Responses to Referee: 1

Comment 1:

One of my major concerns is the applicability of the proposed approach in practice, which is greatly hurt by the assumptions made in the paper, including all failures are deterministic and there are no inter-option constraints. Needless to say, these assumptions generally do not hold true in practice. While the first assumption is mentioned in the paper (Assumption 1 on page 6), the second assumption is not mentioned. This is important because in the presence of inter-option constraints, which invalidate some option setting combinations, some of the propositions and proofs given in the paper seem to become invalid and the approach needs to go through a set of nontrivial modifications. These issues need to be discussed in the paper and at the very least, the authors should elaborate on how to relax these assumptions.

Response: As suggested, we have explicitly clarified the following assumptions made by our approach: 1) the result of test case is deterministic; 2) failures can be distinguished and 3) no inter-option constraints exist. Additionally, we have added one more section (Section 8) to explain the implication of these assumptions and several approaches to relax them.

Comment 2:

I believe what is meant by Assumption 1 in the paper is that all failures are deterministic, rather than all test results are deterministic. If so, this assumption should be restated.

Response: We clarify that Assumption 1 in our paper is to mean that the test results are deterministic. Actually, both assumptions have been used in the existing studies of combinatorial testing. For example, Zhang [1] and Ghandehari [2] are based on the assumption that the test results are deterministic, while Yilmaz [3] and Fouché [4] are based on the assumption that the failures are deterministic.

In this paper, we use the assumption of deterministic test result. This is because it is simpler to handle. Deterministic failures may introduce the problem of test case-aware covering array [5], i.e., a failure may only appear in some test cases with a given test configuration, but not others. This is beyond the scope of this paper. Additionally, we believe that non-deterministic test results are caused by the non-deterministic failures, and hence we emphasize that in the Assumption 1 and later in section 8.

[1] Zhang, Zhiqiang, and Jian Zhang. "Characterizing failure-causing parameter interactions by adaptive testing." Proceedings of the 2011 International Symposium on Software Testing and Analysis. ACM, 2011.
[2] Ghandehari, Laleh Shikh Gholamhossein, et al. "Identifying failure-inducing combinations in a combinatorial test set." Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on. IEEE, 2012.

[3] Yilmaz, Cemal, Myra B. Cohen, and Adam Porter. "Covering arrays for efficient fault characterization in complex configuration spaces." Software Engineering, IEEE Transactions on 32.1 (2006): 20-34.
[4], Sandro, Myra B. Cohen, and Adam Porter. "Incremental covering array failure characterization in large configuration spaces." Proceedings of the eighteenth international symposium on Software testing and analysis. ACM, 2009.
[5] Yilmaz, Cemal. "Test case-aware combinatorial interaction testing." Software Engineering, IEEE Transactions on 39.5 (2013): 684-706.

Comment 3:

Furthermore, it is important to emphasize the fact that the proposed approach is concerned with option-related failures. It would also be interesting to evaluate and/or discuss how the proposed approach could deal with non-option-related failures.

Response: Considering the Reviewer's suggestion, we have emphasized that our approach can only handle the option-related failures (third paragraph of Page 7). Additionally, we discussed the problems of non-option-related failure in Section 9.5.2, as well as some measures that can be used to alleviate this problem.

Comment 4:

Another concern is that a number of definitions, propositions, and proofs are given in the paper, but the relationship between the theory and the proposed approach is not clear at all. How does the proposed approach benefit from these propositions? How does the proposed approach evolve from the given theory?

Response: We agree that the relationship between the formal modeling and proposed approach could be made clearer. In fact, the definitions in Section 3.1 (Definition 3.1 to Definition 3.5) are given to formally define what is MFS. Section 3.2 reveals the relationships between test sets and schemas. These relationships are the foundation of how to identify the MFS, and how masking effects affect the MFS identification. To achieve this objective, Propositions 3.7 and 3.8 show that any test set has its corresponding minimal schemas. These two propositions show the theory to obtain MFS, as MFS is exactly the minimal schemas of failing test sets. Propositions 3.10 and 3.11 show that for two test sets that have an inclusion relationship, their minimal schemas also have some relationships. These two propositions give the foundation of the impact of the masking effects, as the masking effects will make us misjudge the outcomes of test cases, which will result in that the actual failing test set will either be super-set or sub-set of the observed failing test set. Based on this, we can figure out the extent to which the MFS we obtained under the masking effects will be deviated from the real MFS. Other propositions in Section 3.2, i.e., Proposition 3.6, 3.9, are the auxiliary propositions to get these four propositions. Later in Section 5, we formally define the masking effect, and show what impact masking effects will have on the MFS identification based on Proposition 3.10 and 3.11. One key observation from the impact of masking effects is caused by the inability to determine the outcomes of some test cases which have triggered different failures other than the same failure under analysis. What's worse, according to the analysis in Section 5.3, this can result in incorrect inference about the results of some test cases. Hence, a natural idea is to reduce the number of test cases that trigger other failures as much as possible when we identify the MFS for some particular failure. In fact, we do not need to use the fixed test cases that are used for MFS identification (See newly added section 4). Hence, we can replace the test cases that are used to identify the MFS triggering other failures with the test cases that either pass or trigger the same failure under analysis. This is the foundation of the approach we proposed in this paper.

To summarize, these propositions are created to show how MFS is identified, and how and why the MFS identification is affected by the masking effect. The approach is proposed to reduce the impact that may be introduced by the masking effects, so we can obtain a higher-quality MFS. According to the reviewer's comments, we have improved the description of the significance for these propositions in the last paragraph of Section 3.2. And, one more section, i.e., Section 4, is added to show why we do not need to use the fixed test cases to identify the MFS, and shows that the test cases that trigger other test cases may negatively affect, to a significant extent, the result of FII approach (Section 5.3). Later on, we have improved the explanation why we should replace test cases that trigger other test cases in the first paragraph of Section 6.

