ICIC Express Letters
Volume **11**, Number **2**, February **2017**

ICIC International ©2017 ISSN 1881-803X
pp. **277–284**

NEW SUSPICIOUSNESS METRICS TO IMPROVE FAULT RANKING FOR SOFTWARE FAULT LOCALIZATION

Wanchang Jiang[1,2], Jiadong Ren[1] and Yuan Huang[1]

[1]College of Information Science and Engineering
Yanshan University
No. 438, Hebei Avenue, Qinhuangdao 066004, P. R. China
jwchang84@163.com; jdren@ysu.edu.cn; 757918272@qq.com

[2]School of Information Engineering
Northeast Dianli University
No. 169, Changchun Road, Chuanying Dist., Jilin 132012, P. R. China

Received July 2016; accepted October 2016

Abstract. *To improve the reliability of software, it is necessary to investigate how to design suspiciousness computation metrics to increase suspiciousness fault ranking for fault localization. In this paper, different forms and numbers of fractions based on failed execution spectrum and successful non-execution spectrum are designed to balance the influence of each decisive spectrum on the suspiciousness, and three new suspiciousness metrics EF_0NP_4, $EF_{03}NP_3$ and $EF_{034}NP_3$ are proposed to compute suspiciousness of each statement to be the fault. In addition, using the proposed conception of execution trace self-information, weights are designed to weigh each fraction in the above metrics, the influence of two decisive spectra on the suspiciousness is dynamically adjusted, and then three weighted suspiciousness metrics EF_0NP_4W, $EF_{03}NP_3W$ and $EF_{034}NP_3W$ are proposed correspondingly. Then, a suspiciousness metric-based statement ranking algorithm is designed to apply our six proposed metrics to getting suspiciousness ranking of statements to be the fault. Experiments on the Software-artifact Infrastructure Repository show that compared with other metrics, our metrics (especially our weighted metrics) help improve the fault ranking about 15.7% (16.3%) on average, and up to 21.6% (22.4%) in specific case, which is largely independent of types of test suite. As a result, fewer statements need to be examined until the fault is found and the efficiency of fault localization is improved. Furthermore, the ineffectiveness of failed execution spectrum-based metrics can be solved.*

Keywords: Program spectra, Execution trace self-information, Suspiciousness metric, Software fault localization

1. **Introduction.**

Software testing and debugging techniques are utilized to improve the software reliability, which is important for software development. Since the software testing is of high expenditure, it is impossible to test software exhaustively with all test cases. Therefore, a subset of test cases is selected to reduce the cost of software testing [1, 2]. To further improve the effectiveness of software testing, test cases should be scheduled and prioritized [3].

The artificial factor in software system makes it necessary to solve the problem of identifying fault as soon as possible with limited resources. An approach is proposed in [4] to identify the influential functions in complex software network for fault localization. To obtain suspiciousness of statements to be the fault, suspiciousness metric-based fault localization methods are designed by using program spectra [5]. However, most statements are executed in few of failed executions, and failed execution spectrum-based suspiciousness metrics of Wong1 [6], Kulczynski1 [5], Ochiai and Tarantula [7] cannot work when the number of failed executions decreases to zero, which affects the stability of these metrics.

In view of this case, other program spectra are also considered to design suspiciousness metrics, such as Wong2 [6], Ample [7], Sokal and Hamann [5]. However, the fault cannot be ranked well because it is unreasonable to claim that each decisive spectrum has the same effect on the suspiciousness. An approach is proposed, which combines multiple ranking metrics for effective fault localization [8]. However, each metric in the approach is not improved at all.

