The Journal of Systems and Software 129 (2017) 35–57

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A theoretical analysis on cloning the failed test cases to improve spectrum-based fault localization☆

Long Zhang[a,b], Lanfei Yan[a,b], Zhenyu Zhang[a,c,*], Jian Zhang[a,b], W.K. Chan[d,a], Zheng Zheng[e]

[a] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[b] University of Chinese Academy of Sciences, Beijing, China
[c] Institute for Software Research, University of California, Irvine, California, USA
[d] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Hong Kong
[e] School of Automation Science and Electrical Engineering, Beihang University, Beijing, China

ARTICLE INFO

Article history:
Received 25 September 2015
Revised 23 March 2017
Accepted 22 April 2017
Available online 24 April 2017

Keywords:
Software debugging
Fault localization
Class imbalance
Test suite cloning

ABSTRACT

Fault localization is the activity to locate faults in programs. Spectrum-based fault localization (SBFL) is a class of techniques for it. It contrasts the code coverage achieved by passed runs and that by failed runs, and estimates program entities responsible for the latter. Although previous work has empirically shown that the effectiveness of typical SBFL techniques can be improved by incorporating more failed runs, debugging often takes place when there are very few of them.

In this paper, we report a comprehensive study to investigate the impact of cloning the failed test cases on the effectiveness of SBFL techniques. We include 33 popular such techniques, and examine the accuracy of their formulas on twelve benchmark programs, using four accuracy metrics and in three scenarios. The empirical results show that on 22, 21, and 23 of them the fault-localization accuracy can be significantly improved, when the failed test cases are cloned in the single-fault, double-fault, and triple-fault scenarios, respectively. We also analytically show that on 19 of them the improvements are provable for an arbitrary program and an arbitrary test suite, in the single-fault scenario; and moreover, for ten of the rest formulas, their accuracy are proved unaffected in all scenarios.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

Program debugging involves fault localization and repair. It is time-consuming and tedious, and represents major bottlenecks in typical software development projects. To reduce the cost of program debugging, many techniques have been proposed in the literature to (semi-)automate various debugging activities. In particular, there is a large body of research on fault localization. Spectrum-based fault localization (SBFL) is one of them under active study in the recent decades (Jones et al., 2002; Liblit et al., 2015; Zhang et al., 2005; Abreu et al., 2006; 2007; Wong et al., 2007; 2008b; Zhang et al., 2009; Naish et al., 2011; Zhang et al., 2011; Wong et al., 2012; Xie et al., 2013b, 2013a; Xu et al., 2013; Wong et al., 2014).

A program spectrum is the dynamic program behavior extracted from the execution profile of a program run. It can be collected for program entities of selected granularities, such as program statements and branches (Reps et al., 1997; Harrold et al., 1998; Xie et al., 2013a). SBFL techniques accept the program spectra and the test outcomes of a set of test cases as inputs, and recommend suspicious regions in the faulty program that are likely to contain faults responsible for the failed runs of the test cases. A typical SBFL technique computes a suspiciousness score for each program entity in the program using a *risk evaluation formula* (or *formula* for short). The greater the suspiciousness score, the more the associated program entity is assessed to be fault-relevant by that technique. Given a particular suspiciousness score threshold reflecting the debugging efforts a program can afford, the SBFL technique can then output a set of program entities, for each of which the associated suspiciousness score does not exceed the given threshold.

☆ This work was supported by a grant from the National Key Basic Research Program of China (project no. 2014CB340702), a grant from the National Natural Science Foundation of China (project no. 61379045), a grant from the China Scholarship Council (project no. 201604910232), an open project from the State Key Laboratory of Computer Science (project no. SYSKF1608), and grants from the General Research Fund of the Research Grants Council of Hong Kong (project nos.11200015 and 11201114).

* Corresponding author.
E-mail addresses: zlong@ios.ac.cn (L. Zhang), yanlf@ios.ac.cn (L. Yan), zhangzy@ios.ac.cn (Z. Zhang), zj@ios.ac.cn (J. Zhang), wkchan@cityu.edu.hk (W.K. Chan), zhengz@buaa.edu.cn (Z. Zheng).

http://dx.doi.org/10.1016/j.jss.2017.04.017
0164-1212/© 2017 Elsevier Inc. All rights reserved.

The *effectiveness* of a SBFL technique is thus measured as whether the output set of program entities includes the fault(s). For ease of presentation, we refer to the *accuracy* of a formula embedded in a SBFL technique as the minimum proportion of such an output set of program entities to the total number of program entities in the program code.[1] Researchers have proposed many SBFL formulas, such as Jaccard (1901), Ochiai (1957), and Jones and Harrold (2005). Almost all previous works focus on reporting empirical results to compare the effectiveness of the selected SBFL techniques (Jones et al., 2002; Jones and Harrold, 2005; Liblit et al., 2015; Do et al., 2005; Liu et al., 2006; Abreu et al., 2006; 2007; Wong et al., 2007; Zhang et al., 2009; Wong et al., 2010; Gore and Reynolds, 2012). At the same time, some analytical findings among the accuracy of SBFL formulas (Lee et al., 2009; Naish et al., 2011; Xie et al., 2013a) have also been obtained recently. For instance, Lee et al. (2009) proved that the q_e formula and the *Tarantula* formula are equivalent in accuracy in some scenarios. In their follow-up work (Naish et al., 2011), they proposed two risk evaluation formulas, validated more groups of risk evaluation formulas which accuracy was empirically equivalent, and showed that two formulas consistently achieved accuracy higher than, if not equivalent to, some other risk evaluation formulas. Xie et al. (2013a) proposed an analytical framework to investigate risk evaluation formulas in a specific single-fault scenario. They proved that many risk evaluation formulas can be placed into six groups of formulas such that the formulas in each group share the same accuracy as or are always less accurate than the formulas in some other groups. They showed that two of the six groups contain formulas that no other formulas they analyzed can be more accurate than the former formulas.

Apart from the study on the relationships among formulas, there are also studies on investigating the relationships among test cases in a test suite. For instance, Gong et al. (2012) referred to the ratio of the number of passed test cases to the number of failed test cases as the *class balance ratio* of a test suite. A test suite having a class balance ratio of 1 is called a *class-balanced test suite*. They empirically showed that SBFL techniques tend to achieve higher accuracy if the class balance ratio is closer to 1 (i.e., more like a class-balanced test suite). On the other hand, in practice, fault localization may have to start when there is only a small number of failed runs (or failed test cases), even though many passed test runs have been available. Hence, even though using a class-balanced test suite tends to make a SBFL technique empirically more effective, in practice, this notion is difficult to be directly applied. In the preliminary version (Gao et al., 2013) of this paper, we acknowledged this limitation to prevent the research results to be transferred to the industry, and proposed to clone the available failed test cases in a given test suite to construct a class-balanced test suite. It also reported that such test suites constructed by cloning the failed test cases can theoretically improve the effectiveness of two particular SBFL techniques under specific conditions.

In this paper, we first revisit the notion of class imbalance problem in the context of spectrum-based fault localization. Specifically, some SBFL formulas may become less accurate if there is very few failed test cases and the class balance ratio is therefore far from 1. We propose to clone the whole set of failed test cases to enlarge it until it catches up with the set of passed test cases in size. We report a controlled experiment on the formulas of 33 SBFL techniques in both single-fault and multi-fault scenarios on seven small-scale and five medium-scale program benchmark programs. For each faulty version of each subject program, we run the test suite over it, and collect the spectrum information and test outcome (i.e., pass or fail). We use each pair of spectrum information and test results as an input to separately drive each of the 33 SBFL formulas, and evaluate their accuracy accordingly. Next, we repeat the above procedure but apply our cloning strategy to generate a new set of failed test cases to replace the original. We therefore can contrast the accuracy of each SBFL formula on each faulty program with each test suite before and after applying the cloning strategy, and empirically evaluate the impact of the cloning strategy on the accuracy of the studied SBFL formulas. On the other hand, a real-world program may contain more than one fault. To simulate the situation, we synthesize double-fault program versions by incorporating the faults in two randomly chosen faulty versions of the corresponding program. We also repeat a similar procedure to synthesize triple-fault program versions.

The empirical results show that 22, 21, and 23 (out of the 33) SBFL formulas exhibit improvements, in terms of average fault-localization accuracy, after cloning the failed test cases, in the single-fault, double-fault, and triple-fault scenarios, respectively. Furthermore, the observed improvements for each SBFL formula are determined statistically significant by hypothesis testing. To further evaluate the cloning strategy, we incorporate four different accuracy metrics in the experiment and always obtain similar empirical results and consistent observations. At the same time, there are ten SBFL formulas, which fault-localization accuracy is never observed changed in either scenario.

In this paper, we also show through mathematical analysis that the identified non-negative changes in fault localization accuracy are not by chances. We formally classify the impact of cloning the set of failed test cases on the accuracy of a SBFL formula into four types, denoted by Preserved, Improved, Deteriorated, and Non-deterministic. We summarize our findings into four theorems and four corollaries, which are further applied to label each of the 33 SBFL formulas using one of these four types. The analytical study confirms that 19 of the 33 studied formulas are more effective when the failed test cases in a given test suite are cloned in the single-fault scenario. Moreover, the ten formulas, which accuracy is observed unchanged in the experiment, are proved to be unaffected in all scenarios.

This paper significantly extends its preliminary version (Gao et al., 2013) in two main aspects: First, we report a comprehensive and new experiment in both the single-fault scenario and multi-fault scenarios to investigate the use of the above-mentioned cloning strategy on 33 SBFL formulas. It also uses multiple effectiveness metrics to validate the results empirically. Second, we analytically investigate the improvements achieved with the 33 risk evaluation formulas, which reconcile the observations obtained in the empirical evaluation. The scale of the empirical study is one order of magnitude larger than that presented in the preliminary version.

The main contribution of this work is twofold. (i) In both the single-fault and the multi-fault scenarios, the effects of the cloning strategy on improving the formulas of the 33 widely-studied SBFL techniques are reported. (ii) Both empirical results on the extent of the improvements and theoretical analysis to confirm the improvements are reported.

The paper is organized as follows. Section 2 formulates the class-imbalanced problem for spectrum-based fault localization and presents our proposal to address it. It also raises the research questions that we aim to investigate empirically. Section 3 reports a controlled experiment for evaluating the proposal, and summarizes our findings and observations. Section 4 presents our theoretical framework and summarizes the analytical results on confirming the empirical observations on different risk evaluation formulas. Section 5 summarizes the findings, answers the research questions, and discusses threats to the validity of the conclusions.

[1] In the controlled experiment, we incorporate different variants of this accuracy definition to generalize the results.

Section 6 reviews the closely related work. Section 7 concludes the work. Appendix A and B contains the proofs.

2. Research problem

In this section, we revisit the class-imbalance problem and our cloning strategy, and present the research questions.

2.1. Preliminaries

In conducting a typical software test, after the execution, each test case in a given test suite of a program can be classified into one of the two subsets: the set of failed test cases and the set of passed test cases, according to whether the execution of the program over the test case reveals unexpected output or not. The effectiveness of spectrum-based fault localization techniques is influenced by the characteristics of test suites. One aspect of study on such influences is to measure the extent of class balance between the number P of passed test cases and number F of failed test cases (Gong et al., 2012).

2.1.1. The class-imbalance phenomenon

Regarding the class-imbalance phenomenon (Gong et al., 2012) existing in spectrum-based fault localization, we first give the following notations for ease of the reference.

Notation 2.1 (P & F)**.** The notations P and F denote the number of the passed test cases and the number of the failed test cases, respectively, in a given test suite.

We refer to the ratio (P: F) as the *class balance ratio* (Gong et al., 2012). If the class balance ratio of a test suite is not close to 1, the test suite is said to be *class-imbalanced*. It is well-known that the class balance ratios of many practical test suites of real-world programs are not close to 1. In fact, in most cases, the number of failed test cases is often much smaller than the number of passed test cases in the same test suite. Our previous work (Gong et al., 2012) reported that if the class balance ratio was much larger than 1, many SBFL techniques such as Jaccard, Hamann, Wong2, Euclid, Ochiai and Wong3 produced rather low accuracy, which indicated that the fault-localization effectiveness of SBFL techniques when using such a test suite was undesirable.

In general, to handle such a problem, there are at least three basic strategies, which aim to add, remove, or modify test cases, respectively. (i) To achieve a class-balanced test suite, a baseline deletion-based strategy is to remove some passed test cases from the given test suite if the class-balanced ratio is greater than 1 or to remove some failed test cases from a given test suite if the class-balanced ratio is lower than 1. Nonetheless, this strategy will likely produce a reduced test suite containing very few test cases in some cases. This is because in practice, the class-balanced ratio of a given test suite is much larger than 1. For instance, if a given class-imbalanced test suite contains M (e.g., 100) test cases, and only m (e.g., 3) of them are failed test cases, then the resultant class-balanced test suite will only contain $2 \cdot m$ (e.g., 6) test cases. Although the resultant test suite is class-balanced, and is likely to make a SBFL technique more effective than a class-imbalanced test suite of the same suite size (Gong et al., 2012), the resultant accuracy is likely significantly lower than the one using the original test suite due to the disadvantage of the test suite size. (ii) A baseline modification-based strategy is to revise some passed test cases (e.g., changing the input values or strengthening the oracle check encoded in these changed test cases) into some other test cases. However, whether these changed test cases will expose program failures is uncertain, which requires further research so that effective modifications can be resulted in a class-balanced test suite. A way to make the above modification-based strategy feasible is to revise (i.e., re-code) some passed test cases into existing failed test cases to form a class-balanced test suite. Nonetheless, this strategy will reduce the diversity of the program spectra achieved by the original subset of passed test cases in the given test suite, which makes the process of contrasting the program spectra of the remaining passed test cases against those of the failed test cases less effective due to the loss of factual data points. (iii) In the next subsection, we are going to present our addition-based strategy.

2.1.2. Cloning: our proposed solution

A conservative strategy is to treat all the test information as useful and do not reduce test information. We (Gao et al., 2013) argued that a good direction to mitigate the class-imbalance problem stated in the last subsection is to construct a class-balanced test suite from a given class-imbalanced one.

An addition-based strategy is to add test cases to the given test suite. Adding arbitrary test cases to produce a class-balanced test suite is beyond the scope of this work. In this paper, we strategically choose to clone the failed test cases. One may wonders which particular failed test cases to be cloned (or cloned more), and yet the question remains to be answered until sufficient empirical findings are accumulated. Toward understanding the influence of addition-based strategy, we (Gao et al., 2013) have proposed to clone the entire given set of failed test cases for a certain number of times until the size of the enlarged set of failed test cases caught up with the size of the original set of passed test cases, which is formally presented below.[2]

Notation 2.2 (c)**.** $c = \lfloor \frac{P}{F} \rfloor$ is the number of times each failed test case is cloned.[3]

Gao et al. (2013) have already studied the above cloning strategy on the risk evaluation formulas of two SBFL formulas (Abreu et al., 2007) and Wong2 (Wong et al., 2007) in the single-fault scenario through a postmortem analysis. Specifically, for each program entity (e.g., a statement) i in a single-fault program, we computed its suspiciousness scores using each of the two risk evaluation formulas with each applicable test suite. For ease of reference, we denote their suspiciousness scores by $Jaccard(i)$ and $Wong2(i)$, respectively. Since we conducted a postmodern analysis, we knew the location of the fault in the faulty program. We therefore marked the faulty program entity f, and computed the difference $\mathcal{S}usp_{\Delta} = Jaccard(i) - Jaccard(f)$ with respect to the program entity i and the faulty program entity f. In the preliminary version (Gao et al., 2013) of this paper, based on these differences, we constructed three sets of program entities containing every program entity i that $\mathcal{S}usp_{\Delta} > 0$, $\mathcal{S}usp_{\Delta} = 0$, and $\mathcal{S}usp_{\Delta} = 0$, respectively.

Regarding the proposed cloning strategy to address the problem of class imbalance, we further introduce the following notations to simplify the presentation of this paper.

Notation 2.3 (P' & F')**.** P' $(= P)$ and F' $(= c \cdot F)$ denote the number of the passed test cases and the number of the failed test cases, respectively, in a given test suite, **after cloning**.

With P' passed test cases and F' failed test cases in the enlarged test suite, Gao et al. (2013) repeated the above procedure for each program entity i to compute its revised suspiciousness score using each risk evaluation formula. It classified the program entity i into three sets to determine whether each corresponding risk evaluation formula becomes more accurate or not, after the set of failed

[2] The cloning strategy may have variants. For example, if the numbers of test cases P and F are relatively prime, the best strategy is to clone both the set of passed test cases and the set of failed test cases to their least common multiple, to reach the optimal "balance".

[3] Note that we always assume $c > 0$ in this paper.

test cases is cloned using the above cloning strategy. Gong et al. (2012) have also empirically shown that both Jaccard and Wong2 became more accurate with *more balanced* test suites.

2.2. Problem formulation

In this section, we give the problem settings to formulate the proposed cloning solution, together with an accuracy metric to evaluate it.

2.2.1. The fault localization problem settings

Like many existing studies (Xie et al., 2013b, 2013a), in our model, there is a program, a test suite, and a matrix containing the coverage spectrum achieved by each program run of each test case. We adopt the following notations.

Notation 2.4 ($\mathbf{a}_{ep}$, $\mathbf{a}_{np}$, $\mathbf{a}_{ef}$, $\mathbf{a}_{nf}$)**.** The notations $\mathbf{a}_{ep}$ and $\mathbf{a}_{np}$ represent the numbers of passed test cases exercising and not exercising a program entity, respectively. The notations $\mathbf{a}_{ef}$ and $\mathbf{a}_{nf}$ represent the numbers of failed test cases exercising and not exercising a program entity, respectively.

Following the prior studies (Xie et al., 2013b, 2013a), in our model, we evaluate risk evaluation formulas in terms of accuracy. The risk evaluation formula of a typical SBFL technique is defined as a function to map a set of parameters to a real number.

Definition 2.1 (risk evaluation formula & suspiciousness score: $\mathcal{S}usp$)**.** A risk evaluation formula (or **formula**) is the function of a SBFL technique to map the parameters $\mathbf{a}_{ef}$, $\mathbf{a}_{ep}$, $\mathbf{a}_{nf}$, and $\mathbf{a}_{np}$ to a real number, for each program entity in a program. A real number thus mapped is named as a suspiciousness score, expressed as $\mathcal{S}usp$, of the program entity.

Since applying the cloning strategy will alter the test suite, the spectra arguments after the cloning has been applied are given using the following notations.

Notation 2.5 ($\mathbf{a}'_{ep}$, $\mathbf{a}'_{np}$, $\mathbf{a}'_{ef}$, $\mathbf{a}'_{nf}$)**.** We denote the corresponding values of $\mathbf{a}_{ep}$, $\mathbf{a}_{np}$, $\mathbf{a}_{ef}$, $\mathbf{a}_{nf}$, after the cloning strategy has been applied, by $\mathbf{a}'_{ep}$, $\mathbf{a}'_{np}$, $\mathbf{a}'_{ef}$, $\mathbf{a}'_{nf}$, respectively. That is, $\mathbf{a}'_{ep}$ and $\mathbf{a}'_{np}$ denote the numbers of passed test cases in the test suite after cloning, exercising and not exercising the program entity in question, respectively. The notations $\mathbf{a}'_{ef}$ and $\mathbf{a}'_{nf}$ denote the numbers of failed test cases in the test suite after cloning, exercising and not exercising the program entity in question, respectively.

According to the strategy to clone the set of failed test cases, which has been presented above, it is easy to know that $\mathbf{a}'_{ep} = \mathbf{a}_{ep}$, $\mathbf{a}'_{np} = \mathbf{a}_{np}$, $\mathbf{a}'_{ef} = c \cdot \mathbf{a}_{ef}$, and $\mathbf{a}'_{nf} = c \cdot \mathbf{a}_{nf}$.

2.2.2. Measuring the effectiveness of fault localization boosting

A SBFL formula assigns a suspiciousness score to each program entity. Suppose that a risk evaluation formula has assigned each program entity (in a faulty program) a suspiciousness score. A simple way to reference the fault localization result is to arrange all the program entities into a sequence in the descending order of their suspiciousness scores. As a result, the position of the faulty program entity in the ranked list reflects the fault-localization accuracy.

Definition 2.2 (The Avg expense (Wong et al., 2007))**.** The expense to locate a faulty program entity f is measured as the effort a developer spends to check along the ranked list before reaching the fault. It is calculated as the ratio of the rank of f in the ranked list to the length of the ranked list. When f shares identical suspiciousness score with some other program entities, thus forming a tie in the list, it will be checked as the middle one in the tie. Such an expense is referred to as Avg expense in previous work (Li et al., 2014).

We are interested in studying the change in expense by applying the cloning strategy. Thus, we further define the notion of *increase* (see below). It aims to measure the difference in expense before and after the cloning strategy has been applied to a given test suite with respect to the same SBFL formula.

Definition 2.3 (increase & increase ratio)**.** The increase is the expense achieved by a SBFL formula over a given test suite subtracting the expense achieved by the same SBFL formula over a test suite generated by applying the cloning strategy on the original test suite. The increase ratio is defined as the percentage of increase with respect to the expense achieved by the same SBFL formula over the given test suite.

A positive increase ratio and a negative one indicate the accuracy of a SBFL formula improved and deteriorated by cloning, respectively. A zero-valued increase ratio indicates that the accuracy of a SBFL formula is preserved (i.e., not affected) by the cloning strategy. The increase ratio may be different when adopting different expense metrics. Nonetheless, the essence is that the higher the ratio, the more a risk evaluation formula is improved by the cloning strategy.

