The AI Conference Paper Assignment Problem

Judy Goldsmith*
Dept of Computer Science
University of Kentucky

Robert H. Sloan†
Dept of Computer Science
University of Illinois-Chicago

Abstract

The Conference Paper Assignment Problem (CPAP) is the problem of assigning reviewers to conference paper submissions in a manner intended to minimize whingeing. It is assumed that papers are reviewed by members of a preset program committee (PC), each of whom has the opportunity to *bid* on papers prior to the assignment algorithm being run. In this survey, we show that CPAP is in P if the only information given is individual program committee members' preferences for individual papers. However, if both preferences and expertise (based on, say, keywords) are given, the problem is potentially more complex.

Introduction

It is a well-known problem of program committee chairs to assign papers to reviewers in such a way that the papers get reasonable coverage by PC members who are both sufficiently knowledgeable about the papers' topics and are willing to review those particular papers.

Some committees handle the Conference Paper Assignment Problem (CPAP) by sloughing responsibility off on a commercial or semi-commercial program; some write programs of their own. Others assign papers by hand, using some sort of rough heuristic or a backtracking algorithm.

Anyone who has been on a reasonable number of PCs has had the experience of being asked to review papers for which s/he has no expertise and/or no interest. Such reviews may end up perfunctory or may reflect the reviewer's crankiness at being asked to write the review.

Clearly, it is in the best interest of the PC and the research community at large to optimize paper assignments. In this paper, we discuss a variety of criteria for optimization, several algorithms in use by individuals and programs, and give an analysis of the complexity of the problems.

We assume that that PC members have the opportunity to *bid* on papers before they assigned, and that part or all of the bidding consists of rating papers on a scale of preferences, from "I do not want this paper but am willing to review it" up to "I really want to review this paper." In addition, we assume there is an option of saying "I cannot review this paper" (for instance, due to a conflict of interest). We will assume that these ratings are integers in the range of $0, \ldots, c$ for some $c \geq 1$, and that higher numbers indicate greater desire, and that the lowest rating of 0 means that the reviewer absolutely cannot review the paper, and that any higher rating indicates at least some miniscule willingness to rate the paper.

In some bidding processes, PC members also give independent ratings of their *expertise* for each paper. We will discuss the case with expertise ratings later; for now we simply point out that if there is an expertise rating of "utterly incompetent to review in this area," (which, incidentally, there usually is *not*), then for purposes of deciding whether there is any feasible paper assignment at all, that rating can be converted to an interest rating of "cannot review." Notice that here we are talking only of the use of expertise in the *bidding* process; most conferences ask reviewers to rate their expertise as part of their review, but that does not concern us here.

As we discuss later, there is a straightforward algorithm, essentially network flow, for checking whether there is a *feasible* paper assignment, where each paper gets a sufficient number of reviewers, each of which is at least marginally interested in reviewing that paper. If not, it is up to the PC chair to recruit new reviewers.

Except when we discuss how to use network flow to determine if there is a feasible paper, we will assume throughout that the problem instances have feasible solutions. Our goal is to find *good* solutions, for some measure of goodness.

Preferences *without* Expertise Ratings

Before we discuss optimality criteria, we will mention those algorithms that we understand to be in use or to have been used for recent conferences. We omit specific attribution to protect our sources.

Techniques in Current Use

In several cases, the techniques used to assign papers reflect the research community. For instance, a business-school colleague tells us that he always uses Integer Linear Programming (ILP). ILP is NP-complete, but there are some very good ILP solvers, and our colleague claims that they

*This material is based on work supported by NSF Grant No. ITR-0325063.

†This material is based upon work supported by the National Science Foundation under Grant No. CCF-0431059.

work well for CPAP, even though they probably are not finding optimal solutions.

Members of the constraint satisfaction community are said to use straight constraint solvers or weighted-constraint solvers. Again, this is not guaranteed to be fast—or optimal—but it probably works well.

Several of our informants claimed to use "round robin" assignments. We understand this to mean that they assign one paper to each reviewer, and then begin again, perhaps with some form of backtracking. Or perhaps they assign one reviewer per paper, etc.

One informant told us that his system was "based on graphs," and referred us to a paper that contained no useful details.

We did not receive an answer about the algorithm used by ConfMaster, which is quite popular for computer science conferences, and was used, for instance, for AAAI 2007.