Therefore, to increase the suspiciousness fault ranking and solve the ineffectiveness of failed execution spectrum-based metric, three new suspiciousness metrics $\mathrm{EF_0NP_4}$, $\mathrm{EF_{03}NP_3}$ and $\mathrm{EF_{034}NP_3}$ are proposed by constructing different fractions on the basis of failed execution spectrum and successful non-execution spectrum to balance the influence of each decisive spectrum on the suspiciousness. In addition, to further improve the effectiveness of suspiciousness metric for statement ranking, execution trace self-information is proposed to obtain information quantity about each type of execution. On the basis of the above metrics, three weighted suspiciousness metrics $\mathrm{EF_0NP_4W}$, $\mathrm{EF_{03}NP_3W}$ and $\mathrm{EF_{034}NP_3W}$ are designed respectively by using the proposed execution trace self-information to dynamically reflect different influences of each fraction on suspiciousness. And a suspiciousness metric-based statement ranking algorithm is designed to apply our six proposed suspiciousness metrics to obtaining suspiciousness ranking of statements to be the fault. The fault ranking can be increased with different types of test suite and fewer statements need to be examined until the fault is located.

The organization of this paper is as follows. Section 2 introduces the basic concepts. In Section 3, three new suspiciousness metrics are proposed. In Section 4, based on execution trace self-information, weighted suspiciousness metrics are designed. A suspiciousness metric-based statement ranking algorithm is presented in Section 5. Section 6 discusses the experiments on a typical software-artifact. Conclusions are given in the last section.

2. **Preliminaries.**

In this section, concepts of execution trace spectra and program spectra of program running are introduced.

Let the set of statements $\{S_1, S_2, \ldots, S_N\}$ denote a program written in a programming language. As a basic element of the program, a statement can be a simple statement or a compound one. To locate a fault, a program should be executed with a test suite of test cases $\{T_1, T_2, \ldots, T_M\}$, and the execution traces of each program running are collected.

Definition 2.1. *Execution trace spectra. The execution trace spectra are extracted as a matrix, with column denoting statement S_i and row denoting test case T_j, and cell e_{ji} indicating whether S_i is executed or not with T_j. If S_i is executed with T_j, e_{ji} is equal to 1; otherwise, its value is zero. The last column r_j denotes the execution result, namely 1 stands for a failed execution and 0 for a successful execution.*

Definition 2.2. *Program spectra a_{ef}, a_{ep}, a_{nf} and a_{np}. For each program spectrum, the first subscript 'e' or 'n' indicates whether S_i is executed or not, and the second subscript 'p' or 'f' indicates whether the corresponding test case is a passed or failed one. Failed execution spectrum a_{ef} is defined as the number of failed executions covering S_i as follows.*

$$a_{ef} = \sum_{j=1}^{M} \{e_{ji} | e_{ji} = 1 \wedge r_j = 1\} \tag{1}$$

Similarly, successful execution spectrum a_{ep}, failed non-execution spectrum a_{nf} and successful non-execution spectrum a_{np} are defined.

3. **Suspiciousness Metrics Based on Failed Execution Spectrum and Successful Non-Execution Spectrum.**

Compared with other statements, the fault statement would be executed in more failed executions and fewer successful ones. Different a_{ef}-based fractions and a_{np}-based fractions are designed by using four program spectra to exert more influence of a_{ef} on suspiciousness. Thus, three new suspiciousness metrics $\mathrm{EF_0NP_4}$, $\mathrm{EF_{03}NP_3}$ and $\mathrm{EF_{034}NP_3}$ are proposed, based on the decisive factors of failed execution spectrum and successful non-execution spectrum.

A new a_{ef} and a_{np}-based suspiciousness metric $\mathrm{EF_0NP_4}$ is proposed, which consists of a_{ef} and one a_{np}-based fraction. The numerator and denominator of the fraction are a_{np} and the sum of four spectra respectively. As inversely proportional factors in $\mathrm{EF_0NP_4}$, a_{ep} and a_{nf} are taken into account to include in the denominator. And a_{ef} and a_{np} are included in the denominator to further reduce the influence of a_{np} on the suspiciousness.

$$\mathrm{EF_0NP_4} = a_{ef} + \frac{a_{np}}{a_{np} + a_{ef} + a_{ep} + a_{nf}} \tag{2}$$

where 'EF' in $\mathrm{EF_0NP_4}$ denotes a_{ef}-based expression, and '0' indicates that there is no a_{ef}-based fraction. 'NP' denotes a_{np}-based expression, and '4' indicates the number of spectra in the denominator of a_{np}-based fraction.

