ICIC Express Letters ICIC International ©2016 ISSN 1881-803X
Volume **10**, Number **11**, November **2016** pp. **2719–2726**

EFFECTIVE SUSPICIOUSNESS COMPUTATION METRICS FOR FAULT LOCALIZATION IN SOFTWARE SYSTEM

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, Jilin 132012, P. R. China

Received April 2016; accepted July 2016

Abstract. *Suspiciousness metrics based on failed execution spectrum are ineffective for fault localization without any failed execution, and with test suites of different types and sizes, the performance of spectra-based suspiciousness metrics is not stable. In this article, both failed execution spectrum and successful non-execution spectrum are considered as decisive factors, failed non-execution spectrum and successful execution spectrum are used as secondary factors in computing suspiciousness of each block to be the fault, and two new suspiciousness metrics (short as F_4N_1 and F_2N_2) are designed. To obtain more statistical information of the fault, the concept of execution trace self-information is proposed, and then two weights are constructed for weighting the complex expressions respectively in F_4N_1 and F_2N_2. Therefore, two weighted suspiciousness metrics F_4N_1W and F_2N_2W are acquired. Then a fault localization algorithm based on proposed suspiciousness metrics is given to apply these metrics to fault localization. Experiments are conducted on the program in the Siemens Suite with test suites of different types and sizes. It is shown that in most cases, fault has a higher suspiciousness ranking obtained by our metrics, especially by our weighted metrics. And fewer blocks need to be examined until the fault is located.*
Keywords: Software fault localization, Program spectra-based metric, Execution trace self-information, Suspiciousness computation metric

1. **Introduction.** Software testing is important to confirm the reliability of the software system, especially for large software systems. Because of the complexity of the software system, software testing is time-consuming, and executing all test cases is infeasible. Thus, a regression test selection technique [1] and a test selection strategy based on weighted attribute [2] are proposed to decrease the size of test suite. With the given test suites, test cases should be prioritized to increase the effectiveness of the testing [3].

However, software cannot be tested exhaustively. The main aim of researches is how to locate faults as soon as possible. To identify the influential functions in complex software network, an approach is proposed for fault localization [4]. Program spectra are designed to capture the dynamic feature of program. Based on failed execution spectrum, the decisive factor in computing suspiciousness, suspiciousness metrics Jaccard [5], Tarantula [6], Zoltar [7] and Ochiai [8] are proposed. When the fault is covered by no failed executions, these metrics will be ineffective, especially for the small test suite. In view of this, other spectra are also considered for designing metrics, such as metrics Euclid [8], Simple Matching, Sokal and Hamann [9]. A learning-based approach is proposed to combine multiple metrics for fault localization [10]. However, with the assumption that failed execution spectrum and successful non-execution spectrum have the same effect on the suspiciousness, the performance is not good in locating some faults, which affects the stability of these metrics for fault localization.

Therefore, based on failed execution and successful non-execution spectra, two new suspiciousness computation metrics F_4N_1 and F_2N_2 are proposed with the aim of computing the effective and stable suspiciousness result and solving the ineffective problem of metrics based on failed execution spectrum as well. Furthermore, the definition of execution trace self-information is given, and then weighted suspiciousness metrics F_4N_1W and F_2N_2W are designed based on execution trace self-information accordingly.

The remainder of this paper is organized as follows. Preliminaries are presented in Section 2. In Section 3, we describe two suspiciousness metrics F_4N_1 and F_2N_2, and two weighted metrics F_4N_1W and F_2N_2W based on execution trace self-information. Section 4 gives a suspiciousness metric-based fault localization algorithm. The experiments are designed on the typical program in Sections 5. Finally, we conclude the work in Section 6.

2. Preliminaries.

In this section, some notations and preliminaries are presented. A program is given which contains one fault (error or bug), and the program is divided into blocks $\{B_1, B_2, \cdots, B_N\}$. A block can be a single statement or a compound one. To locate the fault, a test suite $\{T_1, T_2, \cdots, T_M\}$ should be executed. Then the execution block spectra $\{e_{ij}\}$ and the corresponding result $\{r_i\}$ are collected, where $1 \le i \le M$ and $1 \le j \le N$. If B_j is covered by the execution of T_i, $e_{ij} = 1$; otherwise $e_{ij} = 0$. If the output for a given test case is different from the expected output, a failed execution occurs, $r_i = 1$; otherwise, the result is successful ($r_i = 0$).