One former PC chair told us that he had used a manual method based on greedily choosing reviewer/paper pairs that have the largest number of keywords in common, using balance as a secondary heuristic/backtracking guide. (We interpret "balance" as a goal of assigning roughly the same number of papers per reviewer.) Notice that this solution is based on both preferences and expertise, and thus, technically, belongs later on with our discussion of CPAP with expertise ratings in bidding.

In short, there are several methods being used. However, none are guaranteed to be both polynomial-time computable and to produce optimal assignments. First we will discuss what it means for an assignment to be feasible, and then we will discuss optimality criteria.

Feasible Solutions: Network Flow

In order to test whether a given instance of the Conference Paper Assignment Problem (CPAP) has a feasible solution, we reduce the problem to an instance of Network Flow, and apply any polynomial-time algorithm to compute the maximum flow through the network.

Given a set of r reviewers and p papers, and given a function $b(x, y) = v$ that takes as input a reviewer and a paper and returns that reviewer's bid for that paper, we create an instance of Network Flow as follows. We assume that each paper requires m reviews and each reviewer will review no more than u papers. Furthermore, we assume that $m * p \leq u * r$.

The network consists of $r + p + 2$ nodes:

- A set of nodes with one node for each reviewer, $\{x_1, \ldots, x_r\}$,
- A set of nodes with one node for each paper, $\{y_1, \ldots, y_p\}$,
- a source s, and a sink t.

Thus the node set is

$$V = \{s\} \cup \{x_1, \ldots, x_r\} \cup \{y_1, \ldots, y_p\} \cup \{t\}.$$

There are the following edges:

- edges of capacity 1 from x_i to y_j if reviewer i is willing, at least marginally, to review paper j;
- edges of capacity u from s to each x_i;
- edges of capacity m from each y_j to t.

We claim that there is a feasible solution for the CPAP instance if and only if the maximum flow through this network is exactly $m \cdot p$.

Optimality Criteria

If there are only two ratings that a PC member can give when she is bidding, "Absolutely will not review," and "at least marginally willing to review," then optimality is identical to feasibility. Henceforth, we implicitly assume that c, the number of possible distinct bids excluding the "Absolutely will not review," is at least 2. In practice, relatively small values, such as $c = 3$, seem to be popular.

We will also assume throughout that every paper is to get *exactly* m reviews. In practice, this means that PC members are only forced to do enough reviews to get all the papers adequately reviewed. Any reviewer who has additional capacity and wants to volunteer to read an additional paper that she is especially interested in is of course free to do so.

It is not clear what the appropriate optimality criterion should be. We list several that seem to be obvious candidates.

MAXIMUM TOTAL Maximize the total value of the bids. More formally, an assignment meets the maximum total criterion if it is a feasible assignment with exactly m reviews per every paper that maximizes the sum

$$\sum_{\text{paper } x \text{ assigned to reviewer } y} b(x, y) \ .$$

MAXIMIZE AVERAGE REVIEWER ASSIGNMENT This is equivalent to MAXIMUM TOTAL, since the average reviewer assignment is simply the total divided by the number of reviewers.

MAXIMIZE MOST WANTED Maximize the total number of assignments of paper x to reviewer y where the bid was $b(x, y) = c$ ("I really want it.").

MAXIMIZE WANTED A generalization of MAXIMIZE MOST WANTED is to maximize the total number of assignments where the bid was at least some particular value $v \leq c$.

MINIMIZE LEAST WANTED Minimize the total number of assignments of paper x to reviewer y where the bid was $b(x, y) = 1$ ("I really don't want it but could review it if I absolutely have to.").

MINIMIZE UNWANTED A generalization of MINIMIZE LEAST WANTED is to minimize the total number of assignments where the bid was at least some particular value $v \leq c$.

We will consider primarily the six criteria we just listed, but there are other possibilities. Note that these criteria all focus on the quality of the reviews, not on individual happiness. For example, one might maximize instead the *number of reviewers* who have at least one "I really want it" paper, or minimize the number that have any "I really don't want this" paper. Alternatively, one could maximize the number of reviewers whose average value over the bid values on all

papers assigned to them exceeds some threshold t. Yet another possibility would be to assign weights to individual members of the PC according to the general noisomeness of their complaints. Algorithms that optimize the weighted sum of the bid totals could be considered, but we do not so so here. There are other, more complex options from the literature of fair division, which we do not address here. In particular, we are looking for polynomial-time computable criteria, and most of the fair division criteria are computationally intensive.

