15-June-2014

Dear Professor Hall, Editors, and Reviewers

Thank you for the detailed and constructive reviews.

Following the two reviewers’ comments and advice, we significantly revised our manuscript. Below we provide a list of our responses to the reviewers’ comments. We are happy to answer any further questions.

We would like to thank you for giving us this opportunity to improve the manuscript. We look forward to hearing from you.

Yours Sincerely,

Xin, Liang, Tien-Duy, David, Lingxiao, and Hongyu

**Responses to Reviewers’ Comments**

**1. Summary of Changes:**

* **Table & Figures.** We adjusted the tables and figures in the paper to better fit the paper.
* **New Sections.** We add two new sections, i.e., Section 2.2.4 Practical Usage, and Section 4.5 Approach Complexity and Robustness.

**2. Changes with respect to specific comments given by the individual reviewers:**

**Reviewer** #**1:**

* 1. **Comments:** My main requirement for a "minor" revision is that this paper very much needs a thorough proofread by a native English speaker. Some issues include the title -- in English "multi-faults" should be replaced by "multi-fault" because multi indicates plural already (e.g. we say "multicore" not "multicores" machine). The problems here do not rise to the level of making the paper very hard to read, but it would be much more pleasant to read after a thorough proofreading pass. There are numerous missing/extra articles and plural problems like the one in the title.

**Response:** Thank you for pointing out the issues. We have changed all the appearance of “multi-faults” to “multi-fault”. Also, we have fixed various English grammar issues in the paper.

* 1. **Comments:** I have one basic complaint about experimental results. You give average positive improvements in text in a few places, can you give average negative also? Also, note, that "improvement" is positive by definition, a language issue. This is somewhat a presentation issue, but it is also a real issue. Figure 11, for example, looks as if when DMS performs worse than RAPTOR, it performs MUCH worse, perhaps, in the multi-fault case. Compare to Figure 7, where it's clear the average gain is much much better than the average gain. Some discussion of this issue would be very nice, and help make the multi-fault aspect more than just some additional empirical data, it would create an understanding of how the multi-fault case really differs from single fault. Fig. 11 in particular makes me think I might prefer RAPTOR for multi-fault cases!

**Response:** Thank you for the questions. We changed "negative improvement" to "negative deterioration".

We have added Table 9 and Table 14 which correspond to the negative deterioration for single-fault and multi-fault programs and explanatory texts in the corresponding places in the paper:

"Table 9 characterizes the degree of negative deterioration of DMS over other techniques. As the first row of the table indicates, for half of the 19.29% faulty versions, Dms deteriorates between 0.03% and 0.60% from Raptor, and for the other half, Dms deteriorates between 0.60% and 1.15%. The average percentage of negative deterioration of DMS over Raptor is 0.54%."

"Table 14 illustrates the degree of negative deterioration of Dms over other techniques. The first row shows that, half of the 31.79% faulty versions for which Dms performs worse than Raptor have deterioration between 0.23% and 2.94%, and the other half have deterioration between 2.94% and 53.30%. The average deterioration of Dms from Raptor is 8.54%."

"The comparison shows that Dms is better than Raptor on 60 versions, out of 115 versions that show differences in cost, but performs worse than the Raptor on 55 versions. The average deterioration (8.54%) of Dms (Table 14 is higher than its average improvement (5.95%) in comparison with Raptor (Table 13), even though Dms reduces the total labelling effort from 98 test cases to 79 (Table 11). We are yet unclear about the reason causing the trade-off in the multi-fault programs. It is very intriguing future work to find ways to balance between labelling effort and diagnostic cost better."

* 1. **Comments**: - On p. 24, line 40 -- is Table 6 the right table #?

**Response:** Thank you for the question. We have modified it.

**Reviewer** #**2:**

* 1. **Comment:** Before being considered for publication, I would like the authors to address the following issues:

- The abstract mentions that fault localization techniques require "non-trivial number of test cases". Previous work, such as Abreu et al. 2009, state that 20 passed and 6 failed runs suffice for fault localization. I do not think that this is a non-trivial number, and therefore this statement should be somehow backed up.

**Response:** Thank you for the question. We have rephrased the abstract as follows:

“Although most fault localization techniques can localize faults relatively accurately even with a small number of test cases, choosing the right test cases and creating oracles for them are not easy.”

We also added the related citations in Section 5.1.4:

"We also find that in our experiments the average debugging cost of using Dms will not reduce noticeably even if more labeled test cases beyond 50 are added further (See Figure 3), which is in line with studies in the literature (e.g. Abreu et al, 2009; Liblit et al, 2005) that tens of passed and failed spectra may suffice for fault localization."

* 1. **Comment:** Labeling in the abstract and introduction is not clear what it refers to. It then becomes clear that it refers to label a test case as pass/fail

**Response:** Thank you for the question. We have added an explanatory phrase "(i.e., effort needed to decide whether the test cases pass or fail)".