Comment 5:

A related concern is that it is not clear at all whether the proposed approach always guarantees to find the minimal failure inducing schema (MFS) as it is defined in Definition 3.5. Note that once a failure inducing schema is found, the number test cases required to prove that this schema is an MFS grows exponentially with the number of options in the schema. It is not clear whether the proposed approach actually tries all the subsets or not. A proof is needed here!

Response: We have added one more section (Section 4) to formally show that, we do not need to execute all the test cases to identify the MFS. The proof is also given. What's more, we show that we can further reduce the number of test cases with some assumptions (Safe value assumption in this paper). Similarly, the formal proof is given in Section 4.

Comment 6:

Masking affects are not only caused by failures. Unaccounted for control dependencies, for example, can also cause masking effects, e.g., the use of –help command line option may make the sut to display a help text and exit without exercising any of the remaining option setting combinations appearing in the underlying configuration. Therefore, the authors can consider restating Definition 4.1. It is also important to mention that Definition 3.5 is not valid in the presence of masking effects when discussing this definition; not a couple of pages later.

Response: It's true that masking effects can be triggered by other events, such as unaccounted for control dependencies. As suggested by this comment, we have restated the definition of masking effects to be “A masking effect occurs when a test case t contains an MFS of a particular failure, but it does not trigger the expected failure because other unexpected event, such as other type of failure or unaccounted control dependency, was triggered earlier that prevents t from being normally checked”. The new definition considers the cause of masking effects more generally such that it is not limited to different failures. Additionally, we have re-emphasized that Definition 3.5 is not valid in the presence of masking effects in the third paragraph in Page 7.

Comment 7:

The proposed approach deserves a section of its own. As it is right now, the description of the approach is spread across a number of sections. For example, Section 5.2 titled “a case study using the replacement strategy” is the first section that attempts to describe how the approach works. However, as also indicated by its title, this section is supposed to introduce a feasibility study and the approach must have been introduced much more earlier. What is the input to the proposed approach? What is the exact algorithm implemented by the approach? When does the search for failure inducing combinations terminate?

Response: We agree that we should make the approach an independent section. Hence, after we describing the test case replacement strategy in Section 6, we added a new section, i.e., Section 7 to present our approach. In this section, we show how our approach works, including the specific inputs and outputs, how to identify the MFS (with which algorithm), when to replace the test cases, and when to terminate our approach.

Comment 8:

The original FIC_BS approach needs to be explained in the paper, which is not that big of a deal, so that readers are not forced to read other papers just to see what is going on. For example, just by reading the paper it is not clear how the “fixed parts” are determined and after reading the FIC_BS paper, one can only guess.

Response: As suggested, we have explained how FIC_BS works in Section 4. We also explain what the “Fixed part” exactly means (Section 4), i.e., the same part as that in the original failing test case (others are the mutated part).

Comment 9:

The third and sixth paragraphs in Section 2 suggest that the proposed approach will use some sort of static analysis, which is not true. Therefore, these paragraphs should be rephrased to avoid misleading the reader.

Response: As suggested, we have clarified that the proposed approach does not use static analysis. For example, we have removed the phrases like “traditional approaches” and “source code”. Also, we have emphases the main target of this paper in italic.

Comment 10:

To generate “desirable” configurations, a suspiciousness score is computed for every option and failure. What is the justification for using such a scoring scheme? Doesn’t it make more sense to compute the scores for combinations of option settings rather than for individual options?

Response: The idea to compute suspiciousness score for every option is first applied in combinatorial testing in [1]. In fact, the suspiciousness score is based on the intuition that, if one option value appears more frequently in the test cases which trigger a particular failure, it is more likely to contribute to that failure. This idea is originally inspired by the general spectra-based fault location technique—Tarantula [2,3].

Besides, as suggested, it is more reasonable to obtain the suspiciousness of combinations of options, i.e., schemas. In fact, our formula (EQ1), exactly gives the simplest way to compute the suspiciousness of a schema, i.e., Linear Averaging of the options in a schema. Although there exist other ways to define the suspiciousness of schema, e.g., considering the weight of different option value, we believe the linear averaging is the most common and straightforward way to compute the suspiciousness of the schema.

[1] Ghandehari, Laleh Shikh Gholamhossein, et al. "Identifying failure-inducing combinations in a combinatorial test set." Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on. IEEE, 2012.
[2] Jones, James A., Mary Jean Harrold, and John Stasko. "Visualization of test information to assist fault localization." Proceedings of the 24th international conference on Software engineering. ACM, 2002.
[3] Jones, James A., and Mary Jean Harrold. "Empirical evaluation of the tarantula automatic fault-localization technique." Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ACM, 2005.

Comment 11:

It is quite confusing that the two strategies introduced by the paper, namely “regarded as one failure” and “distinguishing failures”, are explained using the OFOT approach, which is a different fault characterization approach, than the one used in the experiments, namely FIC_BS. How was FIC_BS augmented with these strategies in the experiments? For example, with the “distinguishing failures” strategy what happens if a configuration reveals a different failure?

Response: It is true that we used the OFOT approach to describe the impact of masking effects on FII approaches, which is different than the FIC_BS approach. In fact, these two approach are very similar in theory, but OFOT is simpler to explain. As suggested, we have revised the explanation about the impact of masking effect such that it is based on the FIC_BS approach. Furthermore, for consistency, all the text that is related to the FII approach in this paper is revised to be based on the FIC_BS approach, including the sections about explaining MFS identification, masking effect, and the approach proposed in this paper.

