, 2]. Similarly, when the same function is applied to column *randomly* in the same Table 3, it returns [12, 12, 12, 6, 12, 12, 6, 6, 6, 6, 6, 12], when applying sorting function, it yields [12, 12, 12, 12, 12, 12, 6, 6, 6, 6, 6, 6]. Recall that the sixth value on the list belongs to the faulty statement. Therefore, the sorting function arbitrarily breaks the tie on the list by sorting the values from the largest to the lowest (descending order)

4.6. Statistical Tests

In this study, we use Wilcoxon signed-rank test to test the significance and Cliff's delta to measure the effect size. We determined if the combined risk evaluation metrics examined fewer program statements than the maximal and non-maximal ones before locating the first fault in the studied programs.

4.6.1. Wilcoxon Signed-Rank Test

Wilcoxon signed-rank test is a suitable alternative to other statistical tests, such as the t-test when a normal distribution of a given population cannot be assumed, particularly when there are matched pairs [13,16]. Since we have a matched pair comparison where we assess whether or not one risk evaluation metric or their combinations performs better than the other, we choose the Wilcoxon signed-rank test with degree freedom of 5%.

4.6.2. Cliff's Delta

Cliff's delta is a measure of effect size quantifying the magnitude of difference between two groups X and Y. Cliff's delta indicates that the dominance probability of observations in one group is larger than in the other group. In the context of this study, it informs us if a risk evaluation metric performs better than the other and its practical usefulness. Cliff's delta d is defined in Equation (7) [52].

$$d = \frac{\#(x_i > y_i) - \#(x_i < y_i)}{mn} \tag{7}$$

where # is the cardinality indicating the number of times, m and n is the size of group X and group Y, respectively. Each observation $x \in X$ is compared against each observation $x \in Y$ to count the number of times $x > y$ and $x < y$. Finally, the difference in counts is divided by total comparisons.

The value of Cliff's delta d is always $-1 \leq d \leq +1$. The extreme values -1 and $+1$ indicate the two groups X and group Y are completely non-overlapping (the two groups are significantly different), while 0 indicates the two overlap completely (the two groups are similar).

We used this statistical tool to compare the mean values of the *Average Wasted Effort* scores computed by the combined risk evaluation metrics, and all other ones studied in this study.

We adopt the interpretations of Cliff's delta d from Romano et al., which are approximated from Cohen's d as follows [53].

- $d = 0 \implies$ there is no difference in the performance of two risk evaluation metrics, and they are essentially the same.
- $d = 0.147 \implies$ there is a small difference in the performance of two risk evaluation metrics.
- $d = 0.33 \implies$ there is a medium difference in the performance of two risk evaluation metrics.
- $d = 0.474 \implies$ there is a large difference in the performance of two risk evaluation metrics.

When comparing two risk evaluation metrics, if the output results in negative, the average mean value of the treatment groups' data which are the combined risk evaluation metrics in our case, is smaller than the average mean value of the control groups' data which are the standalone risk evaluation metrics in this case.

5. Results

This section presents the results of the empirical evaluation of different risk evaluation metrics.

5.1. RQ1: Which Combined Metric Performs the Best among the Combinations?

We evaluate the effectiveness of combined risk evaluation metrics concerning their correlation as mentioned in Section 3.1. We compare their performance using the *Exam* and the *wasted effort* (*WE*) risk evaluation metrics. The evaluations and comparisons are based on the programs of SIR (92 faults) and Defects4J (357 faults).

Table 5 shows the performance of positively and negatively correlated combined risk evaluation metrics in small, medium, and large programs. Unfortunately, it is difficult to see negatively correlated two maximal risk evaluation metrics. Therefore, the few available ones are reported. The first column in Table 5 shows whether the two combined risk evaluation metrics are positively or negatively correlated. Next are the acronyms, as explained in Section 3, and the subsequent columns show the performance of each combined-risk evaluation metric in terms of *Exam* and *wasted effort*. The shaded combined-risk evaluation metrics show the best-performing ones to be compared with the existing risk evaluation metrics.

Table 5. Performance comparison of correlated positive and negative combined-risk evaluation metrics. The shaded metrics performed the best in each group and are shortlisted for further assessments.

		Small programs (3037 LOC)		**Medium programs (5725 LOC)**		**Large programs (14,333 LOC)**	
Group	**Combined Metrics**	**Exam**	**AWE**	**Exam**	**AWE**	**Exam**	**AWE**
Negative	MM_{H^-}	4.51	205	8.43	326	12.1	2163
	MM_{M^-}	4.51	205	8.43	326	12.1	2163
	MM_{L^-}	6.88	305	9.56	351	15.82	2780
	MM_{N^-}	6.88	305	9.56	351	15.82	2780
	NM_{H^-}	23.78	750	25.18	886	35.96	5357
	NM_{M^-}	7.20	235	8.72	322	14.45	2521
	NM_{L^-}	5.31	220	7.03	318	11.31	2057
	NM_{N^-}	7.91	281	11.44	396	17.95	3032
	NN_{H^-}	13.71	405	12.56	560	13.52	2493
	NN_{M^-}	25.41	799	13.38	578	15.45	2818
	NN_{L^-}	25.14	785	32.06	1020	41.64	5631
	NN_{N^-}	4.62	212	10.14	397	11.92	2300
Positive	MM_{H^+}	4.13	130	7.05	389	8.32	1989
	MM_{M^+}	15.12	1104	9.14	562	10.1	2877
	MM_{L^+}	18.31	1214	20.93	646	20.48	3140
	MM_{N^+}	12.02	905	6.69	319	10.46	2020
	NM_{H^+}	4.61	150	7.05	389	8.32	1989
	NM_{M^+}	4.52	149	6.50	511	7.04	2532
	NM_{L^+}	4.18	133	6.20	379	7.70	2023
	NM_{N^+}	20.45	708	35.39	870	42.19	5243
	NN_{H^+}	4.95	194	6.31	510	7.11	2528
	NN_{M^+}	5.02	207	7.48	374	8.83	2034
	NN_{L^+}	4.95	211	6.63	344	6.65	1545
	NN_{N^+}	5.11	224	7.44	322	11.79	2207

The best performing combined-risk evaluation metrics are determined when each shortlisted combined-risk evaluation metric outperforms the rest in two or more subject programs. We select one combined-risk evaluation metric to represent each grouped risk evaluation metrics. Recall that the groups are maximal and maximal (MM)), non-maximal and maximal (NM), and non-maximal and non-maximal (NN).