Proposition 3.1. *When a_{ef} is nonzero, a_{ef} is the factor determining the suspiciousness value. Otherwise, only a_{np}-based fraction plays a role in the metric.*

Proof: The denominator of a_{np}-based fraction equals the number of test cases, that is $a_{ep} + a_{nf} + a_{ef} + a_{np} = M$. In addition, the numerator a_{np} is considered as a component of the denominator; thus, a_{np}-based fraction is less than 1. When a_{ef} is nonzero, that is to say, a_{ef} is at least 1, a_{ef} is larger than the value of a_{np}-based fraction. As a result, a_{ef} plays the decisive role in the computing suspiciousness. With the help of a_{np}-based fraction, the metric can work when a_{ef} decreases to zero, only a_{np}-based fraction plays a role in this case.

When the statement is covered by failed executions, the event gives much information about the fault and increases the possibility of statement to be fault. To increase the importance of a_{ef}, one a_{ef}-based fraction is added besides a_{ef}. And the denominator of the a_{np}-based fraction is reduced to adjust the influence of a_{np} on suspiciousness. Based on the motivation, a_{ef} and the a_{ef}-based fraction are considered as the influence of a_{ef} on suspiciousness, one a_{np}-based fraction is introduced to reflect the influence of a_{np} on the result, and a novel suspiciousness computation metric $\mathrm{EF_{03}NP_3}$ is designed.

$$\mathrm{EF_{03}NP_3} = a_{ef} + \frac{a_{np}}{a_{ef} + a_{ep} + a_{np}} + \frac{a_{np}}{a_{np} + a_{ep} + a_{nf}} \tag{3}$$

The inversely proportional factors in $\mathrm{EF_{03}NP_3}$, a_{ep} and a_{nf} are included in the denominator of a_{ef}-based fraction. Moreover, a_{ef} is also included to decrease the effect of a_{ef}, so the sum of a_{ef}, a_{ep} and a_{nf} is considered as the denominator. Just as a_{ef}-based fraction, the sum of a_{np}, a_{ep} and a_{nf} is used as the denominator of a_{np}-based fraction. When a_{ef} becomes zero, the suspiciousness depends only on a_{np}-based fraction; otherwise, a_{ef} plays a decisive role in computing suspiciousness. On the basis of metric $\mathrm{EF_{03}NP_3}$, another a_{ef}-based fraction is added to further increase the importance of a_{ef}, and a new suspiciousness metric $\mathrm{EF_{034}NP_3}$ is proposed as follows.

$$\mathrm{EF_{034}NP_3} = a_{ef} + \frac{a_{ef}}{a_{ef} + a_{ep} + a_{nf}} + \frac{a_{ef}}{a_{ef} + a_{np} + a_{ep} + a_{nf}} + \frac{a_{np}}{a_{np} + a_{ep} + a_{nf}} \tag{4}$$

a_{ef} and two a_{ef}-based fractions are considered to improve the influence of a_{ef}, the decisive factor of the suspiciousness. With the a_{np}-based fraction, a_{np} has a certain effect on the suspiciousness result.

4. Weighted Suspiciousness Metrics Based on Execution Trace Self-Information.

To obtain information quantity about each type of execution, the conception of execution trace self-information is proposed. On the basis of $\mathrm{EF_0NP_4}$, $\mathrm{EF_{03}NP_3}$ and $\mathrm{EF_{034}NP_3}$, weights are designed for each fraction by using execution trace self-information in order to reflect different influences of each fraction on suspiciousness, and thus three weighted suspiciousness metrics $\mathrm{EF_0NP_4W}$, $\mathrm{EF_{03}NP_3W}$ and $\mathrm{EF_{034}NP_3W}$ are proposed respectively. On the basis of $\mathrm{EF_0NP_4}$, a weighted suspiciousness metric $\mathrm{EF_0NP_4W}$ is designed with execution trace self-information as follows

$$\mathrm{EF_{034}NP_3W} = a_{ef} + \omega_{np} \cdot \frac{a_{np}}{a_{np} + a_{ef} + a_{ep} + a_{nf}} \tag{5}$$

where ω_{np} is the weight of the a_{np}-based fraction.

