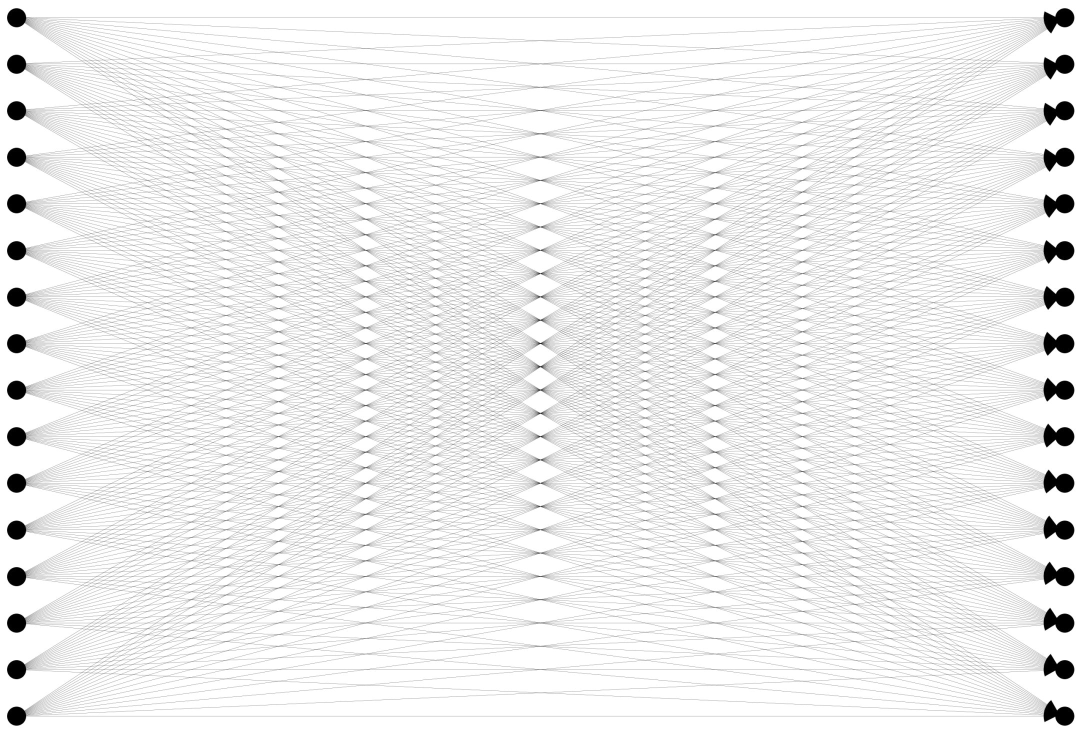
Automated matching and fairness for peer review



Bram Bakker

Layout: typeset by the author using LaTeX.
Cover illustration: A bipartite graph made using NetworkX

Automated matching and fairness for peer review

juggling objectives

Bram Bakker
11721995

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dhr. dr. R. de Haan

Institute for Logic, Language and Computation
Faculty of Science
University of Amsterdam
Science Park 107
1098 XG Amsterdam

Jan, 2021

Abstract

Matching reviewers to the right papers for large scale academic conferences can be tricky, especially when trying to fairly divide reviewer expertise among papers. In this thesis different variations of automated matching of peer review are evaluated. Data from various AI conferences is encoded into linear programming and network flow problems. The effect of optional constraints, like ensuring diversity of academic backgrounds and the use of a minimum score are also examined. Not only different ways to match the data are compared, but also ways of transforming the data itself. Using clustering, the affinity from reviewer to paper is predicted, and with order weighted averages the affinities are changed to give a reviewers ordinal ranking more weight. All these different techniques are then compared under different definitions of fair, like democratic fairness, rank-maximality and the Rawlsian theory of justice.

Contents

Chapter 1

Introduction

One of the most important parts of scientific research is the process of peer review. Having other experts determine the professional performance or quality of research, is a way to make sure the right papers are being published in academic journals, and the right researchers receive academic praise from institutions[Lee, 2013]. It also plays a vital part in rating papers during academic conferences. Peer review at academic conferences will be the focus of this paper.
During the organisation of conferences, arranging the best possible peer reviews can be a challenge. With hundreds of papers and hundreds of reviewers, the question becomes: Who gets to review which papers? The first big concern here is bias. Ideally a perfect reviewer is impartial and will make judgments based on the norms and values pertaining to the arguments made by the paper and nothing else. But beyond the potential conflicts of interests which surely would need to be avoided, the remaining reviewers couldn't possibly all review every paper according to their best ability. Since their expertise is focused on a relatively small area, there is a large amount of papers for which they are probably not fully qualified as assessor. Moreover, because reviews cost time and effort, a reviewer can only manage to review a limited amount of papers without a massive drop in quality. This means that in academic conferences, reviewers would need to be matched to a limited amount of papers each in a way to maximize their expertise.

In the 1940's, mathematician George B. Dantzig came across a similar issue when he was put in charge of the mechanization of logistical programs of the U.S. Air Force[Dantzig, 2002]. With the advent of the computer, planning for large scale logistics finally had the potential to be automated. Previously, planners had to organize logistics through rule of thumb and common sense, since

