Paper Matching with Local Fairness Constraints

Ari Kobren, Barna Saha, Andrew McCallum
{akobren,barna,mccallum}@cs.umass.edu
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, Massachusetts

ABSTRACT

Automatically matching reviewers to papers is a crucial step of the peer review process for venues receiving thousands of submissions. Unfortunately, common paper matching algorithms often construct matchings suffering from two critical problems: (1) the group of reviewers assigned to a paper do not collectively possess sufficient expertise, and (2) reviewer workloads are highly skewed. In this paper, we propose a novel *local fairness formulation* of paper matching that directly addresses both of these issues. Since optimizing our formulation is not always tractable, we introduce two new algorithms, FairIr and FairFlow, for computing fair matchings that approximately optimize the new formulation. FairIr solves a relaxation of the local fairness formulation and then employs a rounding technique to construct a valid matching that provably maximizes the objective and only compromises on fairness with respect to reviewer loads and papers by a small constant. In contrast, FairFlow is not provably guaranteed to produce fair matchings, however it can be 2x as efficient as FairIr and an order of magnitude faster than matching algorithms that directly optimize for fairness. Empirically, we demonstrate that both FairIr and FairFlow improve fairness over standard matching algorithms on real conference data. Moreover, in comparison to state-of-the-art matching algorithms that optimize for fairness only, FairIr achieves higher objective scores, FairFlow achieves competitive fairness, and both are capable of more evenly allocating reviewers.

CCS CONCEPTS

• **Mathematics of computing → Matchings and factors**; **Network flows**; **Approximation algorithms**.

KEYWORDS

Paper Matching, Integer Programming, Network Flow, Approximation Algorithms

ACM Reference Format:
Ari Kobren, Barna Saha, Andrew McCallum. 2019. Paper Matching with Local Fairness Constraints. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330899

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330899

1 INTRODUCTION

In 2014, the program chairs (PCs) of the Neural Information Processing Systems (NeurIPS) conference conducted an experiment that allowed them to measure the inherent randomness in the conference's peer review procedure. In their experiment, 10% of the submitted papers were assigned to *two* disjoint sets of reviewers instead of one. For the papers in this experimental set, the PCs found that the two groups assigned to review the same paper disagreed about whether to accept or reject the paper 25.9% of the time. Accordingly, if all 2014 NeurIPS submissions were reviewed again by a new set of reviewers, about 57% of the originally accepted papers would be rejected [23].

The NIPS experiment is only one of many studies highlighting the poor reliability of the peer reviewing process. For example, another study finds that the rate of agreement between reviewers for a clinical neuroscience journal is not significantly different from chance [25]. This is particularly troublesome given that decisions regarding patient care, expensive scientific exploration, researcher hiring, funding, tenure, etc. are all based, in part, on published scientific work and thus on the peer reviewing process.

Unsurprisingly, previous work shows that experts are able to produce higher quality reviews of submitted publications than non-experts. Experts are often able to develop more "discerning" opinions about the proposals under review [1, 12] and some researchers in cognitive science and artificial intelligence claim that experts can make more accurate decisions than non-experts about uncertain information [11]. Clearly peer review outcomes are likely to be of higher quality if each paper were reviewed exclusively by experts in the paper's topical areas. Unfortunately, since experts are relatively scarce, this is often impossible. Especially for many computer science venues, which are faced with increasingly large volumes of submissions, assigning only experts to each submission is impossible given typical reviewer load restrictions. Further exacerbating the problem, conference decision processes are dictated by a strict timeline. This necessitates significant automation in matching reviewers to submitted papers, highly limiting the extent to which humans can significantly intervene.

Automated systems often cast the paper matching problem as a global maximization of reviewer-paper *affinity*. In particular, each reviewer-paper pair has an associated affinity score, which is typically computed from a variety of factors, such as: expertise, area chair recommendations, reviewer bids, subject area matches, etc. The optimal matching is one that maximizes the sum of affinities of assigned reviewer-paper pairs, subject to *load* and *coverage* constraints, which bound the number of papers to which a reviewer can be assigned and dictate the number of reviews each paper must receive, respectively [3, 28]. While optimizing the global objective has merit, a major disadvantage of the approach is that it can lead

to matchings that contain papers assigned to a set reviewers who lack expertise in that paper's topical areas [7, 26]. This is because in constructing a matching that maximizes the global objective, allocating more experts to one paper at the expense of another may improve the objective score. In order to be fair, it is important to ensure that each paper is assigned to a group of reviewers who instead possess a minimum acceptable level of expertise.

Recent work has attempted to overcome these problems by either (a) introducing strict requirements on the minimum affinity of valid paper-reviewer matches, or (b) optimizing the sum of affinities of the one paper that is worst-off [7, 26]. However, restricting the minimum allowable affinity often renders the problem infeasible as there may not exist any matching that provides sufficient coverage to all papers subject to the threshold. Previously proposed algorithms that maximize the sum affinities for the worst-off paper do result in matchings that are more fair, but they also suffer from two disadvantages: (1) they do not simultaneously optimize for the overall best assignment (measured by sum total affinity), and (2) they are agnostic to lower limits on reviewer loads (which are common in practice) and thus may produce matchings in which reviewers are assigned to dramatically different numbers of papers.

To address these issues, we introduce the *local fairness formulation* of the paper matching problem. Our novel formulation is cast as an integer linear program that (1) optimizes the global objective, (2) includes both upper *and lower* bound constraints that serve to balance the reviewing load among reviewers, and (3) includes *local fairness constraints*, which ensure that each paper is assigned to a set of reviewers that collectively possess sufficient expertise.

The local fairness formulation is NP-Hard. To address this hardness, we present FairIr, the **FAIR** matching via **I**terative **R**elaxtion algorithm that jointly optimizes the global objective, obeys local fairness constraints, and satisfies lower (and upper) bounds on reviewer loads to ensure more balanced allocation. FairIr works by solving a relaxation of the local fairness formulation and rounding the corresponding fractional solution using a specially designed procedure. Theoretically, we prove that matchings constructed by FairIr may only violate the local fairness and load constraints by a small margin while maximizing the global objective. In experiments with data from real conferences, we show that, despite theoretical possibility of constraint violations, FairIr never violates reviewer load constraints. The experiments also reveal that matchings computed by FairIr exhibit higher objective scores, more balanced allocations of reviewers and competitive treatment of the most disadvantaged paper when compared to state-of-the-art approaches that optimize for fairness.

In real-conference settings, a program chair may desire to construct and explore many alternative matchings with various inputs, which demands an efficient fair matching algorithm. Toward this end, we present FairFlow, a min-cost-flow-based heuristic for constructing fair matchings that is faster than FairIr by more than 2x. While matchings constructed by FairFlow are not guaranteed to adhere to a specific degree of fairness (like FairIr or previous work), in experiments, FairFlow often constructs matchings exhibiting fairness and objective scores close to that of FairIr in a fraction of the time. Unlike FairIr and matching algorithms that rely on linear programming, FairFlow operates by first maximizing the global objective and then refining the corresponding solution through a series of min-cost-flow problems in which reviewers are reassigned from the most advantaged papers to the most disadvantaged papers.

This paper is organized as follows. Section 2 presents the standard paper matching formulation that optimizes the global objective. Section 3 covers our main contribution by providing the local fairness formulation of paper matching and describes FairIr and its formal guarantees. Section 4 presents the more efficient FairFlow heuristic. In Section 5, we experimentally show the effectiveness of our approach over other approaches on several datasets coming from real conferences.