How to Solve It

The Network Flow instance described earlier takes into account only whether the reviewer is *willing* to review each paper. It does not take into account the level of willingness, or preference, of the reviewer for each paper. That appears to add a layer of complexity to the problem.

However, it is possible to state the CPAP with maximum-total optimality criterion as a problem called the Minimum Cost Flow Problem (MCFP) (see, e.g., (Cormen *et al.* 2001; Jungnickel 2005)). MCFP generalizes the maximum flow problem. An instance of MCFP is a flow problem plus a cost on each directed edge, and a specified total flow amount. The cost is the cost of shipping one unit over that edge. The problem is to find a total flow of the specified amount that minimizes the total cost while meeting the usual flow graph constraints.

To solve the CPAP for the MAXIMUM TOTAL criterion, we extend our reduction to network flow for feasibility to a reduction to MCFP. We use the reviewers' bids to define the cost of the edges. However, normal encoding of bids assigns higher numbers to higher desirability. For this reduction, we require the inverse: as desirability rises, the (integer) value should decrease. Thus we will assign cost $c - b(x_i, y_j)$ to the edge $\langle x_i, y_j \rangle$. We assign cost 0 to each edge from s and each edge to t. We make the desired flow $m \cdot p$.

We claim that the minimum cost maximum flow (which will have flow $m \cdot p$, by our assumption of feasibility) will exactly correspond to the paper assignment that maximizes the total bid values. Furthermore, there is a polynomial-time algorithm to compute this flow. (See, e.g., (Jungnickel 2005, Chapter 11).)

The MCFP approach can also be adapted for the four criteria: MAXIMIZE MOST WANTED, MAXIMIZE WANTED, MINIMIZE LEAST WANTED, and MINIMIZE UNWANTED.

For example, consider MAXIMIZE MOST WANTED, where our goal is to maximize the total number of assigned paper–reviewer pairs $\langle x_i, y_j \rangle$ where the bid $b(x_i, y_j)$ was c, i.e., "I really want that paper." To maximize the number of such assignments, we assign a cost of 0 to every edge corresponding to a bid of c, and a cost of 1 to all other edges representing nonzero bids. This MCFP must maximize the number of desired assignments.

Notice that there might be many different paper assignments that all optimize the MAXIMIZE MOST WANTED criterion. If we like, we can refine the algorithm for the MAXIMIZE MOST WANTED criterion so that it chooses, among the solutions that optimize that criterion, a solution with maximum total value. To obtain such a solution, we change the cost of the edges corresponding to bids other than c. Instead of making the cost of all those $\langle x_i, y_j \rangle$ edges with $b(x_i, y_j) < c$ be 1, we instead make the cost be a large quantity that gets a little larger as the bid gets lower (less desired). A cost of $c \cdot m \cdot p + c - b(x_i, y_j)$ gives the desired paper assignment.

For the MAXIMIZE WANTED criterion, we assign the 0 cost to all edges $\langle x_i, y_j \rangle$ that have a bid $b(x_i, y_j)$ of at least the threshold, and the high cost to the other edges. MINIMIZE UNWANTED is similar.

Preferences and Expertise: Polyamory

It would seem that adding expertise to preferences in the bidding process would further constrain solutions, and therefore make the CPAP easier. However, that does not appear to be the case.

Strictly from the mathematical modeling point of view, we can again assume that the expertise of each reviewer for each paper is given as some integer in a fixed range.

From a practical point of view, during bidding, one probably does not want to ask reviewers to bid an expertise for each paper. The problem is that expertise ratings could be used to game the system and strengthen or weaken bids. Instead, authors of papers can be required to choose keywords from a preset list, and reviewers can be asked to rank their expertise on the keywords, prior to considering the list of submissions. One also needs some well-defined algorithm for comparing a reviewer's expertise with the paper's keywords and producing a number that represents the reviewer's expertise with respect to that paper.

In any event, one can consider that *papers prefer reviewers with higher expertise over reviewers with lower expertise*. This transforms the problem into a variant of a *bipartite matching* problem, with two sets of optimization variables: the reviewers' preferences and the papers' preferences. There are therefore many possible optimization criteria.