logistical challenges contain way too many variables to ever be optimized using the human mind. Computing power offered a way to possibly maximize total utility gained by job assignment. Dantzig thought to formulate planning problems into a minimization function of a linear form subject to linear equations and inequalities. The day he wanted to present his paper called *Programming in a Linear Structure*(the Air Force referred to their logistical operations as *programs*), he took a stroll with Dutch-American economist T. C. Koopmans along the Santa Monica beach, Koopmans made the suggestion to shorten the title to *linear programming*. This concept of linear programming has been massively influential, and Dantzig's way to solve the minimization, the simplex algorithm, is still what is being built on today. Two different adjusted versions of the simplex algorithm will be used to solve the paper assignment problem in this thesis.
However, our problem isn't as simple as minimizing or maximizing some functions through algorithms. Peer review is a delicate process, one could for example argue that reviews only matter in the sense that they are fair. Is it fair if (even though overall, the amount of experts matched with papers is as high as can be), one paper gets assigned 3 disinterested reviewers whose expertise lies outside of the area of the paper? And the next paper gets 3 enthusiastic reviewers? The way to divide the reviewers among papers would therefore maybe need to have a fairness check, so no paper is extremely disenfranchised, and no single reviewer has to review 10 papers completely out of their field of interest, thereby giving sub-par reviews.
Another goal which should be considered is diversity. By ensuring that a group of reviewers of a paper have different academic backgrounds or specialties, they could notice new interesting things, or perhaps identify mistaken views on a topic the researcher (and the potential reviewer most suited to review the research-topic) wasn't knowledgeable about. This can be translated to having a low inter-rater reliability, or a high amount of disagreeability between reviewers. It could even be normatively desirable[Lee, 2013] to have at least one reviewers who is *not* suited for the job.
Another potential problem in this selection process is that of strategy[Stelmakh, 2020]. If a particular reviewer wants to help a friend by giving a great review or alternatively wants to hurt a competitor and torpedo-review, how can the peer review process prevent that from happening? Does the matching need to be randomized to a certain extent? But how do we ensure fair or varied reviews then?
So far, only the matching itself has been discussed, but what hasn't been discussed is how this 'expertise' which is being divided, is measured. In most academic conferences where peer review is utilized, reviewers simply have to bid on certain papers, to identify which papers they are excited/comfortable with

to review. Machine learning techniques can also be used to infer these preferences[Charlin and Boutilier., 2012], and could be combined with the knowledge of the bids to create a reviewer/paper affinity score. Having a sophisticated system of determining affinity between paper and reviewer carries the potential to increase the fairness/diversity objectives previously mentioned. Making the predictions sophisticated enough, it could bypass the process of bidding completely and massively save time and effort.
All these different objectives: fair, varied or strategyproof matchings, plus learning which reviewers should fit with which papers, must be considered by program chairs of academic conferences. These get a huge list of papers and also a list of reviewers to be assigned papers for peer review. Currently a solution is offered by popular software for organizing conferences, like Easychair. EasyChair can automatically give a good (enough) matching being given a list of reviewers and their stated preferences.
In this thesis, the fundamental questions of the real-world issues concerning peer reviews will be asked. How do we make this system fair? Will this fair solution be computationally feasible? What do we consider fair? What is the cost of having a diverse set of reviewers for each paper? Will our solution be strategy-proof? In order to achieve answers and explore potential tradeoffs between for example fair or quick solutions, multiple algorithms will be pitted against each other on the matching side, and multiple affinity allocations will be tested under these algorithms.

Chapter 2

Related literature

This thesis covers quite a wide array of topics, from the philosophy of fairness, to linear programming, to the general two-sided matching problem formulation, and game-theoretic concepts as strategy-proofness. Therefore, for each of these topics, some extra literature is provided if the reader 's interest is sparked and they would like to dive into a topic in more detail.

2.1 linear programming

As described in the introduction, the modern concept of linear programming was conceived in 1947 when George Dantzig was tasked with optimizing and mechanizing logistical programs of the U.S. Air Force[Dantzig, 2002]. Although mathematicians like Fourier earlier had published ways of solving linear systems[Vanderbei, 2015], it wasn't until the advent of the computer that optimizations like that were thought possible. In order to solve these linear inequalities Dantzig created the simplex method[Dantzig and Wolfe, 1955]. This 1965 monograph[Dantzig, 1965] on the simplex method remains important work to this day. Optimizing systems and finding extremities to solve is hugely influential on economics and game theory. Eventually the simplex method was expanded upon, like in integer programming[Chen and Dang., 2010], and network flows[Orlin, 1997]. Both of these will be used here.

2.2 Predicting preferences

In the conference paper assignment problem, the best way to save time and administrative effort is to automate the bidding process, in which reviewers have to signal which papers they would prefer to review. This is done via expertise modelling[Mimno and McCallum, 2007], detecting what reviewers fit with which papers via reading keywords from the paper, or using TF-IDF vectors for reviewer profiles. This bag-of-words approach looks at an author's work, creating 'persona's'[Rosen-Zvi, 2012] from the types of words they use. Another way to predict preferences, has been attempted using bayesian networks[Charlin and Boutilier., 2012]. These could be used to automate the bidding process, or alternatively, supplement it. Generally research in this area isn't preoccupied with fair or equally divided matchings. The affinities themselves are being modified.

2.3 Fair divisions

What does it mean to make fair assignments of indivisible goods? Because the goods are indivisible, a perfectly fair assignment typically isn't possible, so fairness needs to be approximated. One of the ways to accomplish this is using the Rawlsian theory of justice[Rawls, 2009], put into rules via the leximin principle[Tuucooom, 2000]. The leximin principle is simple, it presumes that the situation most fair is one where the utility is maximized for the agent who is the least well off. In our concept, it would mean the matching where the worst off reviewer couldn't be more satisfied with his received papers without disadvantaging someone else further down.
One could also see a fair situation as one where the majority of the people are 'envy-free'[Erel Segal-Halevi, 2019]. Envy-freeness normally means an agent is at least as well off as anyone else, and it isn't something that is normally feasible using indivisible goods. One can cut a cake in the exact slices that no one sees another person get a bigger cake, but giving each reviewer exactly their preferred paper and having no other reviewer receive more of their choices seems clearly out of reach. If however, most agents would be as well off, and have the same utility compared to most other agents, one could say that it is a democratically fair division.
Another definition is looking at how many people receive their top ranked preference in the division of goods, here a priority is given to the ranks of the things agents prefer. To optimize this, and then optimize the highest amount of second choices, then third etc. is also known as a rank-maximal assignment[Irving, 2006].

2.4 strategy-proof

The game-theoretic concept of strategy-proofness means an agent can't maximize his/her utility by lying. This has been studied extensively for social choice problems[Brandt, 2016] and peer review specifically[Stelmakh, 2020][Haris Aziz, 2019]. This is because lying about one's preferences could be disastrous for the process of peer review where one is supposed to be impartial, and the competitive nature of academia can encourage the opposite behaviour. Langford (2008) for example calls academia inherently adversarial:

It explains why your paper was rejected based on poor logic. The reviewer wasn't concerned with research quality, but rather with rejecting a competitor.
(Langford 2008, found in[Xu, 2018])

Most of the strategyproof algorithms identified for peer review matchings thus use some sort of randomization to make sure that torpedo-ing a competitor is hard to do. Theoretically, achieving both a strategyproof and a fair matching is hard, since the specific torpedo-strategy is encouraged in most fair matchings, where picky reviewers get advantaged to make sure they get the papers they prefer.