2.3. Expectations and research questions

In this section, we first make expectations on the effectiveness of the proposed cloning solution in the general fault localization problem setting. After that, we present the research questions to be studied.

2.3.1. Potential benefits

We recall that the proposed solution is to clone the entire failed set without distinguishing individual test cases. One benefit is that the involved calculation can be simplified by manipulating the formulas rather than including concrete additional failed test cases to run the program. Therefore, the computational complexity of adopting our solution is low. Moreover, the strategy can be analytically studied since there is only a change in weighting the arguments of the formulas. In another word, the increased computational cost is marginal.

Many spectrum-based fault localization techniques are inspired by the fact that the exercising of faulty statements is responsible for the appearance of most program failures. As a result, many of them follow the principle to design the risk evaluation formulas whose $\mathcal{S}usp$ value increases with the argument $\mathbf{a}_{ef}$ or decreases with the argument $\mathbf{a}_{ep}$. In such a way, statements exercised more often in failed runs are intentionally marked as more suspicious by giving higher suspiciousness scores, and statements exercised more often in passed runs are marked as safer by giving lower suspiciousness scores.

Let us further consider an ideal case for a faulty statement f and an ordinary statement i. Generally, exercising a faulty statement f can result in some kind of error and may lead the test run to fail. As a result, in practice, most of the observed passed runs are the consequences of skipping the exercise on f, which is expressed by $\frac{\mathbf{a}^f_{ep}}{P} \approx 0$. On the other hand, most of the observed failed runs are the consequences of exercising f, which is expressed by $\frac{\mathbf{a}^f_{ef}}{F} \approx 1$. On the contrary, since exercising an arbitrary program entity i has no strong relationship with whether or not a test run will fail, we conjecture that in general, there is no significant difference between the probability of i being exercised in passed runs and that in failed runs, which is expressed by $\frac{\mathbf{a}^i_{ef}}{F} \approx \frac{\mathbf{a}^i_{ep}}{P} \in (0, 1)$. Finally,

we have $1 \approx \frac{\mathbf{a}_{ef}^{f}}{F} > \frac{\mathbf{a}_{ef}^{i}}{F} \approx \frac{\mathbf{a}_{ep}^{i}}{P} > \frac{\mathbf{a}_{ep}^{f}}{P} \approx 0$ most of the time. In such a case, cloning the set of failed test cases not only increases $\mathbf{a}_{ef}^{f}$ and $\mathbf{a}_{ef}^{i}$, but also enlarges the gap between $\mathbf{a}_{ef}^{f}$ and $\mathbf{a}_{ef}^{i}$ for c times. As a result, f is ranked higher according to the monotonic increasing nature of the $\mathcal{S}usp$ formulas.

Although existing empirical studies have successfully shown that f can be effectively located using the heuristics, analytical study can be complicated for coincidental correctness cases, where exercising f may have some probability in producing a passed run. Second, no universal or general conclusion can be made on whether a boosting method definitely works for all fault localization formulas. For example, a boosting method improving the formula $\mathbf{a}_{ef} - \mathbf{a}_{ep}$ must deteriorate the formula $\mathbf{a}_{ep} - \mathbf{a}_{ef}$ at the same time. It should be noted that such pairs of opposite formulas can be easily constructed. Third, there are fault localization formulas that are theoretically unaffected by a cloning strategy (e.g., the formula $\frac{\mathbf{a}_{ef}}{F} - \frac{\mathbf{a}_{ep}}{P}$ of technique AMPLE2 (Xie et al., 2013a)). To further understand the accuracy change made by the proposed cloning strategy on SBFL formulas, we propose to study a set of research questions, which are presented in the next subsection.

2.3.2. Research questions

In this study, we include 33 SBFL formulas, which are adopted from existing work (Naish et al., 2011; Xie et al., 2013a). To the best of our knowledge, Naish et al. (2011) and Xie et al. (2013a) conducted the two most comprehensive theoretical investigations on SBFL risk evaluation formulas in the literature. For presentation completeness, the 33 risk evaluation formulas of these studied techniques are listed in Table 1. The groups of these formulas, if any, are adopted from Xie et al. (2013a).

The experiments in prior studies (Debroy and Wong, 2009; Naish et al., 2011; Gong et al., 2012; Naish et al., 2015) sometimes classified faulty programs based on the number of known faults existing in these programs. If there is only one known fault, we name it a *single-fault* scenario. If there are more than one known fault, we name it a *multi-fault* scenario. We also follow this classification of scenarios in this paper.[4]

It is worth noting that although many SBFL techniques are developed in the background of single-fault scenario, these techniques have been empirically shown applicable to locate faults in various multi-fault scenarios. On the other hand, theoretical analyses (Naish et al., 2011; Xie et al., 2013a) are developed in the single-fault scenario, but their analysis results have not been shown generalized to multi-fault scenarios. Gong et al. (2012) showed that in single-fault scenario, six risk evaluation formulas can exhibit higher accuracy by using a class-balanced test suite. Gao et al. (2013) successfully proved that the changes in accuracy of two risk evaluation formulas Jaccard and Wong2 by using the cloning strategy are always non-negative (i.e., with a non-negative increase ratio). Thus, we wonder whether the results obtained so far can be further generalized beyond those two SBFL formulas. Moreover, if this is the case, we would also like to know the extent of improvement that may be observed empirically. Therefore, in this study, we formulate the following research questions.

Q1. In which scenarios and for which risk evaluation formulas, will the cloning strategy improve their fault-localization accuracy? Are the improvements statistically significant?
Q2. To what extent and in what expense metrics can an improvement in accuracy be achieved by each formula? Moreover, is the improvement applicable to the single-fault scenario only, the multi-fault scenario only, or both?
Q3. Are the improvements achieved by the same formula consistent across all possible programs, all possible faults, and all possible test suites?

Research question Q1 aims to study whether the cloning strategy is an plausible approach to improving the accuracy of risk evaluation formulas. However, knowing whether there is an improvement in accuracy is not enough. Research question Q2 further studies the extent of improvement in terms of expense, observed with the studied formulas. We are going to answer these two research questions empirically through a controlled experiment. If the answer to Q2 provides strong evidences that there are consistent improvements with some formulas, we will be further interested in knowing whether the cloning strategy to improve the accuracy of these formulas can be confirmed analytically, which further consolidate the empirical findings from Q2 toward a theory of SBFL-based debugging. We will present an analytical framework in Section 4 and systematically show the results on each formula under the assumption that "every program execution of the program over any failed test case always exercises one or more faulty program entities", to answer Q3.

3. Controlled experiment

In this section, we introduce the experiment used to validate our proposal.

3.1. Experiment setup

We chose the Siemens suite (Software-artifact Infrastructure Repository, 2005), the program "space", and four UNIX utility programs ("flex", "gzip", "grep", and "sed") as the benchmarks in our experiment. We downloaded the benchmarks (including all versions and associated test suites) from the SIR repository (Software-artifact Infrastructure Repository, 2005). The descriptive statistics of these benchmarks are shown in Table 2. Take the first row of the table as an example. The subject "print_tokens" is equipped with five seeded faults. Each faulty version has 194–195 lines of code. The subject also came with a test suite containing 4130 test cases, 1.7% of which were failed ones. We deemed a version without enabling any equipped fault as a golden version of the subject. The data for the remaining benchmark shown in the table can be interpreted similarly.

There are originally 247 faulty versions in the program suite. We ran each faulty version of each subject over each applicable test case, and compared with the program run of the golden version of the same subject over the same test case to mark the test outcome, that is, "pass" or "fail". Following previous experiments (Zhang et al., 2009), we applied the whole test suite as inputs to individual program versions. If a faulty version was not associated with any failed run, like Zhang et al. (2009), we excluded that version from our data analysis.[5] Among the Siemens programs, we discarded the versions 4 and 6 of program "print_tokens" because the faults were located in the header files. Besides, we discarded the version 9 of program "schedule2", versions 6 and 7 of program "schedule", and versions 19, 27, and 32 of program "replace". It was because that there was no failed execution observed after running all the test cases of all these program versions. For the same reason, we also discarded versions 1, 2, 4, 6, 25, 26, 30, 32, 34, 35, 36, and 38 of program "space". Similarly, for the four UNIX programs "flex", "grep", "gzip" and "sed", we finally selected 32, 13,

[4] In both scenarios, we investigate the kind of faults, exercising which in a test run is the cause of the failures observed from that test run. In case of a code omission fault, we mark the closest adjacent program entity or directly affected program entity to be "faulty" to continue (Zhang et al., 2009).

[5] We also excluded those test cases that were associated with crash run, and applied the rest test cases as inputs to individual subject programs.

Table 1
Risk evaluation formulas of studied SBFL techniques.

Group	Formula		Formula	
ER1	Naish1 (Naish et al., 2011)	$\begin{cases} -1 & \text{if } \mathbf{a}_{ef} < f \\ \mathbf{a}_{np} & \text{if } \mathbf{a}_{ef} = F \end{cases}$	Naish2 (Naish et al., 2011)	$\mathbf{a}_{ef} - \frac{\mathbf{a}_{ep}}{P+1}$
ER2	Anderberg (Anderberg, 1988)	$\frac{\mathbf{a}_{ef}}{2F-\mathbf{a}_{ef}+2\mathbf{a}_{ep}}$	Dice (Dice, 1945)	$\frac{2\mathbf{a}_{ef}}{F+\mathbf{a}_{ep}}$
	Goodman (Goodman and Kruskal, 1954)	$\frac{3\mathbf{a}_{ef}-F-\mathbf{a}_{ep}}{\mathbf{a}_{ef}+F+\mathbf{a}_{ep}}$	Jaccard (Jaccard, 1901)	$\frac{\mathbf{a}_{ef}}{F+\mathbf{a}_{ep}}$
	Sørensen-Dice (Duarte et al., 1999)	$\frac{2\mathbf{a}_{ef}}{2F-\mathbf{a}_{nf}+\mathbf{a}_{ep}}$		
ER3	CBI inc. (Liblit et al., 2015)	$\frac{\mathbf{a}_{ef}}{\mathbf{a}_{ef}+\mathbf{a}_{ep}} - \frac{F}{F+P}$	q_e (Lee et al., 2009)	$\frac{\mathbf{a}_{ef}}{\mathbf{a}_{ef}+\mathbf{a}_{ep}}$
	Tarantula (Jones et al., 2002)	$\frac{\mathbf{a}_{ef}/F}{\mathbf{a}_{ef}/F+\mathbf{a}_{ep}/P}$		
ER4	Hamann (Hamming, 1950)	$\frac{2\mathbf{a}_{ef}+2\mathbf{a}_{np}}{F+P} - 1$	Euclid (Krause, 1973)	$\sqrt{\mathbf{a}_{ef}+\mathbf{a}_{np}}$
	Rogers & Tanimoto (Rogers and Tanimoto, 1960)	$\frac{\mathbf{a}_{ef}+\mathbf{a}_{np}}{\mathbf{a}_{nf}+\mathbf{a}_{ep}+F+P}$	Hamming etc. (Naish et al., 2011)	$\mathbf{a}_{ef}+\mathbf{a}_{np}$
	Simple Matching (Meyer et al., 2004)	$\frac{\mathbf{a}_{ef}+\mathbf{a}_{np}}{F+P}$	Wong2 (Wong et al., 2007)	$\mathbf{a}_{ef}-\mathbf{a}_{ep}$
	Sokal (Lourenco et al., 2004)	$\frac{2\mathbf{a}_{ef}+2\mathbf{a}_{np}}{\mathbf{a}_{ef}+\mathbf{a}_{np}+F+P}$		
ER5	Binary (Naish et al., 2011)	$\begin{cases} 0 & \text{if } \mathbf{a}_{ef} < f \\ 1 & \text{if } \mathbf{a}_{ef} = F \end{cases}$	Russel & Rao (Russel and Rao, 1940)	$\frac{\mathbf{a}_{ef}}{F+P}$
	Wong1 (Wong et al., 2007)	$\mathbf{a}_{ef}$		
ER6	Rogot1 (Rogot and Goldberg, 1966)	$\frac{\mathbf{a}_{ef}/2}{\mathbf{a}_{ef}+F+\mathbf{a}_{ep}} + \frac{\mathbf{a}_{np}/2}{\mathbf{a}_{nf}+\mathbf{a}_{np}+P}$		
	Scott (Scott, 1955)	$\frac{4\mathbf{a}_{ef}\mathbf{a}_{np}-2\mathbf{a}_{nf}\mathbf{a}_{ep}-\mathbf{a}_{nf}^2-\mathbf{a}_{ep}^2}{(\mathbf{a}_{ef}+F+\mathbf{a}_{ep})(\mathbf{a}_{np}+\mathbf{a}_{nf}+P)}$		
Non-grouped	AMPLE (Meyer et al., 2004)	$\left\lvert \frac{\mathbf{a}_{ef}}{F} - \frac{\mathbf{a}_{ep}}{P} \right\rvert$	AMPLE2 (Xie et al., 2013a)	$\frac{\mathbf{a}_{ef}}{F} - \frac{\mathbf{a}_{ep}}{P}$
	Arithmetic Mean (Rogot and Goldberg, 1966)	$\frac{2\mathbf{a}_{ef}\mathbf{a}_{np}-2\mathbf{a}_{nf}\mathbf{a}_{ep}}{(\mathbf{a}_{ef}+\mathbf{a}_{ep})(\mathbf{a}_{nf}+\mathbf{a}_{np})+FP}$	Kulcznski1 (Lourenco et al., 2004)	$\frac{\mathbf{a}_{ef}}{F-\mathbf{a}_{ef}+\mathbf{a}_{ep}}$
	Cohen (Cohen, 1960)	$\frac{2\mathbf{a}_{ef}\mathbf{a}_{np}-2\mathbf{a}_{nf}\mathbf{a}_{ep}}{(\mathbf{a}_{ef}+\mathbf{a}_{ep})P+(\mathbf{a}_{nf}+\mathbf{a}_{np})F}$	Kulcznski2 (Lourenco et al., 2004)	$\frac{\mathbf{a}_{ef}}{2F} + \frac{\mathbf{a}_{ef}}{2\mathbf{a}_{ef}+2\mathbf{a}_{ep}}$
	Fleiss (Fleiss, 1965)	$\frac{4\mathbf{a}_{ef}\mathbf{a}_{np}-2\mathbf{a}_{nf}\mathbf{a}_{ep}-\mathbf{a}_{nf}^2-\mathbf{a}_{ep}^2}{2F+2P}$	Ochiai (Ochiai, 1957)	$\frac{\mathbf{a}_{ef}}{\sqrt{F(\mathbf{a}_{ef}+\mathbf{a}_{ep})}}$
	M1 (Everitt, 1978)	$\frac{\mathbf{a}_{ef}+\mathbf{a}_{np}}{F-\mathbf{a}_{ef}+\mathbf{a}_{ep}}$	M2 (Everitt, 1978)	$\frac{\mathbf{a}_{ef}}{2F-\mathbf{a}_{ef}+P+\mathbf{a}_{ep}}$
	Wong3 (Wong et al., 2007)	$\mathbf{a}_{ef} - \begin{cases} \mathbf{a}_{ep} & \text{if } \mathbf{a}_{ep} \leq 2 \\ 2+0.1(\mathbf{a}_{ep}-2) & \text{if } 2 < \mathbf{a}_{ep} \leq 10 \\ 2.8+0.001(\mathbf{a}_{ep}-10) & \text{if } \mathbf{a}_{ep} > 10 \end{cases}$		

Table 2
Descriptive statistics of benchmarks.

Small-scaled	LoC*	# of faulty versions	Fault type	Test suite size	% of failed test cases
print_tokens	194–195	5	Seeded	4130	1.7
print_tokens2	196–200	9	Seeded	4115	5.4
replace	241–246	29	Seeded	5542	2.0
schedule	151–154	5	Seeded	2650	2.4
schedule2	128–130	9	Seeded	2710	3.2
tcas	63–67	41	Seeded	1608	2.4
tot_info	122–123	23	Seeded	1052	5.6
Medium-scaled	**LoC***	**# of faulty versions**	**Fault type**	**Test suite size**	**% of failed test cases**
flex	4002–4035	32	Seeded	567	38.9
grep	3198–3466	13	Seeded	809	11.7
gzip	1740–2041	13	Seeded	214	1.6
sed	1520–3733	22	Seeded	370	13.4
space	3651–3657	26	Real	13,645	14.8

*: lines of executable statements.

13 and 22 versions, respectively, which was in line with the previous experiments (Xie et al., 2013b). Finally, in total 227 faulty versions were used to simulate the single-fault scenario. We also constructed double-fault versions by pair-wisely combining the faults of each benchmark. If two faults could not coexist, we excluded the generation of the corresponding double-fault version from the experiment. Specifically, we randomly chose 50% of all the legitimate pair-wise fault combination for each benchmark, excluded those versions which were not associated with any failed run (as what we did in the single-fault scenario), and finally included a total of 1259 such faulty versions to simulate the double-fault scenario. By following the procedure as what we did to synthesize double-fault programs, we also synthesized and chose 2000 triple-fault program versions for the benchmarks. We then computed the suspiciousness scores of the program entities in each faulty version over each test suite via each risk evaluation formula shown in Table 1. Next, we measured the expense (see Definition 2.2) achieved by each risk evaluation formula (see Definition 2.1), and computed the increase (see Definition 2.3).

Our experiment was carried out on an Ubuntu 13.10 desktop system with a configuration of 8-core i7-870 (2.93G) CPU with 4GB physical memory. We used the tool *gcov* 4.6 to obtain the coverage information of the programs compiled with *gcc* 4.7.

3.2. Experiment results

In this section, we report the accuracy improvements by applying the cloning strategy, together with hypothesis testing to determine whether there are statistically significant differences.

Table 3 shows the increase ratio observed from the data for each risk evaluation formula. All the individual formulas or formula groups are listed in descending order of their accuracy improvements across all benchmarks. Take the formula Wong3 (the 8th row) and the benchmark "print_tokens" (the third column) as an example. The expense of Wong3 on the program print_tokens are 0.0095 before cloning and 0.0080 after cloning. That is, 0.95% of all the program entities has been checked before the fault is reached in the ranked list generated by Wong3. And, by using the test suite after cloning to drive Wong3, 0.80% of all the program entities need to be checked before the fault is reached in the ranked list. Accordingly, the code examination effort is reduced by applying cloning, and the increase ratio (i.e., Inc. Ratio in the table) is computed as $\frac{0.0095-0.0080}{0.0095} = 0.1559$ (or 15.59%). The results of increase ratios on the other benchmarks are similarly computed. By averaging all the 12 benchmarks, the mean increase ratio across all the programs is 0.1157 (or 11.57%). Other cells in the table can be interpreted similarly.

When two formulas manifested identical accuracy on each individual faulty version, we classified them into specific groups. In summary, we observe from the table that in the single-fault scenario, on average, the increase ratio observed on 31 risk evaluation formulas are non-negative, and those of Kulcznski2 and Arithmetic Mean are of mixed results after the cloning strategy has been applied. A close look at the experimental results reveals the following:

- R1. The average increase ratios of ER4, M1, ER6, Fleiss, ER2, Kulczynski1, Cohen, Wong3, Ochiai, and M2 are positive on all the benchmarks. In other words, the accuracy of these formulas have been improved by applying the cloning strategy.
- R2. The increase ratios of Arithmetic Mean are positive on all benchmarks except two ("schedule2" and "flex", which are negative). The overall average of Arithmetic Mean across all benchmarks is also positive. The increase ratio of Kulcznski2 is on average negative and also negative on four ("print_tokens2", "schedule2", "tcas", and "tot_info") of the 12 benchmarks. It indicates that the accuracy of these two formulas can only be partially improved in the single-fault scenario.
- R3. The average increase ratios of AMPLE, AMPLE2, ER1, ER3, and ER5 over different subject programs are always zero. It also means that on average, the accuracy of these formulas are preserved by the cloning strategy.

To verify whether the results observed can be statistically meaningful, we adopt the Wilcoxon signed rank test method. The Wilcoxon signed rank test is a nonparametric test for two populations when the observations are paired (MathWorld, 2010). A null hypothesis is set to stand for the fact that the difference between the two sets of pairs follows a symmetric distribution around zero. A p-value can be calculated for the Wilcoxon signed rank test. It is defined as the probability, under the assumption of the null hypothesis, of obtaining a result equal to or more extreme than what was actually observed. In our experiment, in case an improvement in terms of average accuracy is observed with a SBFL formula, we apply the Wilcoxon signed rank test to measure how much the observed on average improvement might come from two sample populations without significant differences. Here, the two populations are the SBFL formula scores before and after cloning. In general, the smaller the p-value, the less probably the null hypothesis holds. In other words, the smaller the p-value, the more confidence we have that there is a significant improvement by cloning the failed test runs.

Since Table 3 shows that the accuracy of most formulas is improved on most benchmarks in the top eleven rows (except Arithmetic Mean on schedule2 and flex), we want to know whether there is any statistically significant difference with the improvements. The last two columns of Table 3 present the results of the above hypothesis testing in the single-fault scenario. In each table, the p-value column shows the results of the Wilcoxon signed rank test in h-value of 1.0000 and at the significance level of 0.05. Note that for the cases where the increase ratios are on average zero (ER1, ER3, ER5, AMPLE2, AMPLE) or negative (Kulcznski2), we skipped to apply hypothesis testing to check the improvement achieved. The corresponding cells in the table is marked "imapplicable". From the remaining cases in the table, we observe that the null hypotheses are always rejected at the significance level of 0.05. More specifically, we observe that in the single-fault scenario, the fault-localization accuracy of 22 formulas have been improved when the clone strategy is applied.

