Reviewer 1

Comment 1:

One weakness I see in the evaluation is in the choice of case studies. More specifically, a very common case in practice which is highly relevant to the suggested framework is that of a test space that contains many MFS of a low degree. Such a case is not represented in the evaluation. When there are many MFS of a low degree, there are many failing test cases, hence many calls to the identification component. In addition, the probability that a newly generated test case will contain multiple MFS that will cause an identification failure is significantly higher. As illustrated by Figure 2, such a test case will be skipped without contributing anything to the accumulated coverage, and a new test case will be generated instead. The question whether under these conditions the framework still achieves a reduction in the number of multiple MFS and in the total number of test cases should be evaluated.

**Response:**  It is true that our subjects did not include the SUT which contains many MFS with low degree, and we agree it is important to evaluate our framework in such condition. Hence, we added one more experiment to evaluate our framework with SUTs of different number of MFS. By doing so, we can observe whether the performance of our framework is sensitive to the number of low degree MFS. Considering we need to have subjects with various number of MFS of low degree (which makes using real software subjects impossible), we used toy SUT with injected MFS in this experiment so that we can control the number of MFS what is needed to identify.

Comment 2:  
  
One more topic I am missing is addressing test space constraints. Since almost all real-world industrial CT models contain constraints, the question how do they impact the suggested framework is of significant importance. For example, an implicit schema can stem not from the interaction of an MFS with other MFS but rather from the interaction between the MFS and the model constraints. How do the authors account for such lost schemas? It is also not mentioned whether the 5 case studies contain constraints. Incorporating case studies with constraints increases the validity of the evaluation results.

**Response:**  We agree that it is important to discuss the impact of test space constraints on our framework, as well as how to handle them. Hence, in the approach section, we explicitly described the constraints handling part in our framework (see the newly added Section 4.2.2). Specifically, the constraints are handled in the same way as those identified MFS. That is, we labeled them as forbidden schemas, and also computed the implicated constraints. As suggested, we also considered the implicated schemas tem from the interaction from MFS and constraints (the computation of this type of implicated schema is similar as those original implicated forbidden schemas). After this, we will remove these schemas to be covered and forbidden them to appear in the following iterations of our framework. Additionally, as suggested, we explicated showed the constraints in our 5 case studies in the experiments.

Comment 3:

Additional comments:  
- 4.2 before EQ5: the text mentions that the mutation should not include the "currently identified MFS". But the MFS is not identified yet - is the intention here the candidate MFS? This should be explained more clearly.

**Response:** It is true that we did not clarify clearly the term -- “currently identified MFS”. Here we mean the MFS that has been already identified in the previous iteration. in each iteration of our framework, we will identify the MFS in failing test case detected in this iteration. Hence, when we generate test cases in the following iteration, we should let it do not contain those MFS identified in previous iteration. As suggested, we have rephrased the “currently identified MFS” to be “already identified MFS“.

Comment 4:

- 5.3.2 masking effects: I did not understand the possible explanation suggested for the lack of gap between ict and sct in terms of masking effects. Was the intention that since sct covers the same schemas more times, the chances of them reappearing in passing tests hence reducing masking is higher? Please clarify.

**Response:** It is true that we did not clearly clarify the reason why the gap between ict and sct in terms of masking effects is trivial. The reasons are manifold. Specifically, for ict, while forbidding identified MFS in the latter generated test cases can reduce masking effects, but the incorrectly identified MFS may make this effort in vain. That is, if the schemas identified by our framework is not the real MFS, then it will not contribute to the reduction on masking effects. This conclusion can also be manifested in Table 10 (Page), where the f-measure of ict is not always to be 1, indicating that the MFS identified is not always correct. On the other hand, for sct, while it does not forbid any MFS in the test cases generation stage, but it generates more test cases than ict (many of them are redundant and cover the same schemas more times). Hence, sct may obtain more chances to revise their MFS identification. That is, if it incorrectly identifies the MFS in one failing test case, it may obtain the correct MFS in the next failing test case, and this obviously improve the performance on reduction of masking effects. As suggested, we have emphasized this point in this paper (Page, )

Comment 5:

- Potential future work: incorporate bug fixing information into the framework, i.e., MFS combinations becoming non-MFS - how can the framework utilize this information in an efficient and effective way?