2.5 general two-sided matching problem

The general two-sided matching problem occurs everywhere in economics and nature, where individual agents in two sets have to match up with each other according to their preferences. In 1962, Gale and Shapley published their seminal paper on this topic[Gale and Shapley., 1962], identifying an algorithm to find stable matchings for two sided markets. This breakthrough awarded them half of the Nobel Prize in 2012.
The research area of two-sided matching has grown enormously since 1962, as more algorithms were identified, its wide array of applications grew, attracting for example economists like Roth[Roth, 1989], who won the other half of Shapley's Nobel Prize, scientists and mathematicians. A good basis can be found in chapter 14 of the handbook of computational social choice[Brandt, 2016].

Chapter 3

Problem formulation

3.1 data

Two sided matching problems can either be bipartite or non-bipartite. Bipartite being two sets of agents where one or both of them have preferences over the other, non-bipartite being a set of agents where each individual agent has preferences over the set. The bipartite case can be divided further into one sided or two sided. In our case, the conference paper assignment problem is one sided bipartite, since only the reviewers have preferences over the papers. This one sided bipartite problem will from here on be defined as follows: There's a set of reviewers (R), which need to be assigned to a set of papers (P). In order to determine how many reviewers are assigned to how many papers the Program Chair can decide that assignments need to follow a few constraints. First of which is a load constraint (L), the amount of papers maximally/minimally assigned to each reviewer. The load constraint is there so there's not one reviewer who would have to, for example, review 15 papers, and no reviewer who has to review 0. Conversely there is a Coverage (C) constraint, with a minimum and maximum for the papers. So for both C and L there's a maximum and minimum.

$$C_{min} \leq C \leq C_{max}$$
$$L_{min} \leq L \leq L_{max}$$

These max and min bounds could be tweaked according to specific needs or opinions regarding review quality versus workload. The effectiveness of a larger amount of reviewers per paper can be questioned[Squazzoni, 2015], so coverage could be a matter of opinion, and perhaps the minimum and maximum load would need to be close together to make sure the reviewers aren't massively overburdened compared to each other. However, if the load and coverage aren't in balance with the proportion of papers to reviewers, a correct matching won't be possible.

The specific dataset being used contains the bids of reviewers over papers from the 2015 and 2016 Autonomous Agents and Multiagent Systems Conference (MD-04-01, MD-04-02), and also three other unknown AI conference datasets (MD-02-01,MD-02-02,MD-02-03). These are publically available, from preflib(`https://www.preflib.org/data/index.php#md`). The 2015 AAMAS data contains 201 reviewers with bids over 613 papers; The 2016 AAMAS data contains 161 reviewers with bids over 442 papers. The data of the three unknown AI conferences is smaller and as follows: MD-02-01 has 31 reviewers and 54 papers, MD-02-02 24 reviewers and 52 papers, and MD-02-03 146 reviewers and 156 papers. Each individual batch of data is split into three parts, the reviewers, the bids, and the papers. The bids are scored either 'yes', 'no', or 'maybe'. If no bids are made, then it indicates a conflict of interest, and under no circumstances can that paper be matched to that reviewer. In all cases the amount of papers to be reviewed is larger than the amount of reviewers to be assigned to them. The most balanced division is 146 R / 176 P, while the most imbalanced is 201 R / 613 P. This is why, in the following experiments, the load and coverage constraints are chosen in such a way that a reviewer would always need to review more papers than a paper gets assigned reviewers.

3.2 Clusters and diversity

Because the data contains only three scores, the options to maximize or fairly divide these preferences for each reviewer can be quite a challenge. If two reviewers have bidded 'maybe' on a paper, it could be that one of them would be fine to review the paper, just not enthusiastic. Alternatively, they could just barely bid 'maybe' instead of 'no'. Since we have a lot of 'maybe's', it could be helpful to differentiate between a 'maybe yes' and a 'maybe no'. One place of inspiration to achieve this was a paper which examined the role of popularity in two-sided matching[Chakraborty and Ostrovsky., 2010].
In general two sided matching problems where preferences need to be satisfied, people often change their preference to what is most popular. An applicant to a company is more likely to be hired if the company knows that the applicant is popular, and if scientific journals know a researcher has had their paper rejected, they will be less likely to publish that paper. This isn't one to one applicable to the paper assignment problem, since each researcher has their own expertise. However, if we can identify sub-groups of researchers who are quite similar, we can change the scores based on the popularity of a paper in a re-

viewers group. Here some inspiration was taken from the APT test for expertise modelling[Rosen-Zvi, 2012], which identifies multiple 'persona's' paper authors could be placed in. It does so via Natural Language Processing, but similar classes could be made just based on the bids we already have. If all the reviewers in a group have roughly the same preferences, and they say 'yes' to a paper, it seems logical that the 'maybe' bid from another person in that group , leans more to 'maybe yes' than 'maybe no'.
In order to accomplish this, the preferences of each reviewer were placed as a vector into a clustering algorithm. Based on these clusters each of the scores of the papers was changed to reflect the relative popularity of that paper in reviewers from the same class. Now that these classes were identified, they could be used for different objectives. If we remember from the introduction, having low inter-rater reliability could ensure better quality reviews. In order to ensure diverse matching, inspiration was taken from a paper about the housing allocation in Singapore[Benabbou, 2018]. If the connection isn't immediately clear it will be explained now.
Singapore is an island state off the southern tip of the Malay Peninsula, and while being tiny and relatively wealthy, it houses an ethnically diverse population. In order to ensure that the different ethnicities in Singapore get along, and that Malaysians aren't living in a Malaysian enclave, while the Chinese live in a Chinese enclave, there's an ethnic diversity quota on every housing area. The mechanism which needs to be encoded here is one very similar to our peer review mechanism. There's a set of people with preferences over neighborhoods, and there's a set of neighborhoods which can only house so many people but needs to be diverse while maximizing social welfare (the preferences of the inhabitants). Similarly, our reviewers have preferences over papers, these need maximization, and since we have discovered that diversity could also be a desirable trait, that could be an extra constraint just as the linear program used in this paper [Benabbou, 2018].