Overall speaking, the results show that two in every three studied SBFL formulas can be positively affected by the cloning strategy to increase their fault-localization accuracy.

3.2.1. Impact of different scenarios

Definition 2.2 describes the expense metrics with respect to a fault. We therefore measured the expense achieved by each risk evaluation formula to locate each fault in the multi-fault scenario. In the multi-fault scenario, we also measured the expenses and the values of increase by following the procedure used in the single-fault scenario.

Table 4 presents the increase ratios of the 33 risk evaluation formulas on 1259 double-fault program versions used in our experiment to study the double-fault scenario. It also presents the increase ratios of the 33 risk evaluation formulas on 2000 faulty versions used in our experiment to study the triple-fault scenario. We note that when two formulas manifested identical accuracy on each individual faulty version, we classified them into specific groups. A formula in a group in Table 4 does not indicate whether the formula is in some particular group shown in Table 3.

Table 3
Effectiveness of cloning on different formulas over each benchmark.

Group/Formula	Expenses	Small-sized							Medium-sized					Avg. Inc. Ratio	p-value
		prnt_tkns	prnt_tkns2	replace	schedule	schedule2	tcas	tot_info	space	flex	sed	gzip	grep		
ER4	Before	0.0779	0.0734	0.0938	0.0515	0.2795	0.0918	0.1219	0.0311	0.2297	0.0487	0.0544	0.0980		
	After	0.0095	0.0416	0.0267	0.0109	0.2092	0.0507	0.0744	0.0192	0.2032	0.0319	0.0412	0.0846	0.4273	2.5×10^{-22}
	Inc. ratio	0.8779	0.4328	0.7156	0.7884	0.2514	0.4477	0.3900	0.3841	0.1155	0.1362	0.2427	0.3455		
M1	Before	0.0779	0.0734	0.0938	0.0515	0.2795	0.0918	0.1219	0.0311	0.2297	0.0487	0.0544	0.0980		
	After	0.0095	0.0416	0.0267	0.0109	0.2092	0.0507	0.0744	0.0192	0.2032	0.0319	0.0412	0.0846	0.4273	2.5×10^{-22}
	Inc. ratio	0.8779	0.4328	0.7156	0.7884	0.2514	0.4477	0.3900	0.3841	0.1155	0.1362	0.2427	0.3455		
ER6	Before	0.0353	0.0631	0.0565	0.0484	0.2770	0.0884	0.1362	0.0015	0.2297	0.0428	0.0544	0.0743		
	After	0.0095	0.0448	0.0316	0.0131	0.2285	0.0510	0.0823	0.0013	0.2010	0.0347	0.0479	0.0677	0.3203	2.7×10^{-19}
	Inc. ratio	0.7309	0.2899	0.4403	0.7299	0.1750	0.4232	0.3960	0.1356	0.1250	0.0892	0.1199	0.1882		
Fleiss	Before	0.0551	0.2459	0.1715	0.9083	0.9258	0.7751	0.7604	0.0014	0.2241	0.2168	0.0549	0.0800		
	After	0.0158	0.1777	0.0915	0.2946	0.7541	0.4447	0.4576	0.0013	0.1984	0.1901	0.0525	0.0693	0.3056	4.0×10^{-9}
	Inc. ratio	0.7137	0.2773	0.4665	0.6757	0.1855	0.4263	0.3982	0.1101	0.1146	0.1331	0.0428	0.1233		
ER2	Before	0.0165	0.0267	0.0285	0.0112	0.2181	0.0505	0.0562	0.0014	0.2200	0.0218	0.0913	0.0082		
	After	0.0095	0.0181	0.0219	0.0082	0.1814	0.0461	0.0405	0.0012	0.1968	0.0187	0.0818	0.0072	0.2000	1.7×10^{-14}
	Inc. ratio	0.4223	0.3207	0.2326	0.2723	0.1685	0.0878	0.2805	0.1473	0.1053	0.1187	0.1034	0.1407		
Kulcznski1	Before	0.0165	0.0267	0.0285	0.0112	0.2181	0.0505	0.0562	0.0014	0.2200	0.0218	0.0913	0.0082		
	After	0.0095	0.0181	0.0219	0.0082	0.1814	0.0461	0.0405	0.0012	0.1968	0.0187	0.0818	0.0072	0.2000	1.7×10^{-14}
	Inc. ratio	0.4223	0.3207	0.2326	0.2723	0.1685	0.0878	0.2805	0.1473	0.1053	0.1187	0.1034	0.1407		
Cohen	Before	0.0167	0.0490	0.0326	0.0138	0.2150	0.0509	0.0660	0.0014	0.2297	0.0229	0.0544	0.0082		
	After	0.0095	0.0419	0.0262	0.0109	0.1720	0.0496	0.0595	0.0014	0.2097	0.0214	0.0496	0.0080	0.1217	1.6×10^{-11}
	Inc. ratio	0.4296	0.1441	0.1964	0.2127	0.0200	0.0257	0.0988	0.0555	0.0873	0.0327	0.0889	0.0689		
Wong3	Before	0.0095	0.0069	0.0197	0.0109	0.1986	0.0556	0.0335	0.1607	0.2296	0.1871	0.0561	0.0569		
	After	0.0080	0.0064	0.0190	0.0088	0.1699	0.0498	0.0375	0.1576	0.2201	0.1524	0.0463	0.0539	0.1157	1.6×10^{-6}
	Inc. ratio	0.1559	0.0667	0.0373	0.1875	0.1445	0.1047	0.1203	0.0190	0.0415	0.0528	0.1755	0.1856		
Ochiai	Before	0.0108	0.0204	0.0238	0.0091	0.1860	0.0488	0.0461	0.0014	0.2200	0.0201	0.0913	0.0081		
	After	0.0095	0.0165	0.0210	0.0082	0.1791	0.0458	0.0391	0.0013	0.2181	0.0177	0.0827	0.0081	0.0936	1.6×10^{-11}
	Inc. ratio	0.1235	0.1945	0.1164	0.1045	0.0372	0.0605	0.1522	0.1106	0.0083	0.0041	0.0943	0.1168		
Arithmetic Mean	Before	0.0126	0.0459	0.0283	0.0089	0.1832	0.0511	0.0581	0.0014	0.2297	0.0214	0.0544	0.0081		
	After	0.0095	0.0400	0.0236	0.0082	0.2071	0.0491	0.0553	0.0014	0.2299	0.0202	0.0501	0.0081	0.0605	1.0×10^{-6}
	Inc. ratio	0.2477	0.1297	0.1683	0.0786	-0.1307	0.0399	0.0478	0.0070	−0.0007	0.0060	0.0789	0.0539		
M2	Before	0.0095	0.0122	0.0214	0.0082	0.1761	0.0459	0.0387	0.0014	0.2200	0.0173	0.0913	0.0079		
	After	0.0095	0.0098	0.0207	0.0082	0.1606	0.0452	0.0348	0.0014	0.2194	0.0171	0.0912	0.0079	0.0382	7.9×10^{-7}
	Inc. ratio	0.0000	0.1978	0.0305	0.0000	0.0884	0.0157	0.1010	0.0102	0.0026	0.0037	0.0006	0.0077		
AMPLE	Before	0												0	inapplicable
AMPLE2															
ER1	After						0							0	inapplicable
ER3	Inc. ratio	0												0	inapplicable
ER5															
	Before	0.0095	0.0096	0.0210	0.0082	0.1591	0.0442	0.0233	0.0014	0.2200	0.0174	0.0913	0.0081		
Kulcznski2	After	0.0095	0.0140	0.0209	0.0082	0.1656	0.0457	0.0273	0.0012	0.2145	0.0174	0.0833	0.0081	−0.0403	
	Inc. ratio	0.0000	−0.4561	0.0059	0.0000	−0.0405	−0.033	−0.1738	0.0954	0.0248	0.0041	0.0876	0.0020		

We have the following observations on the results summarized in the double- and triple-fault scenarios.[6]

R4. The average increase ratios of ER4, M1, Fleiss, ER6, Cohen, ER2, Kulcznski1, Cohen, Ochiai, Arithmetic Mean, and Kulcznski2 are positive in both the double-fault and the triple-fault scenarios. In other words, on average, the accuracy of these formulas can be improved by applying the cloning strategy.

R5. The average increase ratios of Wong3 and M2 are negative in the double-fault scenario. In other words, their accuracy cannot be improved by cloning, in the double-fault scenario.

R6. The increase ratios of AMPLE, AMPLE2, ER1, ER3, and ER5 are always zero. It means that on average, the accuracy of these formulas are preserved by the cloning strategy.

[6] What are not shown in this table include (a) the increase ratios of Ochiai on program "space" and Arithmetic Mean on program "schedule2" are negative, in both the double-fault and triple-fault scenarios, (b) the increase ratios of Kulcznski2 are negative on programs "replace", "schedule", "schedule2", "tcas", and "tot_info", in the double-fault scenario, (c) the increase ratio of Wong3 is only negative on programs "schedule2", "tot_info", and "grep", in the double-fault scenario, and (d) the increase ratio of M2 is only negative on programs "print_tokens", "schedule2", "tcas", and "tot_info", in the double-fault scenario.

3.2.2. Impact of different expense metrics

In this section, we also include three other metrics to evaluate the effectiveness achieved by the cloning strategy. In Definition 2.2, a tie containing the faulty program entity f in the ranked list is broken in such a way that program entities are checked in order and f is checked as the middle one of the tie. It is also named the *Avg* metric in previous studies (Li et al., 2014). In this section, we let f be checked as the last one in the tie containing it, and follow existing work (Wong et al., 2007) to name such a metric *Max*. Besides Avg and Max, a *Min* expense, which means checking f as the first element in the tie containing it, has also been used in previous studies (Wong et al., 2007). These three metrics differ in how the program entities in a tie will be checked (Wong et al., 2007; Li et al., 2014). However, in this study, we did not include the Min expense. It is because that solely referencing Min may result in an over-estimation of fault localization accuracy.[7]

Table 3 shows that the cloning strategy can improve the accuracy of 22 studied formulas using Avg as the metric. But in practice, developers may only care about the top portion of the fault localization results. For example, if the program has a few

[7] For example, let us consider a fictitious SBFL formula that cannot distinguish any program entities and assigns an identical suspicious score to each of them. For such a formula, the faulty program entity is always given the highest rank by the Min expense metric, resulting in a very high fault-localization accuracy.

Table 4
Effectiveness of cloning in different scenarios.

Group/Formula	Single-fault		Double-fault		Triple-fault	
	Avg. inc. ratio	p-value	Avg. inc. ratio	p-value	Avg. inc. ratio	p-value
ER4	0.4273	2.5×10^{-22}	0.4224	4.7×10^{-16}	0.4402	3.6×10^{-12}
M1	0.4273	2.5×10^{-22}	0.4224	4.7×10^{-16}	0.4402	3.6×10^{-12}
ER6	0.3203	2.7×10^{-19}	0.2321	5.0×10^{-6}	0.2958	6.0×10^{-7}
Fleiss	0.3056	4.0×10^{-9}	0.3217	3.8×10^{-5}	0.3472	4.0×10^{-6}
ER2	0.2000	1.7×10^{-14}	0.1829	3.3×10^{-11}	0.1991	1.6×10^{-9}
Kulcznski1	0.2000	1.7×10^{-14}	0.1829	3.3×10^{-11}	0.1991	1.6×10^{-9}
Cohen	0.1217	1.6×10^{-11}	0.2020	1.8×10^{-5}	0.1489	4.7×10^{-4}
Ochiai	0.0936	1.6×10^{-11}	0.0909	3.7×10^{-7}	0.1432	4.3×10^{-6}
Arithmetic Mean	0.0605	1.0×10^{-6}	0.0802	9.6×10^{-5}	0.0815	1.8×10^{-7}
AMPLE AMPLE2 ER1 ER3 ER5	0	Inapplicable	0	Inapplicable	0	Inapplicable
Wong3	0.1157	1.6×10^{-6}	−0.0622	Inapplicable	0.2025	2.5×10^{-5}
M2	0.0382	7.9×10^{-7}	−1.2200	Inapplicable	0.0124	3.6×10^{-6}
Kulcznski2	−0.0403	Inapplicable	0.0177	1.7×10^{-4}	0.0694	2.0×10^{-4}

Table 5
Effectiveness of cloning in different expense metrics.

Group/Formula	Avg (Definition 2.2)		Max		Top-5		Top-5‰	
	Avg Inc. Ratio	p-value	Avg Inc. Ratio	p-value	Avg Inc. Ratio	p-value	Avg Inc. Ratio	p-value
ER4	0.4273	2.5×10^{-22}	0.4020	1.8×10^{-19}	1.2581	8.6×10^{-10}	0.3878	6.7×10^{-7}
M1	0.4273	2.5×10^{-22}	0.4020	1.8×10^{-19}	1.2581	8.6×10^{-10}	0.3878	6.7×10^{-7}
ER6	0.3203	2.7×10^{-19}	0.3092	7.8×10^{-17}	0.5556	6.5×10^{-5}	0.0984	5.3×10^{-4}
Fleiss	0.3056	4.0×10^{-9}	0.2844	7.3×10^{-7}	0.4894	2.8×10^{-4}	0.0308	2.1×10^{-4}
ER2	0.2000	1.7×10^{-14}	0.1819	3.5×10^{-10}	0.1667	4.7×10^{-8}	0.2143	3.3×10^{-6}
Kulcznski1	0.2000	1.7×10^{-14}	0.1819	3.5×10^{-10}	0.1667	4.7×10^{-8}	0.2143	3.3×10^{-6}
Cohen	0.1217	1.6×10^{-11}	0.1115	7.9×10^{-9}	0.0606	7.4×10^{-5}	0.0806	1.6×10^{-4}
Wong3	0.1157	1.6×10^{-6}	0.0918	5.8×10^{-4}	0.1268	3.5×10^{-5}	0.0476	9.4×10^{-4}
Ochiai	0.0936	1.6×10^{-11}	0.0701	2.2×10^{-9}	0.0845	4.6×10^{-6}	0.1525	7.2×10^{-5}
Arithmetic Mean	0.0605	1.0×10^{-6}	0.0544	9.5×10^{-4}	0.1324	1.8×10^{-4}	0.0806	4.6×10^{-4}
M2	0.0382	7.9×10^{-7}	0.0339	1.9×10^{-5}	0.0132	4.3×10^{-6}	0.0926	8.1×10^{-4}
AMPLE AMPLE2 ER1 ER3 ER5	0	Inapplicable	0	Inapplicable	0	Inapplicable	0	Inapplicable
Kulcznski2	−0.0403	Inapplicable	−0.0475	Inapplicable	−0.0130	Inapplicable	0.0169	1.0×10^{-4}

thousand lines of executable statements, developers may just care about some top few suggested candidates. That is, the other parts may exceed the effort a developer affordable to examine (Song, 2014; Kochhar et al., 2016). If the faulty entity appears only after the top selected few, it can be considered useless and developers may just switch to other fault localization manners to finish their debugging tasks. Hence, we check only a portion of the ranked list rather than using the whole list, and compute the percentage of faulty versions, in which their faults can be located when checking a limited portion of the ranked list. It is natural that when handling programs of a larger scale, a developer may be more tolerant to the accuracy of a fault localization tool and want to check more suspicious candidates than usual before giving up. Thus, we use two notions Top-N and Top-N‰ to replace the notion of expense in Definition 2.2, and accordingly calculate the increase ratios. Here, the Top-N and the Top-N‰ metrics calculate the increase ratios of fault-localization accuracy after cloning, by solely examining the top N elements and the top N‰ elements in the ranked list, respectively. In this study, N is chosen to be 5, i.e., we choose to check the top 5 and the top 5‰ part of a ranked list for a program version, respectively. We measure whether the improvement is observable to developers under these two metrics.

Table 5 shows the Max, Top-5 and Top-5‰ results for each risk evaluation formula in the single-fault scenario. Take the formula ER4 (the second row) and Max (the second column) as an example. The number “0.4020” denotes the average increase ratio (i.e., Avg. Inc. Ratio in the table) in terms of the Max expense. Other cells can be interpreted similarly. We have the following observations.[8]

R7. The average increase ratios of ER4, M1, Fleiss, ER6, Cohen, ER2, Kulcznski1, Cohen, Wong3, Ochiai, Arithmetic Mean, and M2 are always positive in terms of Max, Top-5 and Top-5‰. In other words, on average, the accuracy of these formulas can be improved by applying the cloning strategy.

R8. The average increase ratios of Kulcznski2 are negative in terms of Max and Top-5 and positive in terms of Top-5‰. In other words, the accuracy of Kulcznski2 may not be always improved.

[8] For ease of presentation, the increase ratio of each formula on each individual program is not shown in this table. We note that in summary, they are consistent to the observations we have presented in Table 3 that Kulcznski2 is only negative on programs “replace”, “schedule”, “schedule2”, “tcas”, and “tot_info”, when incorporating the Top-N and Max metrics.

R9. The increase ratios of AMPLE, AMPLE2, ER1, ER3, and ER5 are always zero. It also means that on average, the accuracy of these formulas are preserved by the cloning strategy.

3.3. Summary

In Table 3, we have found that on average, the cloning strategy has aided formulas in groups ER2, ER4, and ER6 as well as formulas M1, Kulcznski1, Cohen, and Fleiss. Moreover, on average, the strategy has not changed the accuracy of the formulas in ER1, ER3 and ER5 as well as the formulas AMPLE and AMPLE2. We have further found that the results do not vary significantly in single-, double- and triple-fault scenarios. Using metrics Max, Top-5, and Top-5‰, the obtained results do not significantly derivate from the results on using the metric Avg.

Overall speaking, the results have shown that a large proportion of studied SBFL formulas can be positively affected by the cloning strategy to increase their fault-localization accuracy. We have also observed that formulas exhibiting positive or zero increase ratios in Tables 4 and 5 are very consistent. Is it a coincidence? Or, is there any theoretical reason behind the scenes? In the next section, we are going to study these formulas from an analytical viewpoint.

4. Theoretical analysis

In this section, we present our results by giving mathematical proofs to the formulas which accuracy was observed to have been consistently improved or preserved by applying the cloning strategy in the experiment reported in the last section. We first formulate the process of the cloning strategy, then define the possible types of a SBFL formula that can be affected by the cloning strategy. Finally, we determine the type of each SBFL formula studied.

4.1. Type of a SBFL formula

We suppose that we are given a program containing fault(s) that have been executed over a given test suite such that the code coverage information has been collected and all test cases in the given test suite have been labeled as either failed or passed. We want to determine a SBFL formula to be one of the four types that the accuracy of the corresponding types of SBFL formulas will be *improved, preserved, deteriorated*, or *nondeterministic* after applying the cloning strategy, respectively. In this section, when we discuss the accuracy of a fault localization formula, it is evaluated using the two expense metrics Avg (Definition 2.2) and Max (Section 2.2.2). We do not include the metrics Top-N or Top-N‰ (Section 2.2.2) because there is no objective criterion in choosing N in these metrics. Nevertheless, the Min (Wong et al., 2007) expense metrics is included to help generalize our theoretical analysis with Avg and Max. To separate from the *expense metrics* used in the controlled experiment in Section 3.2, we consistently list out Avg, Min, and Max as *expense schemes* in this section. The definitions of types are given as follows. It is easy to know they are mutually exclusive, and a SBFL formula must belong to one of them.

Definition 4.1 (Type Preserved)**.** A SBFL formula is said to be Preserved if for any program, any fault (including every fault in a multi-fault scenario), any test suite, and any expense scheme, the increase ratio is always zero.

Definition 4.2 (Type Improved)**.** A SBFL formula is said to be Improved if for any program, any fault (including every fault in a multi-fault scenario), any test suite, and any expense scheme, the increase ratio is always non-negative; and for at least one program, one fault (can be one fault in a multi-fault scenario), one test suite, and one expense scheme, the increase ratio is positive.

Definition 4.3 (Type Deteriorated)**.** A SBFL formula is said to be Deteriorated if for any program, any fault (including every fault in a multi-fault scenario), any test suite, and any expense scheme, the increase ratio is always non-positive; and for at least one program, one fault (can be one fault in a multi-fault scenario), one test suite, and one expense scheme, the increase ratio is negative.

Definition 4.4 (Type Non-deterministic)**.** A SBFL formula is said to be Non-deterministic if for at least one program, one fault (can be one fault in a multi-fault scenario), one test suite, and one expense scheme, the increase ratio is positive; and for at least one program, one fault (can be one fault in a multi-fault scenario), one test suite, and one expense scheme, the increase ratio is negative.

4.2. The determination of types

In this section, we first list out lemma, definitions, and notations used for easy of presentation. After that, we give theorems and proofs on how we determine the types of a SBFL formula.

4.2.1. Lemma, definitions, and notations

Notation 4.1 ($\mathcal{P}rog$, $\mathcal{S}uite$, i, f)**.** We use $\mathcal{P}rog$ to denote a program, and use $\mathcal{S}uite$ to denote a test suite. Further, we use i to denote an arbitrary program entity in $\mathcal{P}rog$, and use f to denote a faulty program entity in $\mathcal{P}rog$.

Notation 4.2 ($\mathcal{S}usp_\Delta$)**.** $\mathcal{S}usp_\Delta = \mathcal{S}usp^i - \mathcal{S}usp^f$.

The notation $\mathcal{S}usp_\Delta$ denotes the result of the subtraction of the suspiciousness score of f from the suspiciousness score of i. A positive $\mathcal{S}usp_\Delta$ for a program entity i indicates that the program entity i is more suspicious than the program entity f. Similarly, a negative $\mathcal{S}usp_\Delta$ indicates that the corresponding program entity i is less suspicious than f. If $\mathcal{S}usp_\Delta = 0$, it indicates that i is equally suspicious as f. The proofs in this paper are based on the concept of mutually exclusive sets (Xie et al., 2013a).