We can view this problem as a variant of the stable marriage problem. In brief, in the classic stable marriage problem, there are two disjoint sets each of the same size, called *men* and *women* and each element of each set has a total preference over elements of the other set. The problem is to find a *matching*, a set of pairs of one man and one woman such that each person belongs to exactly one pair with the *stability* property: there is no unmatched man–woman pair who each prefer one another to the person they are matched to. We refer the reader to their favorite algorithms text for a longer and clearer definition of the problem, and a proof that there is a linear-time algorithm, known as the Gale-Shapley algorithm, for finding stable marriages in the classic stable marriage problem's setting (Gale & Shapley 1962).

Optimality criteria: stability?

Stability has long been taken to be the most important optimization criteria for such real-world applications of stable marriage as medical resident–hospital matching. Stability, suitably generalized to the CPAP setting, seems like a reasonable criteria for goodness of assignments, but it is not

clear whether "stability" is the one best optimization criteria for the this setting. What makes instability so threatening to human marriage is the potential for two people who are not married to each other to run away together. Similarly, if a medical resident and a hospital both prefer each other to the result of the matching algorithm, then they may make a side agreement to ignore the results of the matching algorithm. It is, however, less likely that a conference submission and a reviewer will elope. One can imagine, however, a reviewer sighting a paper on a colleague's desk and announcing, "I know much more about this than she does. I will review this paper, and she can review this paper on the semantics of robot eschatology in my stead."

Incidentally, in the classic Stable Marriage Problem, most instances admit many different stable matches, and there are several notions of optimality of a stable match, that maximize the overall happiness of the participants, though there is a tradeoff concerning which set's happiness is maximized.

Differences from classic stable marriage

There are three ways in which our problem differs from the standard Stable Marriage Problem. All three variants have been studied in the Stable Marriage literature. Unfortunately while each individual variant can be solved in polynomial time by an extension of the Gale-Shapely algorithm, the combination of all three extensions together leads to an NP-complete problem.

Monogamy versus Polyamory The original version of the Stable Marriage Problem is monogamous: one man marries one woman. In our version, each paper has several reviewers and each reviewer covers several papers. This is the problem considered by (Baïou & Balinski 2000) in "Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry)". They showed that there is a polynomial-time algorithm for this case. Incidentally, we are not matching groups to groups, but individuals to many other individuals. That is, for example, even if Sloan and Goldsmith both review the paper on robot eschatology, they may review otherwise disjoint sets of papers. Therefore, we prefer to describe these matchings not as "polygamous" or "polyandrous", but in keeping with modern terminology, as **polyamorous.**

Many-one matchings have been studied in the context of the Scottish residency assignment problem, the married-doctors residency assignment problem, and the roommate assignment problem. One amusing result is the following. If there is a maximal stable matching such that x has fewer than capacity many "mates" then x gets exactly that list of mates in *all* maximal stable matchings (Gusfield & Irving 1989).

Complete versus Incomplete Lists

The next difference between our problem and the original Stable Marriage Problem is that our reviewers refuse certain papers, due to conflicts of interest. This is referred to in the stable-marriage literature as *incomplete lists*.

It is known that the stable marriage problem with incomplete lists can still be solved, if a solution exists, by the Gale-Shapley algorithm.

Baïou and Balinski's polynomial-time algorithm is for polyamorous marriage with incomplete lists. There is a minor difference between their definition and our setting. They assume that every person has a capacity for how many they can be matched to, and prefers be matched to that many from their list, as opposed to being matched to fewer. In our case, reviewers probably do not prefer to maximize their number of papers. However, it is the addition of the third difference with the classic Stable Marriage Problem that really causes us difficulty.

Total orders versus indifference The final difference between our problem and the original one is that our papers and reviewers do not generate a totally ordered ranking of their opposite numbers. Rather, they define a preset number of preference equivalence classes. This is referred to in the stable-marriage literature as either *lists with ties* or *indifference*.

It is known that the stable marriage problem with indifference can still be solved by the Gale-Shapley algorithm, simply by assigning arbitrary orderings within the equivalence classes. However, Stable Marriage with *both* indifference and incomplete lists is NP-complete even in the one-to-one case (Manlove *et al.* 2002). The problem is that once we have ties, two different ways of arbitrarily resolving the tie can lead to two different sizes of maximum matching. Indeed, Manlove et al. show that the one-to-one problem with ties and incomplete list is NP-complete even if there is only a single tie per preference list from one of the two sides (i.e., either the men/papers or women/reviewers), and no ties in the preference lists of the other side.