**Response:** The reason why using bug fixing information can improve the effectiveness of our framework is that it can ensure that the schema identified is real MFS or not. As we had discussed in Section 4.5 (Paragraph ,), it is impossible to guarantee the identified schemas is MFS or not unless we execute all the possible test cases. The incorrectness of MFS identification, however, negatively impact the effectiveness of our framework. For example, if we forbidden a non-MFS in the following iteration of our framework, we lose chances to check whether it will trigger a failure or not. Consequently, using bug fixing information to improve the accurateness of MFS identification is appealing.

On the other hand, it is known that all the existed MFS identification approaches just give an approximation solution to identify MFS, and we need to execute all the test cases to ensure the identified schemas are real MFS or not. Hence, when aiming to improve the quality of the results of MFS identification approaches, using bug fixing information to assist the MFS identification approach (through a feed-back way) is a more effective way than exhaustive testing.

Comment 6:

- Reference number 23 is missing the author names

**Response:** Fixed as suggested.

Comment 7:

- The paper can benefit from proof reading as it contains numerous typos and grammar mistakes

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

**At last, we are grateful for your valuable comments.**

Reviewer 2

Comment 1:

The motivation of covering all t-wise interactions before moving to the debugging phase is not clear. Why not generate tests while identifying failure-inducing interactions then debug and fix the problems and then re-test by augmenting the test suites. This will probably result in reduced test suites.

**Response:** We agree that we did not clearly the motivation. Hence, we have empathized that our framework does not proceed sequentially, that is, start MFS identification after all the t-wise interactions has been covered. Instead, these two procedures interleaves each other in our framework. Additionally, we agree that using debugging information and fix the problems is helpful to our framework, as it will increases the accurateness of the MFS identification process, and hence reduces the number of test cases. As suggested, we have emphasized these two points in the beginning of Section 4 (Page), and the last paragraph of Section 4.1 (Page ), respectively.

Comment 2:

The idea behind the approach is quite similar with what is proposed in code-based fault localization, e.g., Jeremias Rößler, Gordon Fraser, Andreas Zeller, Alessandro Orso: Isolating failure causes through test case generation. ISSTA 2012: 309-319. Although different in context, I think the paper will benefit by discussing it since it relies on the same idea.

**Response:** We agree that there exist some similarities between the work [1] focused on code-based fault localization and our approach, especially at the adaptive part, i.e., both the two approaches combine failure isolation and test cases generation. Hence, we added one section to discuss the relationships and differences between them (See Page. )

[1]Röβler, Jeremias, Gordon Fraser, Andreas Zeller, and Alessandro Orso. "Isolating failure causes through test case generation." In *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, pp. 309-319. ACM, 2012.

Comment 3:

Similarly, the idea of generating CT tests by selected dissimilar tests and prioritizing them at the same time is related with the similarity t-wise test selection used in product lines, i.e., Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, Patrick Heymans, Yves Le Traon: Bypassing the Combinatorial Explosion: Using Similarity to Generate and Prioritize T-Wise Test Configurations for Software Product Lines. IEEE Trans. Software Eng. 40(7): 650-670 (2014).

**Response:** Yes, we agree. The Software Product Lines (SPL) testing problem is a very important field in Combinatorial testing (CT)[1-4]. Many techniques in CT have been applied on SPL testing [4], among which Henard C, et al. [1] considered both test cases generation and prioritizing (by selecting dissimilar tests). Also, our framework can be deemed as one type of handling test cases generation and prioritization problem, which aims at fault localization as well as fault detection. As suggested, we have added one section (Page, ) to discuss the relationships and differences between these two works.

[1] Henard C, Papadakis M, Perrouin G, Klein J, Heymans P, Traon YL. Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test suites for large software product lines. arXiv preprint arXiv:1211.5451. 2012 Nov 23.

[2] Perrouin G, Sen S, Klein J, Baudry B, Le Traon Y. Automated and scalable t-wise test case generation strategies for software product lines. In Software Testing, Verification and Validation (ICST), 2010 Third International Conference on 2010 Apr 6 (pp. 459-468). IEEE.