Definition 4.5 (Mutually exclusive sets $\mathcal{S}et_{\mathrm{H}}$, $\mathcal{S}et_{\mathrm{E}}$, and $\mathcal{S}et_{\mathrm{L}}$ (Xie et al., 2013a))**.** Program entities can be grouped into three **exclusive sets**, according to whether the suspiciousness score of a program entity is higher than, equal to, or lower than the suspiciousness score of f. These three sets are denoted by $\mathcal{S}et_{\mathrm{H}}$, $\mathcal{S}et_{\mathrm{E}}$, and $\mathcal{S}et_{\mathrm{L}}$, respectively.

Here, $\mathcal{S}et_{\mathrm{H}}$, $\mathcal{S}et_{\mathrm{E}}$, and $\mathcal{S}et_{\mathrm{L}}$ can be determined based on $\mathcal{S}usp_\Delta$, as $\mathcal{S}et_{\mathrm{H}} = \{i \mid \mathcal{S}usp_\Delta > 0\}$, $\mathcal{S}et_{\mathrm{E}} = \{i \mid \mathcal{S}usp_\Delta = 0\}$, and $\mathcal{S}et_{\mathrm{L}} = \{i \mid \mathcal{S}usp_\Delta < 0\}$. With $\mathcal{S}et_{\mathrm{H}}$, $\mathcal{S}et_{\mathrm{E}}$, and $\mathcal{S}et_{\mathrm{L}}$, the Min, Max, and Avg schemes (see Section 3.2.2) can be expressed as $\frac{|\mathcal{S}et_{\mathrm{H}}|+1}{|\mathcal{S}et_{\mathrm{H}}|+|\mathcal{S}et_{\mathrm{E}}|+|\mathcal{S}et_{\mathrm{L}}|}$, $\frac{|\mathcal{S}et_{\mathrm{H}}|+|\mathcal{S}et_{\mathrm{E}}|}{|\mathcal{S}et_{\mathrm{H}}|+|\mathcal{S}et_{\mathrm{E}}|+|\mathcal{S}et_{\mathrm{L}}|}$, and $\frac{\lfloor|\mathcal{S}et_{\mathrm{H}}|+1/2+|\mathcal{S}et_{\mathrm{E}}|/2\rfloor}{|\mathcal{S}et_{\mathrm{H}}|+|\mathcal{S}et_{\mathrm{E}}|+|\mathcal{S}et_{\mathrm{L}}|}$, respectively.

Regarding the cloning strategy, we also introduce the following notations to simplify the presentation.

Notation 4.3 ($\mathcal{S}usp'_\Delta$, $\mathcal{S}et'_{\mathrm{H}}$, $\mathcal{S}et'_{\mathrm{E}}$, $\mathcal{S}et'_{\mathrm{L}}$)**.** $\mathcal{S}usp'_\Delta$, $\mathcal{S}et'_{\mathrm{H}}$, $\mathcal{S}et'_{\mathrm{E}}$, $\mathcal{S}et'_{\mathrm{L}}$ denote the corresponding values of $\mathcal{S}usp_\Delta$, $\mathcal{S}et_{\mathrm{H}}$, $\mathcal{S}et_{\mathrm{E}}$, $\mathcal{S}et_{\mathrm{L}}$, after cloning, respectively.

We use $\mathbf{a}'_{\mathrm{ep}}$, $\mathbf{a}'_{\mathrm{np}}$, $\mathbf{a}'_{\mathrm{ef}}$, $\mathbf{a}'_{\mathrm{nf}}$ to compute $\mathcal{S}usp'$ and $\mathcal{S}usp'_\Delta$, use $\mathcal{S}usp'_\Delta$ to determine the sets $\mathcal{S}et'_{\mathrm{H}}$, $\mathcal{S}et'_{\mathrm{E}}$, $\mathcal{S}et'_{\mathrm{L}}$, and finally compute the expense increases and the increase ratios with respect to different expense schemes.

4.2.2. Theorems and corollaries

Lemma 4.1. *In the single-fault scenario,* $(\mathbf{a}_{ef}^{f}, \mathbf{a}_{nf}^{f}) = (F, 0)$. *Xie et al. (2013a)*[9]

Theorem 4.1 (Determining a Preserved-typed formula)**.** *A SBFL formula is of type* Preserved *when all of C1-C3 are satisfied.*

C1: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \Rightarrow \mathcal{S}usp'_{\Delta} > 0)$;
C2: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} = 0 \Rightarrow \mathcal{S}usp'_{\Delta} = 0)$;
C3: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} < 0 \Rightarrow \mathcal{S}usp'_{\Delta} < 0)$.

Proof. Satisfying C1 means that any program entity, which belongs to the set $\mathcal{S}et'_{H}$ before the cloning, also belongs to $\mathcal{S}et_{H}$ after applying cloning. Satisfying C2 means that any program entity, which belongs to the set $\mathcal{S}et'_{E}$ before the cloning, also belongs to $\mathcal{S}et_{E}$ after applying cloning. Satisfying C3 means that any program entity, which belongs to the set $\mathcal{S}et'_{L}$ before the cloning, also belongs to $\mathcal{S}et_{L}$ after applying cloning. As a result, the sizes of the three sets do not vary before and after cloning. So the expenses calculated before and after cloning are the same, and the expense increase and increase ratio are both zero. □

Corollary 4.1. *A SBFL formula is of type* Preserved *when any two of C1-C3 are satisfied.*

C1: $\forall \mathcal{P}rog, \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \Leftrightarrow \mathcal{S}usp'_{\Delta} > 0)$;
C2: $\forall \mathcal{P}rog, \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} = 0 \Leftrightarrow \mathcal{S}usp'_{\Delta} = 0)$;
C3: $\forall \mathcal{P}rog, \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} < 0 \Leftrightarrow \mathcal{S}usp'_{\Delta} < 0)$.

Proof. The result directly follows Theorem 4.1 because the sets $\mathcal{S}et_{H}$, $\mathcal{S}et_{E}$, and $\mathcal{S}et_{L}$ are mutually exclusive. □

Theorem 4.2 (Determining an Improved-typed formula)**.** *A SBFL formula is of type* Improved *when all of C1-C4 are satisfied.*

C1: $\forall \mathcal{P}rog, \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \Leftarrow \mathcal{S}usp'_{\Delta} > 0)$;
C2: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} < 0 \Rightarrow \mathcal{S}usp'_{\Delta} < 0)$;
C3: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \wedge \mathcal{S}usp'_{\Delta} \leq 0)$;
C4: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} \geq 0 \wedge \mathcal{S}usp'_{\Delta} < 0)$.

Proof. Similarly to the proof of Theorem 4.1, satisfying C1 and C2 respectively mean that the Min scheme and the Max scheme never increase in all cases of cloning. On the other hand, satisfying C3 and C4 respectively mean that the Min scheme and the Max scheme decrease in some cases of cloning. Since the Avg scheme is an averaging of the Min scheme and the Max scheme, as a result, satisfying all of C1-C4 means all the three kinds of schemes never increase in all cases, and at least one of them decreases in some cases. □

Corollary 4.2. *A SBFL formula is of type* Improved *when all of C1-C3 are satisfied.*

C1: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} \geq \mathcal{S}usp'_{\Delta})$;
C2: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \wedge \mathcal{S}usp'_{\Delta} \leq 0)$;
C3: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} \geq 0 \wedge \mathcal{S}usp'_{\Delta} < 0)$.

Proof. Since the condition C1 here implies both "$\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \Leftarrow \mathcal{S}usp'_{\Delta} > 0)$" (C1 of Theorem 4.2) and "$\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \Leftarrow \mathcal{S}usp'_{\Delta} > 0)$" (C2 of Theorem 4.2), the corollary follows Theorem 4.2. □

Corollary 4.3. *A SBFL formula is of type* Improved *when both C1 and C2 are satisfied.*

C1: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} \geq \mathcal{S}usp'_{\Delta})$;
C2: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \wedge \mathcal{S}usp'_{\Delta} < 0)$.

Proof. Since the condition C2 here implies both "$\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \wedge \mathcal{S}usp'_{\Delta} \leq 0)$" (C3 of Theorem 4.2) and "$\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} \geq 0 \wedge \mathcal{S}usp'_{\Delta} < 0)$" (C4 of Theorem 4.2), the corollary follows Theorem 4.2. □

[9] Note that the happening of coincidental correctness does not affect this fact.

Corollary 4.4. *A SBFL formula is of type* Improved *when all of C1-C3 are satisfied.*

C1: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp'_{\Delta} = \alpha \mathcal{S}usp_{\Delta} + \beta, \alpha > 0, \beta \leq 0)$;
C2: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \wedge \mathcal{S}usp'_{\Delta} \leq 0)$;
C3: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} \geq 0 \wedge \mathcal{S}usp'_{\Delta} < 0)$.

Proof. From C1, we can obtain "$\mathcal{S}usp'_{\Delta} > 0 \Rightarrow \alpha \mathcal{S}usp_{\Delta} + \beta > 0 \Rightarrow \alpha \mathcal{S}usp_{\Delta} > 0 \Rightarrow \mathcal{S}usp_{\Delta} > 0$" (C1 of Theorem 4.2) and "$\mathcal{S}usp_{\Delta} < 0 \Rightarrow \alpha \mathcal{S}usp'_{\Delta} < 0 \Rightarrow \alpha \mathcal{S}usp'_{\Delta} + \beta < 0 \Rightarrow \mathcal{S}usp'_{\Delta} < 0$" (C2 of Theorem 4.2). In other words, the corollary also follows Theorem 4.2 because C1 here implies "$\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \Leftarrow \mathcal{S}usp'_{\Delta} > 0)$" (C1 of Theorem 4.2) and "$\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} < 0 \Rightarrow \mathcal{S}usp'_{\Delta} < 0)$" (C2 of Theorem 4.2). □

Theorem 4.3. (Determining a. Deteriorated-*typed formula). A SBFL formula is of type* Deteriorated *when all of C1-C4 are satisfied.*

C1: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} < 0 \Leftarrow \mathcal{S}usp'_{\Delta} < 0)$;
C2: $\forall \mathcal{P}rog,\ \forall \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \Rightarrow \mathcal{S}usp'_{\Delta} > 0)$;
C3: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} < 0 \wedge \mathcal{S}usp'_{\Delta} \geq 0)$;
C4: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} \leq 0 \wedge \mathcal{S}usp'_{\Delta} > 0)$.

Proof. Similar to the proof of Theorem 4.1, satisfying C1 and C2 mean that the Min and Max schemes never decrease in all cases of cloning. On the other hand, satisfying C3 and C4 respectively mean that the Min scheme and the Max scheme increase in some cases of cloning. As a result, satisfying all of C1-C4 means that the expense under each of the three kinds of expense schemes never decreases in all cases, and may increase in some cases. □

Theorem 4.4 (Determining a Non-deterministic-typed formula)**.** *A formula is of type* Non-deterministic *when both C1 and C2 are satisfied.*

C1: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} > 0 \wedge \mathcal{S}usp'_{\Delta} \leq 0)$;
C2: $\exists \mathcal{P}rog,\ \exists \mathcal{S}uite, (\mathcal{S}usp_{\Delta} < 0 \wedge \mathcal{S}usp'_{\Delta} \geq 0)$.

Proof. Similar to the proof of Theorem 4.1, satisfying C1 means that the expense can increase in some cases of cloning. Satisfying C2 means that the expense can decrease in some cases of cloning. As a result, satisfying both C1 and C2 means the expenses can either decrease or increase in specified cases of cloning. □

4.3. Results of analysis

By applying the theorems and corollaries in the last subsection, we can analytically determine the type of a SBFL formula, in the single-fault and multi-fault scenarios, respectively.

Xie et al. (2013a) have proved six equivalent groups in the single-fault scenario, namely, ER1 to ER6. We follow their work to list out the groups and formulas in Table 6. As a result, in the single-fault scenario, there are mainly six equivalent groups of formulas and 11 ungrouped formulas. Since Naish et al. (2011) have showed the equivalence for five groups of them with no assumption of single-fault or multi-fault, in the multi-fault scenario, there are another five equivalent groups of formulas, namely, ER2 to ER6. Specifically, formulas originally in ER1 is now known not to be kept under the same group, and Binary is no longer in group ER5. The proofs of these relations reported in Table 6 are reported in Appendix B.11, B.12, and B.15. ER1 and ER5 are previously claimed to be maximal by Xie et al. (2013a). Our results show that formulas in ER1 and ER5 can be generalized to the multi-fault scenario.[10]

[10] From the tables, we not only identify SBFL formulas of type Improved and Preserved, but also identify SBFL formulas of type Non-deterministic, which means their accuracy can be either improved or deteriorated, depending on the program, fault, and test cases.

Table 6
Theoretical analysis of the effectiveness of cloning.

Group/Formula	Scenarios		Proof
	single-fault	multi-fault	
ER2	Improved	Non-deterministic	Appendix A.1, B.1
ER4			Appendix A.2, B.2
ER6			Appendix A.3, B.3
Ochiai			Appendix A.4, B.4
M1			Appendix A.5, B.5
M2			Appendix A.6, B.6
Wong3			Appendix A.7, B.7
Kulcznski1			Appendix A.8, B.8
AMPLE	Preserved	Preserved	Appendix A.9, B.9
AMPLE2			Appendix A.10, B.10
ER1			Appendix A.11, B.11, B.12
ER3			Appendix A.12, B.13
ER5			Appendix A.13, B.14, B.15
Fleiss	Non-deterministic	Non-deterministic	Appendix A.14, B.16
Cohen			Appendix A.15, B.16
Arithmetic Mean			Appendix A.16, B.16
Kulcznski2			Appendix A.17, B.16

The results on the SBFL type are showed in the "Type" column of Table 6 for both the single-fault and multi-fault scenarios. From Table 6, we can obtain the following results.

R10. In total, 19 formulas, including five ungrouped formulas Ochiai, M1, M2, Wong3, and Kulcznski1 and formulas in three groups ER2, ER4, and ER6, are of type Improved in the single-fault scenario, and of type Non-deterministic in the multi-fault scenario.
R11. In total, ten formulas, including two ungrouped formulas AMPLE and AMPLE2 and formulas in three groups ER1, ER3, and ER5, are of type Preserved, in both the single-fault and the multi-fault scenarios.
R12. Formulas Kulcznski2, Arithmetic Mean, Cohen and Fleiss are of type Non-deterministic in both scenarios.
R13. None of the formulas is of type Deteriorated in any scenario.

5. Summary

In this section, we answer the research questions, and discuss threats to the validity.

5.1. Answering research questions

To answer the research questions, we have reported a controlled experiment in Section 3 and a theoretical analysis in Section 4 in both single-fault and multi-fault scenarios. Table 7 summarizes the major findings reported in Tables 3–6. It shows both the empirical and theoretical results in both single-fault and multi-fault scenarios for all the 33 studied formulas. We can thus answer the three research questions as follows:

A1
(a) Improvements in average accuracy are observed on 22, 21, and 23 formulas, in the single-fault scenario, double-fault, and triple-fault scenarios, respectively.
(b) Consistent results are observed when incorporating four different accuracy metrics, and all the observed improvements are determined statistically significant, using hypothesis testing.
(c) At the same time, the accuracy of ten formulas are observed unaffected, in the single-fault, double-fault, and triple-fault scenarios.

A2
(a) The observed improvement ratios in the Avg expense accuracy on the 22, 21, and 23 formulas are in range of [3.82%, 42.73%], [1.77%, 42.24%], and [1.24%, 44.02%], in the single-fault, double-fault, and triple-fault, respectively.
(b) The observed improvement ratios on the 22 formulas in the single-fault scenario are in range of [3.39%, 40.20%], [1.32%, 125.81%], and [3.08%, 38.78%], when incorporating the Max, Top-5, and Top-5‰ accuracy metrics, respectively.
(c) However, in each scenario, some formulas result in adverse effects in accuracy on some benchmarks occasionally (see Tables 3 and 4).

A3
(a) In the single-fault scenario, the improvements on average accuracy on 19 formulas are analytically proved to be consistent across all programs, all faults, all test suites, and all adopted expense metrics.
(b) On the other hand, accuracy of ten of the rest formulas are analytically confirmed unaffected across all programs in all scenarios.

5.2. Threats to validity

In this section, we discuss the factors affecting empirical observations and the issues resulting in different theoretical analysis conclusion.

5.2.1. Different experiment setups

We used twelve benchmarks to reduce the possibility of overfitting of the results. Nonetheless, although some of the subjects of these benchmarks were real-life and medium-scale programs, they only represented several classes of programs. For object-oriented programs, our model can also be applied. However, the characteristics of this kind of language may be quite different from that of procedural languages, such as C, which is the language all the subject programs are written in. Different characteristics of programming languages result in different program structures and that may also produce different results. Using other kinds of programs and more benchmarks in an experiment can strengthen the measure to alleviate this threat to validity.

Besides, the experiment used the SIR test suites and test cases as starting points to generate class-balanced test suites for each benchmark. It may lead to a bias in the experimental results if the test suites and test cases downloaded from the SIR is not totally fair. By cross-validating the results of our theoretical analysis reported in the present paper, we observe a high consistency in whether the increase ratio is zero or positive with the empirical findings. We tend to believe that the use of these test suites and test cases are trustworthy in our experiment.

We followed the SIR documents to manipulate the data set to simulate a software development scenario. Experiments simulating other development scenarios may manifest different observations. We used *gcov* to conduct coverage profiling. The tool gcov is a reliable coverage profiling tool except the inability to handle stack overflow errors. To give a better support for the crashing runs, we used some tactics associated with gcov to retrieve the traces for those exception cases. Previous work Fu and Ryder (2007); Yuan et al. (2010) have investigated this topic that the exception information in runtime contains plenty of error information, providing more information to ease fault localization. It is worth noting that different configurations on a coverage profiling tool may result in different observations.

The main purpose of the multi-fault scenario is to produce a scenario that not all failed test cases in a given test suite refer to the same fault. In this way, the cloning strategy is not limited by the theoretical relations already found in prior theoretical analy-

Table 7
Summary of empirical observations and theoretical analysis.

Group	Formula	Empirical observations			Theoretical analysis	
		Single-fault	Double-fault	Triple-fault	Single-fault	Multi-fault
ER2	Anderberg Dice Godman Jaccard Sørensen-Dice	statistically significant improvements (always > 0)			Improved	
ER4	Euclid Hamann Hamann etc. Rogers & Tanimoto Simple Matching Sokal Wong2					
ER6	Rogot1 Scott					
non-grouped	kulcznski1 M1 M2 Wong3 Ochiai Cohen Fleiss Kulcznski2 Arithmetic Mean AMPLE AMPLE2	on average < 0	occasionally < 0		Non-deterministic	
ER1	Naish1 Naish2	always = 0			Preserved	
ER3	CBI inc. q_e Tarantula					
ER5	Binary Russel & Rao Wong1					

sis studies (e.g., Xie et al., 2013a) in the single-fault scenario. The procedure to produce multi-fault versions is applicable to serve this purpose. We simulated the multi-fault scenario by using pair-wise combinations of existing faults in the SIR benchmarks, and used 50% of pair-wise combination program versions and 2000 of triple-wise combination of program versions due to the constraint of our resources and co-existential conflict among faults. On the other hand, we only used the faulty versions produced or synthesized from the faults available in SIR in the experiment. Observations from the empirical results may vary with programs containing more faults are used (e.g., the subjects in the experiments of Yu et al., 2015). The use of other faults and the use of other ways to simulate the multi-fault scenarios may give different empirical results.

On the other hand, in the double-fault scenario, we only evaluated the accuracy of a formula in locating the first fault. It is because the accuracy of them in locating the second fault has been evaluated in the single-fault experiment. For the same reason, in the triple-fault scenario, we only evaluated the accuracy of a fault localization formula in locating the first fault. It is because that the accuracy of them in locating the second fault had been evaluated in the double-fault experiment, and the accuracy of them in locating the third fault had been evaluated in the single-fault experiment. We intentionally did so to avoid the mix up of empirical results from different scenarios. Another reason supporting us to do so is that in practice a developer never knows whether more than one fault exists in the program under debugging. As a result, the most efficient approach is to test the revised program right after a fault is fixed, since examining the whole or rest part of the ranked list of a previous program version may not be meaningful. In some cases, fixing the first fault may also break the error propagation chain of some other faults, which may coincidentally also fix the latter too. In such a way, referencing the accuracy of such a formula in locating the first fault better reflects the efforts of a developer's work. However, different empirical observations might be obtained when using different evaluation strategies.

There also exist other kinds of tie breaking strategies (e.g., Le et al., 2016). Incorporating other accuracy metrics may also result in different observations.

Table 8
Effectiveness of cloning on the formula *R*.