Negative Correlation

Among the combined two maximal risk evaluation metrics, the highly-correlated maximal risk evaluation metrics, MM_{L^-}, outperformed the others. For the combination of two non-maximal risk evaluation metrics, neutrally-correlated metrics, NN_{N^-} outperformed the rest. The combination of maximal and non-maximal risk evaluation metrics shows that lowly-correlated metric, NM_{L^-}, is promising. Furthermore, in terms of *Exam* scores, MM_{H^-} outperformed all other combinations in small programs and NM_{L^-} outperformed all other combinations in medium and large programs. In terms of *wasted effort*, MM_{H^-} outperformed all other combinations in the small program and NM_{L^-} outperformed all other combinations in medium and large programs. In summary, on average performance, the best negatively correlated combined-risk evaluation metric is NM_{L^-}, which is the combination of lowly-correlated maximal and non-maximal risk evaluation metrics.

Positive Correlation

For positive correlations, in terms of *Exam* scores, highly-correlated two maximal risk evaluation metrics outperformed all the other counterparts in small programs, lowly-correlated maximal and non-maximal risk evaluation metrics performed better in medium programs, and lowly-correlated two non-maximal risk evaluation metrics shows more promising than the others. In terms of *wasted effort*, the performance of two highly-correlated maximal risk evaluation metrics is better in the small program, and lowly-correlated combined-risk evaluation metric performs the best in medium and large programs. On average, lowly-correlated combined-risk evaluation metric, NN_{L^+}, is the best among the positively correlated risk evaluation metrics.

Answer to RQ1:

The overall performance shows that we can combine two positively or negatively correlated risk evaluation metrics for effective fault localization. Two findings hold for this research question; (1) two combined maximal risk evaluation metrics can not optimize fault localization because their combination can not outperform their individual risk evaluation metric; (2) lowly correlated risk evaluation metrics, whether positive or negative, can help in fault localization because their combination can outperform the individual metric. Therefore, this question is answered as follows; lowly correlated risk evaluation metrics are suitable for combination for effective fault localization.

5.2. RQ2: How Do the Best Performing Combined Risk Evaluation Metrics Compare against the Performance of Standalone Maximal and Non-maximal Risk Evaluation Metrics?

Table 6 shows the performance comparison if the best-performing combined metrics and the existing metrics. Similarly, Figure 1a shows the visualization of the Exam scores and Figure 1b shows the visualization of the wasted efforts of the compared risk evaluation metrics. We selected MM_{H^-}, NM_{L^-}, NN_{N^-}, MM_{H^+}, NM_{L^+}, and NN_{L^+} to represent the combined-risk evaluation metrics, maximal and maximal, non-maximal and maximal, and non-maximal and non-maximal risk evaluation metrics because of their outstanding performances against other risk evaluation metrics.

Therefore, we compare these six risk evaluation metrics with the other existing ones.

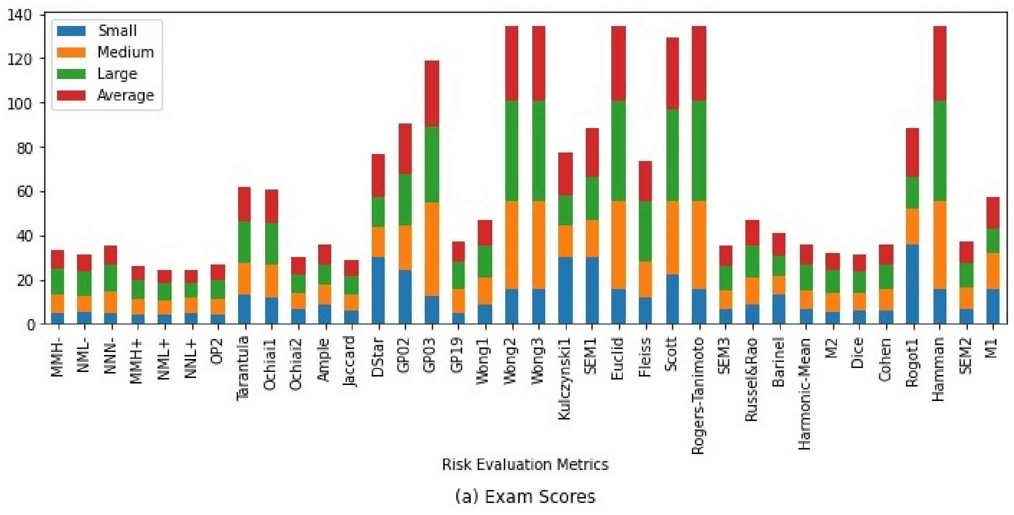
Small Programs: The small program benchmarks assessed the performance of the combined risk evaluation metrics and the existing ones. This experiment once again proved that some existing risk evaluation metrics are optimal in small programs, such as OP2. The OP2 risk evaluation metric is more effective in the small program in terms of *Exam* and *wasted effort* than all the combined risk evaluation metrics. Furthermore, the MM_{H^+} risk evaluation metric, which comprises OP2 and Wong1, performs like OP2, but better than Wong1. Similarly, apart from positively correlated two maximal risk evaluation metrics, all other combinations outperformed the individual risk evaluation metric combined to form them in small programs.

Medium Programs: The NM_{L^+} combined-risk evaluation metric outperformed all other combined-risk evaluation metrics and the existing risk evaluation metrics in terms of *Exam* in the medium program. In terms of *wasted effort*, OP2 risk evaluation metric shows more effective performance than all other compared risk evaluation metrics. The NM_{L^+} combined-risk evaluation metric, which outperformed other risk evaluation metrics in terms of *Exam*, comprises Jaccard and Harmonic-Mean. This combined-risk evaluation metric outperformed the standalone risk evaluation metrics used to form it. Similarly, apart from two highly correlated maximal risk evaluation metrics and low negatively correlated non-maximal risk evaluation metric, all other combinations outperformed their individual risk evaluation metric combined to form them.

