14-Dec-2015

Dear Dr Podgurski,

Manuscript ID STVR-15-0066 entitled "Causal Inference Based Fault Localization for Numerical Software with NUMFL" which you submitted to Software Testing, Verification and Reliability  has been reviewed.  The comments of the referee(s) are included at the bottom of this letter.

We have three reviews from experts in the field. One recommends a minor revision, the other two a major revision. The editors of the special issue have recommended a major revision and I concur. Thus we would like to ask that the paper undergo a major revision for resubmission.

All three reviewers agree that this submission contains enough new material over the ICST 2015 version to justify publication in STVR. There is some disagreement regarding the readability of the paper, most likely because the paper builds on material that is outside the comfort zone of the average software engineering researcher. The reviews provide many detailed comments on which parts of the text cause confusion, and should help guiding the revision towards making the paper more accessible (for instance, by providing more examples and making better use of them).

Besides these presentation aspects, Reviewer 3 also points out a lack of discussion of 1) threats to validity, 2) relevant related work, and 3) the experimental results, and these aspects should be revised. For the latter, Reviewer 2 also has relevant comments, and the omission of Barinel and the use of Sober instead of CBI needs to be either justified or rectified.

Finally, all reviewers raise the lack of data/discussion of the time and complexity of the approach, in particular compared to the much higher costs of developer time.

Thanks once again for your submission. I hope you find the reviewers' comments helpful and that you will submit a revised version within three months. If you need more time, just let me know.

When you submit a revision, it is required that you include a response to reviewers, detailing how you addressed each specific comment. It is best to upload that as a "supplemental file for review," which manuscriptcentral will append to the submission for reviewing.

Please note that submitting a revision of your manuscript does not guarantee eventual acceptance and the same reviewers will be invited to check the revision. Therefore, you should work just as hard on your response to the reviewers as you do the paper. The journal policy is that only one major revision is allowed, so if the revision is not of sufficient quality to be called at least a minor revision, we could not accept it.

Thank you for your submission and I look forward to hearing from you.

Sincerely,

Referee(s)' Comments to Author:

Reviewing: 1

I have several high-level questions and suggestions which the authors perhaps

may want to address:

- It is not clear to me how and why fault localization is different and/or

harder for numerical programs. The authors present this is a fact, but more

explanation would definitely benefit the motivation of the paper.

The fault localization for numerical programs has three challeges:

1 The numerical bugs in the software are often hard to detect because they may not necessarily result in software crashes.

2 It is very difficult to localize the fault by tracing back the execution, because we lack the oracle of intermediate variable values.

3 Testers usually detect failures in numerical program by checking whether the difference between the program output and expected output exceeds a pre- defined tolerance.

- On a similar note, the authors compare to techniques which are not geared

towards numerical programs, yet, they can also find faults reasonably well.

Can your techniques be applied to non-numerical programs? What performance is

expected? I guess my big question here is, what is so different about numerical

programs and why is your technique specialized for them (and not for others?)?

NUMFL can only t be applied to statements whose variables are numerical (int, double , etc). NUMFL is based on regression technique to estimate causal effect estimation. To fit a regression model, numerical variable value is required. In non-numerical program, for example, NanoXML, a java based XML parser, most variables are string type. In this case, NUMFL cannot estimate the causal effect estimation with regression model.

- In the beginning, the authors talk about numerical programs, later it is

about floating-point programs. For which programs exactly is the technique

applicable, and would it be applicable to others (say integer programs) or

if not, why?

The technique is applicable to both integer programs and floating programs.

- A few places in the paper would benefit from more intuitive explanations.

\* In particular, on page 4 there is no intuitive explanation why common

measures are often inadequate to measure the causal effect of a numerical expression.

The common measures like correlation or covariance are inadequate to measure the causal effect of a numerical expression, because they did not control the confounding bias between treatment and outcome.

\* Apparently the causal relationship between X and T illustrated in figure 2

is an issue. Why?

The causal relationship between X and T cause the confounding bias.

\* On pg. 6 you say "Ordinary propensity scores are not applicable to continuous

treatment variables." This is presented as a fact, but it is not clear to me why.

The ordinary propensity score Pr(T=1|X=x) is defined as the conditional probability an individual receive treatment given the confounding variables.  It requires the treatment variable is binary. T=1 means the individual receive the treatment. T=0 means the individual does not receive the treatment.

\* Can you provide some more intuitive explanation of the what GPS and CBPS

do and how they differ, in addition to the formal presentation with equations?

//pending

- Why does the technique need both passing and failing runs? I understand why

the true/correct result of a failing run is needed, but it does not seem to

be explained why passing runs are useful.

In the evaluation section, the NUMFL can be applied to failing runs only(without passing  runs). The passing runs is benefit to the average causal effect estimation, but not required.

