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 evaluation did not include subjects that contain many MFS with low degree, and we agree it is important to include such subjects. Hence, we added one more experiment (Section 5.5 on page 17) to evaluate our framework with SUTs with different numbers of MFS. Doing so allows us to observe whether the performance of our framework is sensitive to the number of low degree MFS. Considering we need to have subjects with various numbers of MFS of low degree (which makes it difficult to use real software subjects), we created SUTs with injected MFS in this experiment so that we can control the number of MFS that are needed to identify. The results of the newly added experiment shows that with the increase of number of MFS in the SUT, the performance of all three approaches (ict, sct, and fda-cit) decreases, but our approach ict still performs better than the other two approaches when the number of MFS is not extremely high, which is the common scenario in practice.

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 on Page 9). In short, the constraints are handled in the same way as those identified MFS. That is, we label them as forbidden schemas, and also compute the implicit constraints. By doing this, the implicit schema stem from the interaction between the MFS and the model constraints can also be found, as the MFS and constraints are handled in the same way (forbidden schema). After this, we remove these schemas to be covered and prevent them from appearing in the subsequent iterations of our framework. Additionally, as suggested, we now show the constraints in our 5 case studies in the experiments (See the second paragraph of the right column on Page 11). Specifically, there are two constraints among the subjects. The first constraint is from Tomcat, of which the error page location must not be empty. The second one is from Hsqldb, of which you can only process with the “next()” method in a non-scrollable result set.

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:** The term, “currently identified MFS”, refers to the MFS that has been already identified in the previous iteration. In each iteration of our framework, we identify the MFS in failing test cases detected in this iteration. Hence, when we generate test cases in the next iteration, we do not allow them to contain those MFS identified in previous iteration. As suggested, we have renamed the term, “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 reason is manifold. Specifically, for ict, while preventing the identified MFS from appearing in the latter generated test cases can reduce masking effects, an 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 reduce on masking effects. This conclusion can be derived from Table 10 (on Page 13), where the f-measure of ict is not always 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, it generates more test cases than ict (many of them are redundant and cover the same schemas multiple times). Hence, sct may be more likely 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 (In the second paragraph from bottom of the left column on Page 15).

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 help to check whether the schema identified is real MFS or not. As we discuss in Section 4.5 (on Page 10), it is impossible to guarantee the identified schemas is MFS or not unless we execute all the possible test cases. Incorrect MFS identification, however, negatively impacts the effectiveness of our framework. For example, if we do not allow non-MFS to appear in the following iteration of our framework, we would not be able to check whether it will trigger a failure or not. Consequently, using bug fixing information to improve the accuracy of MFS identification is appealing.

On the other hand, it is known that all the existed MFS identification approaches just give an approximate 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 result of an MFS identification approach, using bug fixing information to assist the MFS identification approach (through a feed-back way) is more effective 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 explain the motivation. Hence, we have empathized that our framework does not proceed sequentially. That is, it does not start MFS identification after all the t-wise interactions has been covered. Instead, these two procedures interleave each other in our framework. We agree that using debugging information to fix the problems is helpful to our framework, as it will increase the accuracy 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 6), and the last paragraph of Section 4.1 (Page 7), 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. In particular, the adaptive part in [1] also uses feedback from test outcomes to guide test generation, and also leverages test case generation for debugging purposes. Hence, we added one paragraph to discuss this work (See the last paragraph of the right column on Page 20).

[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 to 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 considered to be one solution to the test cases generation and prioritization problem, which aims at fault localization as well as fault detection. As suggested, we have added one paragraph to discuss this work (last paragraph of Section 6 on Page 21).

[1] Henard, C., Papadakis, M., Perrouin, G., Klein, J., Heymans, P., & Le Traon, Y. (2014). Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines. Software Engineering, IEEE Transactions on, 40(7), 650-670.

