View Reviews

Paper ID
489

Paper Title
BugDoc: Debugging Computational Pipelines

Track Name
Research Paper Second-Round

Reviewer #1

Questions

1. Overall Evaluation
Weak Reject

2. Reviewer's confidence
Expert

3. Originality
Medium

4. Importance
Medium

5. Summary of the contribution (in a few sentences)
The paper focuses on debugging computational pipelines. The problem is important and interesting, and the authors follow a principled approach, in particular for the experimental evaluation of their system, BugDoc. The paper is clearly written and easy to follow. There are however two critical issues:

1. The technical contributions appear to be somewhat thin.
2. There is a very close connection to the area of fault localization, which the paper ignores.

6. List 3 or more strong points, labelled S1, S2, S3, etc.
1. The problem is interesting and important.
2. The approach is principled and the writing clear.
3. The evaluation is careful, detailed, and extensive (except for the unexplored connections to fault localization).

7. List 3 or more weak points, labelled W1, W2, W3, etc.
1. An important issue is that the contributions are somewhat thin. The meat of the paper is in Section 3, and the methods there are quite simple. I am concerned about whether the contributions rise to the level of a SIGMOD paper.

2. Another critical problem is that the paper overlooks the vast software engineering literature on fault localization, which appears to be a precise application of this problem setting.

3. There are some other issues with respect to clarity of the problem definitions, though I would expect that these could be fixable through a revision.

8. Detailed evaluation. Number the paragraphs (D1, D2, D3, ...)
The paper focuses on an important and interesting problem. Computational pipelines can fail for a variety of reasons, and we need assistance in tracing where the problem originated. BugDoc focuses on this problem: Given

traces of a computational pipeline (different executions, some failing and some succeeding), the goal is to find the root causes of the errors.

The writing of the paper is good overall, but I find that crucial details are missing, and hinder our understanding early on. For example, Section provides the foundations of the paper's focus, giving definitions are specifying the problem. But already at this point, important aspects are undefined or vague:

* What is the space of computational pipelines that BugDoc captures? (E.g., is concurrency allowed? i assume not, but there is no concrete discussion of the problem space.)

* What does a pipeline instance look like? Does CPi contain all the steps in CP, or could some be missing?

* How is a pipeline CP modeled? What if there are loops and branches in the code? Understanding what CP looks like is fundamental to all the definitions, but the paper assumes that the readers know exactly what this is and build from there. But we really don't know yet what this encapsulates.

* What can parameters be? Are the simply numerical? Can they be categorical? Can they be ranges? This is important because the motivation talked about the errors being due to different types of reasons (e.g., parameters, data). But then, Sec 2 defines everything around parameters, and it is unclear what these parameters encode.

* Contrasting definitions 3 and 4, it appears that a definitive root cause is specified over all possible executions of the pipeline, whereas hypothetical root cause is based on the executions we've seen. I believe this understanding is correct, but I feel on shaky ground, because just before Def 4, the remark emphasizes that we seek to understand a root cause from the existing parameter values and not the universe of all parameter values. This makes me unsure about the definitions that follow. Also, step (3) in Sec 3.1 appears to imply that minimal root cause is determined based on the available executions (not the universe), which seems to contradict Def 4. Overall, I am left confused.

* Minimality says that a root cause is minimal is no proper subset of it is a root cause. This seems clear on its face, but given the form of Cf, I am unsure. For example, is A>10 a subset of A>5?

* Can failures be due to multiple errors? Meaning, some failures are due to A>5 and some others are due to B=3. Or is it assumed that all failed runs have the same root cause?

Ultimately, the technical contributions, which comprise of the debugging strategies and explanation simplification in Sec 3, seem a little thin. They are intuitive enough, but the methods themselves are quite simple. Is there some theoretical grounding to them?