3.3 Fairness

The notion of fairness is a complicated one, and essential in the division of indivisible goods. Matching papers to reviewers according to their affinities gets so complicated that often, it's impossible to divide expertise equally. So since a perfectly fair world isn't realistically possible in this particular problem, what would approximate this? Three evaluations will be considered, one based on democratic fairness, one based on ordinal preference, or rank-maximal choice, and one based on the leximin principle. First a simple egalitarian solution will be tried, assigning an extra constraint that each reviewer needs to get at least a minimum

affinity score, this would mean that no reviewer would be so disenfranchised that they would have to review completely out of their expertise, approximating the leximin principle. The rank-maximal version of fair assignment is approximated by an order weighted average.[Lian, 2018]
The idea behind a maximum order weighted average is this: The first ranked choice is the most important one for many people. A reviewer would probably prefer to review one 'yes' than three 'maybe's'. The Max-OWA technique uses a vector with which each preference score is multiplied according to their place in the score order. This also makes sure that the person who hasn't ranked their choices as high as the others, (never said 'yes', but did say 'maybe' a few times), doesn't suffer as much as in the linear programming maximization.
The third, more sophisticated way to improve fairness was tried using an algorithm with network flows, Fairflow[Kobren and McCallum., 2019]. The high level idea of the Fairflow algorithm, is to un-assign reviewers from the lower scoring papers, and flow their reviewers back to middle scoring papers who share affinity with them, while reassigning new reviewers to lower scoring papers. This is done continuously until convergence is reached.

Chapter 4

Testing

4.1 Linear programming

All of the following solutions were encoded using Python, all code being available here: https://github.com/BramBakker/Thesis_peer_review_fairness.git. To get the basic assignment, linear programming was implemented. This particular linear program definition is taken from Taylor[Taylor, 2008]. An affinity matrix is defined ($A_{i,j}$), of which the rows i are the amount of R's in the dataset, and the columns j, the amount of P's. Thus the bid of the i_{th} reviewer on the j_{th} paper is represented on $A_{i,j}$ with an affinity score, these numbers will be 1 for 'no', 2 for 'maybe', and 3 for 'yes'.
Another matrix is defined which is binary ($B_{i,j}$), containing variables that can be either 1 or 0 with the same dimensions as the affinity matrix. The optimization function is the total sum of A*B per row, per column:

$$\sum_i \sum_j A_{i,j} B_{i,j}$$

. This has to be maximized, while each row of B has to abide by the L and C constraints:

$$L_{min} \leq \sum_j B_{i,j} \leq L_{max} \forall i$$

$$C_{min} \leq \sum_i B_{i,j} \leq C_{max} \forall j$$

The effect of this, is that the resulting B matrix from the maximum optimization function reflects the maximum utilitarian assignment from all papers to all reviewers according to load and coverage constraints, and wherever there's a 1 on the B matrix, the i_{th} reviewer is matched to the j_{th} paper.

Programming was done with Python using the MIP (mixed integer programming) package(https://pypi.org/project/mip/). This package uses the COIN or branch-and-cut algorithm by default. The algorithm relaxes the constraint that B has to be 0 or 1, solves the linear system according to the simplex algorithm originally defined by Dantzig, and then for all the fractional, non integer numbers in B, changes them and tries to bring them closer to integers. It then relaxes again, trying to continuously find a balance between maximizing the optimization function and keeping all numbers in B 0 or 1 until a solution is found.

4.2 optional constraints: diverse and minimum

To achieve a more fair linear programming solution, an optional minimum score for each reviewer was implemented. The trouble was that for each different dataset, a different minimum score was feasible. MD-02-02 contains relatively way more 'yes' bids than MD-04-01. In order to get the best minimum value for each dataset, starting from 0, the constraint kept being increased until the linear program that it resulted in became infeasible. It is to be expected that there won't be much improvement from this constraint, since it is just the utilitarian program, but with the following constraint added:

$$\sum_j A_{i,j} B_{i,j} \geq Minimum \forall i$$

However, diversity of reviewers is also something to consider. So an extra optional constraint was added to the linear program. The constraint makes it so the total sum per class has to be under a certain diversity maximum (D). So using a D of 1, and a C_{max} of 3, each trio of reviewers has to come from different classes, each sum of reviewers of the same class is allowed to be smaller or equal to 1. To achieve this, all different classes of reviewers were identified via the K-means algorithm from Sklearn, using the Kneed (https://pypi.org/project/kneed/) package to find the right number of classes per dataset. These classes were put in a dictionary as keys with the places of their class in the affinity matrix as values. The added specific constraint was that for each R, the maximum sum on the B matrix of the values in each key of that dictionary could maximally be D.

4.3 Max-OWA and clustering

The order weighted averages method was implemented as an adjustment to the affinity matrix A. As explained earlier, the top ranked preferences are given extra

weight in an OWA assignment, then the second, then the third etc. Theoretically, that results in something approaching a rank maximal division. For each vector per reviewer R in the matrix, the top value was identified, and multiplied by the first number in the OWA vector, and then the second highest value was multiplied by the second number in the OWA vector etc. So using an OWA vector of (1,0.5,0.25), all 'yes' bids are given 2 times more weight than the 'maybe' bids, and all the 'maybe' bids 2 times more than the 'no' bids. This from here on will be called OWA-adjusted. Our affinity matrix of A only contains 3 values, so it probably won't have a large effect when all people share their top choice. This is where our persona based cluster affinity scores can come in.
To adjust the affinity matrix to the k-means identified classes, the score of all reviewers in that class was increased by the class-share of yes's times 0.75, and for maybe's by 0.25. This ensures that the scores never go up by one full point, thereby changing a no bid to a maybe bid. For example, if all the reviewers in the same class have the same 'yes' bids, their 3's will be increased to 3.75. This will be called cluster-adjusted, and the combined cluster and then OWA adjusted affinities will be referred to as COWA-adjusted. The new COWA-adjusted matrix then looks something like this.

$$\begin{bmatrix} 3.75 & 1.625 & 0.4375 & ... & 0.75 & 0.25 & 0.25 \\ 0. & 0.25 & 0.25 & ... & 0.28125 & 0.28125 & 0.28125 \\ 0. & 0.25 & 0.5 & ... & 0.28125 & 0.28125 & 0.78125 \\ .. & .. & ... & ... & ... & ... & ... \end{bmatrix}$$

