Thoughts on Improving Review Quality

Paul Francis
Cornell

Abstract

In my mind, one of the biggest problems with PCs today is the quality of the review. The main reason that review quality is often bad is because PCs are overworked. This paper suggests a few ideas for how to reduce the workload on PC members, with a goal of improving the quality of reviews.

1. Introduction

Conferences in computer systems serve four main purposes:

1. They provide a forum where researches can meet.
2. They disseminate research results.
3. They gives papers a stamp of approval.
4. They give researchers a stamp of approval.

While there is in my mind much to dislike about how well the conference system accomplishes all but item 1, this paper makes the assumption that these things won't change easily and so doesn't try. Rather, the focus of this paper is to suggest several ideas on improving the quality of reviews while staying within the current system.

Note that, while I think the quality of reviews (including my own) are often questionable, I don't have a feeling that conferences are bad as a result. I think the reason for this is that there are usually a lot of papers, all of which could reasonably be in a conference, but many of which have to be rejected. Some will be rejected for the wrong reason, and others might even be accepted for the wrong reason, but with occasional exception the set of accepted papers is reasonable. Still, it is frustrating as an author (especially for students) to be given an incorrect review, and it is frustrating as a reviewer to not be able to spend more time on a given paper.

In what follows, I propose three ideas, ranging from easier to harder to implement, and from less to more impact on the system.

2. Idea One: Constrained Author Feedback

This is a very simple but I believe helpful idea. Basically, I would like there to be a system whereby the reviewer can anonymously ask the author a question of the form:

> "Where in the paper is such-and-such a question answered"

And the answer from the author is limited to page/column/line numbers. For example:

> Q: "It appears that, if node R3 in figure 3 were to crash after message 1 is sent but before it is received, you will have a loop. Where in the paper do you describe how this is avoided?"
>
> A: "page 3, column 2, lines 14-24"

The rationale here is that I often find myself rejecting a paper because of a perceived flaw, where it would take considerable effort to convince myself that I understand the paper exactly right and that there really is a flaw. I usually console myself with the thought that, if the paper were better written, I'd more easily understand the algorithm, and therefore I'm on solid ground to reject it. But it'd be good if I could just ask the author. Having said that, I don't want the author to explain things beyond what is in the paper, because the paper should stand on its own. Allowing authors to explain things beyond what is written in the paper effectively means that something more than what has been submitted is being reviewed, and this just confuses the process. Therefore, limiting the authors' answer to material in the paper seems a reasonable compromise between no feedback and too much feedback.

This feature could be pretty easily implemented on any of the online conference management tools, or even through email.

3. Idea 2: Pass Reviews Between PCs

First some background. Today the process of publishing a paper employs the following algorithm:

1. Select a conference.
2. Submit paper.
3. Get back reviews.
4. If accepted, quit.
5. Otherwise, select another conference (probably lower tier).
6. Modify paper appropriately.
7. Goto Step 3.

One problem with this algorithm is that the same paper is reviewed multiple times by (mostly) different PCs. In the lifetime of a paper, it may have a dozen reviews. Worse, sometimes the early submits are hail-mary's[1] designed simply to take the temperature of the paper.

What I suggest is a practice whereby an earlier PC can pass its reviews to a later PC. This is not to be in lieu of subsequent reviews, but rather to make the subsequent reviews easier to write and of higher quality. In the following description, *PC(N)* is the Nth PC to which a paper is submitted, *paper(N)* is the paper submitted to the Nth PC, *reviews(N)* are the reviews written by the Nth PC, *rebuttal(N-1)* is a rebuttal written by the authors about *reviews(N-1)*, and *changes(N)* is a description of how the paper has been modified in response to *reviews(N-1)*.