2 REVIEWER ASSIGNMENT PROBLEM

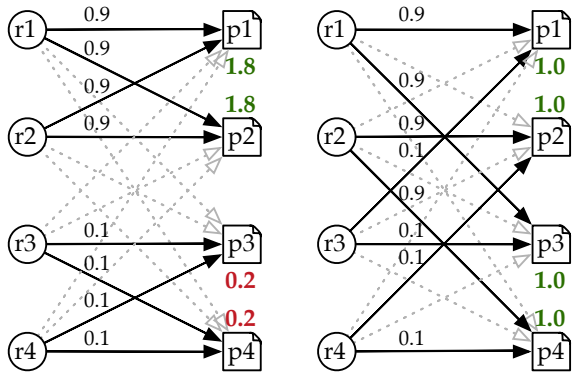
Popular academic conferences typically receive thousands of paper submissions. Immediately after the submission period closes, papers are automatically matched to a similarly sized pool of reviewers. A *matching* of reviewers to papers is constructed using real-valued reviewer-paper *affinities*. The affinity between a reviewer and a paper may be computed from a variety of factors, such as: expertise, bids, area chair recommendations, subject area matches, etc. Previous work has explored approaches for modeling reviewer-paper affinity via latent semantic indexing, collaborative filtering or information retrieval techniques [3–5]. We do not develop affinity models in this work. Instead, we focus on algorithms for matching papers to reviewers given the affinity scores. In the literature, this matching problem is known by many names; we choose *the reviewer assignment problem* (RAP) [14, 26].

The RAP is often accompanied by a two types of constraints: *load constraints* and *coverage constraints* [7]. A load constraint bounds the number of papers assigned to a reviewer; a coverage constraint defines the number of reviews a paper must receive. Typically, all papers must be reviewed the same number of times. Reviewers do not always have equal loads, although a highly uneven load is inherently unfair and may lead to reviewers declining to review or not submitting reviews on time.

Formally, let $R = \{r_i\}_{i=1}^{N}$ be the set of reviewers, $P = \{p_j\}_{j=1}^{M}$ be the set of papers and $\mathbf{A} \in \mathbb{R}^{|R|\times|P|}$ be a matrix of reviewer-paper affinities. The RAP can be written as the following integer program:

$$\begin{aligned} \max \quad & \sum_{i=1}^{|R|}\sum_{j=1}^{|P|} x_{ij}\mathbf{A}_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{|P|} x_{ij} \le U_i, && \forall i = 1, 2, ..., |R| \\ & \sum_{i=1}^{|\mathcal{R}|} x_{ij} = C_j, && \forall j = 1, 2, ..., |P| \\ & x_{ij} \in \{0, 1\}, && \forall i, j. \end{aligned}$$

Here, $\{U_i\}_{i=1}^{|R|}$ is the set of upper bounds on reviewer loads, and $\{C_j\}_{j=1}^{|P|}$ represents the coverage constraints. The matching of reviewers to papers is encoded in the variables x_{ij}, where $x_{ij} = 1$ indicates that reviewer r_i has been assigned to paper p_j. In this formulation, the objective is to maximize the sum of affinities of reviewer-paper assignments (subject to the constraints); it can be solved optimally in polynomial time with standard tools [28].



(a) Unfair Matching. (b) Fair Matching.

Figure 1: Fair & Unfair Matchings. 4 papers and 4 reviewers with $U = L = C = 2$. $r1$ and $r2$ have affinity 0.9 with all papers; $r3$ and $r4$ have affinity 0.1 with all papers. Solid lines indicate assignments, the bold number adjacent to each paper corresponds to its paper score. Matchings in both Figures 1a and 1b achieve equivalent objective scores, but in Figure 1a $p3$ and $p4$ are only assigned to reviewers with low affinity.

In practice, lower bounds on reviewer loads are often invoked in order to spread the reviewing load more equally across reviewers. The formulation above can be augmented to include the lower bounds by adding the following constraints:

$$\sum_{j=1}^{|P|} x_{ij} \geq L_i, \forall i = 1, 2, ..., |R|,$$

where $\{L_i\}_{i=1}^{|R|}$ is the set of lower bounds on reviewer loads. The resulting problem is still efficiently solvable. Note that the formulation above, with and without lower bounds, is currently employed by various conferences and conference management software, for example: TPMS, OpenReview, CMT and HotCRP [2, 26]. We will henceforth refer to the above two formulations as the TPMS RAP, where the inclusion of lower bounds will be clear from context.

3 FAIR PAPER MATCHING

It is well-known that optimizing the TPMS RAP can result in unfair matchings [7, 26]. To see why, consider the example RAP in Figure 1, in which there are 4 papers and 4 reviewers, and define the *paper score* for paper p to be the sum of affinities of reviewers assigned to paper p. In the example, each paper must be assigned 2 reviewers and each reviewer may only be assigned up to 2 papers. Even though the matchings in Figures 1a and 1b obtain equivalent objective scores under the TPMS RAP, the matching in Figure 1a causes papers $P3$ and $P4$ to have much lower paper scores than papers $P1$ and $P2$. In practice, this may indicate that $P3$ and $P4$ have been assigned to a collection of reviewers, none of whom are well-suited to provide an expert evaluation. The assignment in Figure 1b is clearly more equitable with respect to the papers (and reviewers), but the TPMS RAP does not prefer this matching since it seeks to globally optimize affinity.

3.1 Local Fairness Constraints

We propose to prohibit such undesirable matchings by augmenting the TPMS RAP with *local fairness constraints*. That is, we constrain the paper score at each paper to be no less than T [29]. Formally,

$$\sum_{i=1}^{|R|} x_{ij} A_{ij} \geq T, \forall j = 1, 2, ..., |P|.$$

We refer to the resulting RAP formulation as the *local fairness formulation*. While adding local fairness constraints is simple, this formulation is NP-Hard since it generalizes the max-min fair allocation problem [29]. To avoid the hardness of the local fairness formulation, one might instead be tempted to constrain the minimum affinity of valid assignments of reviewers to papers. However, doing so often results in infeasible assignment problems [30].

3.2 FairIr

We present FairIr, an approximation algorithm for solving the local fairness formulation. The algorithm is capable of accepting both lower and upper bound constraints on reviewer loads (as well as coverage constraints). By nature of being approximate, FairIr is guaranteed to return a matching in which any local fairness constraint may be violated by at most $A_{max} = \max_{r \in R, p \in P} A_{rp}$—the highest reviewer-paper affinity, and any reviewer load constraint (upper or lower bound) is violated by at most 1. Moreover, it achieves an 1-approximation (no violation) in the global objective. We call attention to the fact that our guarantees hold even though FairIr is able to accommodate constraints on reviewer lower bounds while optimizing a global objective, unlike most state-of-the-art paper matching algorithms with theoretical guarantees [7, 26]. Note that in practice lower bounds are often an input to the RAP in order to spread the reviewing load more equally across reviewers.

Our algorithm proceeds in rounds. In each round, FairIr relaxes the integrality constraints of the local fairness formulation (i.e., each x_{ij} can take any value in the range $[0, 1]$) and solves the resulting linear program. Any x_{ij} with an integral assignment (i.e., either 0 or 1) is constrained to retain that value in subsequent rounds. Among the x_{ij}s with non-integral values, FairIr looks for a paper such that at most 3 reviewers have been fractionally assigned to it (the paper may have any number of integrally assigned reviewers). If such a paper is found, FairIr drops the corresponding local fairness constraint. If no such paper is found, FairIr finds a reviewer with at most 2 papers fractionally assigned to it and drops the corresponding load constraints. The next round proceeds with the modified program. As soon as a matching is found that contains only integral assignments, that matching is returned. Algorithm 1 contains pseudocode for FairIr.

Theorem 3.1. *Given a feasible instance of the local fairness formulation $\mathcal{P} =< R, P, L, U, C, A, T >$, FairIr always terminates and returns an integer solution in which each local fairness constraint may be violated by at most A_{max}, each load constraint may be violated by at most 1 and the global objective is maximized.*

The proof of Theorem 3.1 is found in the appendix.