	Test cases						ER1 (e.g., Naish1)		ER4 (e.g., Wong2)		R	
	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6	before	after	before	after	before	after
*s*1	•	•	○	○	•	•	2	2	0	2	2	−2
*s*2	○	○	•	•	•	•	2	2	0	2	2	−2
*s*3: faulty	•	•	•	•	•	•	0	0	−2	0	−4	0
*s*4	•	•	•	○	•	○	−1	−1	−2	−1	−5	−3
*s*5	•	○	○	•	•	•	2	2	0	2	2	−2
	pass	pass	pass	pass	fail	fail	rank: 4	rank: 4	rank: 5	rank: 4	rank: 4	rank: 1
							unaffected		catches up		exceeds	

5.2.2. Formulas beyond this study

Since Xie et al. (2013a) have proved that the ER1 formulas are the maximum among the 33 studied formulas in *any* single-fault scenario, they are also the maximum among the 33 formulas after cloning. As a result of theoretical analysis, the cloning strategy improves the accuracy of 19 studied formulas, but can neither help them outperform ER1 nor improve ER1, in the single-fault scenario. Table 3 also shows that the increase ratios are always zero on twelve programs by the cloning strategy. However, by including formulas that *have no partial order with ER1* (Xie et al., 2013a), the conclusions can be different. For example, let us discuss the following artificial formula *R*.

$$R = \begin{cases} -1 - 2 \cdot (\mathbf{a}_{ef} - \mathbf{a}_{ep})^2 & \text{if } \mathbf{a}_{ef} < F \\ P - \mathbf{a}_{ep} - 2 \cdot (\mathbf{a}_{ef} - \mathbf{a}_{ep})^2 & \text{if } \mathbf{a}_{ef} = F \end{cases}$$

We use a program containing five entities, namely *s*1, *s*2, *s*3, *s*4, *s*5, and three test cases, namely *t*1, *t*2, *t*3, where the program entity *s*3 contains a fault, *t*1 and *t*2 are the passed test cases, and *t*3 is the failed test case. The coverage information is shown in Table 8, in which "•" denotes that a statement is executed by a test case, and "○" denotes that the statement is not executed by the test case. The numbers indicate a suspiciousness score which is calculated for a program entity by a formula.

Xie et al. (2013a) have shown that formulas in the ER1 group is more accuracy than formulas in the ER4 group in the single-fault scenario. From Table 8, we can see that Wong2 (from the ER4 group) catches up with Naish1 (from the ER1 group) after the cloning strategy is applied. We can also see that Naish1 outperforms R before cloning, but R becomes outperforming Naish1 after the cloning strategy is applied. Though we show detailed spectra for Naish1 and Wong2 to exemplify ER1 and ER4 in Table 8, the above observation also applies on any formulas from the ER1 group and any fromula from the ER4 group.

In summary, even though the cloning strategy preserves the maximum accuracy of the 33 studied formulas, there exist formulas, which have no partial-order relationship with the ER1 formulas in theory, but also benefit from the cloning strategy and can outperform the ER1 formulas in practice.

6. Related work

Software development process can be tedious and time-consuming because of the appearance and occurrence of faults (also known as bugs). A fault localization task is to locate the root cause of software defects in the source code. At present, typical software fault localization methods include slicing-based fault localization (Zhang et al., 2005), model-based fault localization (Mayer et al., 2007; Xie et al., 2013b), spectrum-based fault localization (Abreu et al., 2006; 2007; Zhang et al., 2008; Baah et al., 2010; Xu et al., 2013), and so on (Dallmeier et al., 2005; Jeffrey et al., 2008; Debroy and Wong, 2009; Kim et al., 2013; Yoo et al., 2013; Masri and Assi, 2014; Lin and Li, 2014). Among them, the spectrum-based fault localization techniques form one of the most heavily studied family of techniques. In recent years, as the requirement of software quality gets higher, software testing is also becoming more and more important. With the advantage of better accuracy and lower cost, spectrum-based fault localization attracts a number of researchers.

Jones et al. (2002) and Jones and Harrold (2005) propose Tarantula. Their proposal uses the mean values of the execution count statistics to compose a formula to assess suspiciousness of statements. Observing that individual executions of statements may have different contributions to indicate suspicious statements, Wong et al. (2007) proposed to calibrate the contribution of each passed execution. Their techniques are shown empirically to outperform Tarantula, and they further defined a series of heuristics (Wong et al., 2010). There are many other such techniques. Russel and Rao (1940), Dice (1945), Ochiai (1957), Rogers and Tanimoto (1960), Anderberg (1988), Sørensen-Dice (Duarte et al., 1999), and Simple Matching and Ochiai2 (Meyer et al., 2004) are originally used in the botany domain. Further, Overlap (Krause, 1973) is a general version of Ochiai (1957). Formulas of the following techniques are also visited in this paper, including Hamming (1950), Goodman and Kruskal (1954), Scott (1955), Cohen (1960), Fleiss (1965), Rogot1, Rogot2, Harmonic Mean, and Arithmetic Mean (Rogot and Goldberg, 1966), Geometric Mean (Maxwell and Pilliner, 1968), Euclid (Krause, 1973), M1, and M2 (Everitt, 1978), Jaccard (1901), Kulczynskil and Kulczynski2 (Lourenco et al., 2004), Hamann and Sokal (Lourenco et al., 2004), Ample (Dallmeier et al., 2005), Zoltar (Sanchez, 2007), and Wong1, Wong2, and Wong3 (Wong et al., 2007). Different from the listed works, the present work does not develop any novel formulas. Rather, we focus on the factors that have impacts on the accuracy across many SBFL formulas.

Lee et al. (2009) used the Siemens Suite and Space as benchmarks. Their experiment suggested that Ochiai is more effective than Jaccard. Naish et al. (2011) have similar conclusions. Wong et al. (2012) proposed crosstab-based technique, and their experimental results showed that the crosstab-based technique is more effective than other techniques, such as Jaccard, SOBER and Liblit05. The formula (Ochiai, 1957) was first introduced from the molecular biology field into spectrum-based fault localization by Abreu et al. (2006). In their work (Abreu et al., 2006), they have shown that Ochiai is more effective than Jaccard, AMPLE and Tarantula. However, they did not discover the basic inherent principles. From the theoretical view, Xie et al. (2013a) analyzed Naish, Jaccard, Tarantula and other related formulas. In the single-fault scenario, they obtained the relationship (best, worst, original order) among different formulas. Tang et al. (2017) managed to relax some assumptions in Xie et al. (2013a) and evaluate the accuracies of formulas. Compared to their works, the present work reports a theoretical analysis in the multi-fault scenario as well.

The class proportion of data sets has been widely studied in the field of machine learning and pattern recognition. Japkowicz and Shaju (2002) establish a relationship between complexity, class proportion level, and size of the training set. They drew a conclusion that the class imbalance phenomenon of the training set brings negative impact to a standard classifier adapting to the class balance situations. Moreover, related studies (Jones and Harrold, 2005; Wong et al., 2008b) also showed that the class imbalance phenomenon of data sets usually has influence on the efficiency of classification. In essence, software fault localization can be viewed as a pattern recognition problem of how to effectively distinguish statements containing latent faults with other ones. As a result, spectrum-based fault localization techniques are influenced by the characteristics of test suites, which can be measured as the extent of class balance between the number of passed test cases and number of failed test cases (Gong et al., 2012). Indeed, this problem has been revisited from different aspects. Applying the formula of Tarantula to rank statements, Baudry et al. (2006) empirically showed that fewer test cases are required to achieve the same fault-localization effectiveness. In recent years, the test suite reduction techniques (Yu et al., 2008; Xuan and Monperrus, 2014), being paid much attention by relevant researchers, tried to improve fault localization accuracy by carefully reducing the size of test suite. Hao et al. (2005) suggested that they can effectively improve the efficiency of fault localization formulas, by cutting similar test cases and reducing information redundancy. Baudry et al. (2006) also came to similar conclusions as the above work. However, there are related studies that reached different conclusions. Yu et al. (2008) with ten test suite reduction methods, by comparing four fault localization techniques, concluded that current test suite reduction techniques may reduce the efficiency of fault localization techniques. Gong et al. (2012) empirically showed that a test suite containing approximately identical numbers of passed and failed test cases can favor existing fault localization techniques. In this paper, we extend the preliminary version of this work by both empirically and theoretically studying the effect of applying the cloning strategy on formulas of 33 SBFL techniques.

Many works aim at exploring the integration of different spectra or fault localization techniques. Santelices et al. (2009) developed an integration technique that exploits the combination of different spectra for fault localization. Empirical studies demonstrate that, on average, combining multiple coverages improves the effectiveness of fault localization, better approximates the ideal choice of coverage per fault, and provides more stable fault localization than using an individual coverage. Debroy and Wong (2013) proposed a consensus-based strategy that combines the results of multiple fault localization techniques, to improve their quality, irrespective of data set. The main idea of the strategy is to combine different rank orderings generated by multiple fault localization techniques in order to obtain a "better ordering". Lucia et al. (2014) proposed a fault localizer fusing approach, assessing the accuracy of several coefficients in software debugging, which consists of three steps: score normalization, technique selection, and data fusion. Besides, a few works are based on machine learning techniques. Wong et al. (2014) proposed a Dstar approach based on an utilization of binary similarity measures and four intuitions. Wong and Qi (2009) and Wong et al. (2008a); 2009) developed two fault localization methods based on BP neural network and RBF neural network, respectively. In their methods, network is trained to learn the relationship between the statement coverage information of a test case and its corresponding execution results, pass or fail.

7. Conclusion

The effectiveness of spectrum-based fault localization techniques can be affected by the number of passed and failed test cases. Previous studies showed that the use of class-balanced test suite, which contained almost identical numbers of passed test cases and failed test cases, can increase the fault-localization effectiveness of such techniques, compared to the use of class-imbalanced test suites.

In the preliminary version of this paper, we proposed a cloning strategy to produce a class-balanced test suite to improve fault localization effectiveness. In this paper, from a theoretical perspective, we have comprehensively analyzed the impact of such a cloning strategy on the accuracy of the formulas of 33 techniques in single-, double- and triple-fault scenarios. We have used twelve subject programs to conduct a controlled experiment to evaluate the accuracy. Empirical results have shown that 22, 21, and 23 formulas show higher average fault-localization accuracy after the failed test cases are cloned, in these three scenarios, respectively. At the same time, consistent empirical observations have been obtained by incorporating four different accuracy metrics. Further, the observed improvements have been validated to be statistically significant using hypothesis testing method. We have also formally defined the criteria that the accuracy of a SBFL formula being improved, deteriorated, preserved, or non-deterministically affected, and give mathematical theorems and proofs on these four types. We have shown by theorems that the accuracy for 19 of the 33 formulas are improved in the single-fault scenario and ten of the 33 formulas are preserved in both single-fault and multi-fault scenarios.

It is encouraging that the empirical results and theoretical analysis agree with each other, improving the reliability of our results. They both show the cloning strategy useful to improve, or at least keep, the accuracy of most of the studied spectrum-based fault localization formulas. Our work has uncovered a previously unknown layer of connection between empirical findings and theoretical results. For future work, we recommend to explore better test suite generation and construction formulas to further improve the precision and practicality of spectrum-based fault localization. It is also a good idea to include more large-scale real-world benchmarks to study debugging techniques so that the research results can be more relevant to the industry.

Appendix A. Proofs in the single-fault scenario

A1. The ER2 group (Improved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Xie et al., 2013a), we take Jaccard as example, for which we have:

$$\mathcal{S}usp_{\Delta} = \frac{\mathbf{a}^i_{ef}}{F+\mathbf{a}^i_{ep}} - \frac{F}{F+\mathbf{a}^f_{ep}} = \frac{F\left(\mathbf{a}^i_{ef}-F\right)+\mathbf{a}^i_{ef}\mathbf{a}^f_{ep}-F\mathbf{a}^i_{ep}}{\left(F+\mathbf{a}^i_{ep}\right)\left(F+\mathbf{a}^f_{ep}\right)}$$

$$\mathcal{S}usp'_{\Delta} = \frac{c\mathbf{a}^i_{ef}}{cF+\mathbf{a}^i_{ep}} - \frac{cF}{cF+\mathbf{a}^f_{ep}} = \frac{c\left[cF\left(\mathbf{a}^i_{ef}-F\right)+\mathbf{a}^i_{ef}\mathbf{a}^f_{ep}-F\mathbf{a}^i_{ep}\right]}{\left(cF+\mathbf{a}^i_{ep}\right)\left(cF+\mathbf{a}^f_{ep}\right)}$$

We next give the following derivation:

$$\mathcal{S}usp'_{\Delta} = \frac{c\left(F+\mathbf{a}^i_{ep}\right)\left(F+\mathbf{a}^f_{ep}\right)}{\left(cF+\mathbf{a}^i_{ep}\right)\left(cF+\mathbf{a}^f_{ep}\right)}\mathcal{S}usp_{\Delta} + \frac{c(c-1)F\left(\mathbf{a}^i_{ef}-F\right)}{\left(cF+\mathbf{a}^i_{ep}\right)\left(cF+\mathbf{a}^f_{ep}\right)}$$

At the same time, the conditions $(\mathcal{S}usp_{\Delta} > 0 \wedge \mathcal{S}usp'_{\Delta} <= 0)$ and $(\mathcal{S}usp'_{\Delta} < 0 \wedge \mathcal{S}usp_{\Delta} >= 0)$ in Corollary 4.4 can be satisfied by $\langle \mathbf{a}^f_{ep} = 2F, \mathbf{a}^f_{ef} = F, \mathbf{a}^i_{ep} = \frac{F}{2}, \mathbf{a}^i_{ef} = \frac{F}{3}\rangle$ and $\langle \mathbf{a}^f_{ep} = 2F, \mathbf{a}^f_{ef} = F, \mathbf{a}^i_{ep} = \frac{F}{2}, \mathbf{a}^i_{ef} = \frac{F}{2}\rangle$, respectively. As a result, Jaccard is of type Improved since $c > 0$ is assumed (see Notation 2.2). □

A2. The ER4 group (Improved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Xie et al., 2013a), we take Wong2 as example, for which we have:

$$Susp_{\Delta} = \mathbf{a}^i_{ef} - \mathbf{a}^i_{ep} - \left(\mathbf{a}^f_{ef} - \mathbf{a}^f_{ep}\right) = \mathbf{a}^i_{ef} - F - \left(\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}\right)$$

$$Susp'_{\Delta} = c\mathbf{a}^i_{ef} - \mathbf{a}^i_{ep} - \left(c\mathbf{a}^f_{ef} - \mathbf{a}^f_{ep}\right) = c\left(\mathbf{a}^i_{ef} - F\right) - \left(\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}\right)$$

We next give the following derivation:

$$Susp'_{\Delta} = (c-1)\left(\mathbf{a}^i_{ef} - F\right) + Susp_{\Delta} \Rightarrow Susp'_{\Delta} \leq Susp_{\Delta}$$

At the same time, the conditions $(Susp_{\Delta} > 0 \wedge Susp'_{\Delta} <= 0)$ and $(Susp'_{\Delta} < 0 \wedge Susp_{\Delta} >= 0)$ of Corollary 4.2 can be satisfied by $\langle \mathbf{a}^f_{ep} = F, \mathbf{a}^f_{ef} = F, \mathbf{a}^i_{ep} = \frac{F}{3}, \mathbf{a}^i_{ef} = \frac{F}{2}\rangle$ and $\langle \mathbf{a}^f_{ep} = F, \mathbf{a}^f_{ef} = F, \mathbf{a}^i_{ep} = \frac{F}{2}, \mathbf{a}^i_{ef} = \frac{F}{2}\rangle$, respectively. Finally, Wong2 is of type Improved. □

A3. The ER6 group (Improved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Xie et al., 2013a), we take Rogot1 as example, for which we have:

$$Susp_{\Delta} = \frac{1}{2}\left(\frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + F + \mathbf{a}^i_{ep}} - \frac{F}{2F + \mathbf{a}^f_{ep}}\right) + \frac{1}{2}\left(\frac{\mathbf{a}^i_{np}}{\mathbf{a}^i_{np} + \mathbf{a}^i_{nf} + P} - \frac{\mathbf{a}^f_{np}}{\mathbf{a}^f_{np} + P}\right)$$

$$Susp'_{\Delta} = \frac{1}{2}\left(\frac{c\mathbf{a}^i_{ef}}{c\mathbf{a}^i_{ef} + cF + \mathbf{a}^i_{ep}} - \frac{cF}{2cF + \mathbf{a}^f_{ep}}\right) + \frac{1}{2}\left(\frac{\mathbf{a}^i_{np}}{\mathbf{a}^i_{np} + c\mathbf{a}^i_{nf} + P} - \frac{\mathbf{a}^f_{np}}{\mathbf{a}^f_{np} + P}\right)$$

We next give the following derivation:

Case 1 ($\mathbf{a}^f_{ep} \leq \mathbf{a}^i_{ep}$): We know that

$$Susp_{\Delta} = \frac{1}{2}\left(\frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + F + \mathbf{a}^i_{ep}} - \frac{F}{2F + \mathbf{a}^f_{ep}}\right) + \frac{1}{2}\left(\frac{P - \mathbf{a}^i_{ep}}{2P - \mathbf{a}^i_{ep} + F - \mathbf{a}^i_{ef}} - \frac{P - \mathbf{a}^f_{ep}}{2P - \mathbf{a}^f_{ep}}\right) < \frac{1}{2}\left(\frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + F + \mathbf{a}^i_{ep}} - \frac{F}{2F + \mathbf{a}^f_{ep}}\right) < 0,$$

$$Susp'_{\Delta} = \frac{1}{2}\left(\frac{c\mathbf{a}^i_{ef}}{c\mathbf{a}^i_{ef} + cF + \mathbf{a}^i_{ep}} - \frac{cF}{2cF + \mathbf{a}^f_{ep}}\right) + \frac{1}{2}\left(\frac{P - \mathbf{a}^i_{ep}}{2P - \mathbf{a}^i_{ep} + cF - c\mathbf{a}^i_{ef}} - \frac{P - z}{2P - \mathbf{a}^f_{ep}}\right) < \frac{1}{2}\left(\frac{c\mathbf{a}^i_{ef}}{c\mathbf{a}^i_{ef} + cF + \mathbf{a}^i_{ep}} - \frac{cF}{2cF + \mathbf{a}^f_{ep}}\right) < 0$$

Case 2 ($\mathbf{a}^f_{ep} > \mathbf{a}^i_{ep}$): We know that

$$\frac{\partial Susp'_{\Delta}}{\partial c} = \frac{1}{2}\left[\frac{\mathbf{a}^i_{ef}\mathbf{a}^i_{ep}}{\left(c\mathbf{a}^i_{ef} + cF + \mathbf{a}^i_{ep}\right)^2} - \frac{F\mathbf{a}^f_{ep}}{\left(2cF + \mathbf{a}^f_{ep}\right)^2} - \frac{\left(P - \mathbf{a}^i_{ep}\right)\left(F - \mathbf{a}^i_{ef}\right)}{\left(2P - \mathbf{a}^i_{ep} + cF - c\mathbf{a}^i_{ef}\right)^2}\right]$$

$$\leq \frac{1}{2}\left[\frac{\mathbf{a}^i_{ef}\mathbf{a}^i_{ep}}{\left(c\mathbf{a}^i_{ef} + cF + \mathbf{a}^i_{ep}\right)^2} - \frac{F\mathbf{a}^f_{ep}}{\left(2cF + \mathbf{a}^f_{ep}\right)^2}\right]$$

Let $G\left(\mathbf{a}^i_{ef}, \mathbf{a}^i_{ep}\right) = \frac{\mathbf{a}^i_{ef}\mathbf{a}^i_{ep}}{\left(c\mathbf{a}^i_{ef} + cF + \mathbf{a}^i_{ep}\right)^2}$, then we can get

$$\begin{cases} \frac{\partial G\left(\mathbf{a}^i_{ef}, \mathbf{a}^i_{ep}\right)}{\partial \mathbf{a}^i_{ef}} = \frac{c\mathbf{a}^i_{ep}\left(F - \mathbf{a}^i_{ef}\right) + {\mathbf{a}^i_{ep}}^2}{\left(c\mathbf{a}^i_{ef} + cF + \mathbf{a}^i_{ep}\right)^3} > 0 \Rightarrow G\left(\mathbf{a}^i_{ef}, \mathbf{a}^i_{ep}\right) \leq G\left(F, \mathbf{a}^i_{ep}\right) \\ \frac{\partial G\left(F, \mathbf{a}^i_{ep}\right)}{\partial \mathbf{a}^i_{ep}} = \frac{2cF^2 - F\mathbf{a}^i_{ep}}{\left(2cF + \mathbf{a}^i_{ep}\right)^3} > 0 \Rightarrow G\left(F, \mathbf{a}^i_{ep}\right) < G\left(F, \mathbf{a}^f_{ep}\right) \end{cases}$$

$$\Rightarrow G\left(\mathbf{a}^i_{ef}, \mathbf{a}^i_{ep}\right) < G\left(F, \mathbf{a}^f_{ep}\right) \Rightarrow \frac{\partial Susp'_{\Delta}}{\partial c} < 0 \Rightarrow Susp'_{\Delta} < Susp_{\Delta}.$$

The conditions $(Susp_{\Delta} > 0 \wedge Susp'_{\Delta} < 0)$ can be satisfied by $\langle c = 10, \mathbf{a}^f_{ep} = 10F, \mathbf{a}^f_{ef} = F, \mathbf{a}^i_{ep} = 9F, \mathbf{a}^i_{ef} = \frac{F}{2}\rangle$. Finally, Rogot1 is of type Improved (by Corollary 4.3). □

A4. The Ochiai formula (Improved)

Proof. First,

$$Susp_{\Delta} = \frac{\mathbf{a}^i_{ef}}{\sqrt{F\left(\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)}} - \frac{F}{\sqrt{F\left(F + \mathbf{a}^f_{ep}\right)}},$$

$$Susp'_{\Delta} = \frac{c\mathbf{a}^i_{ef}}{\sqrt{cF\left(c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)}} - \frac{cF}{\sqrt{cF\left(cF + \mathbf{a}^f_{ep}\right)}}$$