- How realistic is the assumption that a programmer can provide correct results?

In the evaluation the authors start with the correct programs, so the data is

available, but in reality I fear that this will not be case in many applications.

Previously, most numerical libraries are developed by Fortan language. In the development of numerical libraries of other language, like apache common math in java, the developers usually test their program with the help of corresponding libraries developed by Fortran. They run the java program and Fortran program with same inputs. The output of Fortran program is considered as correct result.

- On page 2, the authors cite recent work on SFL using causal inference methodolody,

yet the evaluation at the end cites other work. Is it related, or if not, why

is the evaluation not done with respect to the initially cited similar work?

There are two reasons we did not evaluate Baah’s method in this paper.

The first reason is Baah’s method is a coverage based statistical fault localization. We have already evaluate three coverage based SFL (Ochiai, Dstar, SOBER) in evaluation section.

The second reason is Baah’s method

- The authors only consider the last iteration of a loop in the profiling phase.

Some discussion of the consequences of this decision would be nice (even if they

are not big.)

We have two reasons to use only last iteration of loops:

1 Base on our previous observation, the errors generated by the faulty statement tends to accumulate to the last iteration of the loop.

2 The faulty statement within the loop will influence the program outcome after last iteration of the loop. In other words, the faulty statement pass its value of last iteration to the program outcome.

- Why was the evaluation of GPS vs CBPS done separately from the comparison agains

the five related techniques? I think it would be more useful to have one comparison,

since the evaluation that is performed seems otherwise the same.

This is mainly because we originally name Figure 6 as NUMFL QRMvs Base line techniques. CBPS is not considered as one of base line technique, but another version of NUMFL. So we we compare these two versions of NUMFL separately.

- The authors compare the different techniques in terms of subexpressions examined

until the fault is found. While this is certainly an insightful metric,

I would also appreciate knowing how much time the analysis takes, even if this

is just for information purposes. After all, the technique should be practical

for actual users.

We have added computation cost analysis in section 5.8 of the modified paper.

- On page 17 you say: "... the output errors from passing runs are clustered

around zero." Why is the error not equal to zero? (Especially if the assumption

is that the user provides correct results.)

This is because if an execution’s output error is less than a predefined tolerance, the execution is considered as pass. The predefined tolerance is small, but it is still larger than 0.

A few low-level remarks:

- In Fig. 2 perhaps say/repeat which variables are the treatment, outcome and

confounder.

\\pending

- Fig. 3 Please explain what the consequence of the breaking of the link is,

or otherwise remove the picture because it does not add any new information

in addition to the text.

After breaking the link, confounding variables have no effect on the treatment variable, so we can estimate the causal effect of treatment on outcome without confounding bias.

- Fig. 4: I am not sure what the authors are trying to illustrate here.

The figure shows a piece-wise linear and a quadratic functions, but this is

already clear from the equations in the text.

\\pending

- Pg. 15. There is something missing in this sentence: "There were just 5

versions for which the latter performed better than NUMFL-GPS-QRM."

\\pending

We will fix this.

- The numbers of versions in section 5.7 does not match with the numbers of versions

from the previous comparisons, while the text says that the comparison was performed

on the same set of benchmarks.

In the first paragraph of section 5.7, the word “versions” in the sentence“

There were 13 versions for which NUMFL- GPS-QRM performed better than NUMFL-CBPS-QRM. There were only 3 versions for which NUMFL-CBPS-QRM performed better than NUMFL-GPS-QRM.

”

means the subject programs. We have 16 different subject programs.

At the end of first paragraph of section 5.7, the word “ versions” in the sentence”

NUMFL-GPS-QRM performed better on 61 versions, while NUMFL-CBPS-QRM performed better on 31 versions.

”

means the faulty version of subject programs. We create 92 faulty version of subject programs with fault injection

Reviewing: 2

Comments to the Author

This paper extends the ICST conference paper of the same authors that offers a novel idea in an important area of research. Unfortunately, the authors must improve the presentation of the material before the paper is accepted. As it currently stands, the paper contains many ambiguities and vague descriptions that can be resolved only with significant effort of reading papers from the area of statistical analysis and probabilistic graphical models. The paper should be self-contained. The motivating example introduced in Figure 1 is good, however, the authors make very little use of it! Its real value will come from using it in Section 3, where it can be used to demonstrate how the algorithm works.

We have used the motivating example to demonstrate the algorithm in section 3.3 in the modified paper

The presence of Figure 3 is puzzling, since it differs from Figure 2 only by the presence of X along the X->T edge. In fact, Section 3 is the biggest problem, it uses a lot of mathematical notation that is completely divorced from the example in Figure 1 and from the context of SE research. Since the authors do not propose a new statistical methodology but use the existing one to show how it works in the context of SE fault localization, I expect that the authors make it problem specific.