Theorem 3.1 requires that the instance of the local fairness formulation be feasible. A RAP instance may be *infeasible* if T is too

Algorithm 1 FairIr ($\mathcal{P}$)

```
P' ← relax(P)
X ← {x_ij | i ∈ [|R|], j ∈ [|P|]}
while X is not empty do
  s ←maximize(P')
  for x_ij ∈ X do
    if x_ij == 0 then
      fix x_ij = 0;   X ← X\x_ij
    end if
    if x_ij == 1 then
      fix x_ij = 1;   X ← X\x_ij
    end if
  end for
  if p ∈ P has at most 3 fractional assignments then
    drop-fairness-constraint(p)
  end if
  if no fairness constraints dropped then
    if r ∈ R has at most 2 fractional assignments then
      drop-loads(r)
    end if
  end if
end while
```

large, or if $\sum_{i=0}^{|R|} U_i < \sum_{j=0}^{|P|} C_j$. Checking the second condition is trivial. To check if T is too large, simply check if the corresponding relaxed local fairness formulation is infeasible. By Algorithm 1, if the relaxed program is feasible, then FairIr must return an integer solution for that instance. Formally,

Fact 1. *If an instance of the local fairness formulation, $\mathcal{P}$, is feasible after the integrality constraints on x_{ij}s have been removed, then Algorithm 1 returns an integral (possibly approximate) solution.*

Thus, by Fact 1, testing whether or not FairIr will return an integer solution for an instance of the local fairness formulation requires solving the relaxed program. In practice, a binary search over the feasible range of T is performed and the highest T yielding a feasible program is selected. Such a binary search requires solving the relaxed formulation several times and can add to the computational complexity. Overall, the running time of the algorithm is dominated by the number of times the linear program solver is invoked. Note that during each iteration of FairIr, many constraints may be dropped, which helps to improve scalability without sacrificing the theoretical guarantees. Also, note that by dropping constraints during each iteration the objective score can only increase.

4 FASTER FLOW-BASED MATCHING

For real conferences, paper matching is an interactive process. A PC may construct one matching, and upon inspection, decide to tune the affinity matrix, A, and compute a new matching. Alternatively, a PC may browse a matching and decide that certain reviewers should not be assigned to certain papers, or, that certain reviewers *must* review certain papers. After imposing the additional constraints, ideally, a new matching could be constructed efficiently.

FairIr is founded on solving a sequence of linear programs, and thus may not be efficient enough to support this kind of interactive paper matching when the number of papers and reviewers is large. Other similar algorithms, which consider local constraints, also may not be efficient enough because they too rely on linear program solvers [7, 26]. Therefore, we introduce a min-cost flow-based heuristic for solving the local fairness formulation that is significantly faster than other state-of-the-art approaches. While our flow-based approach does not enjoy the same performance guarantees of FairIr, empirically, we observe that it constructs high quality matches on real data (Section 5).

4.1 Paper Matching as Min-cost Flow

We begin by describing how to solve the TPMS RAP using algorithms for *min-cost flow* (MCF). Our first focus is on RAP instances without constraints on reviewer load lower bounds. Then we describe briefly how load lower bounds can be incorporated.

Construct the following graph, $\mathcal{G}$, in which each edge has both an integer cost and capacity:

(1) create a source node, s, with supply equal to the sum over papers of the corresponding coverage constraint: $\sum_{j=1}^{|P|} C_j$;
(2) create a node for each reviewer $r_i \in R$ and a directed edge between s and each such node with capacity U_i and cost 0;
(3) create a node for each paper $p_j \in P$ and create a directed edge from each reviewer, r_i, to each paper with cost $-A_{ij} \cdot W$, where W is a large positive number to ensure that the cost of each edge is integer. Each such edge has capacity 1;
(4) construct a sink node t with demand equal to the supply; create a directed edge from each paper $p_j \in P$ to t with capacity C_j and cost 0.

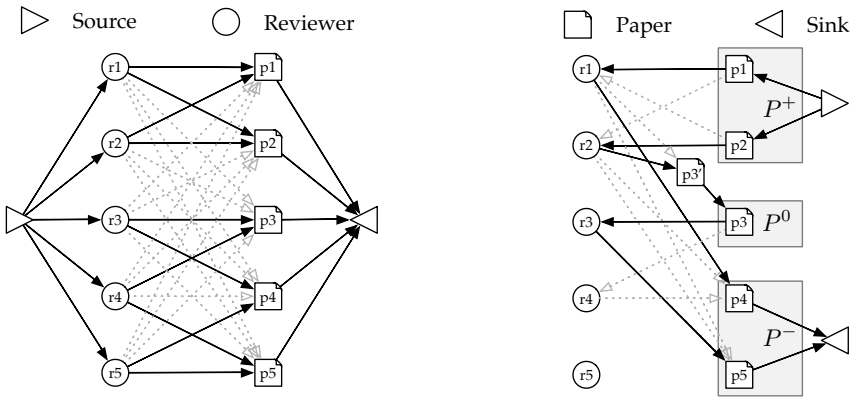
Solve MCF for $\mathcal{G}$, i.e., find the set of edges in $\mathcal{G}$ used in sending flow from s to t such that: the demand at t is satisfied, no more flow is sent through each edge than its capacity, and the sum total cost of all utilized edges is minimal. Note that algorithms like Ford-Fulkerson can be used to solve MCF and many efficient implementations are publicly available. It can be shown that the optimal flow plan on this graph corresponds to the optimal solution for the TPMS RAP. In particular, each edge between a reviewer and paper utilized in the optimal flow plan corresponds to an assignment of a reviewer to a paper. See Figure 2a for a visual depiction of the $\mathcal{G}$.

4.2 Locally Fair Flows

We introduce a MCF-based heuristic, FairFlow, for approximately solving the local fairness formulation via a sequence of MCF problems. Our algorithm is inspired by the combinatorial approach for (approximately) solving the scheduling problem on parallel machines [6]. FairFlow is comprised of three phases that are repeated until convergence. In the first phase, a valid assignment is computed and the papers are partitioned into groups; in the second phase, specific assignments are dropped; in the third phase, the assignment computed in the first phase is refined to promote fairness.

In more detail, in phase 1 of FairFlow, $\mathcal{G}$ is constructed using the 4 steps above (Section 4.1) and an assignment is constructed using MCF. Then, the papers are partitioned as follows:

$$G(p_j) = \begin{cases} P^+ & \sum_{r_i \in R} x_{ij} A_{ij} \geq T \\ P^0 & T > \sum_{r_i \in R} x_{ij} A_{ij} \geq T - A_{max} \\ P^- & \text{otherwise.} \end{cases}$$



(a) RAP as flow. **(b) Refinement Network.**

Figure 2: FairFlow. Darker bold arrows indicate utilized edges, lighter dotted arrows indicate edges that are not utilized. A darker edge originating at a reviewer r and ending at a paper p corresponds to the assignment of r to p. A darker edge originating at a paper p and ending at a reviewer r corresponds to the *"unassignment"* of reviewer r from p. In the first step of FairFlow (Figure 2a), MCF is solved in $\mathcal{G}$. Papers are then grouped into P^+, P^0, and P^- and the refinement network $\mathcal{G}'$ is constructed. Edges are constructed to route flow from P^+ to P^-. Notice dummy node, $p3'$, which serves to limit the flow to $p3$.

In words, the first group, P^+, contains all papers whose paper score is greater than or equal to T; the second group, P^0, contains all papers not in P^+ but whose paper score is greater than T *minus* the maximum score; the third group, P^-, contains all other papers.

In the second phase, for each paper $p \in P^-$, the reviewer assigned to that paper in phase 1 with the lowest affinity is *unassigned*.