Large Programs: The combined-risk evaluation metric, NN_{L+}, comprises two lowly correlated non-maximal risk evaluation metrics, Rogot1 and Barinnel, outperformed all other studied risk evaluation metrics in the large programs, in terms of *Exam* and *wasted effort*. All other combined-risk evaluation metrics outperformed the individual existing-risk evaluation metrics that make up the combined-risk evaluation metric, except the negative negligibly-correlated two non-maximal risk evaluation metrics in terms of *Exam*. Therefore, lowly-correlated two non-maximal risk evaluation metric are suitable for large programs.

Table 6. Performance comparison of the combined risk evaluation metrics with the standalone risk evaluation metrics. The best metrics for each subject program are highlighted in grey colour for easy visualization.

		Small programs (3037 LOC)		**Medium programs (5725 LOC)**		**Large programs (14,333 LOC)**	
Group	**Combined Metrics**	**Exam**	**AWE**	**Exam**	**AWE**	**Exam**	**AWE**
Combined	MM_{H^-}	4.51	205	8.43	326	12.1	2163
	NM_{L^-}	5.31	220	7.03	318	11.31	2057
	NN_{N^-}	4.62	212	10.14	397	11.92	2300
	MM_{H^+}	4.13	130	7.05	389	8.32	1989
	NM_{L^+}	4.18	133	6.20	379	7.70	2023
	NN_{L^+}	4.95	211	6.63	344	6.65	1545
Maximal	OP2	4.12	130	7.20	190	8.67	1999
	Tarantula	12.97	417	14.45	363	18.75	2190
	Ochiai1	12.21	402	14.45	361	18.77	2159
	Ochiai2	6.80	238	6.85	364	8.77	2203
	Ample	8.47	298	9.06	320	9.46	1558
	Jaccard	5.82	156	7.26	390	8.42	2158
	DStar	29.77	1098	14.05	440	13.64	2625
	GP02	24.05	758	20.18	343	23.44	1698
	GP03	12.43	345	42.57	503	34.12	2170
	GP19	4.99	159	10.85	401	12.08	2389
	Wong1	8.63	368	12.3	621	14.29	3325
	Wong2	15.92	498	39.26	605	45.62	2945
	Wong3	15.92	498	39.26	605	45.62	2945
	Kulczynski1	30.12	1111	14.27	441	13.72	2626
Non-maximal	SEM1	30.29	990	16.60	479	19.55	2679
	Euclid	15.92	498	39.26	605	45.62	2945
	Fleiss	11.90	403	16.25	254	27.11	2157
	Scott	21.96	711	33.23	463	41.67	2757
	Rogers-Tanimoto	15.92	498	39.26	605	45.62	2945
	SEM3	6.70	238	8.67	363	11.04	2214
	Russel&Rao	8.63	368	12.30	621	14.29	3325
	Barinnel	12.97	417	8.44	361	9.37	2159
	Harmonic-Mean	6.56	230	8.83	402	11.60	2059
	M2	5.28	176	8.72	389	10.03	2390
	Dice	5.89	190	8.26	380	9.42	2298
	Cohen	6.13	213	9.79	199	11.23	2080
	Rogot1	35.72	1244	16.31	520	14.43	2178
	Hamman	15.92	498	39.26	605	45.62	2945
	SEM2	6.56	230	9.61	380	11.51	2032
	M1	15.89	498	16.16	254	10.89	2290



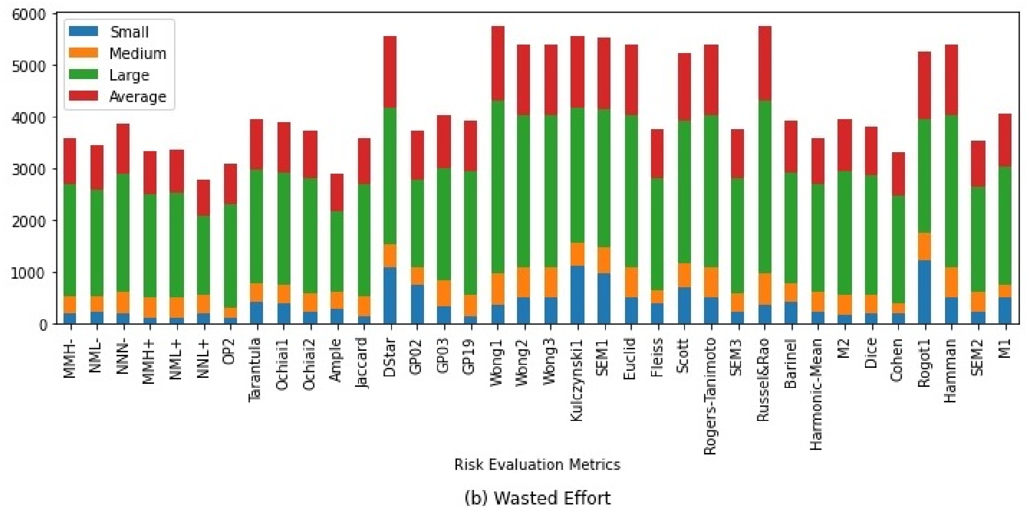


Figure 1. (**a**) shows the stacked bar-chart of Exam Scores of the combined and existing risk evaluation metrics. (**b**) shows the wasted effort computed by the combined and existing risk evaluation metrics.

We further conducted an accuracy level of all the risk evaluation metrics studied in his paper. We use acc@n to calculate the number of faults each metric localizes at the top of the suspiciousness list. The higher the number of faults located by the metric, the better such metric is in fault localization.

Table 7 and Figure 2 show the percentage and its visualization of faults that the combined and standalone risk evaluation metrics localized in small, medium, and large program benchmarks at the statement granularity level.

In small program benchmarks, Ample and GP19 placed more faults at the top of the ranking list than all the studied risk evaluation metrics, including the combined one in the small program benchmarks. Even though the majority of the combined-risk evaluation metrics, except positively low correlated two non-maximal and negatively low correlated maximal and non-maximal ones localized more faults at the top of the ranking list in the small programs than all the other risk evaluation metrics. Furthermore, no individual risk evaluation metric that makes up combined-risk evaluation metrics outperformed any of the combined-risk evaluation metrics.

Table 7. Performance comparison of the combined risk evaluation metrics with the standalone risk evaluation metrics based on acc@n evaluation metric.