4.4 fairflow

The fairflow algorithm functions according to three steps. In the first step a graph is made which starts at a source node, with edges to all reviewers, from all reviewers to all papers, and from all papers to a sink node. Each edge between paper and reviewer has a certain cost with it, which is the minus affinity score of that reviewer and paper, and has a capacity of 1. The capacity between paper and sink is C, and between source and reviewer is L. The flow network is then optimized to have the minimum cost with the maximum flow. So reviewers review as many papers with the minimum ultimate cost between layer 2 and 3 of the network. This max flow min cost algorithm is provided by the NetworkX(https://networkx.org for more documentation) package for python, it is a variation of the simplex algorithm from Dantzig but adjusted for networks[Orlin, 1997]. So the first step in the fairflow algorithm is very much like the linear programming solution, but translated to a graph.
In step two of the algorithm, one reviewer of each of the disadvantaged papers

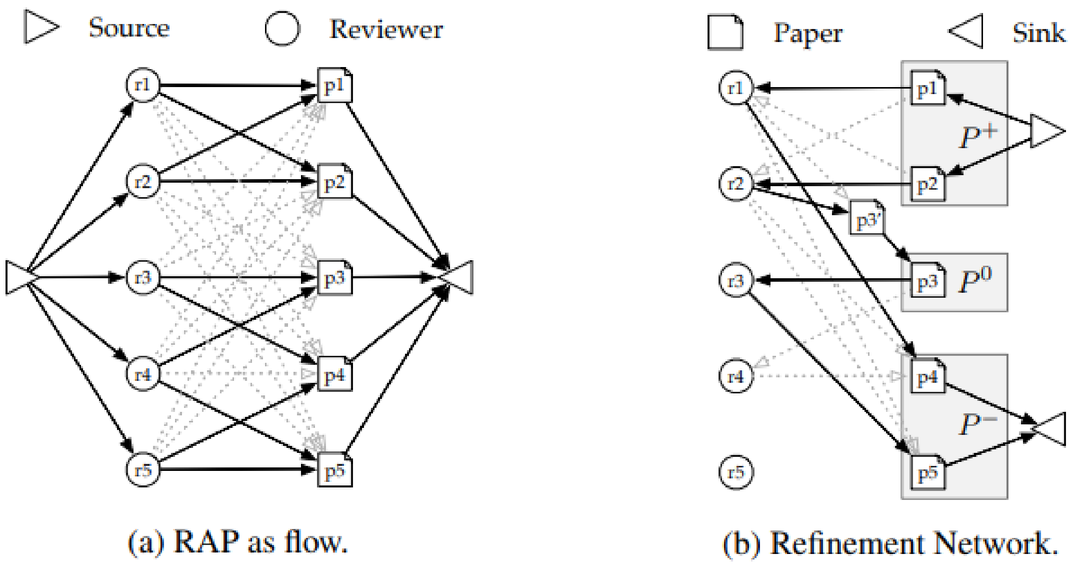


Figure 4.1: Visual representation of step 1 and 3 from[Kobren and McCallum., 2019]

is removed. To determine what paper is advantaged/disadvantaged the papers are split into three parts: P^+, P^0, P^-. These are divided using a suitable value of variable T, of which all P^+ have a larger value than. A paper is assigned to P^0 if its combined score is smaller than T but larger than $T - A_{max}$, meaning T minus the maximum affinity, which for the normal A is 3. So P^0 contains papers only one 'yes' away from being in P^+. Logically, P^- contains papers with a total score lower than both.
After the first flow network is calculated, and the one reviewer per P^- is removed, a second flow network is made, following 9 steps. In these 9 steps a new network is generated and then optimized according to a max flow min cost algorithm. This new network has its 'source' construct an edge to the highest scoring papers, then from those papers to their reviewers, to potential dummy nodes from the middle scoring papers in P^0, from those papers to their previous reviewers. And lastly from all reviewers not matched to P^- to papers in P^-. This means that ideally, some high-scoring papers will send their reviewers to the middle if they fit there, and some reviewers from all scores will go to the most disadvantaged papers. These last 2 steps, of removing a reviewer from the disadvantaged papers, and constructing the new flow network is done multiple iterations. This was achieved in a custom python code, not using the python code based on Gurobi provided by the source paper[Kobren and McCallum., 2019], but networkX.

Chapter 5

Results

To determine the effects of different affinity matrices or optimization methods on fairness, these methods and matrices were implemented as described in the testing chapter. Four different affinity matrices were used, firstly the normal one, A, with scores of 1 for no, 2 for yes and 3 for maybe, second the OWA-adjusted matrix with a vector of (1,0.5,0.25). Third was the cluster adjusted affinity matrix, and fourth the COWA-adjusted matrix. Then the 2 different optional constraints were tested on the normal matrix A, a minimum constraint and a diverse constraint using a D of 1. Lastly the fairflow network flow algorithm was tested using the affinity matrix A. For the first 3 datasets, a maximum coverage of 3 was chosen and maximum load of 5, and the last 2, a maximum coverage of 2 and load of 6 and 7, the minimum coverage in both cases being 2, and minimum load 1.
To determine something approaching rank-maximality, the amount of reviewers who got their top ranks was measured, and the minimum score was measured to approximate the leximin-principle. The standard deviation could be seen as an approximator for democratic fairness, since, if most reviewers get roughly the same score, most of them won't be dissatisfied with the matching and write sub-par reviews (or if they would, at least most papers would get the same low quality of reviews). The total affinity scores of each reviewer were measured too. Though it should be noted, that since our cluster-adjusted matrix only adds scores, it will always end up with a higher amount of total points, that's why these total points are in italics in the score tables. Time was also measured using timeit, not included in the time score, was finding out which minimum value was feasible, and which T score was reasonable, these were both processes of trial and error.
In order to understand the tables better, visual boxplots are provided. Next to each boxplot and above each table, a small box of text is used to help identify/clarify some results.