\\pending

When discussing GPS and CBPS the authors should give a table with concrete values for program variables and show how to apply the proposed formulae to these values.

We have discussed GPS and CBPS with a table of concrete values in section 3.3 and section 4.2.2

GPS and CBPS

The choice of the logistic model must be justified and its use can be shown using the motivating example. Since it is a journal paper, the space limitations do not apply as much as they do for a conference paper, and the authors should work out details of their approach at much deeper depth.

Using motivating example to justified logistic model is difficult. To compute CBPS, we use the R library “CBPS”. It does not provide interfaces to access the fitted logistic model or the values of any intermediate step of CBPS computation. But in section 4.2.2 of modified paper, we do use motivating example to justify the effectiveness of CBPS.

In Section 4, the authors must expand on the Pearl's backdoor criterion which is tightly linked to beliefs about some events and show how this reasoning can be applied to the motivating example. The values of AFCE should be discussed also in the context of the motivating example.

In section 3.3 of the modified paper, we use the motivating example to demenstrate Pearl’s backdoor criterion by showing the effect of confounding variables on treatment variable is significantly reduced given GPS .

In section 4.2.2 of the modified paper, we use the motivating example to explain the estimation of AFCE.

Why don't the authors try to obtain all values from loops and see how the precision of the scores will change? Yes, doing so incurs significant overhead, but as the authors mentioned, the debugging time of developers is much more important than the computation time of a fault localization approach.

We have two reasons to use only last iteration of loops:

Base on our previous observation, the errors generated by the faulty statement tends to accumulate to the last iteration of the loop.

The faulty statement within the loop will influence the program outcome after last iteration of the loop. In other words, the faulty statement pass its value of last iteration to the program outcome.

What is the effect of grouping observations in subclasses whose sizes are not roughly equal? The authors should address this question in their experiments.

\\pending

The discussion on symmetry of numerical errors is confusing, since it is not clear how often it is a big problem. Is it a problem that the values of T are distributed symmetrically around any value or should they be distributed only around zero? Why?

In each subclasses, the values of Te can be either larger than the correct value Tc or smaller than Tc, so we proposed DLRM and QRM which can handle both cases (Te is larger than Tc and Te issmaller than Tc). Both DLRM and QRM are parametric model which requires training data includes observations of both cases.  if in a subclass, all the values of Te are larger than Tc or all the values of Te are smaller than Tc,  DLRM and QRM cannot be well fitted and will give bad esimation of AFCE for expressions. Thus, we assume that the value of Te within each subclasses are roughly symmetrically distributed around the value of Tc.

The algorithm in Figure 5 should be explained at much finer level of details. Please instantiate it for the motivating example to show how it works step by step. Does the algorithm have any invariants or properties that should hold?

We have explained algorithm with motivating example in section 4.2.2 of the modified paper

"likely to characterize the values of numeric variables inadequately" => please explain

Some predicates level SFL like CBI, inserted predicates in a program to enhance SFL. For example, given a numeric variable x in the program, CBI insert three predicates x>0, x=0 and x<0 to decompose the value of x. In real numerical software,  predicates like x>0, x=0 and x<0 are likely to characterize the values of numeric variables inadequately. For example, if x is always larger than 0 in the executions, then  x>0, x=0 and x<0  can not decompose the value space of x.

"which predicates should be inserted" => inserted where? please exaplain

Predicates level SFL like CBI inserts predicates to the statements which defines a numeric variable.

For example, in the motivating example of this paper, the statement at line 3 is x=a\*b. Variable x is not a predicate, so there is no predicate at line 3.  To characterize the values of x, CBI use three predicates x>0, x=0 and x<0 to represent x at line 3, so these 3 predicates are inserted to line3 by CBI.

Reviewing: 3

Comments to the Author

I do think that the paper is generally well written, well structured, and technically sound. However, I do have concerns that need to be addressed by the authors:

- Introduction: I suggest the authors to add the following citation when "spectra" is introduced:

Harrold, Mary Jean. "Testing: a roadmap." Proceedings of the Conference on the Future of Software Engineering. ACM, 2000.

We have cited this paper in the modified version.

- Introduction/Related Work: It seems to be that the authors do not discuss related work that is relevant to the proposed method. The problems NUMFL addresses have also been addressed by other authors in different ways. Examples of works that seem relevant to me are:

Mayer, Wolfgang, and Markus Stumptner. "Evaluating models for model-based debugging." Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2008.

Mayer, Wolfgang, et al. "Prioritising model-based debugging diagnostic reports." Proceedings of the International Workshop on Principles of Diagnosis (DX). 2009.

Abreu, Rui, et al. "Refining spectrum-based fault localization rankings." Proceedings of the 2009 ACM symposium on Applied Computing. ACM, 2009.