		Small program (%)				**Medium program (%)**				**Large program (%)**			
	Metrics	**acc@1**	**acc@3**	**acc@5**	**acc@10**	**acc@1**	**acc@3**	**acc@5**	**acc@10**	**acc@1**	**acc@3**	**acc@5**	**acc@10**
Combined	MM_{H^-}	16	24	30	41	18	30	43	52	15	25	28	37
	NM_{L^-}	10	12	12	20	18	30	43	53	15	25	29	37
	NN_{N^-}	16	15	35	28	22	40	51	58	21	30	33	41
	MM_{H^+}	16	24	30	43	21	34	42	50	15	23	26	34
	NM_{L^+}	16	24	30	43	18	35	45	53	16	25	29	37
	NN_{L^+}	12	18	20	28	17	34	45	54	16	25	29	38
Maximal	OP2	16	24	30	43	17	30	40	49	16	23	26	34
	Tarantula	6	8	9	10	17	30	43	52	16	23	28	36
	Ochiai1	6	6	8	10	17	30	44	54	16	24	28	36
	Ochiai2	16	21	24	36	16	30	43	52	16	24	28	36
	Ample	17	24	30	41	17	29	37	46	15	21	24	31
	Jaccard	16	24	30	41	15	32	44	55	17	25	29	37
	DStar	2	8	10	18	13	28	37	45	14	20	24	30
	GP02	0	1	1	3	6	15	18	21	3	4	4	7
	GP03	13	21	25	32	4	6	11	13	1	4	4	5
	GP19	17	25	26	35	15	22	32	39	11	17	19	24
	Wong1	0	1	1	2	2	9	12	16	1	3	4	7
	Wong2	16	21	25	30	9	11	17	22	8	13	14	18
	Wong3	16	21	25	30	9	11	17	22	8	13	14	18
	Kulczynski1	2	8	10	18	13	27	38	45	14	21	25	30
Non-maximal	SEM1	0	0	0	1	1	3	11	15	0	2	4	5
	Euclid	10	18	23	28	9	11	17	22	8	13	14	18
	Fleiss	10	18	24	29	16	29	43	51	17	24	26	36
	Scott	5	5	7	9	12	17	26	34	11	18	19	23
	Rogers-Tanimoto	10	24	29	30	9	11	17	22	8	13	14	18
	SEM3	11	23	29	30	15	33	43	54	16	24	28	37
	Russel&Rao	0	1	1	2	2	9	12	16	1	3	4	7
	Barinel	6	8	9	10	17	30	44	54	16	24	28	36
	Harmonic-Mean	11	24	30	29	17	29	41	51	16	23	26	35
	M2	12	25	31	31	17	30	40	49	16	23	26	34
	Dice	10	20	25	29	15	32	44	55	17	25	29	37
	Cohen	9	19	25	29	15	30	43	54	17	24	28	36
	Rogot1	2	9	11	20	13	28	37	44	13	20	24	30
	Hamman	12	24	29	28	9	11	17	22	8	13	14	18
	SEM2	13	24	30	27	17	29	41	51	16	23	26	35
	M1	5	8	5	30	16	24	29	40	12	17	25	28

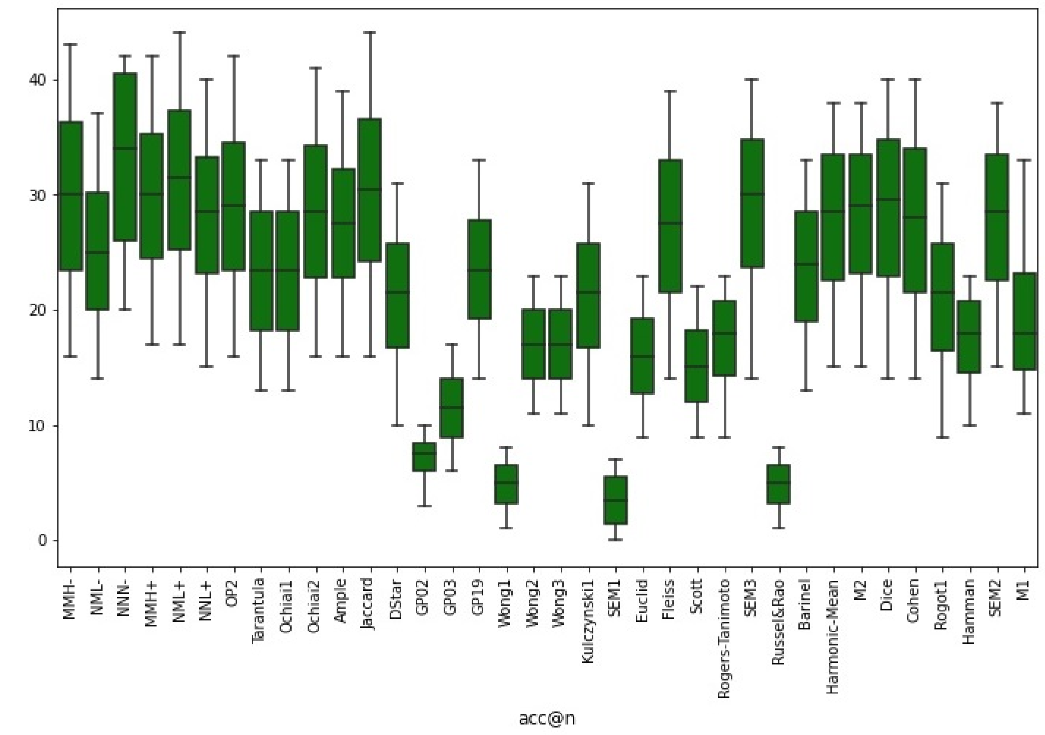


Figure 2. Fault Localization accuracy of the combined and existing risk evaluation metrics

In the medium programs, all the combined-risk evaluation metrics localized more faults than the studied standalone risk evaluation metrics, except the low positively correlated two non-maximal risk evaluation metrics that localized the same faults as some existing risk evaluation metrics.

In large programs, negligible negatively correlated risk evaluation metric, NN_{N-}, localized more faults at the top of the ranking list. We observed that aside from this risk evaluation metric, all other combined-risk evaluation metrics localized faults like or better than all the other studied risk evaluation metrics.

The general observation shows all the combined-risk evaluation metrics localized more faults than all the existing risk evaluation metric at the Top-10, and no individual risk evaluation metric combined to make the combined-risk evaluation metrics localized more fault than combined-risk evaluation metrics.