With the above information, program spectra $\langle a_{ef}, a_{ep}, a_{nf}, a_{np}\rangle$ will be computed for each block, which are the same as the notations in [7]. Failed execution spectrum a_{ef} is the number of failed executions that cover the block, and failed non-execution spectrum a_{nf} denotes the number of failed executions that do not cover the block. Similarly, successful execution spectrum a_{ep} and successful non-execution spectrum a_{np} are defined. On the basis of some program spectra, the suspiciousness metrics are designed for fault localization [9]. And then the fault can be identified through the analysis of program spectra.

3. Suspiciousness Computation Metrics.

To obtain suspiciousness ranking of blocks to be the fault, two new suspiciousness computation metrics F_4N_1 and F_2N_2 are proposed with a_{ef} and a_{np} as the determining factors. To further reflect the importance of each expression in F_4N_1 and F_2N_2, execution trace self-information parameters are defined, and then two weighted suspiciousness computation metrics F_4N_1W and F_2N_2W are presented.

3.1. Suspiciousness computation metrics based on a_{ef} and a_{np} F_4N_1 and F_2N_2.

Based on both failed execution spectrum and successful non-execution spectrum, different expressions are designed for suspiciousness computation formula to reflect the importance of each spectrum, and two new suspiciousness metrics F_4N_1 and F_2N_2 are proposed.

A new suspiciousness metric F_4N_1 is proposed, as shown in Formula (1).

$$F_4N_1 = a_{ef} + \frac{a_{ef}}{a_{nf}} + \frac{a_{ef}}{a_{ep}} + \frac{a_{ef}}{a_{ef} + a_{ep} + a_{nf}} + \frac{a_{np}}{a_{np} + a_{ep} + a_{nf}} \tag{1}$$

where 'F' denotes a_{ef}-based expression, the subscript '4' is the number of expressions in the a_{ef}-based expression, 'N' denotes a_{np}-based expression, and '1' is the number of expressions in a_{np}-based expression.

Both a_{ef} and a_{np} are considered as the decisive factors, and a_{nf} and a_{ep} as the secondary factors. Besides a_{ef} itself, two simple expressions of a_{ef} are introduced to consider the influence of inversely proportional factors a_{ep} and a_{nf} respectively. In addition, one complex expression of a_{ef}, whose denominator is the polynomial $a_{ef} + a_{ep} + a_{nf}$, is introduced to reflect the influence of a_{ep} and a_{nf} on the result together and to reduce the importance of numerator a_{ef}. Unlike that of a_{ef}, only one complex expression of a_{np} is taken into account. The numerator and denominator are respectively a_{np} and the sum of a_{np}, a_{ep} and a_{nf}. Thus, four a_{ef}-based expressions and one a_{np}-based expression are designed to balance the influence of a_{ef} and a_{np} on the suspiciousness value.

Different from F_4N_1, a new suspiciousness metric F_2N_2 is designed wherein, two simple fractions of a_{ef} are excluded and another complex expression of a_{np} is introduced to decrease the importance of the decisive factors a_{ef} and increase the influence of a_{np}.

$$F_2N_2 = a_{ef} + \frac{a_{np}}{a_{np} + a_{ep} + a_{nf}} + \frac{a_{ef}}{a_{ef} + a_{ep} + a_{nf}} + \frac{a_{np}}{a_{np} + a_{ef} + a_{nf}} \tag{2}$$

If a_{ef} is nonzero, namely a_{ef} is larger than or equal to 1, the sum of expressions of a_{ef} will be larger than 1. And for a_{np}-based expression, the numerator a_{np} is included in the denominator, so the expression of a_{np} should be less than 1. Therefore, a_{ef}-based expression plays a main role in computing the suspiciousness. Otherwise, a_{ef}-based expression is zero, and only the expression of a_{np} plays a role in the metric and the suspiciousness still can be computed.

3.2. **Weighted suspiciousness metrics $\mathbf{F_4N_1W}$ and $\mathbf{F_2N_2W}$.** To get the information quantity of each event in execution traces, mainly the event of abnormal behavior of fault, four execution trace self-information parameters are defined.