The smallest dataset MD-02-02 contains relatively many 'yes' scores, so most reviewers score around 15, they review 5 papers which they all said 'yes' to. The loss for the diversity constraint isn't that big here, and all 24 reviewers get at least one of their top ranked choices. The third step of the fairflow algorithm doesn't respect reviewer load rules, so one reviewer ends up with 7 papers and two with 6.

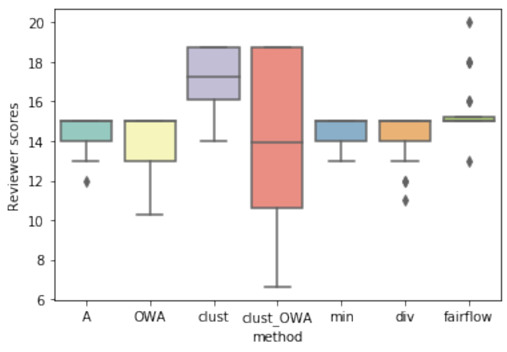


Figure 5.1: MD-02-02

method	time (sec)	total affinity score	mean	std	num top ranked	lowest score
A	0.75	349.0	14.54	0.82	24	12.0
A_{owa}	0.52	340.75	14.2	1.37	24	10.25
$A_{clusters}$	0.6	*409.62*	*17.07*	1.5	24	*14.0*
$A_{clusterowa}$	0.56	332.75	13.86	4.14	24	6.625
A, min	0.57	348.0	14.54	**0.76**	24	**13.0**
A div	1.13	343.0	14.29	1.14	24	11.0
A, fairflow	**0.44**	**374.0**	**15.58**	1.44	24	**13.0**

Table 5.1: MD-02-02

The second smallest dataset, MD-02-01, with 31 reviewers is a bit different, the effect of the COWA adjusted scores compared to just cluster-adjusted is apparent, increasing top ranks from 23 to 26, the smallest standard deviation is achieved here by the fairflow algorithm, but it does worse in minimum score and total utility.

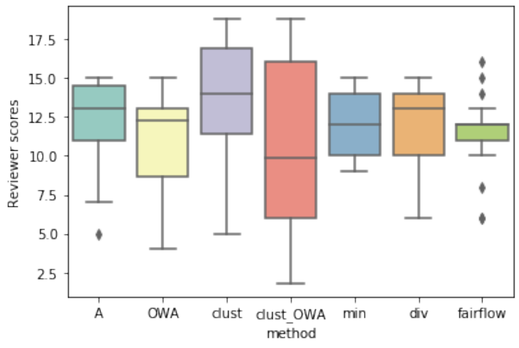


Figure 5.2: MD-02-01

method	time (sec)	total affinity score	mean	std	num top ranked	lowest score
A	0.68	**384.0**	**12.39**	2.52	30	5.0
A_{owa}	0.64	345.0	11.13	3.16	**31**	4.0
$A_{clusters}$	0.91	*432.12*	*13.94*	3.67	23	5.0
$A_{clusterowa}$	0.78	322.29	10.4	5.57	26	1.79
A, min	0.72	379.0	12.23	2.07	**31**	**9.0**
A,div	1.34	370.0	11.94	2.51	**31**	6.0
A, fairflow	**0.6**	356.0	11.48	**2.06**	**31**	6.0

Table 5.2: MD-02-01

The MD-02-03 dataset is more balanced with 146 reviewers and 176 papers. Again, the fairflow algorithm scores best on standard deviation, but on minimum score and total number of top ranked choices the minimum constraint scores the best. And again the OWA adjustment only has a large effect if the matrix was cluster adjusted.

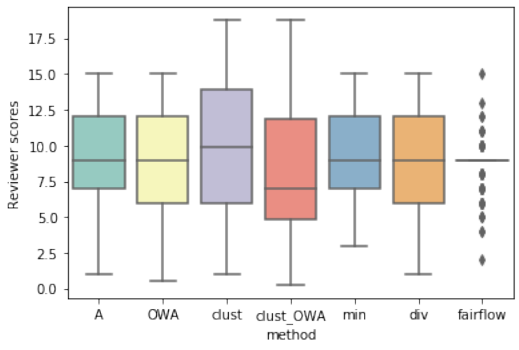


Figure 5.3: MD-02-03

method	time (sec)	total affinity score	mean	std	num top ranked	lowest score
A	8.15	**1406.0**	**9.36**	3.57	138	1.0
A_{owa}	**7.37**	1322.0	9.05	4.0	**140**	0.5
$A_{clusters}$	7.77	*1441.79*	9.88	4.4	122	1.0
$A_{clusterowa}$	7.63	1174.38	8.04	4.22	130	0.25
A, min	7.85	**1406.0**	**9.63**	3.4	**140**	**3.0**
A, div	10.91	1281.0	8.77	4.2	128	1.0
A, fairflow	12.78	1265.0	8.66	**1.62**	134	2.0

Table 5.3: MD-02-03

For the larger imbalanced MD-04-02 dataset, the fairflow algorithm doesn't work as well as in the previous smaller datasets, though again, it can be seen outperforming the minimum constraint with a relatively small standard deviation. The load constraint needed to be raised to 11 to return a feasible result under the diverse constraint.

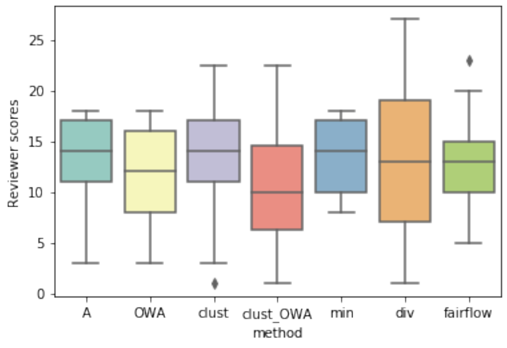


Figure 5.4: MD-04-02

method	time (sec)	total affinity score	mean	std	num top ranked	lowest score
A	31.05	**2153.0**	**13.37**	3.97	157	1.0
A_{owa}	33.98	1885.25	11.71	4.6	156	3.0
$A_{clusters}$	32.69	*2191.36*	*13.61*	4.18	145	4.0
$A_{clusterowa}$	**31.03**	1674.81	10.4	5.16	151	1.0
A, min	32.58	2147.0	13.34	3.62	**159**	**8.0**
A, div	36.27	2080.0	12.92	7.4	145	1.0
A, fairflow	43.9	2028.0	12.59	**3.25**	158	5.0

Table 5.4: MD-04-02

In the largest dataset MD-04-01, with 201 reviewers and 613 papers, the diverse constraint wouldn't work unless the load was increased to 15 for one reviewer. That is why the boxplot goes up so high in that case. Otherwise the same effects can be perceived as in the rest of the dataset, although interesting to note is that in the normal OWA adjustment, the amount of reviewers receiving their top ranked choice went *down*.