Answer to RQ2:
The result shows that the combined-risk evaluation metrics did not only outperform the individual risk evaluation metrics combined to form the combination but also outperformed all the existing ones, except OP2 in some cases. On average performance, low positive correlated maximal and non-maximal risk evaluation metrics performed the best in terms of *Exam*, low positive correlated two non-maximal risk evaluation metrics performed the best in terms of *wasted effort*, and negligible negatively correlated two non-maximal risk evaluation metrics performed the best in terms of acc@1.

5.3. RQ3: Is There Any Statistical Performance Difference between the Combined and Standalone Maximal and Non-maximal Risk Evaluation Metrics?

After comparing the performance of our proposed method and the existing methods, we estimate the statistical reliability of their performances, using the two statistical tools, Wilcoxon signed-rank test and Cliff's delta, that were previously used in [13,54–56].

We aim to confirm if the performance of the two compared risk evaluation metrics is similar or different. If their performances are different, we assume that one risk evaluation metric is better than the other. In this case, Wilcoxon signed-rank test, which can be used to compare two independent samples (*wasted effort* in our case), is an appropriate tool for this purpose. Cliff's delta, on the other hand, is an effect size computational tool that can quantify the differences between two groups of samples beyond the p-value. We use this tool to quantify the performance difference between the studied risk evaluation metrics; refer to RQ3 in Section 4.3 for how we generate the data for this purpose.

Table 8 contain the interpretations of the Wilcoxon statistical analysis. In the context of this study, for a given combined-risk evaluation metric (A) and existing-risk evaluation metric (B), the list of measures would be the list of the *Average Wasted Effort* score values for all the program faulty statements computed by A and B in 30 iterations. For the p-value, if $p \leq \alpha$, and $\alpha = 0.05$, we assume there is a significant difference between the two compared risk evaluation metric, but if $p > \alpha$, we assume there is no significant difference in the performance of the two compared risk evaluation metric. We have 6 combined, 14 maximal, and 16 non-maximal risk evaluation metric for comparison, and therefore, we have 180 pairs of statistical comparisons in this study.

Table 8 also contains Cliff's delta statistics that measure the effect sizes. The advantage of the Cliff's delta statistic is that if the *Average Wasted Effort* of the combinedrisk evaluation metric (A) is smaller than the existing-risk evaluation metric (B), then the d-value of Cliff's delta will return a negative value. This implies that risk evaluation metric (A) outperformed risk evaluation metric (B). The symbol '$<$' is used to denote if (α) of Wilcoxon signed-rank test is less than 0.05, '$>$' if higher than 0.05, and' $=$' if equal to 0.05.

From Table 8, each metric has two rows. The first row shows the Cliff's delta (d) values, and the second row shows Wilcoxon signed-rank test (α). We, therefore, deduce the following: There exist statistically significant differences between the combined-risk evaluation metrics and the existing-risk evaluation metrics with large effect sizes. The OP2 metric's performance against all the combined-risk evaluation metrics is statistically significant, with a large effect size in favour of OP2. Contrarily, the combined-risk evaluation metric, NN_{L^+}, outperformed OP2 with no visible effect size.

Table 8. The statistical comparison of the combined and standalone risk evaluation metrics.

		Maximal and Maximal		**Non-Maximal and Maximal**		**Non-Maximal and Non-Maximal**	
	Metrics	MM_{H^-}	MM_{H^+}	NM_{L^-}	NM_{L^+}	NN_{N^-}	NN_{L^+}
Maximal	OP2	1.00 $>$	0.58 $>$	0.78 $>$	0.58 $>$	1.00 $>$	−0.10 $<$
	Tarantula	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Ochiai1	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Ochiai2	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	0.15 $>$	−1.00 $<$
	Ample	0.94 $>$	0.85 $>$	0.88 $>$	0.85 $>$	1.00 $>$	−0.44 $<$
	Jaccard	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	1.00 $>$	−1.00 $<$
	DStar	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	GP02	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	1.00 $>$	−1.00 $<$
	GP03	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−0.65 $<$	−1.00 $<$
	GP19	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−0.95 $<$	−1.00 $<$
	Wong1	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Wong2	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Wong3	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Kulczynski1	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
Non-maximal	SEM1	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Euclid	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Fleiss	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	1.00 $>$	−1.00 $<$
	Scott	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Rogers-Tanimoto	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	SEM3	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	1.00 $>$	−1.00 $<$
	Russel&Rao	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Barinnel	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	1.00 $<$	−1.00 $<$
	Harmonic-Mean	−0.92 $<$	−0.82 $<$	−0.85 $<$	−0.82 $<$	1.00 $>$	−0.40 $<$
	M2	−1.00 $<$	−1.00 $>$	−1.00 $<$	−1.00 $<$	1.00 $<$	−1.00 $<$
	Dice	−0.94 $<$	−0.84 $<$	−0.88 $<$	−0.85 $<$	0.96 $>$	−0.44 $<$
	Cohen	0.15 $>$	0.10 $>$	0.12 $>$	0.10 $>$	0.45 $>$	−0.88 $<$
	Rogot1	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	Hamman	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$
	SEM2	0.20 $>$	−0.73 $<$	−0.75 $<$	−0.70 $<$	0.32 $>$	−0.31 $<$
	M1	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$	−1.00 $<$

Therefore, the statistical analysis shows that the best performing combined-risk evaluation metric, the positive lowly correlated one, did not only outperform all the existing studied metrics by chance, but the performances are statistically significant with large effect sizes.

Answer to RQ3:

The statistical analysis shows significant differences in the performance of the best combined-risk evaluation metrics and all other standalone risk evaluation metrics, and the effect sizes are large in many cases.

5.4. Discussion

Many techniques have been proposed in software testing, especially in an SBFL, to improve the suspicious statement's ranking in computer software programs. Human design techniques and Genetic Programming techniques have been adopted in the literature to minimize the effort of locating faults in a program. We have shown that combining two of these existing risk evaluation metrics to support fault localization is possible. We have combined different risk evaluation metrics to discover some types that can naturally be combined without machine learning or learning-to-rank algorithms. The majority of the studies in the literature that attempted combining various metrics did that with the aid of the Genetic Algorithm [23] and learning-to-rank [24,28,31] algorithm. Many of these studies combined many formulas without considering their compatibility.