In the third phase, a *refinement network*, $\mathcal{G}'$, is constructed. At a high-level, the refinement network routes flow from the papers in P^+ back through their reviewers and eventually to the papers in P^- with the goal of reducing the number of papers with paper scores less than $T - A_{max}$. The network is constructed as follows:

(1) create a source node, s, with supply equal to the minimum among the number of papers in P^+ and P^-;
(2) create a node for each $p \in P$; for each $p \in P^+$, create an edge from s to p with capacity 1 and cost 0;
(3) create a node for each reviewer $r \in R$;
(4) for each paper $p \in P^+$, create an edge with capacity 1 and cost 0 from p to each reviewer assigned to p;
(5) for each paper $p \in P^0$, create a dummy node, p' and construct an edge from p' to p with capacity 1 and cost 0.
(6) for each reviewer, r, assigned to a paper in P^+, create an edge with capacity 1 and cost 0 to each dummy paper, p', if r was not assigned to the paper to which p' is connected;
(7) for each paper $p \in P^0$ with dummy node p', let S_p be the current paper score at p, let $R(p')$ be the set of reviewers with edges ending at p' and let $R(p)$ be the set of reviewers currently assigned to p. Let A_{min} be the minimum affinity among the reviewers in $R(p')$ with respect to p. For each $r \in R(p)$ construct an edge with capacity 1 and cost 0 from p to each r if $T - A_{max} \leq S_p + A_{min} - A_{rp}$;
(8) for each reviewer, r, construct an edge with capacity 1 to each paper in $p \in P^-$ if r is not currently assigned to that paper. If assigning r to p would cause p's group to change to P^0, the cost of the edge is $-A_{rp} \cdot Z$, where $Z >> W$; otherwise, the cost is $-A_{rp} \cdot W$ (again, Z is a large constant that ensures that edge costs are integral);
(9) create a sink node t with demand equal to the supply at s; for each paper $p \in P^-$ construct an edge from p to t with capacity 1 and cost 0.

A visual illustration of the refinement network appears in Figure 2b.

After the network is constructed, MCF in $\mathcal{G}'$ is solved. The MCF in the refinement network effectively reassigns up to 1 reviewer from each paper in P^+ to a paper in either P^0 or P^-. Additionally, up to 1 reviewer from each paper in P^0 may be reassigned to a paper in P^-. As before, any edge in the optimal flow plan from a reviewer to a paper (or that paper's dummy node) corresponds to an assignment. Any edge from a paper to a reviewer corresponds to *unassigning* the reviewer from the corresponding paper.

Formally, we prove the following fact:

Fact 2. *After modifying an assignment according to the optimal flow plan in $\mathcal{G}'$, no new papers will be added to P^-.*

The proof of Fact 2 appears in the appendix.

After solving MCF in the refinement network, some papers in P^+ and P^- may be assigned $C - 1$ reviewers, which violates the paper capacity constraints. To make the assignment valid, solve MCF in the original flow network (Figure 2a) with respect to the current assignment, the available reviewers, and the papers in violation.

FairFlow can only terminate after a valid solution has been constructed (i.e., after phase 1). The three phases are repeated until either: a) there are no papers in P^- or b) the number of papers in P^- remains the same after two successive iterations.

Load Lower Bounds. Incorporating reviewer load lower bounds can be done by adding a single step to FairFlow. Specifically, in phase 1, first construct a network where the capacity on the edge from s to r_i is L_i (rather than U_i). The total flow through the network is $\sum_{i=1}^{|R|} L_i$ and thus all load lower bounds are satisfied. Once

this initial flow plan is constructed, record the corresponding assignments and update the capacity of each edge between s and r_i to be $U_i - L_i$. Similarly, update the capacity of each edge between p_j and t to be the difference between the paper's coverage constraint and the number of reviewers assigned to p_j in the initial flow plan. The flow plan through the updated network, combined with the initial flow plan, constitute a valid assignment. Afterwards, continue with phases 2 and 3 as normal. The additional step must be performed in each invocation of phase 1.

5 EXPERIMENTS

In this section we compare 4 paper matching algorithms:

(1) **TPMS** - optimal matching with respect to the TPMS RAP.
(2) **FairIr** - our method, Algorithm 1.
(3) **FairFlow** - our min-cost-flow-based algorithm (Section 4.2).
(4) **PR4A [26]** - state-of-the-art flow-based paper matching algorithm that maximizes the minimum paper score. For large problems we only run 1 iteration (PR4A (i1)).

TPMS, FairIr and PR4A are implemented using Gurobi [9]. FairFlow is implemented using OR-Tools[1].

In our experiment we use data from 3 real conferences[2]. Each dataset is comprised of: a matrix of paper-reviewer affinities (paper and reviewer identities are anonymous), a set of coverage constraints (one per paper), and a set of load upper bound constraints (one per paper). One of our datasets also includes load lower bounds. We do not evaluate PR4A on datasets when the load lower bounds are included since it was not designed for this scenario.

We report various statistics of each matching. For completeness, we also include the runtime of each algorithm. However, note that an algorithm's runtime is significantly affected by a number of factors, including: hardware, the extent to which the algorithm has been optimized, dataset on which it is run, etc. All experiments are run on the same MacBook Pro with an Intel i7 processor.

Finding fairness thresholds. Both FairIr and FairFlow take as input a fairness threshold, T. Since the best value of this threshold is unknown in advance, we search for the best value using 10 iterations of binary search. For FairIr, at iteration i with threshold T_i, we use a linear programming solver to check whether there exists an optimal solution to the relaxation of the corresponding local fairness formulation. By Fact 1, if a solution exists, then FairIr will successfully return an integer solution. For FairFlow we do a similar binary search and return the threshold that led to the largest minimum paper score. In our implementation of FairFlow, when we test a new threshold T during the binary search, we initialize from the previously computed matching.

Matching profile boxplots. We visualize a matching via a set of paper score quintiles, which we call it's *profile*. To construct the profile of a matching, compute the paper score of each paper and sort in non-decreasing order. The sorted list of scores is divided into 5 equally-szed groups[3]. Each group of sorted paper scores is further divided into 4 even groups, a, b, c and d (with a and d containing the smallest and largest paper scores, respectively). In each profile visualization that follows, the box in each column is defined by the minimum score in b, b_{min}, and maximum score in c, c_{max} for the corresponding group (i.e, quintile). The lowest horizontal line in a column is defined by the smallest paper score that is greater than or equal to $b_{min} - \frac{c_{max} - b_{min}}{2}$; the highest horizontal line in the column is defined by the largest paper score that is smaller than or equal to $c_{max} + \frac{c_{max} - b_{min}}{2}$. The rest of the points are considered outliers and denoted by red x's. The median paper score among a, b, c and d is represented as an orange line. A matching's profile provides a visual summary of the distribution of paper scores it induces, including the best and worst paper scores.

5.1 Medical Imaging and Deep Learning

In our first experiment we use data from the Medical Imaging and Deep Learning (MIDL) Conference. The data includes affinities of 177 reviewers for 118 papers. The affinities range from -1.0 to 1.0. Each paper must be reviewed by 3 reviewers and each reviewer must be assigned no more than 4 and no fewer than 2 papers (i.e., the data includes upper and lower bounds on reviewer loads).

Figure 3 displays the profiles of matchings computed by the 4 algorithms with and without lower bounds. Without lower bounds, all algorithms produce similar profiles, except that the maximum paper score achieved by PR4A and FairFlow are lowest. Somewhat similarly, these two algorithms achieve lower objective scores, which is likely a result of the fact that neither explicitly maximizes the global sum of paper scores. Interestingly, TPMS constructs a matching that is relatively fair with respect to paper scores even though it is not designed to do so.