Comment 12:

In the first paragraph of Section 5.1, it is not clear what is meant by the following sentence: “The replacement must satisfy the condition that the newly generated ones will not negatively influence the original identifying process.”

Response: This sentence clarifies that the replacement operation should follow some rules. For example, the newly generated test case should keep the same fixed part; otherwise, the original FII approach may not work. As suggested, we have removed this confusing sentence, and replaced it with this sentence: “Normally, when we replace the test case that triggers an unexpected failure with a new test case, we should keep some part of the original test case.”

The comments about the experiments.

Comment 13:

a) The authors need to further justify the use of synthesized subject applications and elaborate more on how these subjects were actually created.

Response: To address this comment, we have improved the description of the use of these synthesized subjects, as well as how they are created. Additionally, we have posted these subjects online, including how to set-up and run the experiments. (Details see: http://gist.nju.edu.cn/doc/multi/index.html and http://gist.nju.edu.cn/doc/multi/runex.html)

Comment 14:

b) What was the MeetEndCriteria used in the experiments? Why? A sensitivity analysis for this parameter would greatly benefit the work.

Response: In the new version of the paper, we removed the dynamic ending function MeetEndCriteria, and replaced it with the static ending condition: all the candidates have been tried (See line 1 of Algorithm 1 in Page 18 of the new version).

Comment 15:

c) The configuration space models used in the experiments are relatively small. Since the authors have already used simulations, they could have also evaluated the proposed approach on larger configuration spaces.

Response: We agree that the configuration models used in the experiments are relatively small. This is mainly because we need to perform the MFS identification for each failing test case. By doing this we can obtain a more general conclusion. If we increase the number of parameters in the testing model, the effort to conduct these experiments will increase exponentially. Additionally, according to the third reviewer’s comment (Comment 2), we removed some of those synthesized subjects, and conduct experiments one more real subject with two versions.

Comment 16:

d) The factors used for the real subject applications seem to be configuration options. Therefore, what were the actual test cases that were executed in the chosen configurations?

Response: It is correct that all the test cases in this paper are in fact test configurations. The actual test cases vary with the subjects. For example, for the HSQLDB program, our test case is one simple SQL command. For JFlex, our test case is a text file. Furthermore, for each subject, a single test case is used for different configuration options.

We note that multiple tests may be needed to check the correctness of the software for each configuration option. Furthermore, different tests may be needed for different configurations [1]. But in this paper, we use a simplified approach, i.e., one test for all the configurations, as we just want to check the masking effects due to configuration options. This is common in other FII approaches studies [2] [3] [4]. The test case-aware combinatorial testing uses multiple test cases, which, however, is beyond the scope of this paper.

[1] Yilmaz, Cemal. "Test case-aware combinatorial interaction testing." Software Engineering, IEEE Transactions on 39.5 (2013): 684-706.
[2] Nie, Changhai, and Hareton Leung. "The minimal failure-causing schema of combinatorial testing." ACM Transactions on Software Engineering and Methodology (TOSEM) 20.4 (2011): 15.
[3] Zhang, Zhiqiang, and Jian Zhang. "Characterizing failure-causing parameter interactions by adaptive testing." Proceedings of the 2011 International Symposium on Software Testing and Analysis. ACM, 2011.
[4] Ghandehari, Laleh Shikh Gholamhossein, et al. "Identifying failure-inducing combinations in a combinatorial test set." Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on. IEEE, 2012.

Comment 17:

e) Please rephrase “to construct many real testing objects [for evaluations] is time-consuming”.

Response: Fixed as suggested.

Comment 18:

f) What is the degree of an MFS?

Response: Since MFS is a schema, the degree of an MFS is the same as the degree of a schema

(Definition 3.3 in Page 6), which is the number of option values in the MFS.

Comment 19:

g) Why was the "ignored number" metric treated differently than the rest of the evaluation metrics? What was the issue with none of the MFS being ignored (paragraph 3 in Section 6.2.2)?

Response: The issue with none of the MFS being ignored is because we merged all the MFS identified in all the failing test case. Hence, there is a strong possibility that we can determine all the MFS. For example, if one MFS cannot be determined in one test case, we may identify the MFS in another test case. Hence, we may not ignore any MFS. Other metrics has no such problem, so they are different. However, as the next comment (**Comment 21**) said, the average performance metrics are more rational than the merged values. Thus, we have revised values for all the metrics, such that only the average values are given.

Comment 20:

h) In the Figures 3a-g, the points that belong to the ILP approach are almost impossible to distinguish from the rest.

Response: We have re-drawn this figure to make it clearer. Additionally, for some sub-figure that is hard to read (Fig. 5e of the new version of paper), we offer additional view of the original data to make the comparison more easily to read.

Comment 21:

One major concern is that the experiments reported in Section 6.2 assume that for a given failure, all the test cases (configurations) revealing the failure are known a priori, which is not realistic at all. In the experiments, given a configuration space, the space is exhaustively tested, and then for each failure, every test case revealing the failure is fed to the proposed approach, and then the MFSs individually obtained for each test case are combined. Therefore, the results reported in this section are over-approximations. To be fair, the proposed approach should not work on any failures that it hasn't discovered by itself, except for the input failure. For example, the average performance metrics obtained from failures could be reported.

Response: It's true that the evaluation for these approach was over-approximation. Instead of merging all the identified MFS for one approach, it's more rational to evaluate the approach against the MFS in the failing test case that it works on. Hence, we have separately evaluated the performance of these approaches in each failing test case, and only focusing on the MFS on that failing test case. At last, we showed the average performance among all the failing test cases.