Definition 3.1. *Using the theory of information, the occurrence probability $P(B_j\overline{R})$ of successful executions covering block B_j is considered together, which can be evaluated with the execution block spectra. The successful execution trace self-information of B_j is proposed as h_{ep}.*

$$h_{ep} = -P(B_j\overline{R}) \log \left(P(B_j\overline{R})\right) \tag{3}$$

And the successful non-execution trace self-information h_{np} is proposed by using the non-occurrence probability $P(\overline{B_jR})$ of successful executions of B_j. Similarly, with the occurrence probability $P(B_jR)$ of failed execution traces of B_j, the failed execution trace self-information h_{ef} is proposed. And with the non-occurrence probability $P(\overline{B_j}R)$ of failed executions of B_j, we get the failed non-execution trace self-information h_{nf}.

Definition 3.2. *With execution trace self-information parameters, the weight $EF\omega$ based on execution trace self-information is designed to weight the complex expression of a_{ef} as a whole, whose structure form is consistent with that of the complex expression.*

$$EF\omega = \frac{h_{ef}}{h_{ef} + h_{ep} + h_{nf}} \tag{4}$$

Definition 3.3. *The weight $NP\omega$ based on execution trace self-information is presented for the complex expression of a_{np}.*

$$NP\omega = \frac{h_{np}}{h_{np} + h_{ep} + h_{nf}} \tag{5}$$

Since each execution self-information parameter has the same sign, it is not necessary to consider the sign factor in $EF\omega$ wherein, h_{ef} is proportional to the weight $EF\omega$, and h_{ep} and h_{nf} are inversely proportional to $EF\omega$. In addition, since h_{ef} is included in the denominator, $EF\omega$ will be less than 1. Similarly, the value of $NP\omega$ will be less than 1.

On the basis of F_4N_1, two complex expressions of F_4N_1 are weighted using $EF\omega$ and $NP\omega$ respectively. Therefore, based on execution trace self-information, a weighted suspiciousness computation metric F_4N_1W is designed.

$$F_4N_1W = a_{ef} + \frac{a_{ef}}{a_{nf}} + \frac{a_{ef}}{a_{ep}} + EF\omega \cdot \frac{a_{ef}}{a_{ef} + a_{ep} + a_{nf}} + NP\omega \cdot \frac{a_{np}}{a_{np} + a_{ep} + a_{nf}} \tag{6}$$

Besides the retaining of all expressions in F_2N_2, the complex expression of a_{ef} is weighted by $EF\omega$, and one expression of a_{np} is weighted by $NP\omega$. A weighted suspiciousness metric F_2N_2W is proposed.

$$F_2N_2W = a_{ef} + \frac{a_{np}}{a_{np} + a_{ep} + a_{nf}} + EF\omega \cdot \frac{a_{ef}}{a_{ef} + a_{ep} + a_{nf}} + NP\omega \cdot \frac{a_{np}}{a_{np} + a_{ef} + a_{nf}} \tag{7}$$

Both h_{ef}-based expression $EF\omega$ and h_{ef}-based expression $NP\omega$ are taken into account for constructing these two metrics, the information quantity can be obtained which is provided by B_j covered by the failed executions and B_j not covered by the successful executions, and then more information about the abnormal behavior of faulty block can be obtained. For h_{ef}, h_{np}, h_{ep} and h_{nf} can be used as the unsigned number, their values exert an influence on the suspiciousness result. A high h_{ef} meaning a high $EF\omega$ and a high h_{np} meaning a high $NP\omega$ are proportional to the suspiciousness result. In addition, small hep and h_{nf} meaning high $EF\omega$ and $NP\omega$ are inverse to the suspiciousness result.

Since both $EF\omega$ and $NP\omega$ are less than 1, the influence of two complex expressions in F_4N_1 or F_2N_2 is dynamically adjusted, and the value of F_4N_1W or F_2N_2W is mainly determined by a_{ef}-based expression when a_{ef} is nonzero. Otherwise, when a_{ef} is zero, the suspiciousness is determined only by a_{np}-based expression.

4. **Software Fault Localization Using Suspiciousness Metric.** For the fault program, a method is described in this section to apply suspiciousness metrics F_4N_1, F_2N_2, F_4N_1W and F_2N_2W to obtaining the suspiciousness ranking result of blocks for locating the fault.

Three phases of work should be conducted for fault localization by using suspiciousness metric. Firstly, the execution traces and outputs of test cases are collected which run on correct and fault versions. Secondly, the execution block spectra will be extracted.