[3] Lopez-Herrejon RE, Javier Ferrer J, Chicano F, Haslinger EN, Egyed A, Alba E. A parallel evolutionary algorithm for prioritized pairwise testing of software product lines. In Proceedings of the 2014 conference on Genetic and evolutionary computation 2014 Jul 12 (pp. 1255-1262). ACM.

[4] Lopez-Herrejon RE, Fischer S, Ramler R, Egyed A. A first systematic mapping study on combinatorial interaction testing for software product lines. In Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on 2015 Apr 13 (pp. 1-10). IEEE.

Comment 4:

Additionally, the paper states, “in the generation stage, testers have no knowledge of the possible MFS, and surely it has opportunities that multiple MFS appear in the same test case”. It seems that dissimilar tests such as those produced by the above paper are the most appropriate to handle such cases.

**Response:** We agree that under some condition, e.g., when those MFS in SUT have similar characteristics (overlapped parameter values), using dissimilar tests prioritization can reduce the possibilities that multiple MFS appear in one test case. However, when those MFS distinct from each other, making tests dissimilar may not guarantee the appearance of multiple MFS. Hence, using the dissimilar tests [1] can be one potential, but not completed, solution to multiple MFS problem, as it depends on the characteristics of the MFS of the SUT. As suggested, we have emphasized this point in this paper ().

[1] Henard C, Papadakis M, Perrouin G, Klein J, Heymans P, Traon YL. Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test suites for large software product lines. arXiv preprint arXiv:1211.5451. 2012 Nov 23.

Comment 5:

I also think that a short discussion of combinatorial test generation approaches and code-based fault localization should be given, in the related work section.

**Response:** As suggested, we discussed some popular code-based fault localization works, e.g., Tarantula[1], The nearest neighbor [2], Delta debugging [3], in the related works (See paragraph in Page), and analyzed the similarities and differences between them with our work. Additionally, we believe these two type of works can be cooperated with each other to obtain better result of fault diagnosis [4].

[1] Jones JA, Harrold MJ, Stasko J. Visualization of test information to assist fault localization. In Proceedings of the 24th international conference on Software engineering 2002 May 19 (pp. 467-477). ACM.

[2] Renieres M, Reiss SP. Fault localization with nearest neighbor queries. In Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on 2003 Oct 6 (pp. 30-39). IEEE.

[3] Zeller A, Hildebrandt R. Simplifying and isolating failure-inducing input. Software Engineering, IEEE Transactions on. 2002 Feb;28(2):183-200.

[4] Ghandehari LS, Lei Y, Kung D, Kacker R, Kuhn R. Fault localization based on failure-inducing combinations. In Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on 2013 Nov 4 (pp. 168-177). IEEE.

Comment 6:

In the approach, why it is mandatory to forbid MFS when augmenting the test suites? It is possible that MFS can interact with input parts, not exercised by the employed test suite, and hide the fault. In other words, why is it assumed that every time that a specific failure-inducing schema is used it triggers the failures. This might not happen under higher strengths. I wonder whether this was observed in the conducted experiments.

**Response:**  We agree that in practice, it is possible that MFS can interact some inputs, so that it may not be observed in some test cases. In such case the schemas identified by our framework will be super-schema of the actual MFS. For example, considering a SUT 2\*3\*3, and assume that the actual MFS is (0, 1, -). Assume that the last factor (-, -, 1) will make the MFS hided, then we can list all the test cases as the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T1 | 0 | 0 | 0 | Pass |
| T2 | 0 | 0 | 1 | Pass |
| T3 | 0 | 0 | 2 | Pass |
| T4 | 0 | 1 | 0 | Fail |
| **T5** | **0** | **1** | **1** | **Pass** |
| T6 | 0 | 1 | 2 | Fail |
| T7 | 0 | 2 | 0 | Pass |
| T8 | 0 | 2 | 1 | Pass |
| T9 | 0 | 2 | 2 | Pass |
| T10 | 1 | 0 | 0 | Pass |
| T11 | 1 | 0 | 1 | Pass |
| T12 | 1 | 0 | 2 | Pass |
| T13 | 1 | 1 | 0 | Pass |
| T14 | 1 | 1 | 1 | Pass |
| T15 | 1 | 1 | 2 | Pass |
| T16 | 1 | 2 | 0 | Pass |
| T17 | 1 | 2 | 1 | Pass |
| T18 | 1 | 2 | 2 | Pass |

As we can see, T5 is pass, but it should fail as it contains (0, 1, -). As a result, the MFS identification result will not be (0, 1, -). This is because not all the test cases contain which (0, 1, -) fail. Instead, the schemas (0, 1, 0) and (0, 1, 2) will be regarded as MFS by definition. In the experiments, all the MFS is obtained according to the MFS definition (See Definition 4 in Page ), and we did not consider the situation that some factors will interact with the MFS and make it hided.