Comment 22:

The discussion on page 24 states that ILP and "distinguishing failures" approaches showed the same or similar performance (i.e., similar exact number, sub number, super number, etc.), but ILP was more costly. I thought that ILP was one of the main contribution of this paper as “distinguishing failures” is a trivial extension for the existing FIC_BS approach. The authors should elaborate more on this.

Response: We actually mean that when compared to the strategy “regarded as same failure”, the results of the two approaches, i.e., ILP and distinguish failures, look similar. However, when comparing the approach ILP and distinguishing failures alone, the difference between them is not trivial. For example, for most subjects in Table XXI in the Appendix of the new version (Page 44), our approach can improve about 10% at metric aggregate against the strategy distinguish failures. As suggested, we have rephrased these sentences.

Comment 23:

In Section 6.3, it is stated that compared to the random test generation strategy, ILP reduced the number of test cases needed by 1 to 2, on average. What is the theoretical and practical significance of this reduction? Doesn’t it suggest that the masking effects used in the experiments were easy to avoid? The approach should be evaluated more rigorously.

Response: Here, we mean that for each failing test case, our approach can save 1-2 test cases. This number is about 7.1% to 14.2% of the overall number of test cases needed for MFS identification. As suggested, we emphasized this in the new version of this paper (last paragraph of Page 32). Additionally, we computed the statistical significance (t-test) of this reduction. The result shows that the reduction is significant.

By the way, the sentence in the paragraph “Aggregative for the five metrics” in Section 9.2.2. (Originally Section 6.2.2 in Page 24) i.e., “the possibility of triggering a masking effect is relatively small”, may mislead readers to believe that “the masking effects used in the experiments were easy to avoid”. In fact, what we mean is that by using “distinguishing failures”, the masking effects can be easily avoid. The reason is that the masking effects are monotonic in the testing objects we constructed for evaluation, which is favorable for the “distinguishing failures” strategy. For example, assume bug A masks bug B; then when we identify the MFS of bug A, the distinguishing failures strategy is the correct strategy, as if there is a test case trigger the bug B, then it must not trigger the bug A (otherwise, bug B will not be triggered). Hence, the probability that the distinguishing failures strategy makes the correct operation is at least 50%. According to this comment, we emphasized this in the new version of the paper (Page 31).

Comment 24:

Furthermore, I strongly believe that all the experiment sections should be rewritten, as it is quite difficult to follow what is going on and to reason about the results.

Response: As suggested, we have rewritten the experiment section. Our effort includes adding new software subjects, removing all the synthesized software, explaining more about how to create these subjects, updating the results of the experiments and the corresponding discussions.

Comment 25:

Section 6.4 compares the proposed approach with FDA-CIT. However, the comparisons don't seem to be fair. First, the authors should clearly mention that FDA-CIT's primary concern is to avoid masking effects and give every t-tuple a fair chance to be tested, not to perform fault characterization and that it can work with non-deterministic failures and in the presence of inter-option constraints. In particular, FDA-CIT uses fault characterization as a tool to reach its goals and it can work with other fault characterization approaches (as also noted by the authors).

Response: According to this comment, we have emphasized the following three points in our paper (See the last paragraph of Section 9.4.2 in Page 36 of the new version for more detail): 1) the main concern of FDA-CIT is to avoid masking effects when generate a covering array. 2) it can handle more practical problems, such as non-deterministic failures and inter-option constraints. 3) it can work with other fault characterization approaches.

Comment 26:

Second, in the experiments, FDA-CIT is used with t-way covering arrays, where 2 <= t <= 4. When working with failures caused by the interactions of t options, the strength of the covering arrays to be used with FDA-CIT should be at least (t+1). When a t-way covering array is used, very few instances of the failure-inducing t-tuple would appear in the array (only one instance in the worst case). When a lower-strength covering array is used, the failure inducing t-tuple may not even appear in the array, which prevents the approach to even reveal the failure. The comparisons are not fair because although there were 6-way failures (Table XIX), the highest strength used was 4. It should have been at least 7. This can clearly be seen in Figures 5a-c, where, as t increased, FDA-CIT's overall performance increased. Considering that the proposed approach required significantly more number of test cases than FDA-CIT (e.g., 172 vs. 626), the extra test cases required for the higher strength covering arrays may not affect the cost relationship much.

Response: Yes, as the reviewer said, it's not fair to just use t-way covering array for FDA-CIT to identify the MFS. Hence, for each experiment, we increased the strength by 1 for FDA-CIT, i.e., (t+1)-way covering array and updated all the results for FDA-CIT. Additionally, we have reduced some redundant test cases generated by our approach ILP (For those failing test cases that contained the existed test cases, we do not necessary conduct the MFS identification again). As a result, the difference in the number of test cases between our approach and FDA-CIT is very small.

Comment 27:

The paper should rigorously address the cost benefit tradeoffs. Assume that we are concerned with highly configurable systems where each factor corresponds to a configuration option. Then, what the paper refers to as a test case will correspond to a configuration. In such systems, a configuration is tested by running a battery of test cases in the configuration. For example, if we have 1000 test cases, given a configuration, we may want to execute all of these test cases in the configuration and determine the failure-inducing option setting combinations on a per test case basis. In such cases, however, the proposed approach needs to be carried separately for every test case (and even for every failure of a test case). That is, the number of configurations required by the proposed approach grows linearly with the number of test cases. However, this is not the case for FDA-CIT as test cases share configurations in FDA-CIT. Considering that the number of test cases required by the proposed approach is already significantly higher than FDA-CIT, this may cause scalability issues in real world scenarios.