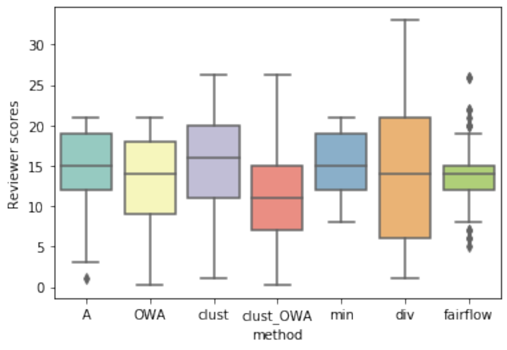


Figure 5.5: MD-04-01

method	time (sec)	total affinity score	mean	std	num top ranked	lowest score
A	**49.96**	**3043.0**	15.14	4.45	199	1.0
A_{owa}	52.53	2701.75	13.44	5.33	197	0.15
$A_{clusters}$	54.22	*3071.74*	*15.28*	4.85	185	1.0
$A_{clusterowa}$	53.6	2257.16	11.23	5.09	191	0.25
A, min	63.16	3042.0	15.13	4.2	**200**	**8.0**
A, div	53.51	2833.0	14.09	8.87	189	1.0
A, fairflow	82.64	2783.0	13.84	**3.12**	**200**	5.0

Table 5.5: MD-04-01

Consistently across datasets, a few observations can be made, adding a minimum constraint to the linear program gets the best lowest score, and also strangely best divides top choice among reviewers, at least in a homogeneous matrix with only 3 values. However, Fairflow has its scores closer together, with a smaller standard deviation.
Another consistent observation, is that the OWA-adjusted scores don't seem to influence the rank-maximality in the normal assignment. The amount of reviewers with their top choice even bizarrely goes down in some datasets. In the COWA-adjustment, it fares as it should. As thought before, when there are less ties in the dataset OWA performs better, and increases the amount of top choices divided among reviewers.
The fairflow algorithm takes longer and performs less consistently in the bigger datasets, plus it doesn't guarantee a minimum load, or a maximum load for reviewers. Timewise, the algorithms run very fast, with fairflow often being the slowest. Not included is the fairflow algorithm on the matrices other than A, because these calculations would take too long for the bigger datasets.

Chapter 6

Discussion and conclusion

6.1 findings

It seems from the results that, according to a democratic definition of fairness, the fairflow algorithm is the most fair. The minimum constraint often performed the best on approximating the leximin principle and the rank-maximal assignment. However, to find a good minimum score takes some trial and error, and a T variable also had to be manually selected. A drawback from the fairflow algorithm is the lack of control on reviewer load in the third phase of the algorithm. The reviewers get assigned continuously more papers until they can't help papers out of the P^- class. So the cost of fairness here could be an increase in the total workload of some reviewers.
Another way some reviewers were tasked with way too high of a workload, was in the larger datasets with unfortunate imbalanced, clustered groups. In the largest dataset (MD 04-02), one group contains half the reviewers, and another just a single reviewer. This makes it so that the solution is infeasible unless the load goes up to 15 for the 1 reviewer in that 1 class. This of course is less than desirable. So the idea of ensuring diversity based on k-means identified clusters of bids can deliver inconsistent results.
The fairflow algorithm was generally slower than the other linear program solutions used here. It should be noted that the fairflow algorithm is very fast compared to other algorithms which try to optimize fairness like P4A[Ivan Stelmakh and Singh, 2019], as tested in the source paper[Kobren and McCallum., 2019]. But in our case, the fairflow algorithm takes too long on the adjusted scores to be usable considering the larger datasets. The rest of the runtimes were in an acceptable range. Seeing as the number 1 reason of using the new adjusted affinity scores came from more fairly dividing them, it is quite disappointing that the fairflow algorithm we implemented takes so long to complete on them.

An expected cost of ensuring diversity was total utility, this was also perceived in the paper on the housing market in Singapore. But in our case the hit social welfare took when people were forced to choose outside their preferences, was smaller than expected. In the larger dataset with imbalanced groups however, it performed pretty badly.
Another thing to note is that, when trying to ensure rank-maximality, the OWA adjustment could be fine if the starting affinity matrix doesn't have a lot of ties. When it does, the OWA assignment isn't that much different from the normal assignment.

6.2 potential improvements

In the future, more creatively could be looked for extra linear programming constraints and uses. The fairflow algorithm could potentially be interpreted as a set of linear programming problems for example. And it could be changed to optimize for reviewers instead of papers. Also, there weren't any alterations or different algorithms tried for solving the linear program. Just the Coin Branch or cut algorithm was used, when there are many options and variations on the simplex algorithm. In the paper using fairflow[Kobren and McCallum., 2019], the researchers optimize their flow network using Gurobi, which maybe could have different results.
Another option to consider is more changes to the affinity system. Like for example, trying to make the 'no' score higher than the 'yes' scores, with 'yes' maybe being zero, and then minimizing the problem. This is perhaps more logical, especially from the reviewers perspective, since reviews cost work to complete, and maximizing to 7 'maybe's' probably isn't appealing.
Strategy-proofness is another avenue for improvement to the assessment of these algorithms, and was originally planned for this thesis. A way to test this could be an addition of artificial reviewers to the data who only have affinity for a single paper, and then it could be checked if those reviewers get the paper per algorithm. If a large amount of reviewers would receive their single preferred paper, the implementation wouldn't be strategy-proof, since that reviewer could act maliciously.
The relationship between the resulting fairness and load and coverage constraints could also be examined more closely. Even though some thought went into selecting our constraints, it was mostly just to deal with the specific dataset sizes and needs when the linear programs became infeasible. Moving beyond that, it would be interesting to view the direct effects of limiting the amount of reviewers and increasing the coverage, in for example, the more balanced dataset.
In both of the results of the fair matchings, there was one variable which needed

to be manually adjusted. It took some time to get a right T value for adjusting the groups in the network flow, and also a right B value, for the minimum affinity for the minimum constraint. Ideally, an automatic assignment would happen completely automatically. Perhaps there is a way to make a B value based on the reviewer-to-paper ratio combined with the ratio for the bids, and maybe a same T value could be computed that way.
Lastly it would also be interesting to measure reviewer satisfaction after matching, to see if the predicted affinities were somewhat correct, was our 'maybe yes' really a 'maybe yes'? Though that would require an interviewing process after an academic conference.