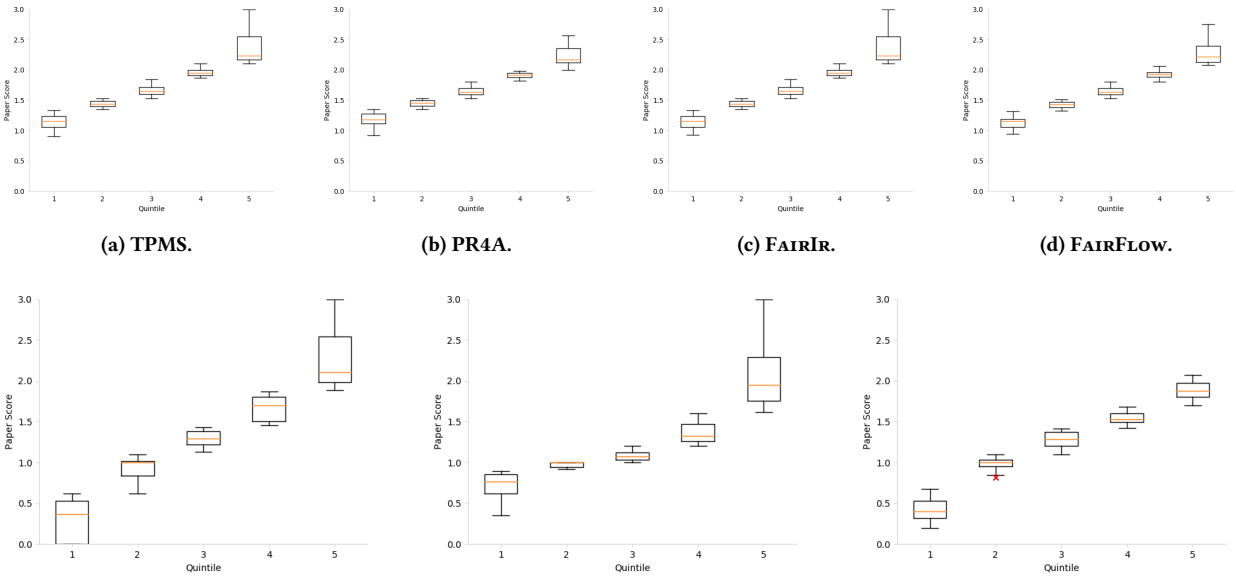
When lower bounds are considered, the algorithms produce much different profiles. First, notice that TPMS constructs a matching in which some papers have a corresponding paper score of 0–signaling an unfair assignment. Of the fair matching algorithms, FairIr's profile includes a higher minimum paper score, a higher maximum paper score, and a higher objective score. However, FairIr is 40% slower than FairFlow. Also note that on this small dataset, we run PR4A with no upper bound on the number of iterations (hence the long runtime). Table 1 (first block) contains matching statistics of the various algorithms for MIDL.

5.2 CVPR

Our next experiment is performed with respect to data from a previous year's Conference on Computer Vision and Pattern Recognition (CVPR). The data includes the affinities of 1373 reviewers for 2623 papers, which amounts to a substantially larger problem than that posed by the MIDL data. All affinities are between 0.0 and 1.0. As before, each paper must be reviewed by 3 different reviewers. Each reviewer may not be assigned to more than 6 papers. Our data does not contain lower bounds. For the purpose of demonstration, we construct a set of synthetic reviewer load lower bounds where all reviewers must review at least 2 papers.

The results are contained in Figure 4 and Table 1 (second block). As before, FairFlow is the fastest fair matching algorithm, achieving 2x speedup over FairIr and an order of magnitude speedup over PR4A when lower bounds are excluded. When lower bounds are included, FairFlow is still 100s (15%) faster than FairIr. PR4A

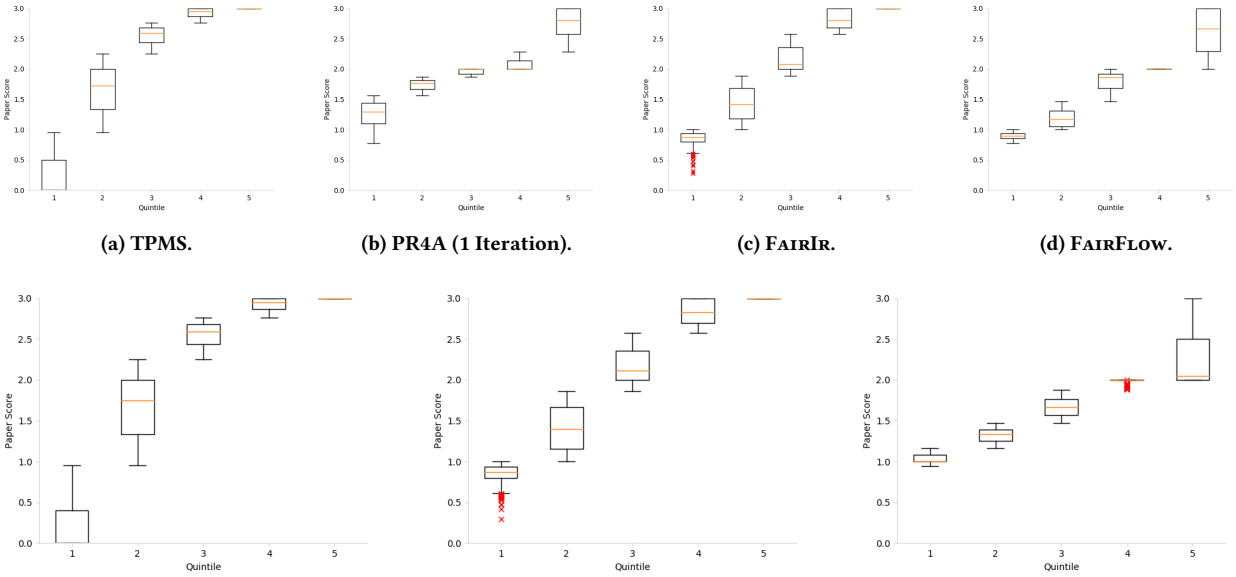
[1] https://developers.google.com/optimization/
[2] Our data is anonymous and kindly provided by OpenReview.net and the Computer Vision Foundation.
[3] Most datasets do not include a number of papers that is divisible by 5; in this case, the last quintile has fewer papers.



(a) TPMS. (b) PR4A. (c) FairIr. (d) FairFlow.

(e) TPMS with lower bounds. (f) FairIr with lower bounds. (g) FairFlow with lower bounds.

Figure 3: MIDL. Figures 3a-3d visualize profiles of matchings computed by TPMS, PR4A, FairIr and FairFlow on the MIDL data with reviewer load lower bounds excluded. All 4 algorithms construct similar profiles. Figures 3e-3g visualize profiles of matchings constructed with respect to the lower bounds. The introduction of lower bounds leads to many papers with low paper scores in the TPMS matching.



(a) TPMS. (b) PR4A (1 Iteration). (c) FairIr. (d) FairFlow.

(e) TPMS with lower bounds. (f) FairIr with lower bounds. (g) FairFlow with lower bounds.

Figure 4: Matching Profiles for CVPR.

and FairIr achieve similar fairness. Interestingly, FairFlow finds the matching with highest degree of fairness when lower bounds on reviewing loads are applied. However, this comes at the expense of a relatively low objective score. FairIr constructs a more fair matching than TPMS, but not than the other two fair matching algorithms. This is unsurprising because FairIr optimizes the global