$$\omega_{np} = \frac{h_{np}}{h_{np} + h_{ef} + h_{ep} + h_{nf}} \tag{6}$$

The structure of ω_{np} is consistent with that of the corresponding a_{np}-based fraction which is weighed as a whole. ω_{np} can be obtained by using execution trace self-information h_{ef}, h_{ep}, h_{np} and h_{nf}, which are proposed to get the information quantity of each type execution for one statement. Take failed execution trace self-information h_{ef} of S_i as an example, and h_{ef} is proposed as

$$h_{ef} = -\mathrm{P}(e_f(S_i)) \log(\mathrm{P}(e_f(S_i))) \tag{7}$$

wherein $\mathrm{P}(e_f(S_i))$ is the probability of S_i executed with failed test cases, and $e_f(S_i)$ is the event that S_i emerges in failed executions. $\mathrm{P}(e_f(S_i))$ can be obtained by using program spectra, the number of failed test cases f and the number of passed test cases p.

$$\mathrm{P}(e_f(S_i)) = \frac{a_{ef}}{f + p} \tag{8}$$

Similarly, successful execution trace self-information h_{ep}, failed non-execution trace self-information h_{nf} and successful non-execution trace self-information h_{np} can be computed. Since each parameter has the same sign, ω_{np} will be less than 1.

Just like $\mathrm{EF_0NP_4W}$, a weighted suspiciousness metric $\mathrm{EF_{03}NP_3W}$ is proposed on the basis of $\mathrm{EF_{03}NP_3}$. h_{ef}, h_{ep}, and h_{nf} are utilized for determining the weight of the a_{ef}-based fraction, and the weight is similarly constructed for the a_{np}-based fraction. The influence of a_{ef} and a_{np} on the suspiciousness is dynamically reflected.

$$\begin{aligned} \mathrm{EF_{03}NP_3W} = a_{ef} &+ \frac{h_{np}}{h_{ef} + h_{ep} + h_{np}} \cdot \frac{a_{np}}{a_{ef} + a_{ep} + a_{np}} \\ &+ \frac{h_{np}}{h_{np} + h_{ep} + h_{nf}} \cdot \frac{a_{np}}{a_{np} + a_{ep} + a_{nf}} \end{aligned} \tag{9}$$

An attempt of changing importance of each fraction in $\mathrm{EF_{034}NP_3}$ is made, three weights are introduced based on execution trace self-information to weigh two a_{ef}-based fractions and the a_{np}-based fraction, and a weighted suspiciousness metric $\mathrm{EF_{034}NP_3W}$ is designed.

$$\begin{aligned} \mathrm{EF_{034}NP_3W} = a_{ef} &+ \frac{h_{ef}}{h_{ef} + h_{ep} + h_{nf}} \cdot \frac{a_{ef}}{a_{ef} + a_{ep} + a_{nf}} \\ &+ \frac{h_{ef}}{h_{ef} + h_{np} + h_{ep} + h_{nf}} \cdot \frac{a_{ef}}{a_{ef} + a_{np} + a_{ep} + a_{nf}} \\ &+ \frac{h_{np}}{h_{np} + h_{ep} + h_{nf}} \cdot \frac{a_{np}}{a_{np} + a_{ep} + a_{nf}} \end{aligned} \tag{10}$$

As shown in the formula, the influence of a_{ef} and a_{np} on the suspiciousness result is further adjusted by weights based on execution trace self-information.

