Assigning Papers to Reviewers

Samir Khuller*
Dept. of Computer Science
University of Maryland, College Park MD 20742.

Richard Matthew McCutchen†

1 Introduction

Assignment problems arise in a variety of settings. For funding agencies such as NSF program directors that co-ordinate panels, assigning proposals to reviewers is a major challenge. It is important that each proposal receive sufficient review by qualified experts, and at the same time we would like to roughly balance the workload across reviewers and to honor the reviewers' preferences for which proposals they would like to read. The same issue arises for a program committee chair, who may have to assign literally hundreds of papers to a program committee consisting of thirty to forty program committee members.

From now on we will focus on the problem of assigning papers to reviewers. We assume that each reviewer is given access to the list of papers to be reviewed, and gives each paper both a "preference" score indicating his/her level of interest in reviewing the paper and an "expertise" score indicating how qualified he/she is to evaluate the paper. (Some organizations may use a single preference score and assume that it also indicates expertise. We believe that making the distinction may better model the real-world objective.) A reviewer may also declare a conflict of interest with a particular paper, meaning that he/she is forbidden to review the paper.

We do not consider stable marriage type preference lists, because a strict ranking of papers would be rather tedious to produce. In this scheme, the papers are essentially grouped into a few categories.

Let N be the number of papers and P be the number of reviewers. Suppose that each paper needs q reviews, so a total of qN reviews need to be generated. Ideally, from the perspective of the papers, we would like to assign each paper the q most qualified reviewers for the paper. Of course, this could lead to a load imbalanced solution where the load on some program committee members is very high, and the load on others is low. On the other hand, we could insist on a perfectly load balanced solution in which the number of papers assigned to each program committee member does not exceed $L = \lceil qN/P \rceil$. However, this may lead to a solution which is not optimal from the perspective of the papers.

One of our goals is to study precisely this tradeoff, and allow each reviewer to be assigned up to $L + C$ papers, where C is the *load tolerance*. We consider the question: is it possible to obtain a high quality assignment with a fairly low value of C? One can also ask whether, in such an assignment, the reviewers receive the papers that they would have most liked to review.

Stinkers are papers that pretty much no-one wanted to review. We would like to spread the load of the stinkers as evenly as possible.

*Research currently supported by CCF-0728839. Email:samir@cs.umd.edu.

†The bulk of this work was done while Matt was at the Dept. of Computer Science, University of Maryland, College Park, and supported by REU Supplement to CCF-0430650. Current email: matt@mattmccutchen.net.

2 Formulation as a Min-Cost Max-Flow Problem

We formulate the assignment problem as a min-cost max-flow problem, where each unit of flow represents one review. The construction is somewhat involved in order to incorporate all the desired incentives. In several places, it makes use of sets of "parallel edges" of different costs connecting a single pair of nodes (x, y) to allow flow to be sent from x to y at a cost that grows faster than linear in the amount of the flow. For example, if there are five unit-capacity edges from x to y of costs 1, 3, 5, 7, and 9, then any integer amount $f \leq 5$ of flow can be sent from x to y at a cost of f^2.

An example of the graph is shown in Figure 1. We have a source s and a sink t. For each paper j we create a set of nodes p_j^1, p_j^2, p_j^3, and for each reviewer i we create a set of nodes r_i^1, r_i^2, r_i^3. (The rationale for these sets is discussed below.) Flow can pass from s through one or more of the nodes r_i^t and one or more of the nodes p_j^t to the sink to represent a review by reviewer i of paper j.

Each paper has an edge of capacity q to the sink, indicating that it needs q reviews. In general, these edges will constitute the min cut, so any max flow will saturate them and thereby provide all the required reviews. We take the min-cost max flow in order to provide the reviews in the "best" possible way.

Each reviewer has a zero-cost edge of capacity L from the source so that he/she can be assigned L papers. If that were all, we would get a perfectly load-balanced solution, but we may be able to improve the quality of the assignments by allowing some imbalance. Therefore, we allow each reviewer to be overloaded by up to C papers (C is the load imbalance parameter) via a set of C additional unit-capacity edges from the source. We make the cost of the lth edge an increasing function $f(l)$ to provide an incentive to balance load across reviewers. Since $2f(1) < f(1) + f(2)$, a solution that loads two reviewers each by $L + 1$ will be preferred to a solution that loads one reviewer by L and the other by $L+2$ unless the load imbalance in the second solution is outweighed by other benefits.