This study combined different metrics by computing their suspiciousness scores for each program statement, and the suspiciousness scores are normalized to the range 0 and 1. The normalized scores are combined to form a single suspiciousness score. The single suspiciousness score is then served as the final suspiciousness score. We compared the effectiveness of the combined-risk evaluation metrics with the existing maximal and non-maximal risk evaluation metrics.

Table 5 shows the performances of negatively and positively correlated risk evaluation metrics. We observed that when combining two risk evaluation metrics, the result will either outperform one of the risk evaluation metrics or not perform like any of them. This can be observed when combining highly positively correlated risk evaluation metrics. This is why we ensured a comprehensive study of different combinations. Controversially, only two maximal risk evaluation metrics that are negative highly or moderately correlated are good for combination, unlike the positive and negative lowly or neutrally correlated ones.

Furthermore, most of the risk evaluation metrics with higher positive correlations outperformed those with lower correlations, even though their combined power can not supersede the individual ones. This bolsters the study by Zou [5], which says two techniques are positively correlated if they are good at locating the same sort of faults. This study further shows that if two techniques of the same family are highly and positively correlated, it does not mean they both have the same localization ability, and their combination will consistently outperform one of the combined techniques. This performance can not enhance the fault localization because the combination aims to optimize the fault localization performance.

Similarly, in addition to the findings above, some non-maximal risk evaluation metrics can also be combined with the maximal ones, provided they are low negatively correlated. Contrarily, positive lowly or neutrally correlated two non-maximal risk evaluation metrics should be considered if the developer intends to combine non-maximal risk evaluation metrics.

Table 6 shows the performance comparisons of the best performing combined-risk evaluation metrics with the existing ones. A combined-risk evaluation metric must outperform all other combined ones in two or more categorized datasets before concluding that it is the best among others. We use one best risk evaluation metric to represent each group (maximal vs maximal), (non-maximal vs maximal), and (non-maximal vs non-maximal). This produces six combined-risk evaluation metrics (MM_{H^-} (Tarantula and Wong2), NM_{L^-} (Euclid and Tarantula), NN_{N^-} (Euclid and SEM2), MM_{H^+} (OP2 and Wong1), NM_{L^+} (Jaccard and Harmonic-Mean), NN_{L^+} (Rogot1 and Barinnel)). We then compare their performances with the existing risk evaluation metrics.

This study shows that only one existing risk evaluation metric, OP2 is more effective than the majority of the combined-risk evaluation metrics, and overall performance of NN_{L+}, which comprises positively correlated two non-maximal risk evaluation metrics (Rogot1 and Barinnel)) outperformed all the compared risk evaluation metrics.

Therefore, to combine two techniques of the same family for effective fault localization, we may use two low-positive correlated non-maximal techniques. Even though it is very hard to obtain high negatively correlated risk evaluation metrics, their availability for combination will improve the accuracy of fault localization.

Therefore, we use the rubric highlighted in Section 1 to summarize our findings in this study.

- *Two positively correlated maximal metrics are already good at locating software faults. Their combination can not outweigh their performance and can not enhance fault localization.*
- *Negatively correlated two maximal risk evaluation metrics are good at locating different sorts of faults. Their combined performances can outweigh their individual fault localization effectiveness, provided their degree of correlation is moderate or high.*
- *Two positively correlated non-maximal metrics are not good at locating software faults, yet their combination can increase the overall fault localization effectiveness, provided their degree of correlation is low or neutral.*
- *Two negatively correlated non-maximal metrics are not good at locating software faults, and their combination can not improve fault localization effectiveness.*
- *Two negatively correlated maximal and non-maximal metrics, where one is good, and the other is not good at locating software faults, can complement each other provided they have low correlation, and their combination can outweigh their individual fault localization effectiveness.*

6. Threats to Validity

Since empirical research faces many risks concerning the validity of results, we identify those measures we took to alleviate them. We organize such threats to validity in three categories as follows.

- Construct validity: Threat to construct validity relates to the program granularity used for spectrum-based fault localization. In this study, we localize the faults at the statement level. Since this is the smallest possible granularity level and captures the program behaviour at a deficient level, this risk to construct validity is minimized. Furthermore, this is also in line with existing works. Many previous studies have localized faults at the statement granularity.
- Internal validity: The threat to internal validity relates to the evaluation metrics used to compare different risk evaluation metrics. One technique might be better than the other for a particular evaluation metric. Therefore, we use three evaluation metrics: the *Exam* score, *Average Wasted Effort*, and acc@n, which concern different aspects. Previous studies have also used these metrics [12,13,16,44]. Since each metric is concerned with different aspects used in previous studies, this threat is reasonably mitigated.
- External validity: The evaluation of risk evaluation metrics in this study depends on the dataset of subject programs used and may not be generalizable. Indeed, the results and findings of many fault localization studies are not directly generalizable.
  The threat to external validity is our method to determine which risk evaluation metrics are suitable for combination. We initially computed performance correlations between different risk evaluation metrics on 30 randomly selected faults from SIR-repository and Defects4J. It is highly likely to obtain different correlation results for another set of randomly selected faults from a different dataset.
  Nonetheless, this study has suggested the best metrics suitable for combination for effective fault localization.

7. Conclusions

This paper explored and empirically evaluated different combinations of maximal and non-maximal risk evaluation metrics based on their correlations. The primary aim of this empirical study was to ascertain which combinations of the risk evaluation metrics may improve the effectiveness of fault localization, more specifically, the spectrum-based fault localization.

The following findings are confirmed in this study.

Finding 1: The highest negatively correlated value of two maximal risk evaluation metrics obtained is -0.129. The combination outperformed the two combined risk evaluation metrics but did not outperform the best existing risk evaluation metric in this study, OP2. This means there are very high chances of getting an effective fault localization performance from two highly correlated maximal risk evaluation metrics with a value of at least -0.70 and above.

Finding 2: Practically, combining maximal and non-maximal risk evaluation metrics with moderate or high correlation power, whether positive or negative, can only outperform one individual risk evaluation metric, but not the two risk evaluation metrics. It is best to consider their low correlation power for effective fault localization.

Finding 3: Combining two non-maximal risk evaluation metrics, with high or moderate and positive or negative correlation power can not outperform the individual risk evaluation metric. Contrarily, the low or negligible positive correlation power of these risk evaluation metric can be considered for effective fault localization.