5. **Suspiciousness Metric-Based Statement Ranking Algorithm for Fault Localization.** A suspiciousness metric-based statement ranking algorithm is designed to apply above proposed suspiciousness metrics to ranking statements for fault localization. For one fault version of a given program, two main steps should be done to realize the suspiciousness metric-based statement ranking. First of all, execution traces are collected to extract the execution trace spectra for a test suite. Then the suspiciousness of statements

can be obtained by using our suspiciousness metrics which are obtained with program spectra and execution trace self-information.

The suspiciousness metric-based statement ranking algorithm is presented as follows.

Algorithm 1: Suspiciousness metric-based statement ranking algorithm

```
Input: fault versions $\{V_k\}$, test suite $\{T_i\}$
Output: sequences $\{S_{i_k}\}_{\mathrm{M}}$ and $\{S_{i_m}\}_{\mathrm{L}}$ of each metric for each version $\{V_k\}$
 1. For each fault version $V_k$
 2.   For each test case $T_j$
 3.     Run program with the test case
 4.     Collect execution trace
 5.     Compare output with the corresponding expected output, get $r_j$
 6.   End For
 7. End For
 8. For each fault version $V_k$ with the test suite
 9.   Extract execution trace spectra $\{e_{ji}\}$
10. End For
11. For each fault version $V_k$
12.   For each statement $S_i$
13.     Compute program spectra $a_{ef}$, $a_{ep}$, $a_{nf}$ and $a_{np}$
14.     Compute execution trace self-information $h_{ef}$, $h_{ep}$, $h_{nf}$ and $h_{np}$
15.     For each metric
16.       Compute the suspiciousness of $S_i$ by the metric
17.     End For
18.   End For
19.   For each metric
20.     Rank statements by using medium line strategy and output sequence $\{S_{i_k}\}_{\mathrm{M}}$
21.     Rank statements by using last line strategy and output sequence $\{S_{i_m}\}_{\mathrm{L}}$
22.   End For
23. End For
```

With our metrics, the statement ranking algorithm is used for fault localization of multiple fault versions of the same program at one time. For each fault version, statements are ranked in descending sequence with one metric. However, several statements may have the same suspiciousness. The medium-line (ML) and last-line (LL) strategies [9] will be used to solve this problem, where the average ranking position of the middle line is used as the ranking of these statements in ML strategy and the last position is used as the ranking of these statements in LL strategy. As a result, two statement sequences $\{S_{i_k}\}_{\mathrm{M}}$ and $\{S_{i_m}\}_{\mathrm{L}}$ are obtained respectively for each metric. Using any one of sequences, top l ranking statements should be examined until the fault is located.

6. **Experiment.**

As the typical software-related artifact for fault localization techniques, the Software-artifact Infrastructure Repository (SIR) [10] is used to conduct experiments to respectively compare our proposed suspiciousness metrics with our weighted metrics, and the proposed suspiciousness metrics with other metrics of Kulczynskil (KUL for short), Sokal (SOK), Hamann (HAN) and Tarantula (TA) for statement ranking.

6.1. **Experimental environment.**

The aircraft collision avoidance system program "t-cas" in SIR is used. Since fault of some versions corresponds to multiple statements or losing statements do not adapt to the experiment, 35 fault versions are chosen from the 41 versions. To verify the stable performance of our metrics for fault localization, test suites of two types "bigrand" and "bigcov" are selected, and four suites of each type are used.

Then the metric-based statement ranking algorithm is realized by Java programming language and then run under Fedora Core system.

6.2. **Experimental results.** With the size 80, test suites of "bigcov" type are generated to achieve branch coverage. Using four test suites of this type, the average suspiciousness ranking of the fault of each version under ML strategy and LL strategy is shown respectively in Figure 1 and Figure 2.