Comment 7:

I missed a discussion on how the input constraints are handled? Additionally, why higher t-wise strengths are not always resulting an improved precision? An explanation should be given.

**Response:**  It is true that we did not clearly clarify the input constraints handling part in our framework of the original paper. As suggested, we have explicitly described this part in the new version (see the newly added Section 4.2.2). Specifically, we formalized the constraints, and computed the possible interaction between them with existed MFS, i.e., the implicated forbidden schemas. After this, we will forbid the appearance of those test cases which satisfy these constraints (as well as those implicated schemas) to make testing valid.

Additionally, as suggested, we have emphasized the discussion why higher t-wise strengths are not always performing better in Page, paragraph. Specifically, the performance of our framework is related to the degree of MFS (i.e., the number of parameter values in the MFS) contained in the SUT. That is, if all the MFS in the SUT is of low degree, a low-wise covering array is enough to detect the MFS. This is because a t-wise covering array can detect all the failures caused by the MFS of t-degree, or less than t-degree. And if a MFS is detected, our framework can identify them as expected. It is surely that a higher-wise can also detect those low degree MFS. But compared to the low-wise covering array, it generates much more test cases. As a result, many failing test cases may contain the same MFS, and what’s worse, it increases the chance that a failing test case contains multiple MFS. This surely decreases the accurateness of MFS identification (As discussed in Section 3.2, Page).

Comment 8:

Why in TCAS the approach is not as good as in the other subjects? Is it because in TCAS the input combinations do not always trigger the mutants? I think that an explanation about this is important as it might indicate limitations of the proposed approach. Additionally, when discussing the results of TCAS the paper states, “Under this condition, both approaches will be transferred to a normal covering array”. Please revise the sentence, as it is unclear.

**Response:** Yes, it is because the test suite for TCAS does not always trigger the failures. In fact, all the MFS of TCAS are of high degree (t > 6), and the covering arrays (t = 2, 3, 4) rarely detect any of them. As see Table 10 (Page ), the results shows that the recall for TCAS is very low ( 0 .0 for all the 2, 3, 4 wise covering array), indicating that the MFS is rarely detected and identified. As suggested, we have emphasized this point in Page, Paragraph.

Additionally, we have revise the sentence “Under this condition, both approaches will be transferred to a normal covering array” to be “Under this condition, both approaches rarely detected the MFS, and hence the overall process will be transferred to be traditional covering array generation (the MFS identification process is omitted)”.

Comment 9:

I think that the paper does a pretty good job in evaluating its propositions on real world subjects. However, I believe that the employed subjects, tests and models should be available in the companion website of the paper. This will enable replication and will help researchers validate their CT approaches on these subjects. Additionally, the manual identification of the MFS introduces a validity threat, which can be reduced by making these data available.

**Response:** As suggested, we posted the subjects (with the setups) on line. See : for more details.

Comment 10:

I also think that the paper can benefit by adding some new results. These involve the performance of the examined approaches and the test size of a CT test suite that ignores the masking effects. The former will indicate the performance impact of the proposed approach on CT test generation and the latter the impact of masking on the test suite size.

**Response:** Yes, we agree. As suggested, we have showed the performance of the examined approaches, i.e., SCT, ICT and FDA-CIT (See page ). Also, we have listed the number of test cases which ignores the masking effects (See page).

**At last, special thanks for your helpful comments.**

Reviewer: 3  
  
Comment 1:

First of all, it is not clear whether the ultimate goal of the proposed approach is to identify failure-inducing option setting combinations or to obtain full coverage under the tested t-way coverage criterion or both. The authors should make this clear in the paper.