6.3 conclusion

Linear programming seems to be a quick and fair solution to the paper assignment problem. The fairflow algorithm better optimizes democratic fairness, but is less versatile in dealing with different constraints and matrices, whereas the linear programming options are easy to implement and generally quicker. Also, the load for some reviewers was very high in the fairflow algorithm. Losing versatility and speed seems to be the price of fair divisions, this is still true, but less so, for the minimum constraint. We also saw the weakness of the OWA adjustment when it came to increasing rank-maximality in homogeneous matrices. Strategy-proofness hasn't been tested yet, but intuitively it makes sense that these fair mechanisms, without randomization, are prone to abuse. Lastly a k-means clustering algorithm could be a way to increase diversity in reviewers and adjust affinity scores, but it delivers inconsistent results when the clusters are imbalanced.
This thesis won't offer a definitive 'best' solution for the Conference paper assignment problem. Some of the strengths and weaknesses of different methods were explored, and every assignment method has its own drawbacks. What matters in this case is the specific needs of an academic conference. Do we care about diversity of expertise among reviewers? Do we care about total utility? Do we care about equal, rank-maximal or leximin reviews? Do we instead want to avoid torpedo-reviews and randomize? These questions can't be fully answered by algorithms, box plots and standard deviations, but they could be informed by them.

Bibliography

[Benabbou, 2018] Benabbou, Nawal, e. a. (2018.). Diversity constraints in public housing allocation. *17th International Conference on Autonomous Agents and MultiAgent Systems*.

[Brandt, 2016] Brandt, Felix, e. a. (2016). *Handbook of computational social choice.* Cambridge University Press.

[Chakraborty and Ostrovsky., 2010] Chakraborty and Ostrovsky., M. (2010). Two-sided matching with interdependent values. *Journal of Economic Theory.*, 145.1:85–105.

[Charlin and Boutilier., 2012] Charlin, Laurent, Z. and Boutilier., C. (2012). A framework for optimizing paper matching. *arXiv preprint*, 1202.3706.

[Chen and Dang., 2010] Chen, Der-San, R. G. B. and Dang., Y. (2010). *Applied integer programming.* Hoboken, NJ.

[Dantzig and Wolfe, 1955] Dantzig, George B., A. O. and Wolfe, P. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5.2:183–195.

[Dantzig, 1965] Dantzig, G. B. (1965). Linear programming and extensions. *Princeton university press*, Vol. 48.

[Dantzig, 2002] Dantzig, G. B. (2002). Linear programming. *Operations research*, 50.1:42–47.

[Erel Segal-Halevi, 2019] Erel Segal-Halevi, W. S. (2019). Democratic fair allocation of indivisible goods. *Artificial Intelligence*, Volume 277.

[Gale and Shapley., 1962] Gale, D. and Shapley., L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69.1:9–15.

[Haris Aziz, 2019] Haris Aziz, O. L. (2019). Strategyproof peer selection using randomization, partitioning, and apportionment, artificial intelligence. *Games and Economic Behavior*, Volume 275:295–309.

[Irving, 2006] Irving, Robert W., e. a. (2006). Rank-maximal matchings. *ACM Transactions on Algorithms (TALG)*, 2.4:602–610.

[Ivan Stelmakh and Singh, 2019] Ivan Stelmakh, N. B. S. and Singh, A. (2019). Peerreview4all: Fair and accurate reviewer assignment in peer review. *Algorithmic Learning Theory.*, PMLR,.

[Kobren and McCallum., 2019] Kobren, Ari, B. S. and McCallum., A. (2019). Paper matching with local fairness constraints. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.*, pages pp. 1247–1257.

[Lee, 2013] Lee, Carole J., e. a. (2013). Bias in peer review. *Journal of the American Society for Information Science and Technology*, 64.1:2–17.

[Lian, 2018] Lian, Jing Wu, e. a. (2018.). The conference paper assignment problem: Using order weighted averages to assign indivisible goods. *Proceedings of the AAAI Conference on Artificial Intelligence.*, Vol. 32. No. 1.

[Mimno and McCallum, 2007] Mimno, D. and McCallum, A. (2007.). Expertise modeling for matching papers with reviewers. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining.*

[Orlin, 1997] Orlin, J. B. (1997). A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78.2:109–129.

[Rawls, 2009] Rawls, J. (2009). A theory of justice. *Harvard university press*.

[Rosen-Zvi, 2012] Rosen-Zvi, Michal, e. a. (2012). The author-topic model for authors and documents. *arXiv preprint:*, 1207.4169.

[Roth, 1989] Roth, A. E. (1989). Two-sided matching with incomplete information about others' preferences. *Games and Economic Behavior*, 1.2:191–209.

[Squazzoni, 2015] Squazzoni, F. B. F. (2015). Is three better than one? simulating the effect of reviewer selection and behavior on the quality and efficiency of peer review. *Winter simulation conference (WSC)*, IEEE.

[Stelmakh, 2020] Stelmakh, I. e. a. (2020). Catch me if i can: Detecting strategic behaviour in peer assessment.. *arXiv preprint*, 2010.04041.

[Taylor, 2008] Taylor, C. J. (2008). On the optimal assignment of conference papers to reviewers.

[Tuucooom, 2000] Tuucooom, B. E. R. T. I. I. (2000). Egalitarianism: is leximin the only option?. *Economics and Philosophy.*, 16:229–245.

[Vanderbei, 2015] Vanderbei, R. J. (2015). *Linear programming.*, volume Vol. 3. Springer, Heidelberg.

[Xu, 2018] Xu, Yichong, e. a. (2018). On strategyproof conference peer review. *arXiv preprint:*, 1806.06266.