What I suggest is that the input to *PC(N)* is as follows:

1. From Author: *paper(N), changes(N)*, and *rebuttal(N-1)*
2. From *PC(N-1): reviews(N-1),* blind, as well as all of the author input to *PC(N-1)*

In short, the PC gets the benefit of previous PCs, as well as the authors' comments on what previous PCs did. Given this input, the reviewer in *PC(N)* can give the paper a quick overview, and then look at previous reviews and well as previous rebuttals to get a good measure of the value of the paper. Presumably the authors will have either disagreed with previous reviews, or agreed with previous reviews and modified the paper accordingly. Either way, the reviewer can quickly focus in on what previous issues were and how they may or may not have been fixed. Of course this doesn't absolve the reviewer from forming his or her own opinion (and there is some danger that it would do just that), but I suspect that this information would allow the conscientious reviewer to do a much better job.

A few general comments. This approach has the benefit that it gives the author a chance to offer a rebuttal, albeit too late to overturn the decision of earlier PCs. The *reviews(N-1)* are kept blind because, if they are not blind, then the PC chairs have a more difficult job; they have to consider conflicts of interest in handing out the previous reviews.

There are a few potential negatives to this system. It substantially increases the complexity of PC mechanics, both because additional materials must be managed, and because materials must be retrieved from the previous PC. It may box in the thinking of the reviewers (they only consider issues introduced by previous reviewers rather than create their own issues). It exposes information that the author might prefer remain private. Finally, it doesn't reduce the workload of the first PC, *PC(1)*, which creates the initial set of reviews. In the third idea presented in this paper, we propose a way to address this last negative.

One general issue that has to be addressed is, how does *PC(N)* become aware of previous PCs? What if the author doesn't want it to be known that the paper was previously rejected? Do we leave it up to the author to reveal previous rejections, or do we create a system whereby the PC can learn about previous rejections regardless of the author's wishes? In the former case, it could easily turn out that the best strategy of authors is usually to not reveal previous rejections, especially when the reasons for the rejections cannot easily be fixed. This suggests that we may need the latter approach.

One way to do this is to develop a repository whereby materials from previous PCs can be loaded into the repository. For each "paper chain" {*paper(1), pa-*

[1] A "hail-mary" is a low-probability, low-risk attempt at something. For instance, in American football, a "hail-mary pass" is a long pass made on the final play of the game in an attempt to score and win. It is as likely to intercepted as not, but the risk is low because at that point the game is lost anyway.

per(2), . . ., paper(N)}, there would be a single record that stores all of the materials over the lifetime of that paper. The record may be indexed by a *record_ID*, title(s), and author names. When an author submits a paper, he or she is expected to also submit the *record_ID* of the paper. If the author claims that this is the first submission, the PC chairs are able to search the repository with the list of authors and/or the title, and try to discover previous submissions. There may be previous submissions that are significantly different but not completely different. In this case, the PC chairs can use their judgment as to whether or not to include previous materials in the review process. After the PC process, if the paper is accepted, then the record is completely removed from the repository. If on the other hand the paper is rejected, the PC chairs upload the materials from the PC into the record.

4. Idea 3: Pre-PC(1) Reviews

(I should note up front that pretty much none of the reviewers like this idea. See Reviewer Comments section below. Never-the-less, here it is.)

A weakness with the Idea 2 is that it doesn't reduce the work of *PC(1)*, which would normally be the Tier 1 conferences. (Actually, it might reduce the work some, because authors may be less reluctant to submit hail-mary papers to top conferences for fear of getting a bad review that is hard to recover from.) Another weakness is that, while Idea 2 increases the number of reviews per conference, it doesn't increase the pool of reviewers per se. Idea 3 is to create a system whereby papers may obtain reviews before being submitted to *PC(1)*.

Specifically, we allow papers to be publicly available before *PC(1)*, and solicit pre-*PC(1)* reviews which are then made available to *PC(1)*. Let's call this process *PC(0)*. The proposal is this. A website is created whereby authors have accounts and can upload unsubmitted papers. The titles, abstracts, keywords, and expected target conference are made public. For each paper uploaded, the authors of that paper must collectively produce a small number of reviews (3 or 4) of other papers. To review a paper, the reviewer searches or browses the titles, abstracts, keywords, and intended conference, commits to reviewing the paper, and only then is able to download the full paper. As with regular PCs, a list of conflicts is maintained, and papers with conflicts are hidden.