We next give the following derivation:

$$Susp'_{\Delta} > 0 \Rightarrow \frac{c\mathbf{a}^i_{ef}}{\sqrt{cF\left(c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)}} > \frac{cF}{\sqrt{cF\left(cF + \mathbf{a}^f_{ep}\right)}}$$

$$\Rightarrow \frac{(1-c)\left(F - \mathbf{a}^i_{ef}\right)}{c{\mathbf{a}^i_{ef}}F} + \frac{\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}/c}{{\mathbf{a}^f_{ef}}^2} < \frac{F + \mathbf{a}^f_{ep}/c}{F^2}$$

$$\Rightarrow \frac{1-c}{c\mathbf{a}^f_{ep}} + \frac{\mathbf{a}^f_{ep} + \mathbf{a}^i_{ep}/c}{{\mathbf{a}^i_{ef}}^2} < \frac{1-c}{cF} + \frac{F + \mathbf{a}^f_{ep}/c}{F^2}$$

$$\Rightarrow \frac{1}{\mathbf{a}^f_{ef}} + \frac{\mathbf{a}^i_{ep}}{{\mathbf{a}^i_{ef}}^2} < \frac{1}{F} + \frac{\mathbf{a}^f_{ep}}{F^2} \Rightarrow T > 0$$

$$Susp_{\Delta} < 0 \Rightarrow \frac{1}{\mathbf{a}^f_{ef}} + \frac{\mathbf{a}^i_{ep}}{{\mathbf{a}^i_{ef}}^2} > \frac{1}{F} + \frac{\mathbf{a}^f_{ep}}{F^2}$$

$$\Rightarrow \frac{1-c}{c\mathbf{a}^f_{ep}} + \frac{\mathbf{a}^f_{ep} + \mathbf{a}^i_{ep}/c}{{\mathbf{a}^i_{ef}}^2} > \frac{1-c}{cF} + \frac{F + \mathbf{a}^f_{ep}/c}{F^2}$$

$$\Rightarrow \frac{(1-c)\left(F - \mathbf{a}^i_{ef}\right)}{c{\mathbf{a}^i_{ef}}F} + \frac{\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}/c}{{\mathbf{a}^f_{ef}}^2} > \frac{F + \mathbf{a}^f_{ep}/c}{F^2}$$

$$\Rightarrow \frac{c\mathbf{a}^i_{ef}}{\sqrt{cF\left(c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)}} < \frac{cF}{\sqrt{cF\left(cF + \mathbf{a}^f_{ep}\right)}} \Rightarrow T' < 0$$

At the same time, the conditions $(Susp_{\Delta} > 0 \wedge Susp'_{\Delta} <= 0)$ and $(Susp'_{\Delta} < 0 \wedge Susp_{\Delta} >= 0)$ can be satisfied by $\langle c = 2, \mathbf{a}^f_{ep} = 4F, \mathbf{a}^f_{ef} = F, \mathbf{a}^i_{ep} = \frac{F}{2}, \mathbf{a}^i_{ef} = \frac{F}{2}\rangle$ and $\langle c = 2, \mathbf{a}^f_{ep} = 3F, \mathbf{a}^f_{ef} = F, \mathbf{a}^i_{ep} = \frac{F}{2}, \mathbf{a}^i_{ef} = \frac{F}{2}\rangle$, respectively. Finally, Ochiai is of type Improved (by Theorem 4.2). □

A5. The M1 formula (Improved)

Proof. First,

$$Susp_{\Delta} = \frac{(F+P)\left(\mathbf{a}^i_{ef} + \mathbf{a}^f_{ep} - F - \mathbf{a}^i_{ep}\right)}{\left(F - \mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\mathbf{a}^f_{ep}},$$

$$Susp'_{\Delta} = \frac{(cF+P)\left(c\mathbf{a}^i_{ef}+\mathbf{a}^f_{ep}-cF-\mathbf{a}^i_{ep}\right)}{\left(cF-c\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\mathbf{a}^f_{ep}}$$

where $\mathbf{a}^i_{nf}+\mathbf{a}^i_{ep}\neq 0$ and $\mathbf{a}^f_{nf}+\mathbf{a}^f_{ep}\neq 0$. We next give the following derivation:

$$\begin{aligned} Susp'_{\Delta} > 0 &\Rightarrow c\left(\mathbf{a}^i_{ef}-F\right)+\mathbf{a}^f_{ep}-\mathbf{a}^i_{ep}>0 \\ &\Rightarrow \mathbf{a}^i_{ef}-F+\mathbf{a}^f_{ep}-\mathbf{a}^i_{ep}>0 \Rightarrow Susp_{\Delta}>0, \\ Susp_{\Delta} < 0 &\Rightarrow \mathbf{a}^i_{ef}-F+\mathbf{a}^f_{ep}-\mathbf{a}^i_{ep}<0 \\ &\Rightarrow c\left(\mathbf{a}^i_{ef}-F\right)+\mathbf{a}^f_{ep}-\mathbf{a}^i_{ep}<0 \Rightarrow Susp'_{\Delta}<0 \end{aligned}$$

At the same time, the conditions $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp'_{\Delta}<0\wedge Susp_{\Delta}>=0)$ can be satisfied by $c=10$, $\langle \mathbf{a}^f_{ep}=10F, \mathbf{a}^f_{ef}=F, \mathbf{a}^i_{ep}=9F, \mathbf{a}^i_{ef}=\frac{F}{2}\rangle$. Finally, M1 is of type Improved (by Theorem 4.2). □

A6. The M2 formula (Improved)

Proof. First,

$$Susp_{\Delta} = \frac{\mathbf{a}^i_{ef}}{2F+P-\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}} - \frac{F}{F+P+\mathbf{a}^f_{ep}},$$

$$Susp'_{\Delta} = \frac{c\mathbf{a}^i_{ef}}{2cF+P-c\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}} - \frac{cF}{cF+P+\mathbf{a}^f_{ep}}.$$

We next give the following derivation:

$$\begin{aligned} Susp'_{\Delta} = &\frac{c\left(2F+P-\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\left(F+P+\mathbf{a}^f_{ep}\right)}{\left(2cF+P-c\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\left(cF+P+\mathbf{a}^f_{ep}\right)}T \\ &+\frac{2c(c-1)F\left(\mathbf{a}^i_{ef}-F\right)}{\left(2cF+P-c\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\left(cF+P+\mathbf{a}^f_{ep}\right)} \end{aligned}$$

At the same time, the conditions $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp'_{\Delta}<0\wedge Susp_{\Delta}>=0)$ can be satisfied by $\langle P=5F, \mathbf{a}^f_{ep}=F, \mathbf{a}^f_{ef}=F, \mathbf{a}^i_{ep}=F, \mathbf{a}^i_{ef}=\frac{4F}{5}\rangle$. Finally, M2 is of type Improved (by Corollary 4.4). □

A7. The Wong3 formula (Improved)

Proof. Case 1 $(\mathbf{a}^i_{ep}\leq 2)$: $T=\mathbf{a}^i_{ef}-F+\delta_1, T'=c\left(\mathbf{a}^i_{ef}-F\right)+\delta_1$,

where $\delta_1 = \begin{cases} -\mathbf{a}^i_{ep}+\mathbf{a}^f_{ep} & \text{if } \mathbf{a}^f_{ep}\leq 2 \\ -\mathbf{a}^i_{ep}+1.8+0.1\mathbf{a}^f_{ep} & \text{if } 2<\mathbf{a}^f_{ep}\leq 10 \\ -\mathbf{a}^i_{ep}+2.79+0.001\mathbf{a}^f_{ep} & \text{if } 2<\mathbf{a}^f_{ep}\leq 10 \end{cases}$

Case 2 $(2<\mathbf{a}^i_{ep}\leq 10)$: $T=\mathbf{a}^i_{ef}-F+\delta_2, T'=c\left(\mathbf{a}^i_{ef}-F\right)+\delta_2$,

where $\delta_2 = \begin{cases} -0.1\mathbf{a}^i_{ep}-1.8+\mathbf{a}^f_{ep} & \text{if } \mathbf{a}^f_{ep}\leq 2 \\ -0.1\mathbf{a}^i_{ep}+0.1\mathbf{a}^f_{ep} & \text{if } 2<\mathbf{a}^f_{ep}\leq 10 \\ -0.1\mathbf{a}^i_{ep}+0.99+0.001\mathbf{a}^f_{ep} & \text{if } 2<\mathbf{a}^f_{ep}\leq 10 \end{cases}$

Case 3 $(\mathbf{a}^i_{ep}>10)$: $T=\mathbf{a}^i_{ef}-F+\delta_3, T'=c\left(\mathbf{a}^i_{ef}-F\right)+\delta_3$, where

$\delta_3 = \begin{cases} -0.001\mathbf{a}^i_{ep}-2.79+\mathbf{a}^f_{ep} & \text{if } \mathbf{a}^f_{ep}\leq 2 \\ -0.001\mathbf{a}^i_{ep}-0.99+0.1\mathbf{a}^f_{ep} & \text{if } 2<\mathbf{a}^f_{ep}\leq 10 \\ -0.001\mathbf{a}^i_{ep}+0.001\mathbf{a}^f_{ep} & \text{if } 2<\mathbf{a}^f_{ep}\leq 10 \end{cases}$

We next give the following derivation,

$$Susp'_{\Delta} = Susp_{\Delta} + (c-1)\left(\mathbf{a}^i_{ef}-F\right)$$

At the same time, the conditions $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp'_{\Delta}<0\wedge Susp_{\Delta}>=0)$ can be satisfied by $\langle c=2, F=10, \mathbf{a}^f_{ep}=10, \mathbf{a}^f_{ef}=10, \mathbf{a}^i_{ep}=1, \mathbf{a}^i_{ef}=9\rangle$ and $\langle c=2, \mathbf{a}^f_{ep}=2, \mathbf{a}^f_{ef}=10, \mathbf{a}^i_{ep}=1, \mathbf{a}^i_{ef}=9\rangle$, respectively. Finally, Wong3 is of type Improved (by Corollary 4.4). □

A8. The Kulcznski1 formula (Improved)

Proof. First,

$$Susp_{\Delta} = \frac{F\left(\mathbf{a}^i_{ef}-F\right)+\mathbf{a}^i_{ef}\mathbf{a}^f_{ep}-F\mathbf{a}^i_{ep}}{\left(F-\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\mathbf{a}^f_{ep}},$$

$$Susp'_{\Delta} = \frac{c\left[cF\left(\mathbf{a}^i_{ef}-F\right)+\mathbf{a}^i_{ef}\mathbf{a}^f_{ep}-F\mathbf{a}^i_{ep}\right]}{\left(cF-c\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\mathbf{a}^f_{ep}},$$

where $\mathbf{a}^i_{nf}+\mathbf{a}^i_{ep}\neq 0$ and $\mathbf{a}^f_{nf}+\mathbf{a}^f_{ep}\neq 0$. We next give the following derivation:

$$Susp'_{\Delta} = \frac{c\left(F-\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\mathbf{a}^f_{ep}}{\left(cF-c\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\mathbf{a}^f_{ep}}Susp_{\Delta} + \frac{c(c-1)F\left(\mathbf{a}^i_{ef}-F\right)}{\left(cF-c\mathbf{a}^i_{ef}+\mathbf{a}^i_{ep}\right)\mathbf{a}^f_{ep}}$$

At the same time, the conditions $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp'_{\Delta}<0\wedge Susp_{\Delta}>=0)$ can be satisfied by $\langle \mathbf{a}^f_{ep}=2F, \mathbf{a}^f_{ef}=F, \mathbf{a}^i_{ep}=\frac{F}{2}, \mathbf{a}^i_{ef}=\frac{F}{3}\rangle$ and $\langle \mathbf{a}^f_{ep}=2F, \mathbf{a}^f_{ef}=F, \mathbf{a}^i_{ep}=\frac{F}{2}, \mathbf{a}^i_{ef}=\frac{F}{2}\rangle$, respectively. The Kulcznski1 is of type Improved (by the Corollary 4.4). □

A9. The AMPLE formula (Preserved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Xie et al., 2013a), we take Wong1 as example, for which we have:

$$Susp_{\Delta} = \left|\frac{\mathbf{a}^i_{ef}}{F}-\frac{\mathbf{a}^i_{ep}}{P}\right| - \left|\frac{F}{F}-\frac{\mathbf{a}^f_{ep}}{P}\right|,$$

$$Susp'_{\Delta} = \left|\frac{c\mathbf{a}^i_{ef}}{cF}-\frac{\mathbf{a}^i_{ep}}{P}\right| - \left|\frac{cF}{cF}-\frac{\mathbf{a}^f_{ep}}{P}\right| = \left|\frac{\mathbf{a}^i_{ef}}{F}-\frac{\mathbf{a}^i_{ep}}{P}\right| - \left|\frac{F}{F}-\frac{\mathbf{a}^f_{ep}}{P}\right|$$

We next have $Susp_{\Delta}=Susp'_{\Delta}$. Finally, AMPLE is of type Preserved. □

A10. The AMPLE2 formula (Preserved)

Proof. First,

$$Susp_{\Delta} = \frac{\mathbf{a}^i_{ef}}{F}-\frac{\mathbf{a}^i_{ep}}{P}-\left(\frac{F}{F}-\frac{\mathbf{a}^f_{ep}}{P}\right),$$

$$Susp'_{\Delta} = \frac{c\mathbf{a}^i_{ef}}{cF}-\frac{\mathbf{a}^i_{ep}}{P}-\left(\frac{cF}{cF}-\frac{\mathbf{a}^f_{ep}}{P}\right) = \frac{\mathbf{a}^i_{ef}}{F}-\frac{\mathbf{a}^i_{ep}}{P}-\left(\frac{F}{F}-\frac{\mathbf{a}^f_{ep}}{P}\right).$$

We next have $Susp_{\Delta}=Susp'_{\Delta}$. Finally, AMPLE2 is of type Preserved. □

A11. The ER1 group (Preserved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Xie et al., 2013a), we take Niash2 as example, for which we have

$$Susp_{\Delta} = \mathbf{a}^i_{ef}-\frac{\mathbf{a}^i_{ep}}{P+1}-\left(\mathbf{a}^f_{ef}-\frac{\mathbf{a}^f_{ep}}{P+1}\right) = \mathbf{a}^i_{ef}-F-\frac{\mathbf{a}^i_{ep}-\mathbf{a}^f_{ep}}{P+1},$$

$$\begin{aligned} Susp'_{\Delta} &= c\cdot\mathbf{a}^i_{ef}-\frac{\mathbf{a}^i_{ep}}{P+1}-\left(c\cdot\mathbf{a}^f_{ef}-\frac{\mathbf{a}^f_{ep}}{P+1}\right) \\ &= c\left(\mathbf{a}^i_{ef}-F\right)-\frac{\mathbf{a}^i_{ep}-\mathbf{a}^f_{ep}}{P+1}. \end{aligned}$$

We next give the following derivation:

$$Susp_{\Delta}>0 \Rightarrow \mathbf{a}^i_{ef}-F>\frac{\mathbf{a}^i_{ep}-\mathbf{a}^f_{ep}}{P+1}$$

$$\Rightarrow \left(\mathbf{a}^i_{ef} - F = 0 \bigwedge \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} < 0\right)$$

$$\Rightarrow c\left(\mathbf{a}^i_{ef} - F\right) > \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} \Rightarrow \mathcal{S}usp'_\Delta > 0,$$

$$\mathcal{S}usp_\Delta = 0 \Rightarrow \mathbf{a}^i_{ef} - F = \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1}$$

$$\Rightarrow \left(\mathbf{a}^i_{ef} - F = 0 \bigwedge \mathbf{a}^i_{ep} - \mathbf{a}^f_{ep} = 0\right)$$

$$\Rightarrow c\left(\mathbf{a}^i_{ef} - F\right) = \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} \Rightarrow \mathcal{S}usp'_\Delta = 0,$$

$$\mathcal{S}usp_\Delta < 0 \Rightarrow \mathbf{a}^i_{ef} - F < \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1}$$

$$\Rightarrow c\left(\mathbf{a}^i_{ef} - F\right) < \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} \Rightarrow \mathcal{S}usp'_\Delta < 0.$$

Finally, Niash2 is of type Preserved (by Theorem 4.1). □

A12. The ER3 group (Preserved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Xie et al., 2013a), we take q_e as example, for which we have:

$$\mathcal{S}usp_\Delta = \frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}} - \frac{\mathbf{a}^f_{ef}}{\mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}}$$

$$= \frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}} - \frac{F}{F + \mathbf{a}^f_{ep}} = \frac{\mathbf{a}^i_{ef}\mathbf{a}^f_{ep} - \mathbf{a}^i_{ep}F}{\left(\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(F + \mathbf{a}^f_{ep}\right)}$$

$$\mathcal{S}usp'_\Delta = \frac{c\mathbf{a}^i_{ef}}{c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}} - \frac{c\mathbf{a}^f_{ef}}{c\mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}} = \frac{c\mathbf{a}^i_{ef}}{c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}} - \frac{cF}{cF + \mathbf{a}^f_{ep}}$$

$$= \frac{c\left(\mathbf{a}^i_{ef}\mathbf{a}^f_{ep} - \mathbf{a}^i_{ep}F\right)}{\left(c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(cF + \mathbf{a}^f_{ep}\right)}$$

We next give the following derivation:

$$\mathcal{S}usp'_\Delta = \frac{c\left(\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(F + \mathbf{a}^f_{ep}\right)}{\left(c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(cF + \mathbf{a}^f_{ep}\right)}$$

$$\mathcal{S}usp_\Delta \Rightarrow \begin{cases} \mathcal{S}usp_\Delta > 0 \Leftrightarrow \mathcal{S}usp'_\Delta > 0 \\ \mathcal{S}usp_\Delta = 0 \Leftrightarrow \mathcal{S}usp'_\Delta = 0 \\ \mathcal{S}usp_\Delta < 0 \Leftrightarrow \mathcal{S}usp'_\Delta < 0 \end{cases}$$

Finally, q_e is of type Preserved (by Theorem 4.1). □

A13. The ER5 group (Preserved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Xie et al., 2013a), we take Wong1 as example, for which we have: $\mathcal{S}usp_\Delta = \mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} = \mathbf{a}^i_{ef} - F$, $\mathcal{S}usp'_\Delta = c\mathbf{a}^i_{ef} - c\mathbf{a}^f_{ef} = c\left(\mathbf{a}^i_{ef} - F\right)$. We next have $\mathcal{S}usp'_\Delta = c\mathcal{S}usp_\Delta$. Finally, Wong1 is of type Preserved (by Theorem 4.1). □

A14. The Fleiss formula (Non-deterministic)

Proof. We can get the functions $\mathcal{S}usp_\Delta$ and $\mathcal{S}usp'_\Delta$ of Fleiss:

$$\mathcal{S}usp_\Delta = \frac{4\mathbf{a}^i_{ef}\left(P - \mathbf{a}^i_{ep}\right) - \left(F - \mathbf{a}^i_{ef} - \mathbf{a}^i_{ep}\right)^2 - 4F\left(P - \mathbf{a}^f_{ep}\right) + {\mathbf{a}^f_{ep}}^2}{2F + 2P}$$

$$\mathcal{S}usp'_\Delta = \frac{4c\mathbf{a}^i_{ef}\left(P - \mathbf{a}^i_{ep}\right) - \left(cF - c\mathbf{a}^i_{ef} - \mathbf{a}^i_{ep}\right)^2 - 4cF\left(P - \mathbf{a}^f_{ep}\right) + {\mathbf{a}^f_{ep}}^2}{2cF + 2P}$$

Here, $(\mathcal{S}usp_\Delta > 0 \wedge \mathcal{S}usp'_\Delta <= 0)$ can be satisfied by $\langle P = 1000, F = 100, \mathbf{a}^f_{ep} = 468, \mathbf{a}^f_{ef} = 100, \mathbf{a}^i_{ep} = 1, \mathbf{a}^i_{ef} = 1\rangle$. And $(\mathcal{S}usp_\Delta < 0 \wedge \mathcal{S}usp'_\Delta >= 0)$ can be satisfied by $\langle P = 1000, F = 100, \mathbf{a}^f_{ep} = 839, \mathbf{a}^f_{ef} = 100, \mathbf{a}^i_{ep} = 888, \mathbf{a}^i_{ef} = 1\rangle$. Finally, Fleiss is of type Non-deterministic (by Theorem 4.4). □

A15. The Cohen formula (Non-deterministic)

Proof.

$$\mathcal{S}usp_\Delta = \frac{2\mathbf{a}^i_{ef}\left(P - \mathbf{a}^i_{ep}\right) - 2\left(F - \mathbf{a}^i_{ef}\right)\mathbf{a}^i_{ep}}{\left(\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)P + \left(F - \mathbf{a}^i_{ef} + P - \mathbf{a}^i_{ep}\right)F} - \frac{2F\left(P - \mathbf{a}^f_{ep}\right)}{\left(F + \mathbf{a}^f_{ep}\right)P + \left(P - \mathbf{a}^f_{ep}\right)F}$$

$$\mathcal{S}usp'_\Delta = \frac{2c\mathbf{a}^i_{ef}\left(P - \mathbf{a}^i_{ep}\right) - 2\left(cF - c\mathbf{a}^i_{ef}\right)\mathbf{a}^i_{ep}}{\left(c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)P + \left(cF - c\mathbf{a}^i_{ef} + P - \mathbf{a}^i_{ep}\right)cF} - \frac{2cF\left(P - \mathbf{a}^f_{ep}\right)}{\left(cF + \mathbf{a}^f_{ep}\right)P + \left(P - \mathbf{a}^f_{ep}\right)cF}$$