For each reviewer i and paper j, there is a unit-capacity edge from i to j allowing that pair to be assigned, unless the reviewer declared a conflict of interest, in which case the edge is not present. The edge cost is based on the preference value a_{ij} stated by reviewer i for paper j. For values on the NSF scale of 1 (best) to 40 (worst), we chose the cost function $(10 + a_{ij})^2$, in an attempt to provide an incentive to avoid really bad matched pairs without completely masking the difference between a good matched pair and an excellent one. This choice seeks only to achieve a natural relationship between a linear preference scale as normally interpreted and the costs to be used in the optimization. We realize that strategic reviewers will take the cost function into account in choosing what preference values to submit, in which case its form matters little.

Alongside these purely additive per-review costs, we want to avoid an individual reviewer getting too many papers he/she does not like. With respect to a reviewer i, we classify papers as "interesting", "boring", or "very boring" based on their preference values; the thresholds for these classes are currently the same for all reviewers. The edge for reviewer i and paper j leaves from r_i^1 if j is interesting, r_i^2 if j is boring, or r_i^3 if j is very boring. Since all edges from the source enter r_i^1, flow for boring and very boring papers is forced to pass through a set of parallel edges from r_i^1 to r_i^2, and flow for very boring papers must pass through an additional set of parallel edges from r_i^2 to r_i^3. In each of these sets, we make the cost of the lth edge an increasing function of l to provide an incentive to balance the load of boring papers in the same way as for overload.

Finally, we would like at least one or two of each paper's reviews to be well qualified, if possible. The method is the same as that for reviewer fairness. With respect to a paper j, we classify reviewers as "expert", "knowledgeable", or "general" by comparing the expertise values