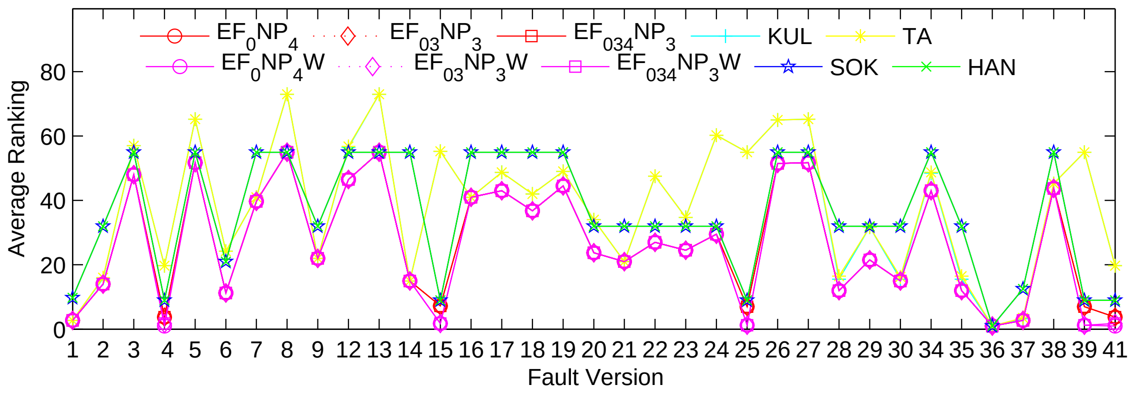

FIGURE 1. The average ranking of the fault based on "bigcov" under ML strategy

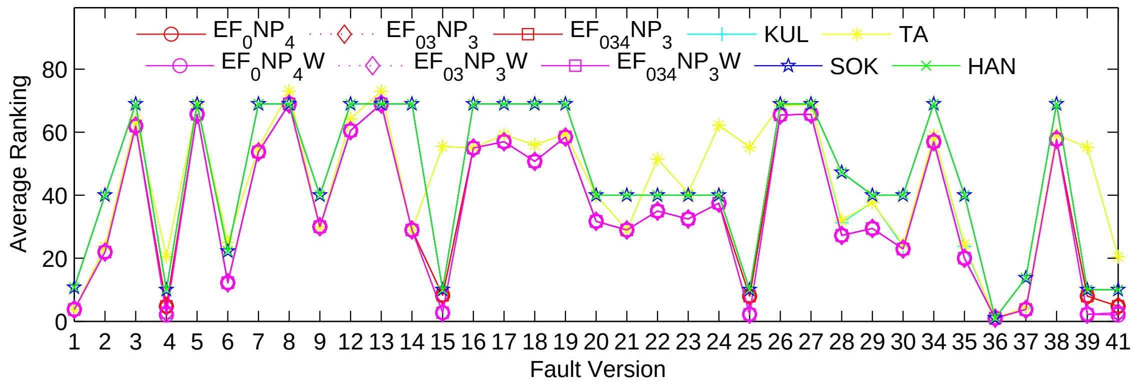

FIGURE 2. The average ranking of the fault based on "bigcov" under LL strategy

As shown in Figure 1, compared with KUL, TA, SOK and HAN, our proposed metrics can rank fault better under ML strategy. Our metrics improve the fault ranking about 14.7% on average over the other metrics, and up to 16.2% in specific case. The performance of KUL and TA is not stable, because they are ineffective for versions 5, 8, 13, 15, 24 and so on when failed execution spectrum is zero. In contrast, our metrics can even work well in this case, and gain an increase of 18.5% and 21.9% for versions 5 and 8.

Under LL strategy, the last number is given as the ranking for juxtaposing statements in the worst case. Although the ranking slightly decreases on the whole, our metrics even perform better than other metrics for such as versions 4, 6, 12, 15 and 17. Taking version 4 as an example, in comparison with KUL and SOK, our metrics gain an average increase of 23.1% and 8.7% respectively. Furthermore, the weighted metric of EF_0NP_4W increases the ranking 25.3% and 10.9% respectively. Test suite of "bigrand" is generated randomly, which has the same size as that of "bigcov". As shown in Figure 3 and Figure 4, the average ranking of the fault under ML strategy and LL strategy respectively is obtained on the basis of four test suites of this type.