For the many-one case, Manlove et al. give a polynomial time 2-approximation algorithm finding either the smallest or the largest stable matching, so the problem is in APX.[1]

Even the one-to-one problem is *not* in PTAS;[2] that is, we cannot improve the 2-approximation to an arbitrarily good approximation ratio (Irving, Manlove, & O'Malley 2006). However, Iwama et al. have polynomial-time algorithms with approximation ratios better than 2 for the one-one case (Iwama, Miyazaki, & Okamoto 2006; Iwama, Miyazak, & Yamauchi 2007). No known approximation algorithms exist for the many-many case.

[1] APX, which appears to be a contraction of "approximation," is the class of all optimization problems that have a polynomial-time approximation algorithm giving *some* guaranteed approximation ratio. An approximation algorithm has an approximation ratio of c if it can be proven that the solution that the algorithm finds is at most c times worse than the optimal solution. See, e.g., (Wegener 2005).

[2] A PTAS, or Polynomial Time Approximation Scheme, is a family of approximation algorithms $\langle A_i \rangle$ for a given optimization problem such that Algorithm A_i has an approximation ratio of $1 + (1/i)$; i.e., we can obtain an approximation ratio arbitrarily close to 1. PTAS is also used for the class of all optimization problems that have such a family of approximation algorithms. See, e.g., (Wegener 2005). It is obviously desirable if an NP-complete problem one wants to solve is in PTAS.

Conclusions

This survey describes work in progress, and is not intended to fully cover the topic of optimal paper assignments. We intend to expand this survey by the time of the workshop.

There are many open questions about the complexity of paper assignment problems under various restrictions and optimality conditions. For instance: How does the number of preference options affect the complexity of optimal assignments? For instance, if the experience rankings were binary ("Yes, I have a clue," or "No clue"), then we are back in the case of only preferences. The computational complexity only arises when there are at least two could-possibly-be-assigned ranks in each of expertise and preference.

It is unclear what criteria should be used for comparing different assignments in the expertise-plus-preference case. Stability is one, and given stability, one can still compare, say, rank maximality (i.e., minimize dissatisfaction over the set of stable matchings).

Other criteria include maximizing the number of papers with at least one expert, or the number of reviewers with at least one (or at least r many) highly preferred papers, or minimizing the number of papers with no experts, or reviewers with no highly preferred papers. These are more finely tuned measures than rank optimality. One could also treat either case (with or without expertise) as a multi-criteria optimization problem, although that would raise the computational complexity significantly.

References

Baïou, M., and Balinski, M. 2000. Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry). *Discrete Applied Mathematics* 101:1–12.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to Algorithms*. Cambridge, MA: MIT Press, second edition.

Gale, D., and Shapley, L. 1962. College admissions and the stability of marriage. *American Mathematics Monthly* 9–15.

Gusfield, D., and Irving, R. W. 1989. *The Stable Marriage Problem: Structure and Algorithms*. The MIT Press.

Irving, R.; Manlove, D.; and O'Malley, G. 2006. Stable marriage with ties and bounded length preference lists. In *Proc. ACID, Volume 7 of Texts in Algorithmics*, 95–106. College Publications.

Iwama, K.; Miyazak, S.; and Yamauchi, N. 2007. A 1.875–approximation algorithm for the stable marriage problem. In *Proc. SODA 2007*.

Iwama, K.; Miyazaki, S.; and Okamoto, K. 2006. A $(2 - c(\log n/n))$-approximation algorithm for the stable marriage problem. *IEICE Trans. Info and Systems* E-89D:2380–2387. Preliminary version appeared in SWAT 2004.

Jungnickel, D. 2005. *Graphs, Networks and Algorithms*, volume 5 of *Algorithms and Computation in Mathematics*. Springer, second edition.

Manlove, D.; Irving, R.; Iwama, K.; Miyazaki, S.; and Morita, Y. 2002. Hard variants of stable marriage. *Theoretical Computer Science* 276:261–279.

Wegener, I. 2005. *Complexity Theory: Exploring the Limits of Efficient Algorithms*. Springer.