Response: We agree with the scalability concern for practical applications. Yes, with no more adaptive control, the number of configurations needed by our approach will grow linearly with the number of test cases. However, in practice, there is no need to execute the test cases under the configuration which contain the identified MFS. This is because, according to the definition of MFS, each configuration containing MFS will fail. Hence, we just need to execute those test cases under other configurations which do not contain identified MFS. In this paper, we do not discuss this because our paper does not focus on the test case-aware testing environment. Also, we can use some adaptive approaches to reduce such redundancy, which is beyond the scope of this paper.

Comment 28:

More details should be provided about how the FDA-CIT approach was implemented. For example, the first paragraph on page 29 mentions about an “over fitting” problem. Were the classification trees computed by using n-fold cross validation to avoid over fitting as suggested by FDA-CIT? What was n? The same paragraph also suggests that every path from the root to a leaf node is considered as MFS. However, FDA-CIT only selects those paths (those MFSs) whose accuracy is above a given cutoff value. What was the cutoff value used in the experiments? How was this cutoff value selected?

Response: As suggested, we have added more details about our experiments. In particular, we have clarified that our implementation of FDA-CIT use n-fold cross validation. The value of n is set as the default value 10 (Weka). Additionally, the cut-off value is 1 in our experiment, that is, only the accuracy is 100 % will be set as MFS, while others are omitted. In fact, for the cutoff value, we have tried other values, but their performances are not good.

Comment 29:

It is good that the authors published the subject applications and the configurations used in the experiments online. However, it is not quite clear how to reproduce the experiments, e.g., how to configure these subject applications and how to run the test cases. Furthermore, the synthesized subject applications and the respective test cases are missing.

Response: As suggested, we have added more content online to explain how to run the experiment and to re-produce the outcomes (including the synthesized subject applications). (Details see:

http://gist.nju.edu.cn/doc/multi/index.html and http://gist.nju.edu.cn/doc/multi/runex.html)

Other Minor issues:

Comment 30:

+ Combinatorial interaction testing is often abbreviated as CIT, not as CT.

Response: Actually both abbreviations have been used in the literature. Specifically, CT is short for combinatorial testing [1][2], CIT is short for Combinatorial Interaction Testing [3][4]. In this paper, they are uniformly cited as Combinatorial testing (CT). To address this comment, we have clarified these two abbreviations in the footnote at Page 2.

[1] Nie, Changhai, and Hareton Leung. "A survey of combinatorial testing." ACM Computing Surveys (CSUR) 43.2 (2011): 11.
[2] Kuhn, D. Richard, Raghu N. Kacker, and Yu Lei. Introduction to combinatorial testing. CRC press, 2013.
[3] Yilmaz, Cemal. "Test case-aware combinatorial interaction testing." Software Engineering, IEEE Transactions on 39.5 (2013): 684-706.
[4] Cohen, Myra B., Matthew B. Dwyer, and Jiangfan Shi. "Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach." Software Engineering, IEEE Transactions on 34.5 (2008): 633-650.

Comment 31:

+ Second paragraph on page 14, the second “the left part” should be “the right part”
Response: Fixed as suggested.

Comment 32:

+ Fifth paragraph on page 14, “an scenario” -> “a scenario”
Response: Fixed as suggested.

Comment 33:

+ Second paragraph in Section 6.1.2, the sentence that starts with “In fact,” is incomplete.
Response: Fixed as suggested.

At last, special thanks for your valuable comments.

Responses to Referee: 2

Comment 1:

The major problem of the paper is the treatment of randomness in the empirical studies. Three of the four empirical studies involve some level of randomness (random replacement for studies 2 and 3, generation of 2-4 way coverage). The authors compute the average of 30 tries, but don't report on notions like confidence interval. Although I am not a statistician, I am convinced that there is not sufficient analysis to provide statistical evidence. Also, in study 4, you are comparing average values; so you should probably do ANOVA analysis or something similar.

Response: We agree that the lack of statistical evidence made our original results not convincing. Hence, for each experiment which contains randomness, for example, the randomly generated covering arrays, we have conducted t-test of the data. Then, the statistical significance value is given to determine whether the results of each experiment are reliable or not.

Detailed remarks

Comment 2:

Page 2 "We can get five two-way suspicious interactions..." five should be six!
Response: Fixed as suggested.

Comment 3:

Page 6 : definition 3.3, you should not use k in v_k_1 or v_k_t because k is intended to represent the number of parameters that influence the SUT.

Response: As suggested, we have replaced v_k_t with v_b_1.

Comment 4:

Page 6: Definition 3.5
The definition should be clarified. It states that "all test cases ... trigger ... failure F". This is not the case with the example of Table II where only 4 test cases trigger Ex1. In most cases, the failure is not triggered by ALL test cases. So you should make clear which is the set where all test cases trigger a failure.

Response: As suggested, we have rephrased the definition of failure-causing schema (Definition 3.5) as follows:
“If for any test case, as long as it contains the schema c, it will trigger the failure of particular fault F. Then we call c the failure-causing schema of F.”.

Comment 5:

page 9: Proposition 3.11 Could you give a proof of this proposition? Probably I don't understand the term "antithesis" correctly, but as far as I understand it, the antithesis is the exact opposite of the thesis. So if both thesis and antithesis are true, everything is true!

Response: We in fact misused the word “antithesis”. As suggested, we now provide a proof of this proposition (Page 9, fourth line from the bottom).

Comment 6:

page 11: typo in section 4.3 "and the other test cases (t2,t3) failed" should be "(t2,t4)".
Response: Fixed as suggested.

Comment 7:

page 12, first line "t1 and t3 should..." should be "t1 and t2 should..."
Response: Fixed as suggested.

Comment 8:

page 13 section 5.1, 4th paragraph, 2nd line "as it may not always BE possible..."
Response: Fixed as suggested.

Comment 9:

page 14 line 7 "the left part" should be "the right part"
Response: Fixed as suggested.