Data	Bounds	Alg	Time (s)	Obj	Min PS	Max PS	Mean PS	Std PS	Min RA	Max RA	Std RA
MIDL	Upper	TPMS	*0.10*	*201.88*	0.90	*3.00*	**1.71**	0.45	0.00	4.00	1.80
	Upper	PR4A	293.83	197.32	0.92	2.57	1.67	***0.38***	0.00	4.00	1.79
	Upper	FairIr	1.60	**201.83**	0.93	***3.00***	**1.71**	0.45	0.00	4.00	1.80
	Upper	FairFlow	**1.15**	197.67	**0.94**	2.75	1.68	0.41	0.00	4.00	1.80
	Lower + Upper	TPMS	*0.17*	*150.04*	0.00	*3.00*	*1.27*	0.69	2.00	2.00	0.00
	Lower + Upper	FairIr	3.01	**145.56**	**0.35**	***3.00***	**1.23**	**0.50**	2.00	2.00	0.00
	Lower + Upper	FairFlow	**2.17**	143.12	0.19	2.07	1.21	**0.50**	2.00	2.00	0.00
CVPR	Upper	TPMS	*47.24*	*5443.64*	0.00	3.00	*2.08*	1.07	0.00	6.00	0.82
	Upper	PR4A (i1)	3251.37	5134.08	**0.77**	3.00	1.96	***0.52***	0.00	6.00	1.24
	Upper	FairIr	594.51	**5373.39**	0.27	3.00	**2.05**	0.84	0.00	6.00	0.83
	Upper	FairFlow	**225.29**	4444.95	**0.77**	3.00	1.69	0.64	2.00	6.00	***0.61***
	Lower + Upper	TPMS	*49.62*	*5443.64*	0.00	3.00	*2.08*	1.07	2.00	6.00	0.78
	Lower + Upper	FairIr	694.03	**5373.23**	0.29	3.00	**2.05**	0.84	2.00	6.00	0.87
	Lower + Upper	FairFlow	**587.69**	4339.60	***0.94***	3.00	1.65	***0.48***	3.00	6.00	***0.63***
CVPR2018	Upper	TPMS	*256.73*	*112552.11*	1.37	**29.24**	*22.23*	5.52	0.00	9.00	2.97
	Upper	PR4A (i1)	8683.79	108714.98	***12.68***	29.13	21.48	**3.86**	0.00	9.00	2.97
	Upper	FairIr	3785.64	**112263.94**	7.19	**29.24**	**22.18**	4.75	0.00	9.00	2.96
	Upper	FairFlow	**910.08**	91029.66	11.12	29.19	17.98	4.49	0.00	9.00	**2.91**
	Lower + Upper	TPMS	*636.01*	*108634.18*	0.00	**29.24**	*21.46*	6.28	2.00	9.00	1.66
	Lower + Upper	FairIr	4666.27	**108083.00**	7.17	**29.24**	**21.35**	5.06	2.00	9.00	1.67
	Lower + Upper	FairFlow	**1790.71**	86166.07	***10.52***	22.79	17.02	**2.77**	2.00	9.00	**1.61**

Table 1: MIDL, CVPR and CVPR2018 matching statistics. PS denotes a paper score and RA denotes the number of papers assigned to a reviewer. Values in *italic* represent the best performer; bold values indicate the best of the fair matching algorithms. FairFlow is always the fastest of the fair algorithms while FairIr always achieves the highest objective score. FairFlow is always competitive with or outperforms PR4A in terms of the fairness with respect to the most disadvantaged paper.

objective, unlike the other algorithms, which more directly optimize fairness. FairIr's balance between fairness and global optimality is illustrated by FairIr's profile (Figure 4f), which contains a handful outliers with low scores, but many papers with comparatively high paper score in quintiles 3, 4 and 5.

5.3 CVPR2018

In our final experiment, we use data from CVPR 2018 (CVPR2018). The data contains the affinities of 2840 reviewers for 5062 papers–a substantial increase in problem size over CVPR. Affinities range between 0.0 and 11.1, with many scores closer to 0.0 (the mean score is 0.36). Each paper must be reviewed 3 times. Reviewer load upper bounds vary by reviewer and range between 2.0 and 9.0. Again, the data does not include load lower bounds and so we construct synthetic lower bounds of 2.0 for all reviewers. Because of the size of the problem, the binary search for a suitable value of T did not terminate within 5 hours. Therefore, we select T by summing the minimum paper score found by FairFlow and $\frac{1}{2}A_{max}$. The reported run time includes the run time of FairFlow.

Table 1 (third vertical block) reveals similar trends with respect to speed (FairFlow is most efficient) and fairness (PR4A and FairIr are the most fair). Figure 5, included in the appendix because of space considerations, displays the corresponding matching profiles.

6 RELATED WORK

Our work is most similar to previous studies that develop algorithms for constructing fair assignments for the RAP. Two studies propose to optimize for fairness with respect to the least satisfied reviewer, which can be formulated as a maximization over the minimum paper score with respect to an assignment [7, 26]. The first algorithm, to which we compare, is PR4A [26]. PR4A iteratively solves maximum-flow through a sequence of specially constructed networks, like our FairFlow, and is guaranteed to return a solution that is within a bounded multiplicative constant of the optimal solution with respect to their maximin objective. As demonstrated in experiments, FairFlow is faster than PR4A and achieves similar quality solutions on data from real conferences. We note that the work introducing PR4A also presents a statistical study of the acceptance of the *best* papers among a batch submitted; we do not focus on paper acceptance in this work.

The second work proposes a rounding algorithm and prove an additive, constant factor approximation of the optimal assignment, like we do [7]. We note that both their algorithm and proof techniques are different from ours. However, their algorithm requires solving a new linear program for each reviewer during each iteration, which is unlikely to scale to large problems. Moreover, PR4A directly compares favorably to this algorithm [26].

With respect to fairness, the creators of TPMS perform experiments that enforce load equity among reviewers (i.e., each reviewer should be assigned a similar number of papers) via adding penalty terms to the objective [3]. These researcher, and others, explore formulations that maximize the minimum affinity among all assigned reviewers, which is different from our fairness constraint [22, 30]. Others have posed instances of the RAP that require at least one

reviewer assigned to each paper to have an affinity greater than T. In this setting, one classic piece gives an algorithm for constructing assignments that maximizes T by modeling the RAP as a transshipment problem [10]. Other objectives have been considered for the RAP, but these tend to be global optimizations with no local constraints that can lead to certain papers being assigned groups of inappropriate reviewers [8, 17, 30].

Some previous work on the RAP models each paper as a binary set of topics and each reviewer as a binary set of expertises (the overall sets of topics and expertises are the same). In this setting the goal to maximize coverage of each paper's topics by the assigned reviewers' expertises [13, 19, 20]. A generalized settings allows paper and reviewer representations to be real-valued vectors rather than binary [15, 27]. The resulting optimization problems are solved via ILPs, constraint based optimization or greedy algorithms. While representing papers and reviewers as topic vectors allows for more fine-grained characterization of affinity, in practice, reviewer-paper affinity is typically represented by a single real-value–like the real-conference data we use in experiments.

A significant portion of the work related to the RAP explores methods for modeling reviewer-paper affinities. Some of the earliest work employs latent semantic indexing with respect to the abstracts of submitted and previously published papers [5]. More recent work models each author as a mixture of personas and each persona as a mixture of topics; each paper written by an author is generated from a combination of personas [21]. Other approaches use reviewer bids to derive the affinity between papers and reviewers. Since reviewers normally do not bid on all papers, collaborative filtering has been used for bid imputation [4]. Finally, some approaches model affinity using proximity in coauthorship networks, citations counts, and the venues in which a paper is published [16, 18, 24].

7 CONCLUSION

This work introduces the local fairness formulation of the reviewer assignment problem (RAP) that includes a global objective as well as local fairness constraints. Since it is NP-Hard, we present two algorithms for solving this formulation. The first algorithm, FairIr, relaxes the formulation and employs a specific rounding technique to construct a valid matching. Theoretically, we show that FairIr violates fairness constraints by no more than the maximum reviewer-paper affinity, and may only violate load constraints by 1. The second algorithm, FairFlow, is a more efficient heuristic that operates by solving a sequence of min-cost flow problems. We compare our two algorithms to standard matching techniques that do not consider fairness, and a state-of-the-art algorithm that directly optimizes for fairness. On 3 datasets from recent conferences, we show that FairIr is best at jointly optimizing the global matching while statisfying fairness constraints, and FairFlow is the most efficient of the fairness matching algorithms. Despite a lack of theoretical guarantees, FairFlow constructs highly fair matchings.