Another important issue is that the paper overlooks the vast software engineering literature on fault localization. I believe that the problem setting of this paper applies directly to FL techniques. These typically focus on identifying lines of code as causes of bugs based on faulty execution, but techniques focusing on inputs exist, and the distinction is not that crucial, as inputs can be provided as lines of code setting the particular parameters.

Excluding the omission of the connections to fault localization, the experimental evaluation is careful and thorough. The authors compare with two other explanation techniques, and BugDoc performs very well in a variety of experiments. Some of this behavior is not entirely surprising. For example, Data X-Ray would allow configurations that lead to bad instances by design, as its objective function permits that based on the problem settings it was designed for. Similarly, Explanation Tables use sampling that may miss some problem instances if they are rare, and their objective of information gain maximization may also not be best-suited the particular problem setting. Conceivably, both of these techniques may be adapted to the problem better with appropriate changes to their objective functions.

This is not to say that the evaluation treats these unfairly, but it would be worthwhile to clarify that their objective may be by design not well-suited for the problem setting of BugDoc, which does not permit a bug to ever lead to successful executions.

Coming back to fault localization, I believe that BugDoc is essentially a bug localization technique, so the authors need to look into the related literature and augment their evaluation appropriately. If the authors posit that fault localization is not applicable (but I doubt it), this needs to be discussed carefully in the related work.

Overall, this is good, principled work, but unfortunately suffers from a few issues that are quite critical.

9. Candidate for a Revision? (Answer yes only if an acceptable revision is possible within one month.)
Yes

10. Required changes for a revision, if applicable. Labelled R1, R2, R3, etc.
Not sure if a revision is reasonable. The main concerns are the technical depth and connections to fault localization; I fear that these may be beyond the scope of a revision, but I could be swayed.

Reviewer #2

Questions

1. Overall Evaluation
Weak Reject

2. Reviewer's confidence
Expert

3. Originality
Medium

4. Importance
Medium

5. Summary of the contribution (in a few sentences)
The authors introduce BugDoc for systematically debugging computational pipelines. BugDoc analyzes the executions of computational pipelines, based on the outcome of these pipelines and makes suggestions for executions with different parameter sets, in order to identify the root cause(s) of failing executions.
BugDoc implements two algorithms to identify the root cause(s). The “Minimal Pairs” algorithm identifies a single parameter-value combination as root cause. The “Debugging Decision Trees” algorithm identifies multiple parameters and parameter-value inequalities as root causes.
The authors compare BugDoc to combinations of the existing solutions Data X-Ray, Explanation Tables, and SMAC. They show BugDoc’s superior precision and recall in root cause identification even when computational resources are restricted. For that purpose, they implement for synthetic computing pipelines and two real world computing pipelines (Linguistic Dataset, Classification Pipeline).

6. List 3 or more strong points, labelled S1, S2, S3, etc.
S1) The paper presents a novel approach for finding root causes in parameterized computational pipelines.

S2) The authors compare BugDoc to multiple combinations of existing state-of-the art solutions for evaluation purposes.

S3) The paper is well written and presents a clear line of thought. It is easy to follow and thus to understand.

7. List 3 or more weak points, labelled W1, W2, W3, etc.

W1) The paper lacks technical depth.

W2) The paper does not provide a clear formalism on the computational pipeline and the parameter space of the same.

W3) My overall impression of the paper is that it is only applicable to certain computational pipelines (i.e. those which clearly fail or succeed).

W4) The paper does not define a clear failure model.

W5) The paper does not define a provenance model.

W6) Experiments could be improved and do not validate all claims.

8. Detailed evaluation. Number the paragraphs (D1, D2, D3, ...)

D1/S1) The authors present a systematic approach for root cause analysis in computational pipelines using BugDoc. BugDoc's systematic approach is especially helpful when debugging parameterized computational pipelines. Debugging these pipelines is typically quite tedious. Thus, the authors present indeed a novel and relevant idea.

D2/S2) Another strong point of this work is that the authors compare their solution to other state-of-the art solutions, such as Data X-Ray, Explanation Tables, and SMAC. The comparison shows BugDoc's superior "performance".