Here, $(\mathcal{S}usp_\Delta > 0 \wedge \mathcal{S}usp'_\Delta <= 0)$ can be satisfied by $\langle P = 1000, F = 100, \mathbf{a}^f_{ep} = 992, \mathbf{a}^f_{ef} = 100, \mathbf{a}^i_{ep} = 1, \mathbf{a}^i_{ef} = 1\rangle$. And $(\mathcal{S}usp_\Delta < 0 \wedge \mathcal{S}usp'_\Delta >= 0)$ can be satisfied by $\langle P = 1000, F = 100, \mathbf{a}^f_{ep} = 999, \mathbf{a}^f_{ef} = 100, \mathbf{a}^i_{ep} = 118, \mathbf{a}^i_{ef} = 12\rangle$. Finally, Cohen is of type Non-deterministic (by Theorem 4.4). □

A16. The arithmetic mean formula (Non-deterministic)

Proof.

$$\mathcal{S}usp_\Delta = \frac{2\mathbf{a}^i_{ef}\left(P - \mathbf{a}^i_{ep}\right) - 2\left(F - \mathbf{a}^i_{ef}\right)\mathbf{a}^i_{ep}}{\left(\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(F - \mathbf{a}^i_{ef} + P - \mathbf{a}^i_{ep}\right) + PF} - \frac{2F\left(P - \mathbf{a}^f_{ep}\right)}{\left(F + \mathbf{a}^f_{ep}\right)\left(P - \mathbf{a}^f_{ep}\right) + PF}$$

$$\mathcal{S}usp'_\Delta = \frac{2c\mathbf{a}^i_{ef}\left(P - \mathbf{a}^i_{ep}\right) - 2c\left(F - \mathbf{a}^i_{ef}\right)\mathbf{a}^i_{ep}}{\left(c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(cF - c\mathbf{a}^i_{ef} + P - \mathbf{a}^i_{ep}\right) + cPF} - \frac{2cF\left(P - \mathbf{a}^f_{ep}\right)}{\left(cF + \mathbf{a}^f_{ep}\right)\left(P - \mathbf{a}^f_{ep}\right) + cPF}$$

Here, $(\mathcal{S}usp_\Delta > 0 \wedge \mathcal{S}usp'_\Delta <= 0)$ can be satisfied by $\langle P = 1000, F = 100, \mathbf{a}^f_{ep} = 999, \mathbf{a}^f_{ef} = 100, \mathbf{a}^i_{ep} = 1, \mathbf{a}^i_{ef} = 1\rangle$. The condition $(\mathcal{S}usp_\Delta < 0 \wedge \mathcal{S}usp'_\Delta >= 0)$ can be satisfied by $\langle P = 1000, F = 100, \mathbf{a}^f_{ep} = 989, \mathbf{a}^f_{ef} = 100, \mathbf{a}^i_{ep} = 3, \mathbf{a}^i_{ef} = 18\rangle$. Finally, Arithmetic Mean is of type Non-deterministic (by Theorem 4.4). □

A17. The Kulcznski2 formula (Non-deterministic)

Proof. First,

$$\mathcal{S}usp_\Delta = \frac{1}{2}\left(\frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + \mathbf{a}^i_{nf}} + \frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}}\right) - \frac{1}{2}\left(\frac{\mathbf{a}^f_{ef}}{\mathbf{a}^f_{ef} + \mathbf{a}^f_{nf}} + \frac{\mathbf{a}^f_{ef}}{\mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}}\right)$$

$$= \frac{\mathbf{a}^i_{ef} - F}{2F} + \frac{1}{2}\left(\frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}} - \frac{F}{F + \mathbf{a}^f_{ep}}\right),$$

$$\mathcal{S}usp'_\Delta = \frac{1}{2}\left(\frac{c\mathbf{a}^i_{ef}}{c\mathbf{a}^i_{ef} + c\mathbf{a}^i_{nf}} + \frac{c\mathbf{a}^i_{ef}}{c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}}\right)$$

$$-\frac{1}{2}\left(\frac{c\mathbf{a}_{ef}^{f}}{c}\mathbf{a}_{ef}^{f}+c\mathbf{a}_{nf}^{f}+\frac{c\mathbf{a}_{ef}^{f}}{c\mathbf{a}_{ef}^{f}+\mathbf{a}_{ep}^{f}}\right)$$
$$=\frac{\mathbf{a}_{ef}^{i}-F}{2F}+\frac{1}{2}\left(\frac{c\mathbf{a}_{ef}^{i}}{c\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{i}}-\frac{cF}{cF+\mathbf{a}_{ep}^{f}}\right).$$

At the same time, conditions $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp_{\Delta}<0\wedge Susp'_{\Delta}>=0)$ can be satisfied by $\langle c=2,\mathbf{a}_{ep}^{f}=2F,\mathbf{a}_{ef}^{f}=F,\mathbf{a}_{ep}^{i}=0,\mathbf{a}_{ef}^{i}=\frac{F}{2}\rangle$ and $\langle c=2,\mathbf{a}_{ep}^{f}=F,\mathbf{a}_{ef}^{f}=F,\mathbf{a}_{ep}^{i}=0,\mathbf{a}_{ef}^{i}=\frac{F}{2}\rangle$, respectively. Kulcznski2 is of type Non-deterministic (by Theorem 4.4). □

Appendix B. Multi-fault program

B1. The ER2 group (Non-deterministic)

Proof. Since all the formulas in the group are equivalent (Naish et al., 2011), we take Jaccard as example:

$$Susp_{\Delta}=\frac{F\left(\mathbf{a}_{ef}^{i}-\mathbf{a}_{ef}^{f}\right)+\mathbf{a}_{ef}^{i}\mathbf{a}_{ep}^{f}-\mathbf{a}_{ef}^{f}\mathbf{a}_{ep}^{i}}{\left(F+\mathbf{a}_{ep}^{i}\right)\left(F+\mathbf{a}_{ep}^{f}\right)},$$
$$Susp'_{\Delta}=\frac{c\left[cF\left(\mathbf{a}_{ef}^{i}-\mathbf{a}_{ef}^{f}\right)+\mathbf{a}_{ef}^{i}\mathbf{a}_{ep}^{f}-\mathbf{a}_{ef}^{f}\mathbf{a}_{ep}^{i}\right]}{\left(cF+\mathbf{a}_{ep}^{i}\right)\left(cF+\mathbf{a}_{ep}^{f}\right)}$$

$(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp_{\Delta}<0\wedge Susp'_{\Delta}>=0)$ can be satisfied by $\langle\mathbf{a}_{ep}^{f}=F,\mathbf{a}_{ef}^{f}=F,\mathbf{a}_{ep}^{i}=\frac{F}{3},\mathbf{a}_{ef}^{i}=\frac{F}{2}\rangle$ and $\langle\mathbf{a}_{ep}^{f}=\frac{F}{3},\mathbf{a}_{ef}^{f}=\frac{F}{2},\mathbf{a}_{ep}^{i}=F,\mathbf{a}_{ef}^{i}=F\rangle$, respectively. So, ER2 is of type Non-deterministic (by Theorem 4.4). □

B2. The ER4 group (Non-deterministic)

Proof. Since all the formulas in the group are equivalent (Naish et al., 2011), we take Wong2 as example, for which we have: $Susp_{\Delta}=\mathbf{a}_{ef}^{i}-\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}+\mathbf{a}_{ep}^{f}$, $Susp'_{\Delta}=c(\mathbf{a}_{ef}^{i}-\mathbf{a}_{ef}^{f})-\mathbf{a}_{ep}^{i}+\mathbf{a}_{ep}^{f}$. $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp_{\Delta}<0\wedge Susp'_{\Delta}>=0)$ can be respectively satisfied by the tuples $\langle\mathbf{a}_{ep}^{f}=F,\mathbf{a}_{ef}^{f}=F,\mathbf{a}_{ep}^{i}=\frac{F}{3},\mathbf{a}_{ef}^{i}=\frac{F}{2}\rangle$ and $\langle\mathbf{a}_{ep}^{f}=\frac{F}{2},\mathbf{a}_{ef}^{f}=\frac{F}{2},\mathbf{a}_{ep}^{i}=F,\mathbf{a}_{ef}^{i}=F\rangle$. Finally, ER4 is of type Non-deterministic (by Theorem 4.4). □

B3. The ER6 group (Non-deterministic)

Proof. Since all the formulas in the group are equivalent (Naish et al., 2011), we take Rogot1 as example, for which we have:

$$Susp_{\Delta}=\frac{1}{2}\left(\frac{\mathbf{a}_{ef}^{i}}{\mathbf{a}_{ef}^{i}+F+\mathbf{a}_{ep}^{i}}-\frac{\mathbf{a}_{ef}^{f}}{\mathbf{a}_{ef}^{f}+F+\mathbf{a}_{ep}^{f}}\right)+\frac{1}{2}\left(\frac{P-\mathbf{a}_{ep}^{i}}{2P-\mathbf{a}_{ep}^{i}+F-\mathbf{a}_{ef}^{i}}-\frac{P-\mathbf{a}_{ep}^{f}}{2P-\mathbf{a}_{ep}^{f}+F-\mathbf{a}_{ef}^{f}}\right)$$
$$Susp'_{\Delta}=\frac{1}{2}\left(\frac{c\mathbf{a}_{ef}^{i}}{c\mathbf{a}_{ef}^{i}+cF+\mathbf{a}_{ep}^{i}}-\frac{c\mathbf{a}_{ef}^{f}}{c\mathbf{a}_{ef}^{f}+cF+\mathbf{a}_{ep}^{f}}\right)+\frac{1}{2}\left(\frac{P-\mathbf{a}_{ep}^{i}}{2P-\mathbf{a}_{ep}^{i}+cF-c\mathbf{a}_{ef}^{i}}-\frac{P-\mathbf{a}_{ep}^{f}}{2P-\mathbf{a}_{ep}^{f}+cF-c\mathbf{a}_{ef}^{f}}\right)$$

ER6 is of type Non-deterministic (by Theorem 4.4) because $\langle c=10,\mathbf{a}_{ep}^{f}=10F,\mathbf{a}_{ef}^{f}=F,\mathbf{a}_{ep}^{i}=9F,\mathbf{a}_{ef}^{i}=\frac{F}{2}\rangle$ satisfies $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$. And $\langle c=10,\mathbf{a}_{ep}^{f}=9F,\mathbf{a}_{ef}^{f}=\frac{F}{2},\mathbf{a}_{ep}^{i}=10F,\mathbf{a}_{ef}^{i}=F\rangle$ satisfies $(Susp_{\Delta}<0\wedge Susp'_{\Delta}>=0)$. □

B4. The Ochiai formula (Non-deterministic)

Proof. First,

$$Susp_{\Delta}=\frac{\mathbf{a}_{ef}^{i}}{\sqrt{F\left(\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{i}\right)}}-\frac{\mathbf{a}_{ef}^{f}}{\sqrt{F\left(\mathbf{a}_{ef}^{f}+\mathbf{a}_{ep}^{f}\right)}},$$
$$Susp'_{\Delta}=\frac{c\mathbf{a}_{ef}^{i}}{\sqrt{cF\left(c\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{i}\right)}}-\frac{c\mathbf{a}_{ef}^{f}}{\sqrt{cF\left(c\mathbf{a}_{ef}^{f}+\mathbf{a}_{ep}^{f}\right)}}$$

At the same time, the conditions $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp_{\Delta}<0\wedge Susp'_{\Delta}>=0)$ can be satisfied by $\langle c=2,\mathbf{a}_{ep}^{f}=4F,\mathbf{a}_{ef}^{f}=F,\mathbf{a}_{ep}^{i}=\frac{F}{2},\mathbf{a}_{ef}^{i}=\frac{F}{2}\rangle$ and $\langle c=2,\mathbf{a}_{ep}^{f}=\frac{F}{2},\mathbf{a}_{ef}^{f}=\frac{F}{2},\mathbf{a}_{ep}^{i}=4F,\mathbf{a}_{ef}^{i}=F\rangle$, respectively. Finally, Ochiai is of type Non-deterministic (by Theorem 4.4). □

B5. The M1 formula (Non-deterministic)

Proof. First,

$$Susp_{\Delta}=\frac{(F+P)\left(\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{f}-\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}\right)}{\left(F-\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{i}\right)\left(F-\mathbf{a}_{ef}^{f}+\mathbf{a}_{ep}^{f}\right)},$$
$$Susp'_{\Delta}=\frac{(cF+P)\left(c\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{f}-c\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}\right)}{\left(cF-c\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{i}\right)\left(cF-c\mathbf{a}_{ef}^{f}+\mathbf{a}_{ep}^{f}\right)}$$

At the same time, the conditions $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp_{\Delta}<0\wedge Susp'_{\Delta}>=0)$ can be satisfied by $\langle\mathbf{a}_{ep}^{f}=F,\mathbf{a}_{ef}^{f}=F,\mathbf{a}_{ep}^{i}=\frac{F}{3},\mathbf{a}_{ef}^{i}=\frac{F}{2}\rangle$ and $\langle\mathbf{a}_{ep}^{f}=\frac{F}{2},\mathbf{a}_{ef}^{f}=\frac{F}{2},\mathbf{a}_{ep}^{i}=F,\mathbf{a}_{ef}^{i}=F\rangle$, respectively. Finally, M1 is of type Non-deterministic (by Theorem 4.4). □

B6. The M2 formula (Non-deterministic)

Proof. First,

$$Susp_{\Delta}=\frac{\mathbf{a}_{ef}^{i}}{2F+P-\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{i}}-\frac{\mathbf{a}_{ef}^{f}}{2F+P-\mathbf{a}_{ef}^{f}+\mathbf{a}_{ep}^{f}},$$
$$Susp'_{\Delta}=\frac{c\mathbf{a}_{ef}^{i}}{2cF+P-c\mathbf{a}_{ef}^{i}+\mathbf{a}_{ep}^{i}}-\frac{c\mathbf{a}_{ef}^{f}}{2cF+P-c\mathbf{a}_{ef}^{f}+\mathbf{a}_{ep}^{f}}$$

At the same time, the conditions $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ and $(Susp_{\Delta}<0\wedge Susp'_{\Delta}>=0)$ can be satisfied by $\langle P=5F,\mathbf{a}_{ep}^{f}=F,\mathbf{a}_{ef}^{f}=F,\mathbf{a}_{ep}^{i}=F,\mathbf{a}_{ef}^{i}=\frac{4F}{5}\rangle$ and $\langle P=5F,\mathbf{a}_{ep}^{f}=F,\mathbf{a}_{ef}^{f}=\frac{4F}{5},\mathbf{a}_{ep}^{i}=F,\mathbf{a}_{ef}^{i}=F\rangle$, respectively. Finally, M2 is of type Non-deterministic (by Theorem 4.4). □

B7. The Wong3 formula (Non-deterministic)

Proof. First,

$$Susp_{\Delta}=\begin{cases}\mathbf{a}_{ef}^{i}-\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}+\mathbf{a}_{ep}^{f} & \text{if } \mathbf{a}_{ep}^{f}\le 2\\ \mathbf{a}_{ef}^{i}-\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}+1.8+0.1\mathbf{a}_{ep}^{f} & \text{if } 2<\mathbf{a}_{ep}^{f}\le 10\\ \mathbf{a}_{ef}^{i}-\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}+2.79+0.001\mathbf{a}_{ep}^{f} & \text{if } 2<\mathbf{a}_{ep}^{f}\le 10\end{cases}$$
$$Susp'_{\Delta}=\begin{cases}c\mathbf{a}_{ef}^{i}-c\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}+\mathbf{a}_{ep}^{f} & \text{if } \mathbf{a}_{ep}^{f}\le 2\\ c\mathbf{a}_{ef}^{i}-c\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}+1.8+0.1\mathbf{a}_{ep}^{f} & \text{if } 2<\mathbf{a}_{ep}^{f}\le 10\\ c\mathbf{a}_{ef}^{i}-c\mathbf{a}_{ef}^{f}-\mathbf{a}_{ep}^{i}+2.79+0.001\mathbf{a}_{ep}^{f} & \text{if } 2<\mathbf{a}_{ep}^{f}\le 10\end{cases}$$

The condition $(Susp_{\Delta}>0\wedge Susp'_{\Delta}<=0)$ can be satisfied by $\langle c=2,F=10,\mathbf{a}_{ep}^{f}=10,\mathbf{a}_{ef}^{f}=10,\mathbf{a}_{ep}^{i}=1,\mathbf{a}_{ef}^{i}=9\rangle$. The

condition $(\mathcal{S}usp_\Delta < 0 \wedge \mathcal{S}usp'_\Delta >= 0)$ can be satisfied by $\langle c = 2, F = 10, \mathbf{a}^f_{ep} = 1, \mathbf{a}^f_{ef} = 9, \mathbf{a}^i_{ep} = 10, \mathbf{a}^i_{ef} = 10\rangle$. Finally, Wong3 is of type Non-deterministic (by Theorem 4.4). □

B8. The Kulcznski1 formula (Non-deterministic)

Proof. First,

$$\mathcal{S}usp_\Delta = \frac{F\left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef}\right) + \mathbf{a}^i_{ef}\mathbf{a}^f_{ep} - \mathbf{a}^f_{ef}\mathbf{a}^i_{ep}}{\left(F - \mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(F - \mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}\right)},$$

$$\mathcal{S}usp'_\Delta = \frac{c\left[cF\left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef}\right) + \mathbf{a}^i_{ef}\mathbf{a}^f_{ep} - \mathbf{a}^f_{ef}\mathbf{a}^i_{ep}\right]}{\left(cF - c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(cF - c\mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}\right)}$$

At the same time, the conditions $(\mathcal{S}usp_\Delta > 0 \wedge \mathcal{S}usp'_\Delta <= 0)$ and $(\mathcal{S}usp_\Delta < 0 \wedge \mathcal{S}usp'_\Delta >= 0)$ can be satisfied by $\langle \mathbf{a}^f_{ep} = F, \mathbf{a}^f_{ef} = F, \mathbf{a}^i_{ep} = \frac{F}{3}, \mathbf{a}^i_{ef} = \frac{F}{2}\rangle$ and $\langle \mathbf{a}^f_{ep} = \frac{F}{3}, \mathbf{a}^f_{ef} = \frac{F}{2}, \mathbf{a}^i_{ep} = F, \mathbf{a}^i_{ef} = F\rangle$, respectively. Finally, Kulcznski1 is of type Non-deterministic (by Theorem 4.4). □

B9. The AMPLE formula (Preserved)

Proof. First,

$$\mathcal{S}usp_\Delta = \left|\frac{\mathbf{a}^i_{ef}}{F} - \frac{\mathbf{a}^i_{ep}}{P}\right| - \left|\frac{\mathbf{a}^f_{ef}}{F} - \frac{\mathbf{a}^f_{ep}}{P}\right|,$$

$$\mathcal{S}usp'_\Delta = \left|\frac{c\mathbf{a}^i_{ef}}{cF} - \frac{\mathbf{a}^i_{ep}}{P}\right| - \left|\frac{c\mathbf{a}^f_{ef}}{cF} - \frac{\mathbf{a}^f_{ep}}{P}\right|$$

We next have $\mathcal{S}usp_\Delta = \mathcal{S}usp'_\Delta$. Finally, AMPLE is of type Preserved. □

B10. The AMPLE2 formula (Preserved)

Proof. we can get the functions $\mathcal{S}usp_\Delta$ and $\mathcal{S}usp'_\Delta$ of the formula of AMPLE2:

$$\mathcal{S}usp_\Delta = \frac{\mathbf{a}^i_{ef}}{F} - \frac{\mathbf{a}^i_{ep}}{P} - \left(\frac{\mathbf{a}^f_{ef}}{F} - \frac{\mathbf{a}^f_{ep}}{P}\right),$$

$$\mathcal{S}usp'_\Delta = \frac{c\mathbf{a}^i_{ef}}{cF} - \frac{\mathbf{a}^i_{ep}}{P} - \left(\frac{c\mathbf{a}^f_{ef}}{cF} - \frac{\mathbf{a}^f_{ep}}{P}\right) = \frac{\mathbf{a}^i_{ef}}{F} - \frac{\mathbf{a}^i_{ep}}{P} - \left(\frac{\mathbf{a}^f_{ef}}{F} - \frac{\mathbf{a}^f_{ep}}{P}\right)$$

We next have $\mathcal{S}usp_\Delta = \mathcal{S}usp'_\Delta$. Finally, AMPLE2 is of type Preserved. □

B11. The Naish1 formula (Preserved)

Proof. First,

$$\mathcal{S}usp_\Delta = \begin{cases} 0 & \text{if } \mathbf{a}^i_{ef} < F \text{ and } \mathbf{a}^f_{ef} < F \\ P - \mathbf{a}^i_{ep} + 1 & \text{if } \mathbf{a}^i_{ef} = F \text{ and } \mathbf{a}^f_{ef} < F \\ -P + \mathbf{a}^f_{ep} - 1 & \text{if } \mathbf{a}^i_{ef} < F \text{ and } \mathbf{a}^f_{ef} = F \\ \mathbf{a}^f_{ep} - \mathbf{a}^i_{ep} & \text{if } \mathbf{a}^i_{ef} = F \text{ and } \mathbf{a}^f_{ef} = F \end{cases},$$

$$\mathcal{S}usp'_\Delta = \begin{cases} 0 & \text{if } c\mathbf{a}^i_{ef} < cF \text{ and } c\mathbf{a}^f_{ef} < cF \\ P - \mathbf{a}^i_{ep} + 1 & \text{if } c\mathbf{a}^i_{ef} = cF \text{ and } c\mathbf{a}^f_{ef} < cF \\ -P + \mathbf{a}^f_{ep} - 1 & \text{if } c\mathbf{a}^i_{ef} < cF \text{ and } c\mathbf{a}^f_{ef} = cF \\ \mathbf{a}^f_{ep} - \mathbf{a}^i_{ep} & \text{if } c\mathbf{a}^i_{ef} = cF \text{ and } c\mathbf{a}^f_{ef} = cF \end{cases}$$