[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 possibility that multiple MFS appear in one test case. However, when those MFS are distinct from each other, making tests dissimilar may not guarantee the appearance of multiple MFS. Hence, using the dissimilar tests [1] may not always solve the multiple MFS problem, as it depends on the characteristics of the MFS of the SUT. As suggested, we have emphasized this point in paragraph (last paragraph of Section 6 on Page 21).

[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 have added one paragraph to discuss combinatorial test generation approaches in the second paragraph of Section 6 (Page 20). Following that paragraph, we also added one paragraph to discuss existing works on MFS identification.

We also agree that code-based fault localization is correlated to the MFS identification approaches. In fact, two MFS identification approaches [1][2] are directly inspired by the delta debugging ideas [3]. As suggested, we discussed some of the code-based fault localization work in the related works (See the second paragraph from bottom of the right column on Page 20). Additionally, we believe these two types of works can be integrated to obtain better results of fault diagnosis [4].

[1] Z. Zhang and J. Zhang, “Characterizing failure-causing parameter interactions by adaptive testing,” in Proceedings of the 2011 International Symposium on Software Testing and Analysis. ACM, 2011, pp.331–341.

[2] J. Li, C. Nie, and Y. Lei, “Improved delta debugging based on combinatorial testing,” in Quality Software (QSIC), 2012 12th International Conference on. IEEE, 2012, pp. 102–105.

[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 with 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 hidden, 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 passes, 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 (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 are obtained according to the MFS definition (See Definition 4 on Page 3), and we did not consider the situation that some factors will interact with the MFS and make it hidden.

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 of our framework in the original paper. As suggested, we have explicitly clarified how they are handled in the new version (see the newly added Section 4.2.2 on Page 9). Specifically, we formalize the constraints, and compute the possible interaction between them and existed MFS, i.e., the implicit forbidden schemas. After this, we will exclude the test cases which satisfy these constraints (as well as those implicit schemas) to make testing valid.

Additionally, as suggested, we have added the discussion why higher t-wise strengths are not always performing better in the last paragraph of the right column on Page 12. Specifically, the performance of our framework depends on the degree of MFS (i.e., the number of parameter values in the MFS) in the SUT. That is, if all the MFS in the SUT are of low degree, a low-strength 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-strength covering array can also detect those lower degree MFS. But compared to the lower-strength covering array, it generates many more test cases. As a result, many failing test cases may contain the same MFS. Furthermore, it increases the chance that a failing test case contains multiple MFS. This decreases the accuracy of MFS identification (As discussed in Section 3.2, Page 5).

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 shown in Table 10 (Page 13), the results shows that the recall for TCAS is very low (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 the second paragraph of Section 5.2.2.

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 made the subjects (with the setups) available online. See:<http://gist.nju.edu.cn/doc/ict> 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 shown the performance of the examined approaches (See Table 11 on Page 13). Also, we have listed the number of test cases that do not suffer from the masking effects (See Table 13 on Page 15, and Table 17 on Page 18).

**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 provides a more efficient framework than the traditional, sequential framework (i.e., first generate and execute covering array and then conduct MFS identification). This framework allows test case generation and MFS identification to be integrated more closely, so that it can reduce the number of generated test cases and also improve the quality of MFS identification. As suggested, we emphasized this in the first paragraph of Section 4 (Page 6).

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 proposes a general framework that combines MFS identification with CA generation. In the framework, the MFS generation can be either OFOT [1], FIC [2], TRT [3], or other MFS identification approaches that can be used to identify the failure-inducing combinations in a failing test case. CA generation can be performed using AETG [4], DDA [5], or other covering array generation approaches that generate one test case at a time.

This framework can be considered to be adaptive combinatorial testing [6]. However, it works in a different way than all the existing adaptive combinatorial testing approaches [6][7][8]. There are three major differences: 1) we do not generate a complete t-way covering array up front; 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 impact 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 have augmented our original framework with measures (See Section 4.2.1 at Page 8) 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 added one paragraph (last paragraph in Section 2, Page 4) to emphasize this point.