Comment 10:

page 16 section 5.2 "fixed part needed to be testED in each iteration"
Response: Fixed as suggested.

Comment 11:

page 17, Figure 2, The suspiciousness matrix is related to e2 and e3 and not e1 and e2.
Response: Fixed.

Comment 12:

page 17, Figure 2. I don't understand the choice of t4'. Why don't you choose "00001221", which has a better suspiciousness score?

Response: This is a very good point. It is true that (0 0 0 0 1 2 2 1) has a better suspiciousness score. But in our approach, some additional rules need to be followed when choosing a test case. Generally speaking, some part of the original failing test case should not be changed in the later generated test case. This part is called “fixed part” in this paper. For t4, (0, -, -, -, -, -, -, -) is the fixed part. Hence, t4 should keep this part, and the remaining part can be any value, as long as it is different from the original failing test case t0. All the test cases that replace t4, i.e., t4' and t4'', must keep this fixed part. This is why we choose (0 2 2 2 2 1 1 2) instead of (0 0 0 0 1 2 2 1), even though (0 0 0 0 1 2 2 1) has a better suspiciousness score.

Comment 13:

page 18, table XVI, what is the "bug pairs", how did you choose these pairs, why are they not the same in the versions of the same program? (were they fixed with the new version?).

Response: Yes, the bugs in the older version were fixed. Here, "bug pairs" are the IDs of two bugs in the same version of a software. We have removed the notion of "bug pairs". Instead, we use the notion "bugs' id". These bugs are collected in the bug tracker of the corresponding software.

Comment 14:

page 19, section 6.1.2 and table XVIII. How many tests of the "failures" count do contain the MFS? From your text, it seems that all failures did contain the MFS, and that none of the tests did contain the second fault alone. It also seems that there are only two faults in each version.

Response: It is true that each version of the original software only has two faults. It is also true that all the failures contain the MFS. But some tests only contain the second fault. This is because, if no test case can detect the second fault alone, then we do not know if there is another fault. Consequently, we cannot know if there exist any masking effect. To be more precise, we have re-written the original Table XVIII on page 18 to be the new Table XVIII in Page 26 (new version), which is also included below. In this table, we use the expression (#n) to represent the *n*th bug. Hence we can easily find how many test cases are triggered with a particular failure. In the masked column, we use the expression (#n -> #m#m') to directly show how many test cases should trigger failure (#m) and (#m') but is masked with failure (#n).

Software	Version	All tests	Failure			Masking	Total
HSQLDB	2.0rc8	18432	#1(2304)	#2(1152)	#3(1152)	#1>#2#3(768)	768 (16.7%)
	2.2.5	6912	#1(1728)	#2(1728)	-	#1>#2(576)	576 (16.7%)
	2.2.9	6912	#1(2304)	#2(768)	#3(384)	#1>#2#3(960) #2>#3(768)	1728 (50%)
JFlex	1.4.1	36864	#1(12288)	#2(12288)	-	#1>#2(6144)	6144 (25%)
	1.4.2	73728	#1(18432)	#2(18432)	-	#1>#2(6144)	6144 (16.7%)
Grep	2.6.3	384	#1(128)	#2(64)	#3(72)	#1>#2#3(80) #2>#3(16)	96 (36.4%)
	2.22	576	#1(192)	#2(64)	#3(80)	#1>#2#3(80) #2>#3(16)	6144 (28.6%)

Comment 15:

page 20 section 6.2 "we observed that the number of parameters...". On which case studies did you make these observations? the ones of table XVIII? Probably not the case studies of table XVIII as the largest one has 15 parameters (and not 30). So you should make more precise from which observations you deduced the number of parameters.

Response: We intended to say that the number of parameters is up to 15. However, by mistake we wrote 30. In fact, all the synthesized subjects we created have the number of parameters up to 15. This number is based on the observation of the original table XVII. As suggested, we fixed this sentence.

Comment 16:

page 22: last two lines testing object 2 has no best strategy, testing object 3 has distinguishing failures as best strategy. So the numbers do not correspond between this paragraph and figure 3.

Response: Yes, we agree. We have updated the figure and the corresponding description.

At last, we are grateful for your helpful comments.

Responses to Referee: 3

Comment 1:

The improvement over simpler strategies is not substantial. For example, using the suspiciousness score only reduces 1-2 test cases on average compared to using a random selection approach. Given the overhead of computing and maintaining the suspiciousness scores, random selection may be a more cost-effective solution. The benefits of using mutation and ILP do not seem to go beyond the very simple distinguish-failure strategy much, which simply does not consider the failing cases that do not belong to the same fault. The two have almost the same performance for 4/5 metrics in Fig. 3. Even for the metric that the proposed method shows benefits, I wonder if one can easily improve the performance of the distinguish-failure strategy by having more test cases.

Response: We agree that it was not clear that whether there exist significant advantages of our approach over those simpler strategies. Hence, we rephrased some of these sentences to make the conclusion clearer, which includes the following points.

First, when comparing to random approach, we said that our approach can save 1-2 test cases. Here, we mean that for each failing test case, our approach can save 1-2 test cases. This number is about 7.1% to 14.2% of the overall number of test cases needed for MFS identification. As suggested, we emphasized this in the new version of this paper (last paragraph of Page 32).