We next have $\mathcal{S}usp_\Delta = \mathcal{S}usp'_\Delta$. Finally, Naish1 is of type Preserved. □

B12. The Naish2 formula (Preserved)

Proof. First,

$$\mathcal{S}usp_\Delta = \mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} - \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1}, \mathcal{S}usp'_\Delta = c\left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef}\right) - \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1}$$

We next have,

$$\mathcal{S}usp_\Delta > 0 \Rightarrow \mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} > \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1}$$
$$\Rightarrow \left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} = 0 \bigwedge \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} < 0\right) \bigvee \left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} > 0\right)$$
$$\Rightarrow c\left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef}\right) > \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} \Rightarrow \mathcal{S}usp'_\Delta > 0$$

$$\mathcal{S}usp_\Delta = 0 \Rightarrow \mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} = \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1}$$
$$\Rightarrow \left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} = 0 \bigwedge \mathbf{a}^i_{ep} - \mathbf{a}^f_{ep} = 0\right)$$
$$\Rightarrow c\left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef}\right) = \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} \Rightarrow \mathcal{S}usp'_\Delta = 0$$

$$\mathcal{S}usp_\Delta < 0 \Rightarrow \mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} < \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} \left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} = 0 \bigwedge \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} > 0\right)$$
$$\bigvee \left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef} < 0\right)$$
$$\Rightarrow c\left(\mathbf{a}^i_{ef} - \mathbf{a}^f_{ef}\right) < \frac{\mathbf{a}^i_{ep} - \mathbf{a}^f_{ep}}{P+1} \Rightarrow \mathcal{S}usp'_\Delta < 0.$$

So Naish2 is of type Preserved (by Theorem 4.1). □

B13. The ER3 group (Preserved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Naish et al., 2011), we take q_e as example, for which we have:

$$\mathcal{S}usp_\Delta = \frac{\mathbf{a}^i_{ef}}{\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}} - \frac{\mathbf{a}^f_{ef}}{\mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}}, \mathcal{S}usp'_\Delta = \frac{c\mathbf{a}^i_{ef}}{c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}} - \frac{c\mathbf{a}^f_{ef}}{c\mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}}$$

We next give the following derivation:

$$\mathcal{S}usp'_\Delta = \frac{c\left(\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(\mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}\right)}{\left(c\mathbf{a}^i_{ef} + \mathbf{a}^i_{ep}\right)\left(c\mathbf{a}^f_{ef} + \mathbf{a}^f_{ep}\right)}$$

$$\mathcal{S}usp_\Delta \Rightarrow \begin{cases} \mathcal{S}usp_\Delta > 0 \Leftrightarrow \mathcal{S}usp'_\Delta > 0 \\ \mathcal{S}usp_\Delta = 0 \Leftrightarrow \mathcal{S}usp'_\Delta = 0 \\ \mathcal{S}usp_\Delta < 0 \Leftrightarrow \mathcal{S}usp'_\Delta < 0 \end{cases}$$

Finally, q_e is of type Preserved (by Theorem 4.1). □

B14. The ER5 group (Preserved)

Proof. Since all the formulas in the group are equivalent in the single-fault scenario (Naish et al., 2011), we take Wong1 as example, for which we have: $\mathcal{S}usp_\Delta = \mathbf{a}^i_{\text{ef}} - \mathbf{a}^f_{\text{ef}}$, $\mathcal{S}usp'_\Delta = c(\mathbf{a}^i_{\text{ef}} - \mathbf{a}^f_{\text{ef}})$. Next, we have $\mathcal{S}usp'_\Delta = c\mathcal{S}usp_\Delta$. So ER5 is of type Preserved(by Corollary 4.1). □

B15. The binary formula (Preserved)

Proof.

$$\mathcal{S}usp_\Delta = \begin{cases} 0 & \text{if } \left(\mathbf{a}^f_{\text{ef}} < F \text{ and } \mathbf{a}^i_{\text{ef}} < F\right) \\ & \text{or } \left(\mathbf{a}^f_{\text{ef}} = F \text{ and } \mathbf{a}^i_{\text{ef}} = F\right) \\ -1 & \text{if } \mathbf{a}^f_{\text{ef}} = F \text{ and } \mathbf{a}^i_{\text{ef}} < F \\ 1 & \text{if } \mathbf{a}^f_{\text{ef}} < F \text{ and } \mathbf{a}^i_{\text{ef}} = F \end{cases} \tag{B.1}$$

$$\mathcal{S}usp'_\Delta = \begin{cases} 0 & \text{if } \left(c\mathbf{a}^f_{\text{ef}} < cF \text{ and } c\mathbf{a}^i_{\text{ef}} < cF\right) \\ & \text{or } \left(c\mathbf{a}^f_{\text{ef}} = cF \text{ and } c\mathbf{a}^i_{\text{ef}} = cF\right) \\ -1 & \text{if } c\mathbf{a}^f_{\text{ef}} = cF \text{ and } c\mathbf{a}^i_{\text{ef}} < cF \\ 1 & \text{if } c\mathbf{a}^f_{\text{ef}} < cF \text{ and } c\mathbf{a}^i_{\text{ef}} = cF \end{cases} \tag{B.1}$$

We next have $\mathcal{S}usp_\Delta = \mathcal{S}usp'_\Delta$. Finally, Binary is of type Preserved (by Corollary 4.1). □

B16. The Fleiss, Cohen, arithmetic mean, and Kulczynski2 formulas (Non-deterministic)

Proof. There formulas are of type Non-deterministic in the single-fault scenario ($\mathbf{a}^f_{\text{ef}} = F$), and a fault example including coverage parameters $\mathbf{a}_{\text{ef}}$, $\mathbf{a}_{\text{nf}}$, $\mathbf{a}_{\text{ep}}$, $\mathbf{a}_{\text{np}}$ are given in previous proofs. Let us clone the fault in each fault example to generate a two-fault version. It is easy to know that the accuracy to locate two faults in the generated multi-fault version, by using each formula, can be either improved or deteriorated. As a result, these formulas are also of type Non-deterministic in the multi-fault scenario (by Theorem 4.4). □

References

Abreu, R., Zoeteweij, P., van Gemud, A.J.C., 2007. On the accuracy of spectrum-based fault localization. In: Proceedings of TAICPART-MUTATION, pp. 89–98.

Abreu, R., Zoeteweij, P., van Gemund, A.J.C., 2006. An evaluation of similarity coefficients for software fault localization. In: Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing (PRDC 2006), pp. 39–46.

Anderberg, M.R., 1988. Cluster analysis for applications. NY Publication: Probability and Mathematical Statistics. Academic Press, New York.

Baah, G.K., Podgurski, A., Harrold, M.J., 2010. Causal inference for statistical fault localization. In: Proceedings of the 19th International Symposium on Software Testing and Analysis (ISSTA 2010), pp. 73–84.

Baudry, B., Fleurey, F., Traon, Y., 2006. Improving test suites for efficient fault localization. In: Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE 2006), pp. 82–91.

Cohen, J., 1960. A coefficient of agreement for nominal scales. In: Educational and Psychological Measurement, pp. 37–46.

Dallmeier, V., Lindig, C., Zeller, A., 2005. Lightweight defect localization for java. In: Proceedings of the 19th European Conference on Object-Oriented Programming (ECOOP 2005), pp. 528–550.

Debroy, V., Wong, W.E., 2009. Insights on fault interference for programs with multiple bugs. In: Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE 2009), pp. 165–174.

Debroy, V., Wong, W.E., 2013. A consensus-based strategy to improve the quality of fault localization. In: Software Practice and Experience (SPE 2013), pp. 989–1011.

Dice, L.R., 1945. Measures of the amount of ecologic association between species. Ecology 297–302.

Do, H., Elbaum, S., Rothermel, G., 2005. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. In: Empirical Software Engineering (ESE 2005), pp. 405–435.

Duarte, J.M., dos Santos, J.B., Melo, L.C., 1999. Comparison of similarity coefficients based on RAPD markers in the common bean. Genet. Mol. Biol. 427–432.

Everitt, B.S., 1978. Graphical Techniques for Multivariate Data. North-Holland, New York.

Fleiss, J.L., 1965. Estimating the accuracy of dichotomous judgments. Psychometrika 469–479.

Fu, C., Ryder, B.G., 2007. Exception-chain analysis: revealing exception handling architecture in java server applications. In: Proceedings of the 29th International Conference on Software Engineering (ICSE 2007), pp. 230–239.

Gao, Y., Zhang, Z., Zhang, L., Gong, C., Zheng, Z., 2013. A theoretical study: the impact of cloning failed test cases on the effectiveness of fault localization. In: Proceedings of the Symposium on Engineering Test Harness 2013 (TSE-TH 2013), in conjunction with the 13th International Conference on Quality Software (QSIC 2013), pp. 288–291.

Gong, C., Zheng, Z., Li, W., Hao, P., 2012. Effects of class imbalance in test suites: an empirical study of spectrum-based fault localization. In: Proceedings of Computer Software and Applications Conference Workshops (COMPSACW 2012), pp. 470–475.

Goodman, L.A., Kruskal, W.H., 1954. Measures of association for cross classification. J. Am. Stat. Assoc. 732–764.

Gore, R., Reynolds Jr., P., 2012. Reducing confounding bias in predicate-level statistical debugging metrics. In: Proceedings of the 34th International Conference on Software Engineering (ICSE 2012), pp. 463–473.

Hamming, R.W., 1950. Error detecting and error correcting codes. Bell Syst. Tech. J. 147–160.

Hao, B., Ying, Y., Zhang, L., Zhao, W., Mei, H., Sun, J., 2005. A similarity-aware approach to testing based fault localization. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), pp. 291–294.

Harrold, M., Rothermel, G., Wu, R., Yi, L., 1998. An empirical investigation of program spectra. In: Proceedings of the 1st ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, pp. 83–90.

Jaccard, P., 1901. Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin del la Socit Vaudoise des Sciences Naturelles 547–579.

Japkowicz, N., Shaju, S., 2002. The class imbalance problem: A systematic study. In: Intelligent Data Analysis, pp. 429–449.

Jeffrey, D., Gupta, N., Gupta, R., 2008. Fault localization using value replacement. In: Proceedings of the 2008 International Symposium on Software Testing and Analysis (ISSTA 2008), pp. 167–178.

Jones, J.A., Harrold, M.J., 2005. Empirical evaluation of the tarantula automatic fault--localization technique. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), pp. 273–282.

Jones, J.A., Harrold, M.J., Stasko, J., 2002. Visualization of test information to assist fault localization. In: Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), pp. 467–477.

Kim, D., Nam, J., Song, J., Kim, S., 2013. Automatic patch generation learned from human-written patches. In: Proceedings of the 2013 International Conference on Software Engineering (ICSE 2013), pp. 802–811.

Kochhar, P.S., Xia, X., Lo, D., Li, S., 2016. Practitioners' expectations on automated fault localization. In: Proceedings of the 25th International Symposium on Software Testing and Analysis, (ISSTA 2016), pp. 165–176.

Krause, E.F., 1973. Taxicab geometry. Mathematics Teacher 695–706.

Le, T.D.B., Lo, D., Goues, C.L., Grunske, L., 2016. A learning-to-rank based fault localization approach using likely invariants. In: Proceedings of the 25th International Symposium on Software Testing and Analysis, (ISSTA 2016), pp. 177–188.

Lee, H., Naish, L., Ramamohanarao, K., 2009. Study of the relationship of bug consistency with respect to performance of spectra metrics. In: Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2009), pp. 501–508.

Li, H., Liu, Y., Zhang, Z., Liu, J., 2014. Program structure aware fault localization. In: Proceedings of International Worshop on Innovative Software Development Methodologies and Practices (INNOSWDEV 2014), in conjunction with the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014), pp. 40–48.

Liblit, B., Aiken, A., Naik, M., Zheng, A.X., 2015. Scalable statistical bug isolation. In: Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (PLDI 2015), pp. 15–26.

Lin, C., Li, Y., 2014. Rate-based queueing simulation model of open source software debugging activities. In: IEEE Transactions on Software Engineering (TSE 2014), pp. 1075–1099.

Liu, C., Fei, L., Yan, X., Han, J., Midkiff, S.P., 2006. Statistical debugging: A hypothesis testing-based approach. In: IEEE Transactions on Software Engineering (TSE 2006), pp. 831–848.

Lourenco, F., Lobo, V., Bacão, F., Binary-based similarity measures for categorical data and their application in self-organizing maps, (2004).

Lucia, L., Lo, D., Xia, X., 2014. Fusion fault localizers. In: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE 2014), pp. 127–138.

Masri, W., Assi, R., 2014. Prevalence of coincidental correctness and mitigation of its impact on fault localization. In: ACM Transactions on Software Engineering Methodology (TOSEM 2014), p. 28. Article 8.

MathWorld, W., The web's most extensive mathematics resource, 2010Access on: http://mathworld.wolfram.com.

Maxwell, A.E., Pilliner, A.E., 1968. Deriving coefficients of reliability and agreement for ratings. Br. J. Math. Stat. Psychol. 105–116.

Mayer, W., Stumptner, M., Gupta, N., Gupta, R., 2007. Abstract interpretation of programs for model-based debugging. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 471–476.

Meyer, A.d.S., Garcia, A.A.F., de Souza, A.P., de Souza Jr., C.L., 2004. Comparison of similarity coefficients used for cluster analysis with dominant markers in maize (zea mays l). In: Genetics and Molecular Biology, pp. 83–91.

Naish, L., Lee, H.J., Ramamohanarao, K., 2011. A model for spectra-based software diagnosis. In: ACM Transactions on Software Engineering Methodology (TOSEM 2011), p. 32. Article 11.

Naish, L., Neelofar, Ramamohanarao, K., 2015. Multiple bug spectral fault localization using genetic programming. In: Asia Wine Service & Education Centre (AWSEC2015), pp. 11–17.

Ochiai, A., 1957. Zoogeographic studies on the solenoid fishes found in japan and its neighbouring regions. Bull. Jpn. Soc. Fish Sci. 526–530.

Software-artifact Infrastructure Repository, 2005University Of Nebraska-Lincoln. Access on: http://sir.unl.edu/php/index.php.

Reps, T., Ball, T., Das, M., Larus, J., 1997. The use of program profiling for software maintenance with applications to the year 2000 problem. In: Proceedings of the 6th European Software Engineering Conference held jointly with the 5th ACMSIGSOFT International Symposium on Foundations of Software Engineering (FSE 1997). 432–C449.

Rogers, D.J., Tanimoto, T.T., 1960. A computer program for classifying plants. Science 1115–1118.

Rogot, E., Goldberg, I.D., 1966. A proposed index for measuring agreement in test-retest studies. J. Chronic. Dis. 991–1006.

Russel, P.F., Rao, T.R., 1940. On habitat and association of species of anopheline larvae in south-eastern madras. J. Malarial Inst. India 153–178.

Sanchez, A. G., 2007. Automatic Error Detection Techniques based on Dynamic Invariants. Master's Thesis.

Santelices, R., Jones, J.A., Yu, Y., Harrold, M.J., 2009. Lightweight fault-localization using multiple coverage types. In: Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), pp. 56–66.

Scott, W.A., 1955. Reliability of content analysis: the case of nominal scale coding. Public Opin. Q. 321–325.

Song, S., 2014. Estimating the effectiveness of spectrum-based fault localization. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering(FSE-2014), pp. 814–816.

Tang, C.M., Chan, W.K., Yu, Y.T., Zhang, Z., 2017. Accuracy graphs of spectrum-based fault localization formulas. IEEE Transactions on Reliability (TR 2017). To appear.

Wong, W.E., Debroy, V., Choi, B., 2010. A family of code coverage-based heuristics for effective fault localization. J. Syst. Software (JSS 2010) 188–208.

Wong, W.E., Debroy, V., Gao, R., Li, Y., 2014. The DStar method for effective software fault localization. In: IEEE Transactions on Reliability (TR 2014), pp. 290–308.

Wong, W.E., Debroy, V., Golden, R., Xu, X., Thuraisingham, B., 2009. Effective software fault localization using an RBF neural network. In: IEEE Transactions on Reliability (TR 2009), pp. 149–169.

Wong, W.E., Debroy, V., Xu, D., 2012. Towards better fault localization: A crosstab-based statistical approach. In: IEEE Transactions on System Man and Cybernetics, Part C: Applications and Reviews (2012), pp. 378–396.

Wong, W.E., Qi, Y., 2009. BP neural network-based effective fault localization. In: International Journal of Software Engineering and Knowledge Engineering (IJSEKE 2009), pp. 573–597.

Wong, W.E., Qi, Y., Zhao, L., Cai, K., 2007. Effective fault localization using code coverage. In: Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), pp. 449–456.

Wong, W.E., Shi, Y., Qi, Y., Golden, R., 2008a. Using an RBF neural network to locate program bugs. In: Proceedings of the 19th International Symposium on Software Reliability Engineering (ISSRE 2008), pp. 27–36.

Wong, W.E., Wei, T., Qi, Y., Zhao, L., 2008b. A crosstab-based statistical method for effective fault localization. In: Proceedings of the 1st International Conference on Software Testing, Verification, and Validation (ICST 2008), pp. 42–51.

Xie, X., Chen, T.Y., Kuo, F., Xu, B., 2013a. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization. In: ACM Transactions on Software of Engineer and Methodology (TOSEM 2013), p. 40. Article 31.

Xie, X., Wong, W.E., Chen, T.Y., Xu, B.W., 2008b. Metamorphic slice: An application in spectrum-based fault localization. In: ACM Information and Software Technology (IST 2013), pp. 866–879.

Xu, J., Zhang, Z., Chan, W.K., Tse, T.H., Li, S., 2013. A general noise-reduction framework for fault localization of java programs. In: Information and Software Technology (IST 2013), pp. 880–896.

Xuan, J., Monperrus, M., 2014. Test case purification for improving fault localization. In: 22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2014), pp. 52–63.

Yoo, S., Harman, M., Clark, D., 2013. Fault localization prioritization: Comparing information-theoretic and coverage-based approaches. In: ACM Transactions on Software Engineering Methodology (TOSEM 2013), p. 29. Artical 19.

Yu, Y., Jones, J., Harrold, M., 2008. An empirical study of the effects of test-suite reduction on fault localization. In: Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), pp. 201–210.

Yu, Z., Bai, C., Cai, K.Y., 2015. Does the failed test execute a single or multiple faults? an approach to classifying failed tests. In: 37th IEEE/ACM International Conference on Software Engineering (ICSE 2015), pp. 924–935.

Yuan, D., Mai, H., Xiong, W., Tan, L., Pasupathy, S., 2010. Sherlog: error diagnosis by connecting clues from run-time logs. In: Proceedings of the 15th Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2010), pp. 143–154.

Zhang, X., He, H., Gupta, N., Gupta, R., 2005. Experimental evaluation of using dynamic slices for fault location. In: Proceedings of the sixth International symposium on Automated Analysis-driven Debugging (AADEBUG 2005), pp. 33–42.

Zhang, Z., Chan, W.K., Tse, T.H., Jiang, B., Wang, X., 2009. Capturing propagation of infected program states. In: Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE 2009), pp. 43–52.

Zhang, Z., Chan, W.K., Tse, T.H., Yu, Y.T., Hu, P., 2011. Non-parametric statistical fault localization. J. Syst. Software (JSS 2011) 885–905.

Zhang, Z., Jiang, B., Chan, W.K., Tse, T.H., 2008. Debugging through evaluation sequences: A controlled experimental study. In: Proceedings of the 32nd Annual International Computer Software and Applications Conference (COMPSAC 2008), pp. 128–135.

Long Zhang is a PhD candidate at the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. He obtained his bachelor degree in Computer Science and Technology from Hefei University of Technology. His research interests include program debugging and program verification.

Lanfei Yan is an MSc student at the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. He obtained his bachelor degree from Wuhan University. His research interests are software testing and program parallelization.

Zhenyu Zhang is an associate professor at the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. He obtained his PhD degree from The University of Hong Kong, and master and bachelor degree from Tsinghua University. He is with the Institute for Software Research, University of California, Irvine, in the years 2016 and 2017. His research interests are debugging and testing for software and systems as well as the reliability issues of web-based services and cloud-based systems. He has published research results in venues such as Computer, TSE, TSC, T-Rel, ICSE, FSE, ASE, and WWW.

Jian Zhang is a professor at the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. His current research interests are program analysis, combinatorial testing, and other topics related to testing and analysis. He has published research results in venues such as ICSE, ISSTA, JSS, and so on. He is an editorial board member of Chinese Journal of Computers, Journal of Computer Science and Technology, Frontiers of Computer Science, and Journal of Frontiers of Computer Science and Technology.

W.K. Chan is an associate professor at City University of Hong Kong. His current main research interest is program analysis and testing for concurrent software and systems. He is the Special Issues Editor of Journal of Systems and Software. He has published more than 100 papers in venues such as TOSEM, TSE, TPDS, TSC, T-Rel, CACM, Computer, ICSE, FSE, ISSTA, ASE, WWW, ICWS, and ICDCS.

Zheng Zheng received his PhD degree in computer software and theory from Chinese Academy of Science. He is currently an associate professor and a PhD supervisor at Beihang University. Since 2014, he was with Department of Electrical and Computer Engineering at Duke University, working as a research scholar collaborating Kishor Trivedi. His research interests include software fault localization and software dependability modeling.