* 1. **Comment:** Introduction's second paragraph mentions the work of Baah et al. 2011 as being an example of Spectrum-based fault localization. That is not really just spectrum-based fault localization in my opinion.

**Response:** Thank you for the question. We have cut the reference for Baah et al. 2011

* 1. **Comment:** The work of Campos et al. on test suite generation for better fault localization has been published at ASE'13, and should be cited in the introduction and related work.

**Response:** Thank you for the pointer. We have added Campos et al. work in Both Introduction and Related Work sections.

In Introduction: "Campos et al (2013) use probability concepts to generate new test cases that could minimize the entropy of fault localization results. Although their approach reduces the diagnosis costs of fault localization results, it does not directly aim to minimize the number of test cases generated that may require manual labelling."

In related work:

"Campos et al (2013) propose ENTBUG which applies entropy theory to guide test case generation.”

* 1. **Comment:** To increase readability, I suggest authors to increase the size of most figures (and tables too), in particular Fig. 2 (this figure is particularly difficult to read also because of the like style 29 and s7). I've had, really, a hard time to read them..

**Response:** Thank you for pointing out the issue. We have changed the style and enlarged the figures a bit.

* 1. **Comment:** What is the rationale of the change potential computation? How did the authors end up with Equation 2? Being central to the approach, it should be discussed in detail.

**Response:** Thank you for the questions. We have added the rationales in Section 4.1:

Before Equation 2:

“In order to speed up the overall evolution process, our approach needs to select next test case that keeps elements with monotonic trends (high change-potential trends) evolving their rankings. In other words, we do not care about changing elements' ranking with unstable trends. In order to identify those high change-potential elements, we need a metric to evaluate and compare trends of different elements……"

After Equation 2:

"Rationale of Equation 2: We want to evolve the ranks of statements in a fast, monotonic way. In linear models, A fast changing monotonic trend should have a larger slope B1 as well as a smaller deviation ei from the linear model. Using this metric in Equation 2 that uses the estimated slope in the numerator and the estimated deviation in the denominator, we may isolate trends that evolve in faster and more stable (less oscillation) ways."

* 1. **Comment:** Does the approach require execution (e.g., to collect the program spectra) before being able to prioritize them? The authors should discuss - perhaps in Section 2 - how this approach could be used in practice. I.e., suppose I want to use it in my company now, what should I do before being able to make use of this technique?

**Response:** Thank you for the questions. We have added a section 2.2.4 “Practical Usage”:

"To use the above mentioned test case prioritization techniques in practice, a program needs to be instrumented first and executed with test cases first to collect the program spectra (execution traces) of the test cases. Then, developers can apply one of the test case prioritization techniques to select top-n ranked test cases, and manually judge whether each of the test cases passes or fails. Based on these selected test cases and their corresponding labels (passed or failed), a fault localization technique (e.g., Nainar et al, 2007; Jones and Harrold, 2005) can be applied to locate faults. Our test case prioritization technique in this paper can also be applied in the same fashion as the above mentioned test case prioritization techniques."

* 1. **Comment:** Regarding the approach: Why does it start with just one failed test case? What if there are more failed test cases? Fig 3. (why not Alg. 1) seems to only use one failed test... Moreover, if there are multiple failed test, which one is selected at this point?

**Response:** Thank you for the questions. We have changed the reference to Algorithm 1. We also added a footnote:

"If there is more than one test that fails, Dms randomly selects one of them to begin with."

* 1. **Comment:** Regarding the approach: the rationale behind equations 2,3,4, and 5 should be discussed in detail as it is the central point of the approach/contribution. Consider explaining them theoretically but also using the running example.

**Response:** Thank you for the questions. We added our rationale and heuristics for designing the equations in Sections 4.1, 4.2, and 4.3, respectively.

"Rationale of Equation 2: We want to evolve the ranks of statements in a fast, monotonic way. In linear models, a fast changing monotonic trend should have a larger slope as well as a smaller deviation from the linear model."

"Rationale of Equation 3: A group with high change-potential elements should be given a higher priority to break. We want to diversify the rankings of elements in the suspicious group that has a high change-potential score. To identify those high change-potential groups, we measure the sum of change-potential scores of its member elements."

"Since we want to stabilize the ranks of all program elements as quickly as possible with as fewer test cases as possible, we heuristically choose the test case that can maximize the difference and thus the reduction of change potentials as the next one for labeling."