Second, for the simple distinguish-failure strategy, it seems that the benefits of using mutation and

ILP do not seem to be significant in the original paper. This because when compared to the strategy "regarded as same failure", the results of these three approaches, i.e., random, ILP and distinguish failures, look similar. However, when comparing the approach ILP and distinguishing failures alone, the difference between them is not trivial. For example, for most subjects in Table XXI in the Appendix of the new version (Page 44), our approach can improve about 10% at metric *aggregate* against the strategy "distinguish failures". By the way, the reason why the strategy "regarded as same failure" does not work as well as strategy distinguishing failures is that that the masking effects are monotonic in the testing objects we constructed for evaluation. That is, our study includes only the case that bug A always mask bug B, and does not consider cases that bug A can mask bug B and bug B can also mask bug A. This condition is favorable for the distinguishing failures strategy. For example, assume bug A masks bug B; then when we identify the MFS of bug A, the distinguishing failures is the correct strategy, as if a test case triggers the bug B, then it must not trigger the bug A (otherwise, bug B will not be triggered). Hence, the probability that distinguishing failures strategy makes the correct operation is at least 50%. According to this comment, we emphasized this in the new version of the paper (Page 31).

Third, we agree that by increasing the number of test cases, the performance of strategy distinguish-failures can be improved. In fact, our approach ILP is one of the versions that augment the distinguishing failures strategy by increasing the number of test cases. The additional test cases generated by ILP are used to replace some test cases that triggered other failures. We believe that there exist other approaches that can do this work. However, it may be challenging to choose which test cases to be added for MFS identification. This is because not all test cases are useful for MFS identification, e.g., test cases that cover the schemas which have already been covered by some other test cases may be redundant.

Comment 2:

The evaluation can be improved. It is currently evaluated on only 2 programs with 5 versions, each version having two bugs. I wonder why not collect more bugs for each version and have more programs (and versions)? I would rather see the author's trade the current space used in the over-detailed experiment set-up for more programs and bugs. The synthetic programs are not that useful.

Response: We agree that more real-life subjects with more bugs would make the evaluation of our approach more solid. Hence, we conduct experiments on one more real subject--Grep, with two different versions (each one contains two bugs). Furthermore, we have revised the description of how to set up these subjects, so that these experiments can be reproduced. As suggested, we have removed 5 synthetic programs in the paper.

Comment 3:

It is unclear what is the termination condition of this failure inducing interaction identification process. The paper seems to indicate that it terminates when the MFS is computed. But from my point of view, the computation of MFS is determined by the test suite you have. This seems to make it a chicken-and-egg problem.

Response: We agree that we did not properly describe the end-condition of our approach. In fact, the failure-inducing interaction identification is the process to filter those interactions that are not failure-inducing. To filter those interactions, we need to find some passing test cases that contain them. Then the ending condition is that no more interaction in the original failing test case can be filtered. A formal description of this ending condition is to generate and execute test cases until Lemma 4.2 (See the newly added Section 4) is satisfied.

According to the comment, we have improved the description of this ending condition in the Section 7.1 (lines 27 – 29 of Algorithm 2).

Comment 4:

Part of the technique hinges on properly classifying failures, which is a hard challenge in general. The authors should discuss how they achieve this.

Response: Yes, it is true that our approach is based on the failures classification, and it is a very hard challenge. However, in this paper, our key point is to discuss the impact of different failures on MFS identification. Hence, we do not focus on how to achieve this. In fact, in these experiments, we just simply treat these bugs with the same exception traces as the same failure, others as different. More complex techniques to handle this problem, such use the clustering techniques to classify the failures according to available information [1][2][3], are beyond the scope of this paper and not be discussed. As suggested, we have emphasized this in Section 8.2 at Page 23.

[1] Zheng, Alice X., et al. "Statistical debugging: simultaneous identification of multiple bugs." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.
[2] Podgurski, Andy, et al. "Automated support for classifying software failure reports." Software Engineering, 2003. Proceedings. 25th International Conference on. IEEE, 2003.
[3] Jones, James A., James F. Bowring, and Mary Jean Harrold. "Debugging in parallel." Proceedings of the 2007 international symposium on Software testing and analysis. ACM, 2007.

Comment 5:

I don't understand why ILP is needed. A simple linear search algorithm shall do the work. Please explain.

Response: It is true that a simpler linear search algorithm can also be applied in our approach to find a proper test case. However, as this problem to search a proper test case is related to Integer (a test case consists of discrete parameters with discrete values), we believe using Integer Linear Programming (ILP) technique is more appropriate. As suggested, we have emphasized this point in Section 6.1 at Page 19.

Comment 6:

The paper still contains a lot of grammatical problems. It has to go through very rigorous proof-reading.

Response: As suggested, we have tried to fix all these grammatical problems and have repeatedly checked the use of English in the paper.

Specifics

Comment 7:

Abstract: "theory lack"=> "theory lacks"
Response: Fixed as suggested.

Comment 8:

page 3: "newly regenerated" => "newly generated"
Response: Fixed as suggested.

Comment 9:

page 3: what is a factor?
Response: A factor is a parameter value in a test case (Or a schema).

Comment 10:

page 3: "suffered multiple failures" => "encountered multiple failures"
Response: Fixed as suggested.

Comment 11:

page 3: "import masking effects" => "induce masking effects"
Response: Fixed as suggested.

Comment 12:

Table II, there seems to be a soundness issue here. For ex2, it is possible that the programs just fails with
<7,2,4,5> and <11,2,4,5> but not <5,2,4,5>. But know the technique seems to think that <-,2,4,5> would fail
the program. How can your technique address this in general?

Response: In this motivation example, as the first parameter v_a just has two possible values, i.e., 7 or 11. Hence if this program fails with <7,2,4,5> and <11,2,4,5> for ex2, then <-, 2, 4, 5> must be a failure-inducing schema for ex2. On the other hand, if v_a has the third value 5, and <5, 2, 4, 5> does not fail, then we can just determine <7,2,4,5> and <11,2,4,5> are the failure-inducing schemas for ex2, not the schema <-, 2, 4, 5>. The determination of a schema to be a failure-inducing schema is just to find that if any test case, as long as contain this schema, will fail with one failure.