The review is double blind. However, the reviewer identity is made available to the PC chairs of *PC(1)*, and optionally to the reviewers of *PC(1)*, but are kept blind to all subsequent PCs. This is necessary in order for *PC(1)* to gauge the quality of *PC(0)* reviews as well as to determine outright conflicts, "friends and family" types of reviews, or intentional "torpedo" reviews.

A PC may choose to require that there be a certain number of pre-existing reviews, either from *PC(0)* or from a previous conference.

Some of the advantages of a *PC(0)* process are as follows. The main advantage is that it should reduce the work of *PC(1)* in two ways. First, it should hopefully reduce the number of hail-mary submissions. The submitters will have gotten the early feedback they need, and if the paper is weak, this may discourage them from attempting a top-tier conference in the first place. Secondly, it provides input to *PC(1)*, thus acting as a kind of "round 1" set of reviews for triage. Another advantage is that it allows reviewers to review papers they would want to read anyway, thus both wasting their time less, and improving the quality of the review.

One option of the *PC(0)* process is to go ahead and make the full papers public (or, to give the author the option of making the full paper public). The advantage of this is that it can serve to timestamp the contribution. This might also serve as a "hot topics" kind of distribution in both the sense of getting the idea out more quickly, and in the sense of giving the authors early feedback on an idea before they put too much effort into it. What's more, a conference might be less inclined to view such a paper as already published, since it won't have officially been accepted by a workshop. This may alleviate some of the awkwardness of, say, trying to determine if a Sigcomm submission is different enough from an earlier Hotnets submission.

A *PC(0)* process also has a number of negatives. The main one is that it is clearly more subject to abuse. It could be expected that authors would solicit sympathetic reviews, or that rivals would search for competing papers and try to kill them. It would be up to *PC(1)* reviewers to be on the lookout for overly sympathetic or overly critical reviews.

Another negative is that, if a conference requires pre-existing reviews, then this effectively sets the deadline for submission earlier. For instance, if I want to submit to Sigcomm, and I don't want to make my paper public, and Sigcomm requires say two pre-existing review, then I have to submit my paper some weeks before the Sigcomm deadline. (Of course, this would have the advantage of forcing my students not to wait until the last minute!)

Obviously the same repository that is used in support of Idea 2 could be expanded to support Idea 3.

4. Small Experiments

The fact that these ideas don't change the current overall structure of the conference/workshop/peer review system means that we could start experimenting with this system on a small scale without, for instance, requiring the online repository system. Two or a few conferences could agree to accept each other's reviews (i.e. Sigcomm could accept Hotnets reviews, Infocom and Sigcomm could accept each other's reviews, and so on). This would allow us to gauge the ideas value and tweak their operation before trying something more ambitious.

5. Reviewer Comments

This section briefly outlines the reviewer comments on this paper.

Regarding Idea 1 (constrained author feedback) most reviewers thought this was a good idea. One reviewer thought that this is in any event possible today (ask the PC chairs to forward emails back and forth). In my mind, not wanting to bother the PC chairs is a substantial disincentive to get author feedback, so I think it'd be nice to have an automated way to do this.

Regarding Idea 2 (passing reviews between PCs), reviewers generally liked the idea. The main concern was that this would allow reviewers to be lazy — they would simply look at previous reviews, and regurgitate the same information. My feeling is that if a reviewer is that lazy, then they aren't likely to write a good review in the first place. I'd prefer to do things to make a diligent reviewer's life easier even if it means making a lazy reviewer's life easier.

Regarding Idea 3, pretty much nobody liked the idea. The main concern is that it is too easy to game, and it wasn't perceived as really helping that much.

6. Conclusion

This paper discusses a few ideas designed to reduce the workload of PCs and improve the quality of reviews. I hope it generates an interesting discussion at the workshop.