Algorithm 1: Suspiciousness metric-based fault localization algorithm

Input: The correct version V_0 and fault versions $\{V_k\}$, the test suite $\{T_i\}$
Output: Ranked blocks $\{B_{i_k}\}$ for each version
For each test case T_i in $\{T_i\}$
    Run the test case on the correct version V_0
    Collect the output
End For
For each fault version V_k
  For each test case T_i
    Run the test case
    Collect execution trace and output r_i
  End For
End For
For each fault version V_k
  Extract execution block spectra $\{e_{ij}\}$ for $\{B_j\}$
End For
For each fault version V_k
  For each block B_j
    Get program spectra a_{ef}, a_{ep}, a_{nf} and a_{np}
    Compute execution trace self-information parameters h_{ef}, h_{ep}, h_{nf}, h_{np}
    Compute the suspiciousness by using suspiciousness metric
  End For
  Output the sequence $\{B_{i_k}\}$ by using the suspiciousness result for version V_k
End For

Thirdly, the suspiciousness ranking of blocks is computed using the extracted program spectra and execution trace self-information.

The concrete steps of the suspiciousness metric-based fault localization algorithm are described in Algorithm 1. Using the algorithm, a sequence $B_{i_1}, B_{i_2}, \cdots, B_{i_N}$ is output according to from high to low suspiciousness. When more than one block corresponds to the same suspiciousness, the middle-line strategy in [6] is used. The examination of blocks starts from high-ranking ones until the fault is located.

5. **Experiment.** To compare the effectiveness for fault localization of our metrics F_4N_1, F_2N_2, F_4N_1W and F_2N_2W with that of other metrics Tarantula (short as TA), Jaccard (JACC), Simple Matching (SIMP), Sokal (SOK), and Hamann (HAN), three groups of experiments are conducted by using the software program "tcas" which has most fault versions in the Software-artifact Infrastructure Repository (SIR) [9].

5.1. **Experimental environment.** "tcas" stands for "aircraft collision avoidance system", which has 41 fault versions. All versions adapt for the experiment except the version where fault lacks code and the one where fault is related to more than one block. It is hard to collect information about fault of versions of above two types. Since the fault of the macro definition cannot be executed, the block of calling macro is located for the version. As a result, 35 fault versions are selected.

The type and size of test suite may affect the performance of suspiciousness metric. To investigate how well our metrics perform with test suites of different types and sizes, test suites of three types "bigcov", "cov" and "bigrand" are used.

An open source software infrastructure WET [11] is referred, the suspiciousness metric-based fault localization algorithm is realized by Java programming language, and our experiments are conducted under Fedora Core system environment.

5.2. **Experimental results.** With test suites of "bigcov", "cov" and "bigrand", three groups of experimental results of the suspiciousness metrics-based fault localization are discussed.

Test suites of "bigcov" are generated for coverage, whose size is about 80. We randomly use five "bigcov" suites, the average ranking of the fault of each version is obtained by using the given metric, and it corresponds to each curve as shown in Figure 1.

a_{ef}-based metrics TA and JACC are ineffective for many versions such as 5, 6, 8, 12, 13, 15, 20, because a_{ef} has no effect on the suspiciousness without any failed execution. Although the improvement of SIMP brings out the metric SOK, metrics SIMP and SOK based on a_{ef} and a_{np} almost have the same performance. Though HAN is constructed based on four spectra, the ranking is not increased obviously. Compared with other metrics, our