Author Contributions: Conceptualization, A.A. and T.S.; methodology, A.A. and T.S.; software, A.A.; data curation, A.A., T.S. and L.G.; original draft preparation, A.A.; writing-review and editing, A.A., L.G. and T.S.; visualization, A.A.; funding acquisition, T.S. and Z.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Zhejiang Provincial Natural Science Foundation of China under Grant No.LY22F020019, the Zhejiang Science and Technology Plan Project under Grant No.2022C01045, and the Natural Science Foundation of China under Grants 62132014 and 61101111.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jones, J.A.; Harrold, M.J.; Stasko, J. Visualization of test information to assist fault localization. In Proceedings of the 24th International Conference on Software Engineering. ICSE 2002, Orlando, FL, USA, 25 May 2002; pp. 467–477.
2. Abreu, R.; Zoeteweij, P.; Golsteijn, R.; Van Gemund, A.J.C. A practical evaluation of spectrum-based fault localization. *J. Syst. Softw.* **2009**, *82*, 1780–1792.
3. Wong, C.P.; Santiesteban, P.; Kästner, C.; Le Goues, C. VarFix: Balancing edit expressiveness and search effectiveness in automated program repair. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, 23–28 August 2021; pp. 354–366.
4. Ye, H.; Martinez, M.; Durieux, T.; Monperrus, M. A comprehensive study of automatic program repair on the QuixBugs benchmark. *J. Syst. Softw.* **2021**, *171*, 110825.
5. Zou, D.; Liang, J.; Xiong, Y.; Ernst, M.D.; Zhang, L. An Empirical Study of Fault Localization Families and Their Combinations. *IEEE Trans. Softw. Eng.* **2019**, *47*, 332–347. https://doi.org/10.1109/TSE.2019.2892102.
6. Srivastava, S. A Study on Spectrum Based Fault Localization Techniques. *J. Comput. Eng. Inf. Technol.* **2021**, *4*, 2.
7. Ghosh, D.; Singh, J. Spectrum-based multi-fault localization using Chaotic Genetic Algorithm. *Inf. Softw. Technol.* **2021**, *133*, 106512.
8. Jiang, J.; Wang, R.; Xiong, Y.; Chen, X.; Zhang, L. Combining spectrum-based fault localization and statistical debugging: An empirical study. In Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 11–15 November 2019; pp. 502–514.

9. Dallmeier, V.; Lindig, C.; Zeller, A. Lightweight defect localization for java. In *European Conference on Object-Oriented Programming*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 528–550.
10. Ajibode, A.; Shu, T.; Said, K.; Ding, Z. A Fault Localization Method Based on Metrics Combination. *Mathematics* **2022**, *10*, 2425.
11. Wong, C.P.; Xiong, Y.; Zhang, H.; Hao, D.; Zhang, L.; Mei, H. Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis. In Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 29 Septembe–3 October 2014; pp. 181–190.
12. Yoo, S. Evolving human competitive spectra-based fault localisation techniques. In *International Symposium on Search Based Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 244–258.
13. Ajibode, A.A.; Shu, T.; Ding, Z. Evolving Suspiciousness Metrics From Hybrid Data Set for Boosting a Spectrum Based Fault Localization. *IEEE Access* **2020**, *8*, 198451–198467.
14. Wu, T.; Dong, Y.; Lau, M.F.; Ng, S.; Chen, T.Y.; Jiang, M. Performance Analysis of Maximal Risk Evaluation Formulas for Spectrum-Based Fault Localization. *Appl. Sci.* **2020**, *10*, 398.
15. Heiden, S.; Grunske, L.; Kehrer, T.; Keller, F.; Van Hoorn, A.; Filieri, A.; Lo, D. An evaluation of pure spectrum-based fault localization techniques for large-scale software systems. *Softw. Pract. Exp.* **2019**, *49*, 1197–1224.
16. Wong, W.E.; Debroy, V.; Gao, R.; Li, Y. The DStar method for effective software fault localization. *IEEE Trans. Reliab.* **2013**, *63*, 290–308.
17. Naish, L.; Lee, H.J.; Ramamohanarao, K. A model for spectra-based software diagnosis. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2011**, *20*, 1–32.
18. Liblit, B.; Naik, M.; Zheng, A.X.; Aiken, A.; Jordan, M.I. Scalable statistical bug isolation. *Acm Sigplan Not.* **2005**, *40*, 15–26.
19. Chen, M.Y.; Kiciman, E.; Fratkin, E.; Fox, A.; Brewer, E. Pinpoint: Problem determination in large, dynamic internet services. In Proceedings of the Proceedings International Conference on Dependable Systems and Networks, Washington, DC, USA, 23–26 June 2002; pp. 595–604.
20. Abreu, R.; Zoeteweij, P.; Van Gemund, A.J.C. An evaluation of similarity coefficients for software fault localization. In Proceedings of the 2006 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06), Riverside, CA, USA, 18–20 December 2006; pp. 39–46.
21. Wong, W.E.; Qi, Y.; Zhao, L.; Cai, K.Y. Effective fault localization using code coverage. In Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), Beijing, China, 24–27 July 2007; Volume 1, pp. 449–456.
22. Yoo, S.; Xie, X.; Kuo, F.C.; Chen, T.Y.; Harman, M. Human competitiveness of genetic programming in spectrum-based fault localisation: Theoretical and empirical analysis. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2017**, *26*, 1–30.
23. Wang, S.; Lo, D.; Jiang, L.; Lau, H.C.; Others. Search-based fault localization. In Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, 6–10 November 2011; pp. 556–559.
24. Xuan, J.; Monperrus, M. Learning to Combine Multiple Ranking Metrics for Fault Localization. In Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 6 December 2014; pp. 191–200. https://doi.org/10.1109/ICSME.2014.41.
25. Zhang, X.Y.; Jiang, M. SPICA: A Methodology for Reviewing and Analysing Fault Localisation Techniques. In Proceedings of the 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), Luxembourg, 27 September–1 October 2021; pp. 366–377.
26. Xie, X.; Chen, T.Y.; Kuo, F.C.; Xu, B. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2013**, *22*, 1–40.
27. Xie, X.; Kuo, F.C.; Chen, T.Y.; Yoo, S.; Harman, M. Provably optimal and human-competitive results in sbse for spectrum based fault localisation. In *International Symposium on Search Based Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 224–238.
28. B. Le, T.D.; Lo, D.; Le Goues, C.; Grunske, L. A learning-to-rank based fault localization approach using likely invariants. In Proceedings of the 25th International Symposium on Software Testing and Analysis, Saarbrücken, Germany, 18–20 July 2016; pp. 177–188.
29. Ernst, M.D.; Cockrell, J.; Griswold, W.G.; Notkin, D. Dynamically discovering likely program invariants to support program evolution. *IEEE Trans. Softw. Eng.* **2001**, *27*, 99–123.
30. Just, R.; Jalali, D.; Ernst, M.D. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In Proceedings of the 2014 International Symposium on Software Testing and Analysis, San Jose, CA, USA, 21–25 July 2014; pp. 437–440.
31. Sohn, J.; Yoo, S. Fluccs: Using code and change metrics to improve fault localization. In Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara, CA, USA, 10–14 July 2017; ACM: New York, NY, USA, 2017; pp. 273–283.
32. Kim, Y.; Mun, S.; Yoo, S.; Kim, M. Precise learn-to-rank fault localization using dynamic and static features of target programs. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2019**, *28*, 1–34.
33. Jones, J.A.; Harrold, M.J. Empirical Evaluation of the Tarantula Automatic Fault-Localization Technique. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering; Long Beach, CA, USA, 7–11 November 2005; Association for Computing Machinery: New York, NY, USA, 2005; ASE '05; p. 273–282. https://doi.org/10.1145/1101908.1101949.