D3/S3) Further, the paper has a clear structure and is well written. It is easy to understand and follow the ideas presented in the paper, since the paper practically contains no typos or overly complicated sentences.

D4/W1) However, the paper lacks technical depth. The authors describe the presented algorithms "Minimal Pairs" and "Debugging Decision Trees" in an informally written text. Presenting these algorithms in pseudo-code with clear definitions of the input, output, and parameters would significantly improve the quality of the paper. Further, the authors do not sufficiently describe BugDoc's decision process for choosing either of the two algorithms. Or does a user actually has to decide upfront? An algorithmic description about the decision process would allow the reader to understand the concepts of BugDoc at large.

D5/W2) In order to present the algorithms in their full technical depth, the paper would clearly benefit from a more concise formalism. For instance, the definition of a computational pipeline does not restrict any properties of the pipeline, i.e., whether the pipeline can contain cycles, branches, or modules with multiple inputs. The definition of the parameter-value pairs further indicates that parameters may not only be those of modules in the pipeline but the choice of module itself. These imprecise formal definitions lead to confusion about the implemtation, applicability, and novelty of BugDoc.

D6/W3) The running example and the evaluation section evoke the impression that BugDoc is particularly designed for machine learning pipelines (which is fine if clearly defined), since it is simple to distinguish the failure or success of such a pipeline. BugDoc does not seem to be applicable to any arbitrary computational pipelines transforming data as stated by the authors. It may not be possible to reduce the result of an arbitrary computational pipeline to success or failure. However, this is often the case for pipelines generating output data from input data. Clarification of the applicability and use cases of BugDoc is needed.

D7/W4) The authors clearly state that BugDoc addresses root cause analysis from among existing parameter-value combinations. It is not designed for software testing. However, it remains unclear to the reader, which failure model BugDoc addresses precisely. For instance, does BugDoc require the computational pipeline to produce the same result, given the same set of parameters?

D8/W5) The authors claim to make use of provenance in BugDoc. If I get it right, they consider the choice of parameters together with the evaluation status (success/failure) provenance. However, this is just a guess. They never explicitly explain a) what provenance they collect (which artifacts, what type of provenance, its granularity) and b) how they query the provenance. That is why the paper would benefit from a concise definition of their provenance model. Currently, the paper evokes the impression that provenance does not play a noticeable role in BugDoc (which is fine, but then there is no point mentioning it repeatedly).

D9/W6) Even though the evaluation consumes the most space in the paper, it still leaves open questions about the scalability of BugDoc. The experiments run on pipelines with 15 parameters at most. Complex computational pipelines can easily exceed one hundred parameters and allow for more than 40 parameter values. Further, the authors claim that the “Minimal Pairs” algorithm is faster than the “Debugging Decision Trees” algorithm, when there is just one minimal root cause. The paper would improve if a runtime comparison between the two algorithms were presented in order to confirm the claim.

D10 In the following, I list some general remarks about the paper. Some elements in the paper consume significant space, but yield hardly any contribution in the way they are currently presented.

D11 The pipeline in Figure 1 and the pipeline in Figure 10 seem to be describing the same process. Probably, it is possible to save space when Figure 10 replaces the pipeline in Figure 1.

D12 Figure 2 significantly benefits from additional context. Instead of using p and v to describe parameters and values, a tree generated from the running example definitely improves the quality of the tree.

D13 The plain, unmodified definition of the f-measure may not necessarily be displayed. It may be sufficient to mention that you make use of the f-measure.

D14 Further, Figure 9 shows nine bars of the same height. Probably, the sentence describing this figure suffices to tell that all the solutions are equally good.

D15 Even though you intend to publish computational pipelines in the evaluation section together with the code of BugDoc, a detailed description or definition of them in the appendix of the paper would have been beneficial. At the moment, I can only assume that the pipelines look similar to the one described in the running example.