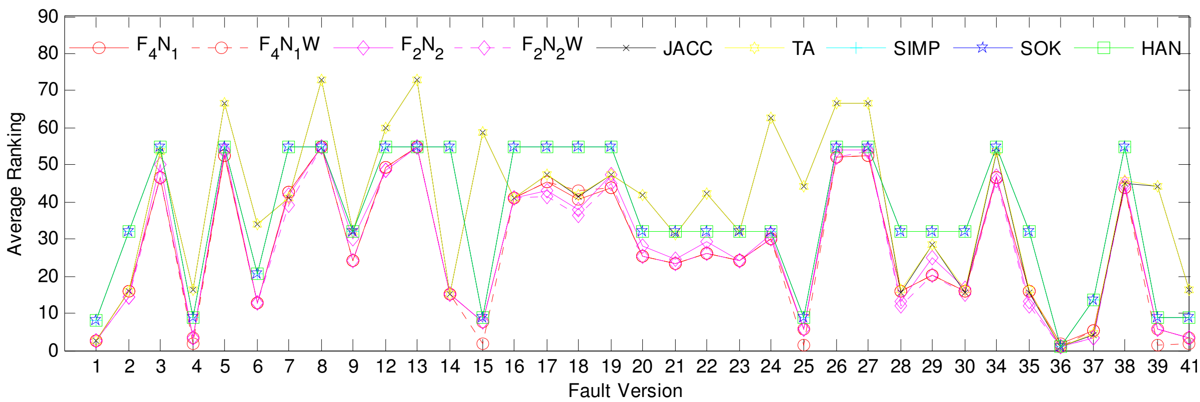

Figure 1. The average ranking of the fault with "bigcov" suites

metrics have better performance for most versions such as 3, 4, 6, 9, 12, 15, 17, 18. In comparison with TA, JACC and SIMP, our metric F_4N_1 gains an average increase of 15%, 14.8% and 11.6% respectively, F_4N_1W gains an average increase of 15.6%, 15.5% and 12.3% respectively, F_2N_2 gains an average increase of 14.3%, 14.1% and 10.9%, and F_2N_2W gains an average increase of 15.8%, 15.7% and 12.5%. In addition, with weights based on execution trace self-information, F_4N_1W performs better than F_4N_1 for versions 4, 15, 25, 30, 36, 39, 41, and F_2N_2W performs better than F_2N_2 for versions 3, 5, 7, 9, 17, 18, 19, 20, 21, 22 and so on.

Five "cov" suites are utilized, which are generated to achieve branch coverage, and the size is reduced to about 7%∼10% of the "bigcov" suite. The average ranking of the fault is shown as Figure 2.

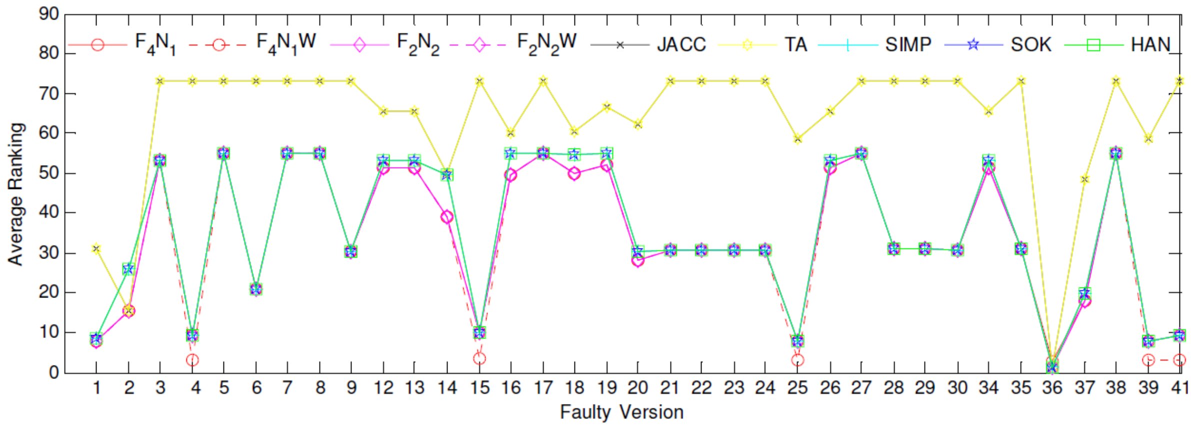

FIGURE 2. The average ranking of the fault with "cov" suites

Since less or none of failed test cases are included in these suites, a_{ef} is zero for most versions. It is obvious that depending on the type of test suite, TA and JACC turn into the worst case, which is shown in Figure 3. Other metrics have relatively close ranking, which are all better than TA and JACC. Our four metrics are superior to metrics SIMP, SOK and HAN for versions 1, 2, 12, 13, 14, 16 and so on. By means of weights based on the execution trace self-information, F_4N_1W performs better than F_4N_1 for versions 4, 15, 25, 36, 39, 41.

Tests are selected randomly to generate "bigrand" suites, which have the same size as "bigcov" suites. The average ranking result of the fault is described as Figure 3, which is obtained based on five "bigrand" suites.