34. Ochiai, A. Zoogeographic studies on the soleoid fishes found in Japan and its neighbouring regions. *Bull. Jpn. Soc. Sci. Fish.* **1957**, *22*, 526–530.
35. Choi, S.S.; Cha, S.H.; Tappert, C.C. A survey of binary similarity and distance measures. *J. Syst. Cybern. Informatics* **2010**, *8*, 43–48.
36. Van de Vijver, F.J.; Leung, K. *Methods and Data Analysis for Cross-Cultural Research*; Cambridge University Press: Cambridge, UK, 2021; Volume 116.
37. Golagha, M.; Pretschner, A.; Briand, L.C. Can we predict the quality of spectrum-based fault localization? In Proceedings of the 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Porto, Portugal, 24–28 October 2020; pp. 4–15.
38. Zheng, W.; Hu, D.; Wang, J. Fault localization analysis based on deep neural network. *Math. Probl. Eng.* **2016**, *2016*.
39. Lo, D.; Jiang, L.; Budi, A.; et al. Comprehensive evaluation of association measures for fault localization. In Proceedings of the 2010 IEEE International Conference on Software Maintenance, Timisoara, Romania, 12–18 September 2010; pp. 1–10.
40. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
41. Rothermel, G.; Elbaum, S.; Kinneer, A.; Do, H. Software-artifact infrastructure repository. 2006. Available online: http://sir.unl.edu/portal accessed on 10 December 2020).
42. Zhang, X.; Gupta, N.; Gupta, R. Locating faults through automated predicate switching. In Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, 20–28 May 2006; pp. 272–281.
43. Laghari, G.; Murgia, A.; Demeyer, S. Fine-Tuning Spectrum Based Fault Localisation with Frequent Method Item Sets. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, Singapore, 3–7 September 2019; Association for Computing Machinery: New York, NY, USA, 2016; ASE 2016; p. 274–285. https://doi.org/10.1145/2970276.2970308.
44. Pearson, S.; Campos, J.; Just, R.; Fraser, G.; Abreu, R.; Ernst, M.D.; Pang, D.; Keller, B. Evaluating and improving fault localization. In Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, 20–28 May 2017; pp. 609–620.
45. Just, R.; Parnin, C.; Drosos, I.; Ernst, M.D. Comparing developer-provided to user-provided tests for fault localization and automated program repair. In Proceedings of the ISSTA 2018, Proceedings of the 2018 International Symposium on Software Testing and Analysis, Amsterdam, The Netherlands, 16–21 July 2018; pp. 287–297.
46. Chen, Z.; Kommrusch, S.J.; Tufano, M.; Pouchet, L.; Poshyvanyk, D.; Monperrus, M. SEQUENCER: Sequence-to-Sequence Learning for End-to-End Program Repair. *IEEE Trans. Softw. Eng.* **2019**; *47*, 1943–1959. https://doi.org/10.1109/TSE.2019.2940179.
47. Zhang, Z.; Chan, W.K.; Tse, T.H.; Jiang, B.; Wang, X. Capturing propagation of infected program states. In Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Amsterdam, The Netherlands, 24–28 August 2009; pp. 43–52.
48. Debroy, V.; Wong, W.E.; Xu, X.; Choi, B. A grouping-based strategy to improve the effectiveness of fault localization techniques. In Proceedings of the 2010 10th International Conference on Quality Software, Zhangjiajie, China, 14–15 July 2010; pp. 13–22.
49. Keller, F.; Grunske, L.; Heiden, S.; Filieri, A.; van Hoorn, A.; Lo, D. A critical evaluation of spectrum-based fault localization techniques on a large-scale software system. In Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, Czech Republic, 25–29 July 2017; pp. 114–125.
50. de Souza, H.A.; Chaim, M.L.; Kon, F. Spectrum-based software fault localization: A survey of techniques, advances, and challenges. *arXiv* **2016**, arXiv:1607.04347.
51. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **2020**, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2.
52. Cliff, N. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychol. Bull.* **1993**, *114*, 494.
53. Romano, J.; Kromrey, J.D.; Coraggio, J.; Skowronek, J.; Devine, L. Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen'sd indices the most appropriate choices. In Proceedings of the Annual Meeting of the Southern Association for Institutional Research, Arlington, TX, USA, 14-17 October 2006; pp. 1–51.
54. Zhang, L.; Yan, L.; Zhang, Z.; Zhang, J.; Chan, W.; Zheng, Z. A theoretical analysis on cloning the failed test cases to improve spectrum-based fault localization. *J. Syst. Softw.* **2017**, *129*, 35–57.
55. Ribeiro, H.L.; de Araujo, P.R.; Chaim, M.L.; de Souza, H.A.; Kon, F. Evaluating data-flow coverage in spectrum-based fault localization. In Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Brazil, 19–20 September 2019; pp. 1–11.
56. Vancsics, B.; Szatmári, A.; Beszédes, Á. Relationship between the effectiveness of spectrum-based fault localization and bug-fix types in javascript programs. In Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), London, ON, Canada, 18–21 February 2020; pp. 308–319.