**Response:** We agree that we did not clearly clarify the ultimate goal of the proposed approach. In fact, our approach aims to give a more efficient framework than traditional sequential procedure (first to generate covering array and then conduct MFS identification), which can make test cases generation and MFS identification cooperated with each other in a more rational way, so that it can reduce the number of generated test cases, as well as improve the quality of MFS identification. Hence, the ultimate goal of the proposed approach is to give a more efficient framework. MFS identification and tested-t-way coverage are just two components of this framework. At same time,

As suggested, we emphasized this in Page .

Comment 2:  
  
Regardless of the ultimate goal, one major concern is the contribution of this work. Using a greedy, one-configuration-at-a-time approach to compute covering arrays, changing one option setting at a time (OFOT) for fault localization, expressing failure inducing combinations as constraints to avoid previously known failing sub-spaces, and feedback-driven, adaptive CIT are not new ideas at all. The proposed approach simply combines OFOT with covering array generation in a rather trivial way, such that likely failing sub-spaces are avoided and that previously covered combinations are not required to be covered repeatedly.

**Response:**  This paper does not simply propose an approach that combines OFOT with CA generation. In fact, we propose a novel framework which combines MFS identification with covering array generation. The MFS generation can be either OFOT [1], FIC [2], TRT [3], or other MFS identification approaches as long as it can identify the failure-inducing combinations in a failing test case. And the CA generation approach can be AETG [4], DDA [5], or other covering array generation approaches (as long as it generates one test case at one time). This framework, although belong to adaptive combinatorial testing [6], but works in a different way than all the existed adaptive combinatorial testing works [6][7][8]. There are three main differences: 1) we do not generate a complete t-way covering array at first; instead, when a failure is triggered by a test case, we immediately terminate test cases generation and turn to MFS identification. 2) our process is fine-grained. That is, both the test cases generation and MFS identification must satisfy the “one test case at one time” criteria. 3) our framework tries to alleviate the impacts of the three problems, i.e., test cases redundancy, multiple MFS appearing in one failing test case and masking effects, in combinatorial testing.

Additionally as suggested by Comment 7, we had augmented our original framework with the measures to handle multiple MFS, which improves the contribution of this paper.

[1] 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.

[2] 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.

[3] Niu, Xintao, et al. "Identifying failure-inducing combinations using tuple relationship." Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on. IEEE, 2013.

[4] Cohen, David M., et al. "The AETG system: An approach to testing based on combinatorial design." Software Engineering, IEEE Transactions on 23.7 (1997): 437-444.

[5] Colbourn, Charles J., Myra B. Cohen, and Renée Turban. "A deterministic density algorithm for pairwise interaction coverage." IASTED Conf. on Software Engineering. 2004.

[6] Nie, Changhai, Henry Leung, and Kai-Yuan Cai. "Adaptive combinatorial testing." Quality Software (QSIC), 2013 13th International Conference on. IEEE, 2013.

[7] Dumlu, Emine, et al. "Feedback driven adaptive combinatorial testing." Proceedings of the 2011 International Symposium on Software Testing and Analysis. ACM, 2011.

[8] Yilmaz, Cemal, et al. "Moving forward with combinatorial interaction testing." Computer 2 (2014): 37-45.

Comment 3:  
  
Another issue is that throughout the paper it is claimed that the proposed approach determines minimal failure-causing schema (MFS) as it is defined in Definition 4. However, as also discussed in Section 4.5, the proposed approach does not guarantee to find MFS. Therefore, either the definition or the terminology used throughout the paper should be changed, because the proposed approach can in general determine \*\*likely\*\* failure-inducing option setting combinations, nothing more.

**Response:**  Yes, we agree. As suggested, we have changed this terminology used throughout the paper to be “likely MFS” or “likely failure-inducing combinations”.

Comment 4:  
It is good that the authors summarized the shortcomings of traditional CIT approaches in the presence of masking effects and multiple MFS in a single configuration. However, it is not clear how masking effects differ from multiple MFS. I would say that multiple MFS may cause masking effects. If so, these are not two separate concepts (in the sense that one causes the other) and they should be treated accordingly in the paper.

**Response:**  Yes, we agree that multiple MFS may cause masking effects. As suggested, we re-treated the two concepts through all this paper, i.e., making masking effects as a sub-effect of the multiple MFS (See Page,).

