Comments from AE:

Thank you for this revision. The reviewers need some minor revisions. Please include a detailed list of your revisions with your next draft.

Reviewer #1: This paper presents a method for prioritizing test cases to maximize diversity for purposes of fault localization. There is a nice practical motivation in limiting how many test cases must be labeled in the absence of an oracle. The gain looks real, because the number of test cases that must be selected to get "pretty good" results is small enough that you can imagine hand labeling -- if the technique "worked" but required hundreds of labeled tests, I would say that it basically is typically useless. The technique is well presented and compared to a variety of relevant standards in the literature. The gain on ASE 2012 is primarily more detailed exposition and (the real gain) considering the multi-fault case. On the one hand, the technique from ASE 2012 already would work on multi-fault, there's no real method change here. On the other hand, there was no evaluation of whether it was useful for multi-fault cases, so this is a useful thing to know.

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.

Xin： Change all Multi-Faults to multi-fault.

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!

Xin: Add negative improvement for multi-fault program. For single-fault program, we lost the data.

Moreover, from our experiment results, it seems RAPTOR performs well than our method in multi-fault cases…

Gong Liang also add the negative experiments in Table 9. Table 9 characterizes the degree of negative improvement of

\textsc{Dms} over \textsc{Raptor}. As the table indicates, half of the 19.29\% faulty versions with positive improvement values have improvements between 0.03\% and 0.60\%, and the other half

have improvements between 0.60\% and 1.15\%. The average

negative improvement of \textsc{Dms} over \textsc{Raptor} is 0.54\%.

Detailed:

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

Xin: Changed

Reviewer #2: This is an interesting and relevant paper on test suite prioritization for improving fault localization. I do like the idea of including the developer in the loop, as this solves the oracle problem in many situations. It is in general well-written (although I would suggest to re-write the empirical evaluation section because it is not really straight-forward).

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.

Xin: I don’t how to address this 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

Xin: Change labeling as “label a test case as pass/fail”.

- 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.

Xin: Cut the reference for Baah et al. 2011

- 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.

Xin: Add Campos et al. work in Introduction and Related Work Section.

Add a sentence in related work: Campos et al. propose ENTBUG which applies entropy theory to guide test case generation~\cite{campos2013entropy}.

- 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..

Xin: Change the style of the figure.

- 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.

Xin: Gong Liang help to address this comments.

\noindent{\em Change Potential Computation.} 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.

- 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?

Xin: Add a section “\subsubsection{Practical Usage}

To use the above mentioned test case prioritization techniques in practice, developers first collect the program spectra (execution trace) for all of the test cases. Then, they run the test case prioritization techniques to select top-n ranked test cases, and manually judge whether it is passed or failed. Based on these ranked test cases and their corresponding labels (passed or failed), they apply fault localization techniques\citep[e.g.][]{NainarCRL07,JH05} to locate the faults.”

- 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?

Xin: Changed Fig.3 as Algorithm 1. Add a footnote “If there are more than one test fails, Dms randomly select one of them to begin with.”

- 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.

Xin: Gong Liang help to address this comments.

\noindent\textbf{Rational of Equation \ref{eq:trend\_metric}:} \textit{A good trend should have a larger slope as well as a smaller deviation from the linear model.}

In our empirical model, fast changing monotonic model has a bigger slope and thus gets a bigger numerator. Meanwhile a stable and monotonic trend has a smaller deviation from the linear model and thus gets a smaller denominator.

\noindent\textbf{Rational of Equation \ref{eq:elem\_potential}:} \textit{Group with high change-potential elements should be given 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.

\noindent\textbf{Rational of Equation \ref{eq:groupset\_potential} and \ref{eq:select\_metric}:} \textit{Test cases that breaks more groups with higher change-potential should be given higher priority.}

Equation \ref{eq:groupset\_potential} 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 $g\_{1}$ and $g\_{2}$. Group $g\_1$ has two high change-potential elements with change-potential score 0.3 and 0.4.

Group $g\_{2}$ has two low change-potential elements with change-potential score 0.1 and 0.2.

According to equation \ref{eq:groupset\_potential}, $\mathcal{H}\_G = (0.3+0.4)^{2} + (0.1 + 0.2)^2 = 0.58$. After choosing a test case that 1) breaks $g\_1$ only and 2) does not change the change-potential score of any elements, then according to equation \ref{eq:select\_metric}, the new change-potential would be $0.3^{2} + 0.4^{2} + (0.1 + 0.2)^2 = 0.34$. However, if we choose another test cases that 1) breaks $g\_{2}$ only and 2) does not change the change-potential score of any elements, the new change-potential would be $ (0.3+0.4)^{2} + 0.1^{2} + 0.2^{2} = 0.54$. As a result, the test case that breaks the high change-potential group (\textit{i.e.,} $g\_{1}$) leads to a larger overall change-potential decrease and thus will be given a higher priority to be selected.

- 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?

Xin: I think it would cause no problem of the value k and w. But I don’t know how to write the sentences in the algorithm…

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

Xin: Changed

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

Xin: add a footnote “$t\_8$ has the maximum ambiguity reduction values.”

- 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

- 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.

Xin: I don’t know how to address this 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?

Xin: Add some sentences “Since Dms would output a ranking list of suspicious code, we compute the diagnostic cost $c\_n$ for Dms when we just inspect top $n$ ($n \in \{1,2,\cdot \cdot \cdot, |D|\}$) suspicious elements. When $c\_n$ is less than $c\_x$, we record $n$ as the number of labeling effort.”

- 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.

Xin: Changed

- 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).

Xin: cut the sentence “due to limited space, we only show the comparison between DMS and these methods in detail. Add the negative improvement table for multi-fault program. For single-fault program, we don’t have the data now.

- 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?

Xin: I think we should down-tone this sentence a bit. However, I don’t know how to detail address this comments.

- 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.

Xin: Added

Minor issues:

- Introduction: can require -> may require.

- Introduction: need human -> need a human.

Xin: changed

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

Xin: add a 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.

Xin: changed