D16 To conclude, the paper proposes a novel approach to root cause analysis in computational pipelines, especially machine learning pipelines. However, it lacks technical depths and formal definitions. In most parts the paper is so imprecise or general that it is barely possible to appreciate the technical contributions of this paper.

9. Candidate for a Revision? (Answer yes only if an acceptable revision is possible within one month.)
No

10. Required changes for a revision, if applicable. Labelled R1, R2, R3, etc.
A revision would have to address all comments listed above.

Reviewer #3

Questions

1. Overall Evaluation
Reject

2. Reviewer's confidence
Expert

3. Originality
Low

4. Importance
Medium

5. Summary of the contribution (in a few sentences)
The paper proposes techniques for debugging workflows in a blackbox manner by analyzing past workflows, and surgically synthesizing new parameter configuration for experimental workflows to narrow down the root cause of workflow failure. The core idea is to used past instances of successful and failed workflows to derive a concise explanation of failure for the failed workflows. In order to further improve the precision of the diagnosis, the technique also tries to synthesize erroneous configurations consisting of candidate bad configurations, actually deploying these workflows, and confirming the diagnosis.

6. List 3 or more strong points, labelled S1, S2, S3, etc.
S1: Root cause diagnosis of workflow failures is hard especially when the failure spans multiple systems and dependencies, and is hard to localize to a particular step in the flow. Automated tools for diagnosis based on historical runs of the workflow seems like a good approach.

S2: The authors propose a couple of different algorithms for efficiently pruning the configuration space, and identifying root causes of workflow failures. The second algorithm also launches experiments with configurations that include the candidate failure condition to further confirm the failure diagnosis using more runs.

S3: Some experiments that compare parts of the technique against existing approaches like Data XRay and Explanation Table.

7. List 3 or more weak points, labelled W1, W2, W3, etc.
W1: The paper neither states the class of workflows that it targets nor does it do a good job at formally defining the problem. This makes it hard to estimate the impact of the work since it is not clear under what circumstances such techniques would be applicable. See detailed evaluation for more.

W2: The algorithms proposed by the paper seem fairly straightforward. The only new idea seems to be actually synthesizing new parameter configuration for the workflow based on candidate causes and running further experiments. It is not exactly clear under what circumstances this is feasible. See detailed evaluation for more.

W3: The experimental results and case studies do not provide any concrete qualitative evaluation of the root causes derived by these techniques. There are a bunch of visualization techniques that can also be used to easily navigate through a history of parameter configurations and arrive at similar results. What I would have liked to see is a complex predicate that involves multiple components that was too hard to debug manually but was very easily detected using this technique.

8. Detailed evaluation. Number the paragraphs (D1, D2, D3, ...)
D1: Real workflows outside of academic context can become complex with conditionals, loops, etc. Furthermore, the configuration for each component in the workflow may have a certain structure to it (as a simple example, the number of hidden layers used is a parameter that only makes sense if the you use a deep network for training). It is not exactly clear if the proposed techniques of the paper work for such non-trivial workflows and complex and

structured configuration spaces.

D2: The parameter space for configurations of workflows can become large quite easily. For instance, the high-level behavioral configuration for the steps of the workflow are only one of the many parameters to the workflow. There are other aspects like the environment in which the workflow is deployed, the compiler versions used to create binaries for the jobs, the version of software dependencies available in the environment, etc. that are also important for detecting root causes. In fact, most of the trickier to diagnose issue arise out of such incompatibilities that are hard to detect. Considering all such values can easily lead to an exponentially larger search space making some of the techniques like launching experimental workflows with erroneous configurations cost-prohibitive.

D3: It is not exactly clear what the experimental results actually illustrate. Ideally, I would have liked to seen at least one instance of a known workflow failure that involved a hard to diagnose issue, but was easily detected by BugDoc.

D4: The presentation of the paper can be improved. The problem definitions should be more formal and so should be the algorithms.

9. Candidate for a Revision? (Answer yes only if an acceptable revision is possible within one month.)
No