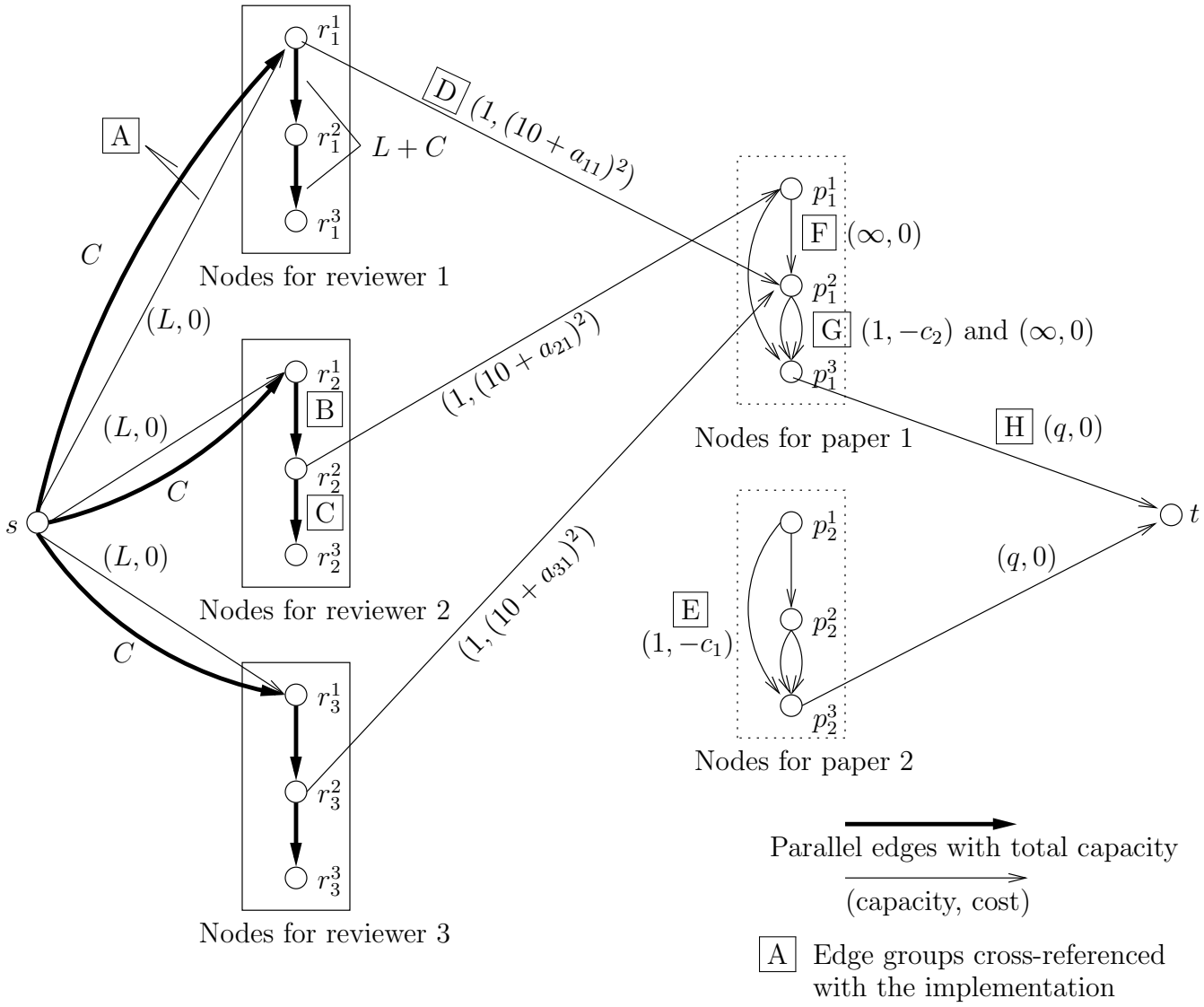

Figure 1: Flow Construction.

to uniform thresholds. (Since this is the only place the expertise values are used, we effectively assume expertise takes on only these three values.) Edges representing expert reviews enter p_j^1, edges for knowledgeable reviews enter p_j^2, and edges for general reviews enter p_j^3; the edge to the sink leaves p_j^3. A paper's first knowledgeable (or expert) review scores a bonus c_2 by traversing a unit-capacity edge of cost $-c_2$ from p_j^2 to p_j^3, and an additional expert review scores another bonus c_1 by traversing a unit-capacity edge of cost $-c_1$ from p_j^1 to p_j^3. In addition to the bonus edges, there are edges of zero cost and unlimited capacity that reviews can follow from p_j^1 to p_j^2 and from p_j^2 to p_j^3 in order to reach the sink. The choice to offer bonuses for two reviews was based on the value $q = 3$; this would be easy to change for other values of q.

In the example in Figure 1, paper 1 is interesting to reviewer 1 and boring to reviewers 2 and 3. Reviewer 2 is expert on paper 1, with reviewers 1 and 3 merely knowledgeable. (Reviewer edges for paper 2 are not shown.) This illustrates how, in principle, the preference and expertise relations might differ. Each is taken into account at a different stage of the construction.

The cost of a flow (assignment) is the sum of its reviewer overload costs, per-review costs, and reviewer boring / very boring load costs, minus paper bonuses. Any one of these components can be traded off against the others.

3 Evaluation

We have implemented the matching algorithm, based on the construction above, in Haskell. The construction is intended to illustrate ways to model real concerns that we find reasonable a priori. We do not have enough experience with real-world instances to be confident that each part of the construction serves its intended purpose or that the parameter values we have chosen are suitable. Fortunately, the parameter values are easy to change in the source code of our implementation, and even substantive changes to the graph structure are not too hard to make. At a minimum, we believe that min-cost max-flow provides a reasonable framework for global optimization of an assignment.

We have worked with Michael Hicks, program chair of POPL 2012, to apply our method to the paper assignment problem for that conference. The set of POPL 2012 reviewers consisted of the program committee (PC) and an external review committee (ERC), where the ERC served two purposes:

- To provide up to one knowledgeable (or expert) review per paper if prior knowledge of the topic was hard to find on the PC.
- To provide all reviews of papers with an author on the PC ("PC papers"), which were considered to pose an implicit conflict of interest to all PC members.

The special policy for ERC reviews of non-PC papers was realized via a more complex paper-side gadget, not described here. Based on an evaluation tool he wrote as well as manual inspection, Dr. Hicks was satisfied that the decisions made by the matching tool closely reflected the best he could have done manually.

We are looking for additional opportunities to apply the matching tool. Anyone interested is invited to contact us so we can help adapt it to the scenario and document the experience gained.

4 Getting the Tool

A distribution containing the source code for the matching tool as well as this document may be browsed or downloaded at:

`https://mattmccutchen.net/match/`

There are currently two branches:

- `master` has the tool as originally designed for NSF, with no distinction between preference and expertise.
- `popl2012` is the basis of the version used for POPL 2012. The main differences are that it has separate preference and expertise, support for "fixing" previously chosen reviewer-paper pairs (buggy, however), and the special ERC gadget.

We regret that we do not currently have a single well-maintained version of the tool to recommend.

References

[1] R. Ahuja, T. Magnanti and J. Orlin. Network Flows: Theory and Applications. *Prentice Hall.*