Wotawa, Franz, Mihai Nica, and Iulia Moraru. "Automated debugging based on a constraint model of the program and a test case." The journal of logic and algebraic programming 81.4 (2012): 390-407.

We have discussed above papers in the section of related work of the modified paper.

Wotawa and colleagues have also done extensive constraint-based debugging of spreadsheets (e.g., CiBSE, ISSRE). That body of work also needs to be discussed in this paper because spreadsheets are in fact numerical programs. Actually, most of the work also have the (strong?) assumption of requiring the exact expected output to be able to reason about observed failures.

We will discuss the above related work

Most of the works mentioned do also reason in terms of multiple faults. There is also Barinel, a spectrum-based reasoning approach that reasons in terms of multiple faults that might be more interesting to compare with in Section 5.6.

\\pending

- Section 2: The authors discuss two potential problems of a naive approach. There is another one that the authors might be worthwhile to discuss: ambiguity groups. The notion of ambiguity groups has been discussed in several related work, including

Gonzalez-Sanchez, Alberto, et al. "Prioritizing tests for fault localization through ambiguity group reduction." Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on. IEEE, 2011.

\\pending

- Section 3.2: Perhaps I am being nitpicking, but I suggest the authors to add the following as I prefer this notation for the equation on page 5

\frac{1}{1+exp(-X\_i'\beta)}

(Ignore this comment if you don't consider this addition to be useful)

\\pending ignore

- Section 3.2, page 6: I don't follow the example explaining propensity scores break. Why is this the case? Can the authors provide a more realistic example?

In section 3.3 of the modified paper, we use the motivating example to demonstrate Pearl’s backdoor criterion by showing the effect of confounding variables on treatment variable is significantly reduced given GPS .

- Section 5.1: Why was m set to 10? How does one decide which is the best value for a software under analysis? Including a study showing the sensitivity of the method to m would highly improve this paper. Perhaps another item for the threat to the validity discussion.

\\ Pending

- Figure 5: There are multiple problems with the algorithm:

   \* T\_e and X\_e are both inputs and variables defined in lines 2 and 3. Why?

   \* What is the "ith subexpression" in lines 2 and 3? Don't you need to make i part of the for-loop condition

   \* m is used, but never defined. Include it as part of the input?

- Section 5.2: The study is done using injected faults. How realistic are these faults? The study has threats to the validity that need to be discussed. The following paper may help the authors identify all the threats:

Steimann, Friedrich, et al. "Threats to the validity and value of empirical assessments of the accuracy of coverage-based fault locators." Proceedings of the 2013 International Symposium on Software Testing and Analysis. ACM, 2013.

\\ Pending

- Section 5.2: Why only 12 faulty versions for >=1500 LOC and 2 for <=1500?

When we create a faulty version, we do not inject faults to the statements, which defines a variable with the type of Strings, Char etc. This is because our technique is based on the values of numerical expressions.

For the large subject programs (programs >=1500 LOC), there are about 200~650 numerical sub-expressions.For those small subject programs in our experiment(programs <=1500 LOC), the number of sub-expressions is less than 100.

We randomly inject the faults into numerical expressions, so most of time, the injected faults make the programs crush. In this study, we expect an incorrect output instead of program crush, because we need the values of outcome variable to fit the regression model. Usually, we need to randomly generate 20 faulty versions of subject program to get one faulty versions that do not make the program crush. Thus, creating 12 faulty versions for programs >=1500 LOC and 2 for programs <1500 are actually the best we can do with the subject programs on hand.

- Section 5.3: Why using Sober instead of CBI, if CBI is known to outperform Sober?

This is because R Gore mentioned that ESP outperforms CBI in statistical debugging in the paper "Statistical debugging with elastic predicates."  We have already choose ESP-SIV and ESP-SCP as baselines, so we do not choose CBI.

But we do borrow the idea of CBI when we applied SOBER to value based fault localization. We have also cited the paper of CBI in references 31.

- Section 5.4: I would rather see the results of DLRM in the tables instead of just mentioning that DLRM outperformed the 5 baseline metrics.

//Pending

- Section 5.4: Figure 6 is unreadable.

We have updated Figure 6 in the modification of the paper.

- Section 5: A major issue I have with this section is that it just outlines results, without comprehensive discussion about the reasons for the results. As an example, what are the characteristics of the programs/tests for those programs in which Ochiai/DStar outperforms NUMFL (and vice-versa). Just summarizing the results gives no insight at all regarding why one method outperforms the other.

//Pending

- Section 5: I would have prefered to read section 5.7 first.

//Pending

- The Introduction mentions an assumption viz. developer-time being more critical resource in debugging than is computation-time. Although I do agree with the authors, I was left wondering about the computation-time complexity of the approach

We have added computation cost analysis in section 5.8 of the modified paper.