Comment 5:

On a related note, dealing with masking effects/multiple MFS is crucial for the proposed approach, as both the effectiveness and the efficiency of the proposed approach can greatly suffer in the presence of them (as also noted by the authors). The paper claims that the proposed approach can deal with multiple MFS, thus masking effects, in a single configuration.  However, it turns out that this is due to a heuristic, which aims to reduce the likelihood of having multiple MFS in a single configuration, which in turn is due to the way the proposed approach operates, i.e., one failure at a time. That is, the proposed approach does not guarantee to avoid all multiple MFS/masking effects, as was also observed in the experiments. For example, in the example given in Figure 5, which is used to illustrate the proposed approach, if testing started with configuration “0001” instead of “0000”, none of the failure inducing combinations would have been found, thus the full coverage under tested t-way coverage criterion would not have been obtained. Considering the current level of contribution of the paper, developing approaches for resolving multiple MFS/masking effects once they surface themselves, can greatly improve the contribution of the paper.

**Response:**  We agree the point that our approach cannot guarantee to avoid all multiple MFS/masking effects. That is, we cannot ensure the appearance of test cases which contain multiple MFS. As suggested, we additionally developed a measure when facing such condition (when encounter multiple MFS in one test case). (See Page ). Specifically, based on the description in the Section 3.2 (Page), we can easily detect if a failing test case containing multiple MFS (cannot obtained a passing test case through mutating any parameter value in this failing test case). Under this condition, we just need to omit this test case, i.e., non-schemas in this test case would be set as covered. Furthermore, we will forbid the appearance of this test case in the latter iteration as it contains multiple MFS. In other word, we will just handle the test case which contains single MFS and ignore those test cases which contains multiple MFS.

It is noted that this measure we proposed now does not completely handling multiple MFS problem. If there exists two MFS in one failing test case, and they have overlapped part, then we cannot determine if this test case contains single MFS or multiple MFS (as if the overlapped part is mutated, the two MFS in this failing test case will be broken at the same time, and it will pass). However, the multiple MFS which has overlapped parts has already been solved in our prior work [1], so we do not introduce this in this work.

[1] Niu, Xintao, et al. "Identifying failure-inducing combinations using tuple relationship." Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on. IEEE, 2013.

Comment 6:  
  
In the experiments, the proposed approach is compared to FDA-CIT – a feedback driven, adaptive CIT process. However, there are several issues that need to be addressed with these experiments:  
First, the proposed approach assumes that all failures are deterministic, which should be mentioned and discussed early in the paper, as this greatly reduces the practicality of the approach.

**Response:**  We agree that our approach is based on the assumption that all failures are deterministic and it should be mentioned early in the paper. As suggested, we discussed this assumption in Page ().

Comment 7:

Second, it is not clear how the first configuration to start the proposed approach was chosen. This is important because the performance of the proposed approach depends on the first configuration (especially in the presence of multiple MFS).

**Response:**  All the test configurations (include the first one) generated by our approach is according to Algorithm 1 (Page). Specifically, we just choose the test configuration that covers the most uncovered schemas, and make sure that it does not contain any constraints. Note that as it is the first configuration, we do not need to avoid the identified MFS (As the MFS identification process is not started yet). It is also noted that, this first test configuration may vary with the covering array generation algorithm we use. Even though we use the same covering array generation algorithm, it may also different for each run of our approach (as the algorithm may contain some random aspects).

Additionally, we agree that if the first configuration is not properly selected (for example, it contains multiple MFS), the performance our approach may be influenced. As suggested by Comment 5, we have weaken this problem.

Comment 8:

Third, the number of configurations required by the identification part of the proposed approach grows linearly with the number of configuration options. However, in the experiment the maximum number of options used was 13, which is quite small. For example, the size of a 2-way covering array created for 10 binary options, can be 6, whereas that created for 6435 binary options can be 16. That means that while the proposed approach will require 10 additional test cases for locating a single MFS in the first case, it will require 6435 additional test cases for the same failure in the second case. On the other hand, FDA-CIT will use 16 rather than 10 test cases to determine the likely failure causes, as FDA-CIT does not require additional test cases for identification. When this coupled with the fact that most of the test cases required by the proposed approach were used by the identification part (Table 8), it necessitates that, to perform a fair comparison, the empirical studies reported in the paper should be repeated by systematically increasing the number of configuration options.