Comment 13:

page 7: "L: The number of failures ...". You should distinguish faults and failures (throughout the paper).
They have separate meaning in software engineering but the paper simply uses failures in all cases.

Response: We have revised use of the two terms throughout the paper. Additionally, we emphasize the relationship between the two notions in 7th paragraph of Page 6. In short, failure is what we observe when executing a test case, fault is what causes failure.

Comment 14:

page 7: "T(c)<=T_Fm", should it not be "T(c) >=T_Fm"?

Response: No, since T_Fm are all the test cases that fail with Fm, and c is just one MFS of Fm, T(c) <= Fm. This is because there may be multiple MFS for Fm. For the following example (2^3), T_F1 = (0, 0, 0), (0, 0, 1), (0, 1, 0). Two MFS are (0, 0, -) and (0, -, 0). Then we can find that T_F1 > T ((0, 0, -)), which are (0, 0, 0), (0, 0, 1).   According to this comment, we have emphasized this point.

Test case	Outcome
(0, 0, 0)	F1
(0, 0, 1)	F1
(0, 1, 0)	F1
(0, 1, 1)	Pass
(1, 0, 0)	Pass
(1, 0, 1)	Pass
(1, 1, 0)	Pass
(1, 1, 1)	Pass
MFS for F1	
(0, 0, -)	(0, -, 0)

Comment 15:

pages 8-9: The proofs are not that useful. They are quite obvious.

Response: We have removed some proofs that are obvious (PROPOSITION 3.6, LEMMA 4.2 and COROLLARY 4.4 of the revised version of the paper).

Comment 16:

page 9: "impacts of masking..." => "impact of masking", the same problem occurs a few times.
Response: Fixed.

Comment 17:

page 10: "one failure-fail"=> "one-fault failure"
Response: Fixed.

Comment 18:

page 11: "significantly impact on" => "has significant impact on", the same occurs a few times.
Response: Fixed.

Comment 19:

page 11: "We offer..." => "Consider"
Response: Fixed.

Comment 20:

page 11: "The pass of..." => "The passings"
Response: Fixed.

Comment 21:

page 12: "In other word",=> "In other words", the same happens a few times
Response: Fixed, all the "In other word" are replaced with "In other words".

Comment 22:

page 14: "between test case" => "between test cases"
Response: Fixed.

Comment 23:

page 14: "triggers other failure"=> "triggers other failures"
Response: Fixed.

Comment 24:

page 14: what do you mean by "the maximal possible failure"?

Response: We changed the term “maximal possible failure” to be “the most likely failure”. For example, in table XIII in page 18 (of the revised version) for test case t_3, F_2 is such a failure, as the suspiciousness between t_3 and F_2 is higher than F_3.

Comment 25:

page 14: "the corresponding failure", do you mean "all other failures" here?

Response: No. Here we mean the “most likely failure it can trigger” as discussed in the response to comment 24.

Comment 26:

page 16: "number of attempts is"

Response: We have revised this to be “the number of attempts to find a proper test case is”.

Comment 27:

page 19: "SUT have" => "SUTs have"
Response: Fixed.

Comment 28:

The observations in 6.1 are not new. So you are not gaining much out of this experiment

Response: Yes, the observation in Section 6.1 (Now Section 9.1) is similar to the existing study [1]. However, there is an essential difference, as we stated in the fourth paragraph of Section 9.1.2. The main difference between that work and ours is the way that the masking effect is quantified. In that work, the masking effect is the number of τ-degree schemas that only appear in the test cases that triggered other failures, where, the τ-degree schemas can be either MFS or not. Our work, however, quantifies the masking effect as the number of test cases that are masked by different failures. These test cases should contain some MFS, i.e., they should have triggered the expected failure if they did not trigger any other failure. As a result, the observation that “masking effects exist widely” does not have the same meaning in the two studies. Hence, we believe that the observation in Section 6.1 (Now Section 8.1 in the revised version) is new.

[1] Yilmaz, Cemal, et al. "Reducing Masking Effects in Combinatorial Interaction Testing: A Feedback Driven Adaptive Approach." Software Engineering, IEEE Transactions on 40.1 (2014): 43-66.

Comment 29:

page 20: "can happened"

Fixed. We changed the original sentence
"we did not build more complex testing scenarios such as the masking effects can happened in the form e1 > e2; e2 > e3; e3 > e1 or even e1 > e2; e2 > e1."
to be
"we did not build more complex testing scenarios such as masking effects in the form e1> e2, e2> e3, e3> e1or even e1> e2, e2> e1"

Comment 30:

page 24: what do you mean by "One issue is the redundancy...", please rephrase.

Response: We have rephrased that sentence to be the following:

"We believe this is an improvement, because too many sub- or super-schemas in fact point to the same actual MFS, which results in duplication and makes it hard to identify the actual MFS.".

Comment 31:

page 26: "produce the generated..."

Response: Has been rephrased to be "guarantee that the generated test cases should cover all the…".

Comment 32:

page 32: "of constrains"
Response: We rephrased this sentence to be: "Further study was conducted [Petke et al. 2013] to show that with consideration of **constraints**, higher-strength covering arrays with early failure detection are practical."

Comment 33:

page 32: "covering array with considering" => "covering array while considering"
Response: Fixed.

Comment 34:

page 32: "this constraint" => "these constraints"
Response: Fixed.

Comment 35:

page 32: "First, the work that aims..", not a sentence.
Response: We have rephrased it to be "First, we discuss the works that aim to identify the MFS in

the SUT”.

At last, we appreciated your comments, which are very useful in improving the quality of this paper.