8 ACKNOWLEDGMENTS

This material is based upon work supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, and in part by the Chan Zuckerberg Initiative under the project "Scientific Knowledge Base Construction." B. Saha was supported in part by an NSF CAREER award (no. 1652303), in part by an NSF CRII award (no. 1464310), in part by an Alfred P. Sloan Fellowship, and in part by a Google Faculty Award. Opinions, findings and conclusions/recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] C. F. Camerer and E. J. Johnson. 1997. 10 The process-performance paradox in expert judgment: How can experts know so much and predict so badly? *Research on judgment and decision making: Currents, connections, and controversies* (1997).
[2] L. Charlin and R. S. Zemel. 2013. The Toronto paper matching system: an automated paper-reviewer assignment system. In *ICML*.
[3] Laurent Charlin, Richard S Zemel, and Craig Boutilier. 2012. A framework for optimizing paper matching. *arXiv preprint arXiv:1202.3706* (2012).
[4] D. Conry, Y. Koren, and N. Ramakrishnan. 2009. Recommender systems for the conference paper assignment problem. In *conference on Recommender systems*.
[5] S. T. Dumais and J. Nielsen. 1992. Automating the assignment of submitted manuscripts to reviewers. In *Research and development in information retrieval*.
[6] M. Gairing, B. Monien, and A. Woclaw. 2007. A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. *Theoretical Computer Science* (2007).
[7] N. Garg, T. Kavitha, A. Kumar, K. Mehlhorn, and J. Mestre. 2010. Assigning papers to referees. *Algorithmica* (2010).
[8] J. Goldsmith and R. H. Sloan. 2007. The AI conference paper assignment problem. In *AAAI Workshop on Preference Handling for Artificial Intelligence, Vancouver*.
[9] Inc. Gurobi Optimization. 2015. Gurobi Optimizer Reference Manual. http://www.gurobi.com
[10] D. Hartvigsen, J. C. Wei, and R. Czuchlewski. 1999. The conference paper-reviewer assignment problem. *Decision Sciences* (1999).
[11] Eric J Johnson. 1988. Expertise and decision under uncertainty: Performance and process. *The nature of expertise* (1988).
[12] P. E. Johnson, F. Hassebrock, A. S. Durán, and J. H. Moller. 1982. Multimethod study of clinical judgment. *Organizational behavior and human performance*.
[13] M. Karimzadehgan and C. Zhai. 2009. Constrained multi-aspect expertise matching for committee review assignment. In *Conference on Information and knowledge management*. ACM, 1697–1700.
[14] N. M. Kou, U. L. Hou, N. Mamoulis, and Z. Gong. 2015. Weighted coverage based reviewer assignment. In *International Conference on Management of Data*.
[15] N. M. Kou, N. Mamoulis, Y. Li, Y. Li, Z. Gong, et al. 2015. A topic-based reviewer assignment system. *Proceedings of the VLDB Endowment* (2015).
[16] X. Li and T. Watanabe. 2013. Automatic paper-to-reviewer assignment, based on the matching degree of the reviewers. *Procedia Computer Science* (2013).
[17] J. W. Lian, N. Mattei, R. Noble, and T. Walsh. 2018. The conference paper assignment problem: Using order weighted averages to assign indivisible goods. In *AAAI Conference on Artificial Intelligence*.
[18] X. Liu, T. Suel, and N. Memon. 2014. A robust model for paper reviewer assignment. In *Conference on Recommender systems*.
[19] C. Long, R. C.-W. Wong, Y. Peng, and L. Ye. 2013. On Good and Fair Paper-Reviewer Assignment. In *International Conference on Data Mining*.
[20] J. J. Merelo-Guervós and P. Castillo-Valdivieso. 2004. Conference paper assignment using a combined greedy/evolutionary algorithm. In *International Conference on Parallel Problem Solving from Nature*.
[21] D. Mimno and A. McCallum. 2007. Expertise modeling for matching papers with reviewers. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 500–509.
[22] R. O'Dell, M. Wattenhofer, and R. Wattenhofer. 2005. The paper assignment problem. *Technical report/Swiss Federal Institute of Technology Zurich, Department of Computer Science* (2005).
[23] Eric Price. 2014. The NIPS Experiment. http://blog.mrtz.org/2014/12/15/the-nips-experiment.html. Guest Post.
[24] M. A. Rodriguez and J. Bollen. 2008. An algorithm to determine peer-reviewers. In *Conference on Information and knowledge management*. ACM, 319–328.
[25] P. M. Rothwell and C. N. Martyn. 2000. Reproducibility of peer review in clinical neuroscience. *Brain* 123, 9 (2000), 1964–1969.
[26] I. Stelmakh, N. B. Shah, and A. Singh. 2018. PeerReview4All: Fair and Accurate Reviewer Assignment in Peer Review. *arXiv:1806.06237* (2018).
[27] W. Tang, J. Tang, and C. Tan. 2010. Expertise matching via constraint-based optimization. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, Vol. 1. IEEE, 34–41.
[28] C. J. Taylor. 2008. *On the Optimal Assignment of Conference Papers to Reviewers*. Technical Report.
[29] V. V. Vazirani. 2013. *Approximation algorithms*. Springer Science & Business Media.
[30] F. Wang, B. Chen, and Z. Miao. 2008. A survey on reviewer assignment problem. In *New frontiers in applied artificial intelligence*. Springer, 718–727.

A REPRODUCIBILITY

All code for experiments is available here: https://github.com/iesl/fair-matching. Anonymized data is either included in the repository or available upon request from the first author.

B CVPR2018 PROFILES

We include the profiles of the CVPR2018 matchings for completeness in Figure 5.

C FAIRIR GUARANTEES

We restate and then prove Theorem 3.1.

Theorem. *Given a feasible instance of the local fairness formulation $\mathcal{P} =< R, P, L, U, C, A, T >$, FairIr returns an integer solution in which each local fairness constraint may be violated by at most A_{max}, each load constraint may be violated by at most 1 and the global objective is maximized.*

The local fairness formulation, $\mathcal{P}$, is comprised of a set of reviewers, R, a set of papers, P, reviewer load lower and upper bounds, L and U, respectively, coverage constraints, C, a paper-reviewer affinity matrix, A, and a local fairness threshold, T. To prove this theorem we rely on three lemmas. The first guarantees that FairIr does not violate a load constraint by more than 1; the second guarantees that FairIr will never violate a local fairness constraint by more than A_{max}; the third guarantees that FairIr will always terminate if the input problem is feasible.

Lemma C.1. *Given a feasible instance of the local fairness formulation, FairIr never violates a load constraint by more than 1.*

Proof. FairIr only drops load constraints if a reviewer is assigned fractionally to at most 2 papers. Clearly, if a reviewer is assigned to exactly one paper, the load constraint can be violated by at most one. Therefore, let r_i be a reviewer, assigned fractionally to p_j and p_k only. Then,

$$L_i \leq x_{ij} + x_{ik} + \alpha \leq U_i.$$

where α is the total load on r_i excluding x_{ij} and x_{ik}. Since r_i is only fractionally assigned to 2 papers, α must be integer; since $x_{ij}, x_{ik} \in (0, 1)$, $x_{ij} + x_{ik} < 2$. Thus,

$$L_i - 1 \leq \alpha \leq U_i - 1.$$

If the load constraints are dropped and r_i is neither assigned to p_j nor p_k, then r_i will retain a load of α, which is at least as large as 1 less than L_i. On the other hand, if r_i is assigned to both p_j and p_k, then r_i will exhibit a load of $\alpha + 2 \leq U_i + 1$. □

Lemma C.2. *Given a feasible instance of the local fairness formulation, FairIr never violates a local fairness constraint by more than A_{max}.*

Proof. FairIr only drops a paper's local fairness constraint if that paper has at most 3 reviewers fractionally assigned to it. Clearly, if a paper has only one reviewer fractionally assigned to it, the local fairness constraint can be violated by at most A_{max}. Assume during an iteration of FairIr a paper has exactly 2 reviewers fractionally assigned to it. Call that paper p_k and those reviewers r_i and r_j. During each iteration of FairIr, a feasible solution to the relaxed local fairness formulation is computed. Therefore,

$$C' + x_{ik} + x_{jk} = C_k,$$

where C' is load the on p_k aside from the load contributed by reviewers r_i and r_j. Recall that $x_{ik}, x_{jk} \in (0, 1)$ and r_i and r_j are the only reviewers fractionally assigned to p_k. Therefore $x_{ik} + x_{jk} = 1$. Moreover,

$$x_{ik}A_{ik} + x_{jk}A_{jk} \leq x_{ik}A_{max} + x_{jk}A_{max} = A_{max}.$$

Now, consider the paper score at p_k, and let T' be the total affinity between p_k and all its assigned reviewers, except for r_i and r_j. Then,

$$\begin{aligned} T' + x_{ik}A_{ik} + x_{jk}A_{jk} &\geq T \\ T' &\geq T - x_{ik}A_{ik} - x_{jk}A_{jk} \\ &\geq T - A_{max}. \end{aligned}$$

Since either r_i or r_j must be assigned integrally to p_k (lest the coverage constraint be violated), dropping the local fairness constraint at p_k can only lead to a violation of the local fairness constraint at p_k by at most A_{max}.