As shown in Figure 3, compared with the performance with test suites of "bigcov" under ML strategy, the performance with "bigrand" suites is reduced. Using EF_0NP_4,

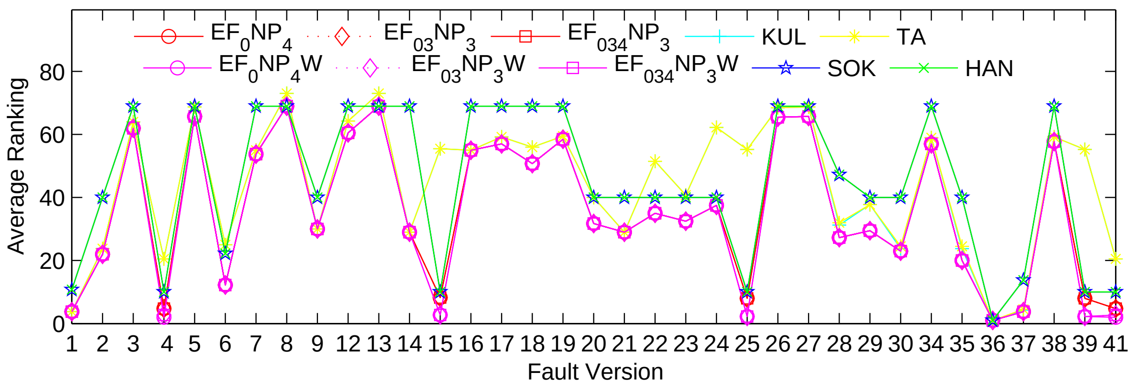

FIGURE 3. The average ranking of the fault based on "bigrand" under ML strategy

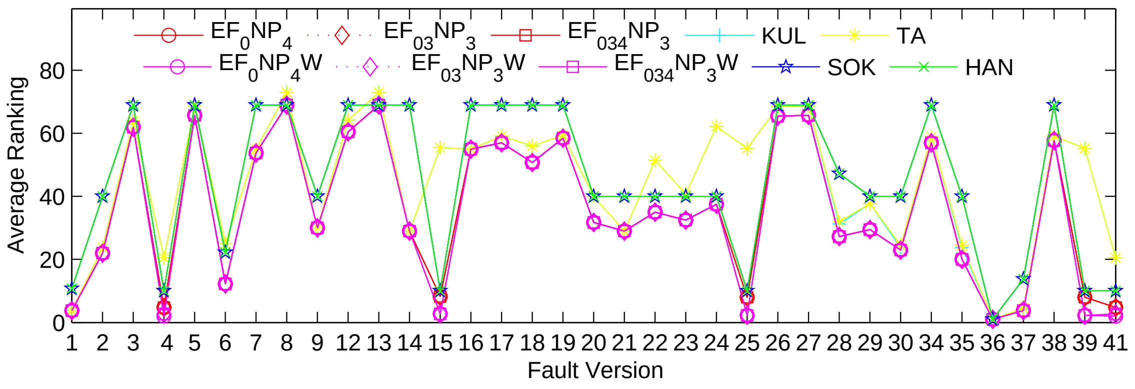

FIGURE 4. The average ranking of the fault based on "bigrand" under LL strategy

$EF_{03}NP_3$ and $EF_{034}NP_3$ makes fault rank better than other metrics KUL, TA, SOK and HAN 20.9%, 21%, 11.1% and 11.1% respectively. In addition, our weighted metrics EF_0NP_4W, $EF_{03}NP_3W$ and $EF_{034}NP_3W$ outperform corresponding non-weighted metrics to some extent, about 1.47%, 1.65% and 1.18% respectively.

Take two of our metrics as example. Under LL strategy, in comparison with KUL, TA, SOK and HAN, our metric of EF_0NP_4 gains an average increase of 17.2%, 17.3%, 11.1% and 11.1% respectively, and our weighted metric of EF_0NP_4W gains an average increase of 18.6%, 18.7%, 12.6% and 12.6% respectively.