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. These two concepts are concerned with different aspects of combinatorial testing. The masking effects is mainly concerned with the test adequacy of CT, which can be regarded as a metric to evaluate how many schemas are actually tested [1]. While for multiple MFS problem, it is mainly concerned with the quality of MFS identification. For clarify of presentation, we separately discuss these two concepts later in this paper. As suggested, we have added one paragraph (last paragraph in Section 3, Page 6) to emphasize this point.

[1] C. Yilmaz, E. Dumlu, M. Cohen, and A. Porter, “Reducing masking effects in combinatorial interaction testing: A feedback driven adaptive approach,” Software Engineering, IEEE Transactions on,vol. 40, no. 1, pp. 43–66, Jan 2014

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 that our approach cannot guarantee to avoid all multiple MFS/masking effects. As suggested, we have developed a measure to deal with such condition (when encountering multiple MFS in one test case). (See Section 4.2.1 on Page 8). This measure is inspired by the interim method proposed by Zhang [1], we find the method FIC– a mutated version of OFOT, can work well under the multiple MFS condition. The mechanism of FIC is very similar to OFOT. Specifically, when identifying the MFS in a failing test case, it also mutates one factor at a time to generate one additional test case. The only difference is that it will not always rollback to the original value it has mutated when it goes on mutating other values (only when a passing test case appears, it will roll back to the original value). This operation will break multiple MFS in one test case and finally only one MFS remains to identify. We added an example to explain the details of this measure (See Table 5, Page 8)

It is noted that this measure does not completely handling multiple MFS problem. For example, if additional test case introduced new MFS, we still cannot get the correct MFS. The newly introduce MFS problem has already been discussed in our previous paper [2], in which we find that more test cases are needed to be generated to alleviate the impact caused by the newly introduced MFS. This point is, however, beyond the scope of this paper.

[1] Z. Zhang and J. Zhang, “Characterizing failure-causing parameter interactions by adaptive testing,” in Proceedings of the 2011 International Symposium on Software Testing and Analysis. ACM, 2011, pp.331–341.

[2] 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:**  As suggested, we discussed this assumption in the last paragraph of Section 2. (Page 3, the paragraph before Section 2.1).

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 follow Algorithm 1 (Page 8). 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 has 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 be different for each run of our approach (as the algorithm may contain some random aspects).

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 weakened 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 very 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 a large number of configuration options. As suggested, we conducted one more experiment to evaluate the sensitivity of both approaches in terms of the number of configuration options in SUT (See the newly added section 5.5.3, page 19). 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. In this case, FDA-CIT is a better choice, as it does not need additional test cases for MFS identification and can handle multiple test cases, i.e., test case-aware condition. Considering that all the subjects just have one test case for each configuration in the experiments, we added one paragraph (last paragraph in Section 5.4, page 17) 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 a 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:** The f-measure in our paper is computed according to the correctly/incorrectly identified failure causing schemas. The reason why fda-cit not performs as good as ict is mainly because fda-cit's primary concern is to avoid masking effects and to give every t-degree schema a fair chance to be tested, not to perform fault characterization. On the other hand, for the classification tree method, when only a very small set of test cases fail, it will result in the input data for CTA to be highly unbalanced [1]. Another point is that all the MFS identified by the classification tree method should contain the same parameter value on the root, which result in the schemas identified by fda-cit will tend to be super-schema of the real MFS. Although this leads to the f-measure of MFS identification being lower than that of ict, it does not have much negative impact on the masking effects reduction. This is because forbidding the appearance of super-schema of real MFS will not reduce the number of schemas that is not MFS to be tested. As a result, the tested-t-way coverage is not changed.

To be more clear, we have emphasized this point in the last paragraph of the right column on Page 16.

[1] J. Zhang, F. Ma, and Z. Zhang, “Faulty interaction identification via constraint solving and optimization,” in Theory and Applications of Satisfiability Testing–SAT 2012. Springer, 2012, pp. 186–199.

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 as a 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 keep 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:** As suggested, 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.**