"Rationale of Equation 4 and 5: A test case that breaks more groups with higher change potentials should be given a higher priority. Equation 4 measures the overall change potential score of all suspicious groups and its square form manifests the diversity of elements ranking. As an example, suppose there are two groups g1 and g2. Group g1 has two high change-potential elements with change-potential score 0.3 and 0.4 respectively. Group g2 has two low change-potential elements with change-potential score 0.1 and 0.2 respectively. According to Equation 4, HG = (0.3+0.4)2 + (0.1 + 0.2)2 = 0.58. After choosing a test case that 1) breaks g1 only and 2) does not change the change-potential score of any element, then according to Equation 5, the new change-potential would be 0.32 + 0.42 + (0.1 + 0.2)2 = 0.34. However, if we choose another test case that 1) breaks g2 only and 2) does not change the change-potential score of any elements, the new change-potential would be (0.3+0.4)2 + 0.12 + 0.22 = 0.54. As a result, the test case that breaks the high change-potential group (i.e., g1) leads to a larger overall change-potential decrease and thus will be given a higher priority to be selected."

* 1. **Comment:** Other question related to the approach is how does one define k and w (important for the bootstrap phase)? What is the impact of a wrongly selected value for these variables?

**Response:** Thank you for the questions. We added a discussion about the parameters in a new section 4.5 about the complexity and robustness of our approach:

"The results of our approach may not be deterministic either since there are random factors (e.g., the first failed test case to choose) and several user-defined parameters (e.g., the maximal number of test cases to select) employed in the approach. Thus, we repeat running our approach for multiple times in our evaluation to take average performance. The maximal number of test cases to select (i.e., k in Algorithm 1) and the switching threshold w in the bootstrap phase would affect how much improvements our approach can achieve over other existing techniques, but they do not matter for the main focus of our evaluation, which is to evaluate whether Diversity Maximization Speedup (Dms) is effective for reducing manual labelling effort and diagnostic cost. Although we expect different optimal k and w for different kinds of programs and faults, we use a consistent setting k <= 500 and w = 10 in our evaluation and would suggest potential users to starts with a small setting where they can tolerate the manual labelling cost."

* 1. **Comment：**  Pg. 14: consider rewriting the two paragraphs before the last one. All these formulae inline make it difficult to get the message absorbed.

**Response:** Thank you for the question. We have changed these two paragraphs.

* 1. **Comment：** Pg. 14: authors must better explain why RAPTOR would choose T8 for next best test to execute.

**Response:** Thank you for the question. We have added an explanatory phrase: "since t8 has the maximum ambiguity reduction value."

* 1. **Comment：** Authors should discuss the time/space complexity of the proposed approach. Does it scale to rather large programs? In this particularly important since it requires to execute and re-execute test cases several times

**Response:** Thank you for the questions. We have added a new section 4.5 Approach Complexity and Robustness.

"The time and space complexity of our approach depend on many various factors. It takes linear time with respect to the number of iterations or test cases we want to select, takes quadratic time with respect to the total number of available test cases, takes cubic time with respect to the number of program elements, takes cubic time with respect to the number of suspicious groups in each iteration. Our approach would also store all of the test spectra in the memory for convenience. In comparison with the other existing approach Raptor, the trend analysis step in our approach may be more computationally expensive. However, the step would still take short absolute amount of time since the number of selected test cases can be limited to tens or hundreds to achieve effective fault localization…"

* 1. **Comment：** Section 5.1.1: to foster reproducibility, the experiments could comply with what has been advocated in the following paper:

Steimann, Friedrich, Marcus Frenkel, and Rui Abreu. "Threats to the validity and value of empirical assessments of the accuracy of coverage-based fault locators." Proceedings of the 2013 International Symposium on Software Testing and Analysis. ACM, 2013.

The cost metric could also be in line with what has been defined in this paper.

**Response:** Thank you for the great pointer! We added more explanations about the meaning of the definition of the measure and a footnote to explain why we use the measure:

"There are a number of alternative ways to define diagnostic costs and accuracies, such as using the number of faults located when up to a certain percentage of program elements are inspected (e.g. Wong et al, 2014; Debroy and Wong, 2013; Cleve and Zeller, 2005; Baah et al, 2010; Jones et al, 2002; Lucia et al, 2014), or assuming a random ordering for elements with the same score and incorporating their expected rank (|{j|fTS(dj)=fTS(d∗)}|+1) / 2 in calculating cost (e.g. Ali et al, 2009; Steimann et al, 2013; Steimann and Frenkel, 2012; Xu et al, 2011). We use the one in Equation 7 and 8 as they are commonly used and easy to understand, and our focus is on evaluating whether different test case prioritization techniques change the diagnostic costs, instead of measuring the absolute costs. We believe that if our technique shows significant improvements over one kind of diagnostic cost, it should also show improvements over other kinds of diagnostic cost."