Next, consider the case that p_k has 3 reviewers fractionally assigned to it, r_h, r_i and r_j. Since the coverage constraint at p_k must be met with equality, one of the two cases below must be true:

$$x_{hk} + x_{ik} + x_{jk} = 1$$

or

$$x_{hk} + x_{ik} + x_{jk} = 2.$$

As before, let T' be the paper score at p_k, excluding affinity contributed from fractionally assigned reviewers. If the first case above is true, then $x_{hk}A_{hk} + x_{ik}A_{ik} + x_{jk}A_{jk} \leq A_{max}$. Furthermore,

$$\begin{aligned} T' + x_{hk}A_{hk} + x_{ik}A_{ik} + x_{jk}A_{jk} &\geq T \\ T' &\geq T - x_{hk}A_{hk} - x_{ik}A_{ik} - x_{jk}A_{jk} \\ &\geq T - A_{max}. \end{aligned}$$

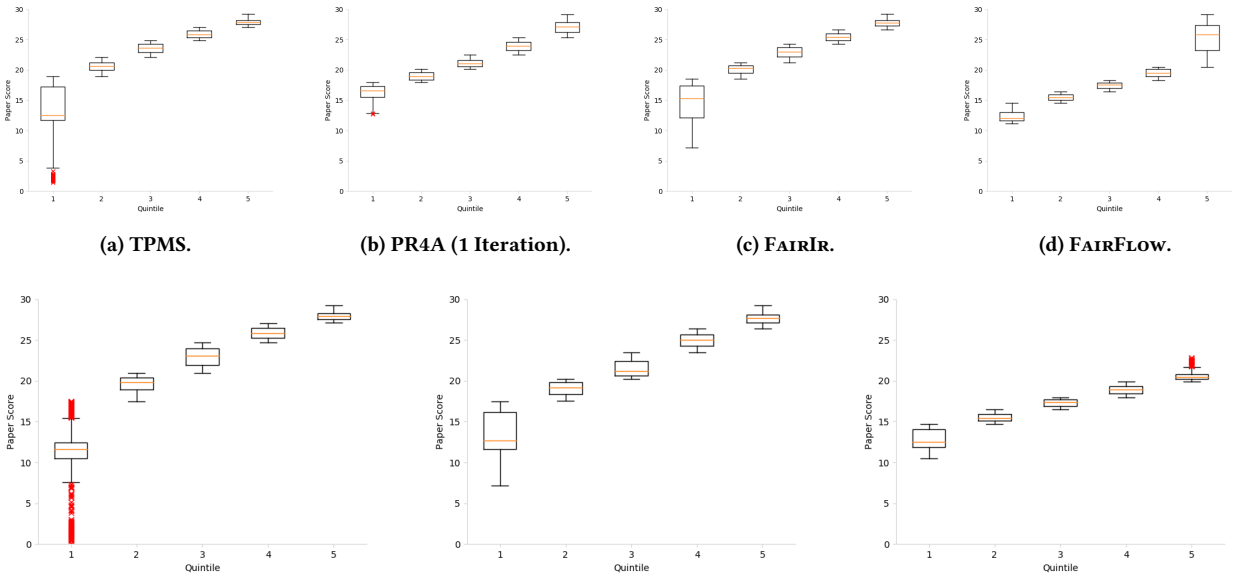
This means that even if all three reviewers were unassigned from p_k (which would make satisfying the coverage constraint at p_k impossible), the local fairness constraint would only be violated by at most A_{max}. Now, consider case 2 above, where $x_{hk}A_{hk} + x_{ik}A_{ik} + x_{jk}A_{jk} \leq 2A_{max}$. In order to satisfy the coverage constraint at p_k, at least two of the three reviewers must be assigned integrally to p_k. Without loss of generality, assume that

$$A_{hk} = \max[A_{hk}, A_{ik}, A_{jk}] \leq A_{max}.$$

Even if r_h is unassigned from p_k, the change in paper score at p_k is at most A_{max} and the local fairness can be violated at most by A_{max}. The same is also true if either r_i or r_j is unassigned from p_k. □

Lemma C.3. *Given a feasible instance of the local fairness fromulation, FairIr always terminates.*

The goal in proving Lemma C.3 is to show that during each iteration of FairIr, either: a constraint is dropped or an integral solution is found. Before proving Lemma C.3 recall that the solution, $x^\star$, of a linear program is always a *basic feasible solution*, i.e., it has n linearly independent tight constraints. Formally,



(a) TPMS. (b) PR4A (1 Iteration). (c) FairIr. (d) FairFlow.

(e) TPMS with lower bounds. (f) FairIr with lower bounds. (g) FairFlow with lower bounds.

Figure 5: Matching Profiles for CVPR2018.

Corollary C.4. *If $x^\star$ is a basic feasible solution of linear program $\mathcal{P}$, then the number of non-zero variables in $x^\star$ cannot be greater than the number of linearly independent active constraints in $\mathcal{P}$.*

Proof. According to Algorithm1, FairIr drops constraints during any iteration in which it constructs a solution exhibiting at least one paper with at most 3 reviewers fractionally assigned to it or at least one reviewer assigned fractionally to at most 2 papers. If FairIr is able to drop a constraint or round a new variable to integral, it makes progress. Therefore, FairIr could only fail to make progress if each reviewer was assigned fractionally to at least 3 papers and each paper was assigned fractionally to at least 4 reviewers. In the following, we show that this is impossible, using a particular invocation of Corollary C.4.

Assume for now that each reviewer is fractionally assigned to exactly 3 papers and each paper is assigned fractionally to exactly 4 reviewers. Therefore, the total number of fractional assignments can be written as follows:

$$\frac{1}{2}[3|R| + 4|P|].$$

An instance of the local fairness paper matching problem contains an upper and lower bound constraint for each reviewer, 1 coverage constraint for each paper, and 1 local fairness constraint for each paper yielding $2|R| + 2|P|$ total constraints. Note that for a reviewer r, only one of its load constraints (i.e., upper or lower) may be tight–assuming that the upper and lower bounds are distinct. Thus, an upper bound on the number of *active* constraints is $|R| + 2|P|$. However, this means that the number of fractional variables is larger than the number of constraints:

$$\frac{1}{2}[3|R| + 4|P|] = \frac{3}{2}|R| + 2|P| > |R| + 2|P|$$

which violates Corollary C.4. When reviewers may be fractionally assigned to at least 3 papers and each paper is assigned fractionally to at least 4 reviewers, the number of nonzero fractional variables could only be larger. Note that, when there is no local fairness constraint FairIr returns an integral solution since the underlying constraint matrix becomes totally unimodular. □

Now to end the proof of the theorem, we note that the global objective value never decreases in subsequent rounds, as we always relax the formulation by dropping constraints and fix those integrality constraints for which $x_{i,j}$s have been returned as integer. Thus, FairIr maximizes the global objective.

D PROOF OF FACT 2

Proof. By definition, papers that are members of P^+ have paper score greater than T. Therefore, unassigning a reviewer from a paper in P^+ may reduce the corresponding paper score by at most A_{max} yielding a paper score of at least $T - A_{max}$, which makes the paper either a member of P^0 or P^+. Now, consider the papers in P^0. By step 7 above, a reviewer r can only be unassigned from a paper $p \in P^0$ if the flow entering p from p' is large enough to make p's resulting paper score at least as large as $T - A_{max}$. Thus, the papers in P^0 either remain in P^0 or become members of P^+, which completes the proof. □