**Response:** We agree that using OFOT as the MFS identification approach in our framework is not efficient, as it grows linearly with the number of configuration options. We also agree that FDA-CIT needs a small number of test cases even though the SUT has large number of configuration options. As suggested, we conducted one more experiment to evaluate the sensibility of both approaches in terms of the number of configuration options in SUT (See page ). The result shows that FDA-CIT has a better performance under such condition.

Comment 9:  
  
Fourth, the proposed approach assumes that only one test case is used for testing. Here, I distinguish between configurations and actual test cases used in these configurations to test the system under test. For example, what if you have hundreds of test cases to run in each of the selected configurations. Note that each test case can have different failing patterns. It seems like the proposed process should be carried out separately for each test case, as it may not be safe to invalidate a failure-inducing combination discovered for a particular test case when running other test cases. If so, the number of additional configurations needed will grow linearly with the number of configuration options times the number of failing test cases. Therefore, for a fair comparison, the performance of the proposed approach should be compared to that of FDA-CIT in the presence of multiple test cases.

**Response:** Yes, we agree that if one configuration has multiple test cases, we should separately handle each of them as different test cases may contain different MFS. Under this case, FDA-CIT is a better choice, as it does not need additional test cases for MFS identification and can handle the multiple test cases, i.e., test case-aware condition. Considering that all the subjects just has one test case for each configuration in the experiments, we added one paragraph to discuss the impact of multiple test cases on our approach, and we also discussed the differences between our approach and FDA-CIT at handling such problem.

Comment 10:

Fifth, it is strange to see that while the F-measures obtained from FDA-CIT were so low (Table 13), the tested t-way coverage measures for FDA-CIT were similar or better than those obtained from the proposed approach (Table 14), especially for large values of t, e.g., t=3 and t=4. Could this be because of the way the F-measures were computed? Seems like automatically identified failure-inducing combinations were symbolically compared with actual combinations, which may be misleading. For example, FDA-CIT can determine a portion of the actual failure-inducing combination at each iteration. Therefore, all the portions related to the same failure should be combined before any performing any comparison. Furthermore, in FDA-CIT, superfluous options can crept into the classification models to protect the integrity of the models, for example to ensure that the classification tree has a single root. Therefore, it may make more sense to compute precision, recall, and f-measures in terms of the correctly/incorrectly identified failure causing schemas of degree t, rather than symbolically comparing the option setting combinations.

**Response:** Yes, in the original version we just compared the failure-inducing combinations obtained at the last iteration. As suggested, we changed the way that F-measure is computed for FDA-CIT. Specifically, we combined the portions obtained at each iteration, and then compared them with the actual MFS. The new results show that there exists an improvement on FDA-CIT as expected.

Comment 11:  
  
Furthermore, it is not clear how the faulty versions of the subject applications used in the experiments were chosen. For example, only one faulty version marked as #55905 seems to have been chosen for Tomcat (Table 6). Why and how?

**Response:** The faulty version of each software is selected through searching the bug-tracker, with key-words: configurations and options. We just selected the version with faults which are option-related, because they can be easily modeled into combinatorial testing scenario.

Comment 12:

Section 4.2 can greatly be shortened as it simply describes a greedy, one-configuration-at-a-time covering array construction approach. The equations introduced in this section do not really help, as they are not used in the remainder of the paper.

**Response:** As suggested, we have shortened this paragraph. However, we kept those equations, as it can give a formal and accurate description of our approach.

Comment 13:  
  
Section 2.1: For a better taxonomy of construction methods for covering arrays, the author should refer to Nie et al.’s survey (ref [38] in the paper.)

**Response:** Fixed as suggested.

Comment 14:

Author names are missing from references [15] and [23].

**Response:** We have added the names of these two references.

Comment 15:

Line 26, second column, page 4: a space character is needed before parenthesis.

**Response:** Fixed as suggested.

Comment 16:

Line 34, first column, page 6: “a validate schema” -> “a valid schema”

**Response:** Fixed as suggested.

Comment 17:

Line 10, second column, page 6: “that was” -> “that were”

**Response:** Fixed as suggested.

Comment 18:

Line 46, second column, page 14: “One the other hand” -> “On the other hand”

**Response:** Fixed as suggested.

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