In conclusion, with test suites of types "bigrand" and "bigcov" under the medium-line and last-line strategies, our metrics help statement ranking algorithm increase fault ranking about 15.7% on average in comparison with other metrics, and up to 21.6% in specific case. Furthermore, our weighted metrics gain an increase of 16.3% on average, and 22.4% for the best case. As a result, fewer statements need to be examined until fault is located.

7. **Conclusions.**

We propose three suspiciousness metrics EF_0NP_4, $EF_{03}NP_3$ and $EF_{034}NP_3$ on the basis of a_{ef} and a_{np} for computing the suspiciousness of statement to be fault. In addition, to adjust the influence of each fraction in each metric on the suspiciousness, we design weights for each fraction by using the proposed conception of execution trace self-information, and then three weighted metrics EF_0NP_4W, $EF_{03}NP_3W$ and $EF_{034}NP_3W$ are designed respectively. A suspiciousness metric-based statement ranking algorithm is designed to apply our metrics to obtaining the suspiciousness ranking of statements to be the fault. Experiments show that our metrics, especially our weighted metrics rank fault well with test suites of different types. Furthermore, the ineffectiveness of a_{ef}-based metrics is solved. Compared with the result of other metrics, fewer statements should be examined according to the ranking until the fault is located. The efficiency of fault localization is improved.

Besides program spectra and execution trace self-information, the relationship of neighbor running statements will be considered to compute suspiciousness of statements in the future work in order to increase the suspiciousness ranking of fault.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (Nos. F020512, F020204), the Natural Science Foundation of Hebei Province (No. F2014203152) and the Education Department of Jilin Province [2016]96.

REFERENCES

[1] S. Yoo and M. Harman, Regression testing minimization, selection and prioritization: A survey, *Software Testing, Verification and Reliability*, vol.22, no.2, pp.67-120, 2012.

[2] Y. Wang, Z. Chen, Y. Feng, B. Luo and Y. Yang, Using weighted attributes to improve cluster test selection, *Proc. of IEEE the 6th Int'l Conf. on Software Security and Reliability*, Washington D.C., USA, pp.44-58, 2012.

[3] C. Fang, Z. Chen, K. Wu and Z. Zhao, Similarity-based test case prioritization using ordered sequences of program entities, *Software Quality Journal*, vol.22, no.2, pp.335-361, 2014.

[4] J. Ren, C. Wang, H. He and J. Dong, Identifying influential nodes in weighted network based on evidence theory and local structure, *International Journal of Innovative Computing, Information and Control*, vol.11, no.5, pp.1765-1777, 2015.

[5] L. Naish, H. J. Lee and K. Ramamohanarao, A model for spectra-based software diagnosis, *ACM Trans. Software Engineering and Methodology*, vol.20, no.3, pp.1-32, 2011.

[6] W. Wong, Y. Qi, L. Zhao and K. Cai, Effective fault localization using code coverage, *Proc. of the 31st Annual Int'l Computer Software and Applications Conference*, Beijing, China, pp.449-456, 2007.

[7] R. Abreu, P. Zoeteweij and A. J. C. van Gemund, On the accuracy of spectrum-based fault localization, *Proc. of the Testing: Academic and Industrial Conf. Practice and Research Techniques*, Windsor, UK, pp.89-98, 2007.

[8] J. Xuan and M. Monperrus, Learning to combine multiple ranking metrics for fault localization, *Proc. of the 30th IEEE Int'l Conf. on Software Maintenance and Evolution*, Victoria, Canada, pp.191-200, 2014.

[9] P. Daniel, K. Y. Sim and S. Seol, Improving spectrum-based fault-localization through spectra cloning for fail test cases, *Contemporary Engineering Sciences*, vol.7, no.14, pp.677-682, 2014.

[10] S. Ali, J. H. Andrews, T. Dhandapani and W. Wang, Evaluating the accuracy of fault localization techniques, *Proc. of IEEE/ACM Int'l Conf. on Automated Software Engineering*, Auckland, New Zealand, pp.76-87, 2009.