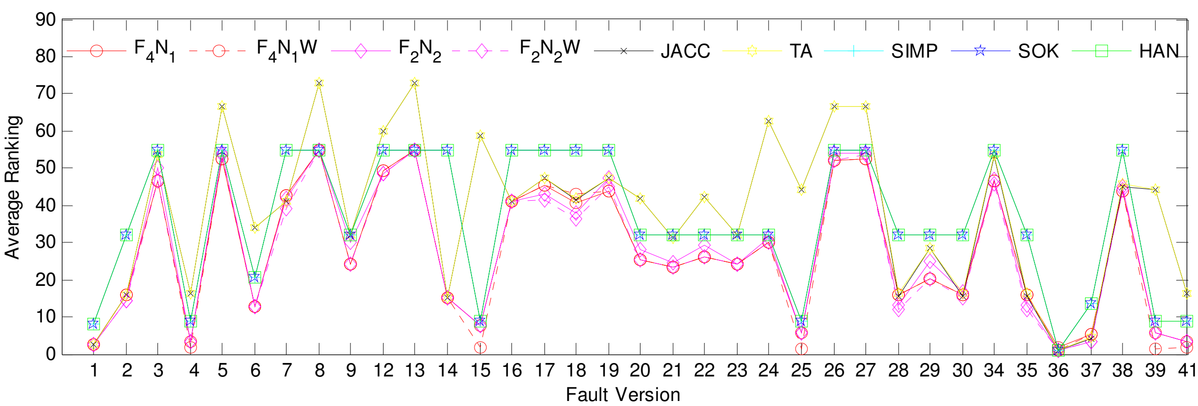

FIGURE 3. The average ranking of the fault with "bigrand" suites

With an even big suite, a_{ef}-based metrics JACC and TA cannot compute suspiciousness for versions 4, 5, 6, 8, 9, 13, 15, etc., and the ranking becomes unstable. The ranking of SIMP, SOK and HAN is between that of a_{ef}-based metrics and our metrics. By contrast, F_4N_1, F_2N_2, F_4N_1W and F_2N_2W have a higher ranking for most versions. Furthermore, F_4N_1W displays better performance than F_4N_1 for versions 4, 6, 15, 22, 25, 36, 39, 41. In a similar way, F_2N_2W performs better than F_2N_2 for versions 9, 12, 34, 38.

To sum up, the results show that F_4N_1, F_2N_2, F_4N_1W and F_2N_2W (especially F_4N_1W and F_2N_2W) have a stable and effective suspiciousness ranking for fault with test suites of different sizes and types, that they are capable in the circumstances of fewer test cases, that it is possible for them to locate fault as soon as possible, and that what is more important is the solution of ineffectiveness problem of a_{ef}-based metrics.

6. **Conclusions.** We propose two new suspiciousness metrics F_4N_1 and F_2N_2 for fault localization based on a_{ef} and a_{np}. We put forward the concept of execution trace self-information, and then design two weighted suspiciousness metrics F_4N_1W and F_2N_2W respectively on the basis of F_4N_1 and F_2N_2. Then we design the suspiciousness metric-based fault localization algorithm to illustrate the application of our suspiciousness metrics in locating fault. The experiment results show that the ineffectiveness problem of a_{ef}-based metrics can be solved, and our metrics F_4N_1, F_2N_2, F_4N_1W and F_2N_2W (especially F_4N_1W and F_2N_2W) generally have a higher suspiciousness ranking of fault with test suites of different types and sizes. As a result, fewer blocks need to be examined for fault localization in software system and the effectiveness of fault localization can be improved.

The performance of suspiciousness metric-based fault localization method should be improved for cases of complex fault in the program, such as the fault of missing multiple lines of code. To effectively locate complex faults, it is necessary to consider the information of execution time of each statement for each test case, and then the suspiciousness metric-based fault localization method will be extended in the future work.

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] C. Zhang, Z. Chen, Z. Zhao, S. Yan, J. Zhang and B. Xu, An improved regression test selection technique by clustering execution profiles, *Proc. of the 10th Int'l Conf. on Software Quality*, Zhangjiajie China, pp.171-179, 2010.

[2] Y. Wang, Z. Chen, Y. Feng, B. Luo and Y. Yang, Using weighted attributes to improve cluster test selection, *Proc. of 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] H. He, J. Wang and J. Ren, Measuring the importance of functions in software execution network based on complex network, *International Journal of Innovative Computing, Information and Control*, vol.11, no.2, pp.719-731, 2015.

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

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

[7] T. Janssen, R. Abreu and A. J. C. Van Gemund, Zoltar: A spectra-based fault localization tool, *Proc. of the 2009 ESEC/FSE Workshop on Software Integration and Evolution Runtime*, Amsterdam, Netherlands, pp.23-29, 2009.

[8] 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.

[9] 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.

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

[11] V. Nagarajan, D. Jeffrey, R. Gupta and N. Gupta, ONTRAC: A system for efficient online tracing for debugging, *Proc. of Int'l Conf. on Software Maintenance*, Paris, France, pp.445-454, 2007.