"There are also many other kinds of threats to validity affecting fault localization techniques in general as listed in a recent study by Steimann et al (2013), such as heterogeneity of test cases, biases in injected faults, unrealistic assumptions about locating and understanding faults, etc. Although we focus on evaluating test case prioritization techniques, instead of fault localization techniques, our work inevitably inherits the threats to validity for fault localization techniques since our evaluation of prioritization techniques is done through the evaluation of fault localization. We hope in future work the threats to validity for both fault localization and test case prioritization techniques can be addressed together."

* 1. **Comment:**  I fail to understand the x% base line effectiveness equation. What does x% stand for at this point? Is this the percentage of test cases that need to be labelled in order to get the same diagnostic cost? Yet another issue: where does the 101% effectiveness comes from?

**Response:** Thank you for the questions. We revised the explanations about how we define the measure in Section 5.1.1:

"If labeling all test cases and performing fault localization on all program spectra results in an average diagnostic cost c, we call it the base line cost. If a test prioritization technique or fault localization technique leads to a diagnostic cost c', then we say the technique achieves x% of base line effectiveness, where x is defined as follows:

x = c'/ c × 100 (9)

To be fair, the number of reduced test cases by each prioritization technique should be measured when the technique achieves 100% of base line effectiveness. However, in reality, it is hardly possible to directly control the cost to be exactly 100% of base line. So, we allow 1% deviation; i.e., in the following evaluation results, we measure the numbers of reduced test cases when at most 101% of base line effectiveness is achieved."

We also added explanations for the calculation of the measure:

"Since Dms would output a ranked list of suspicious program elements, we compute the diagnostic cost cn for Dms when we just inspect top-n (n in {1,2,…,|D|}) suspicious elements. We record the maximum n that cn is still within 101% of base line cost as the amount of labeling effort."

* 1. **Comment:** It seems that there is a cross-ref problem in the document. As an example, on pp. 18 the authors mention table 11, when they mean table 7. Carefully revise references.

**Response:** Thank you for pointing out the problems. We have corrected the cross references.

* 1. **Comment:** The authors mention that "due to limited space, we only show the comparison between DMS and these methods in detail". Being a journal submission, space is not a problem. For instance, Table 8 shows the distribution of positive improvements. It would be interesting to include a similar table but for which RAPTOR is better. This would help in fully understanding the results. (Actually, the paragraph discussing these results mentions table 6 and not table 8).

**Response:** Thank you for pointing out the problems. We have cut the sentence, and added the negative deterioration tables 9 and 14 for single-fault and multi-fault programs.

* 1. **Comment:** DMS vs. RAPTOR (both for single and multiple faults): the sentence that DMS is better than RAPTOR is too strong, and should be down-toned a bit. For instance, does 1% improvement really means that one is better than the other? It terms of cost, the user study by Parnin and Orso (ISSTA'11) seems to say otherwise. Comparison to RAPTOR, the best technique out there, should be made clearer, also in terms of runtime complexity. Does it really improve over raptor in general? What is the trade-off to get better results?

**Response:** Thank you for pointing out the over-claims. We removed all sentences that simply claim Dms is better than Raptor or other techniques, and revised the related sentences with clearer and more specific comparisons. E.g.,

"…The comparison shows that Dms outperforms Raptor on 20 versions by at least 1% cost, and on only 5 versions, it is worse than Raptor by over 1% cost."

"…our method outperforms no worse than other methods for the majority of faulty program versions."

"…suggests that the improvements of Dms over other existing techniques are statistically significant at 95% confidence interval."

Also, added a discussion about runtime complexity in Section 4.5:

"…In comparison with the other existing approach Raptor, the trend analysis step in our approach may be more computationally expensive. However, the step would still take short absolute amount of time since the number of selected test cases can be limited to tens or hundreds to

achieve effective fault localization."

* 1. **Comment:** Related work should consider work on test suite generation for fault localization, e.g.,

Campos, José, et al. "Entropy-based test generation for improved fault localization." Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on. IEEE, 2013.

**Response:** Thank you for the pointer. We have added this reference in both Introduction and Related Work sections.

* 1. **Comment:** Minor issues:

- Introduction: can require -> may require.

- Introduction: need human -> need a human.

**Response:** Thank you for pointing out the grammar issues. We have modified accordingly.

- Section 2: I would suggest to add an introductory text between section 2 and subsection 2.1.

**Response**: Thank you for pointing out the writing issue. We added a transitional paragraph:

"In this section, we summarize relevant materials on software fault localization and test case prioritization that we use in our empirical evaluation."

- Consider using ambiguity group instead of suspicious group to be in line with related work.

**Response**: Thank you for the pointer! We considered it careful and felt that the two different terms show a bit different perspectives of the groups, so we added a footnote to explain the difference:

"We call such groups as suspicious groups since we simply want to state the fact that every group may contain potentially suspicious elements. Some other studies (e.g. González-Sanchez et al, 2011a) call them ambiguity groups as that term may emphasize more on the fact that the elements in the groups have the same but ambiguous suspiciousness scores."