Implementation of Rank Aggregation with Proportionate Fairness in Java

Mark Mori, Chris Pierre-Paul, Brian Poblete
{mcm20c | cp20fl | bap21k}@fsu.edu

Abstract

1. Introduction

1.1 Preliminary

1.1.1 Problem statement

Rank aggregation is a method used to combine multiple rank orders over a set of items. In recent years, it has become more necessary than ever to ensure that the proportionate representation of different groups, especially people groups, is found to be fair. Some current applications of this necessity include college admissions, search engine rankings, and various recommendation systems. As there is an abundance of groups in various real-world applications, our paper, “Rank Aggregation with Proportionate Fairness”, makes an attempt to tackle the Rank Aggregation problem by creating a fair ranking system that does not unfairly apply negating preferences by combining Individual p-fairness (IPF) and Rank aggregation subject to p-fairness (RAPF). Our paper was published by the Sigmod conference in 2022 with the following authors: Dong Wei, Md Moinul Islam, Baruch Schieber, Senjuti Basu Roy.

1.1.2 Definition of terms

A *protected attribute* is any attribute in a list of entries, like a table in a database, that cannot be used in decision making. The reason for this rule may vary but may be attributed to anti-discrimination laws. Many papers often use employees as an example of such a list with *gender* or *seniority level* being protected attributes that should not influence the decision-making process. It is important to note that the value of a protected attribute may be binary (*gender* may be *male* or *female*) or multi-valued (*seniority level* may be *Junior, Mid-career,* or *Senior)*.

Proportionate fairness (p-fairness) is a fairness metric that “ensures proportionate representation of every group based on a protected attribute in every position of the aggregated ranked order” [3]. Essentially, a ranking that satisfies p-fairness must (1) create minimal changes to the ordering of a list containing existing rankings and (2) equally represent the proportion of entries with protected attribute p in the top k entries.

1.2 Introduction of main algorithms (IPF, RAPF)

In their paper titled "Rank Aggregation with Proportionate Fairness," Wei et al. introduce the idea of “p-Fairness" to address this issue. P-Fairness allows for entries from different groups, based on a protected attribute, to be represented proportionally in the final ranked order. The terms Individual p-Fairness (IPF) and the rank aggregation problem subject to p-fairness (RAPF) further define and formalize this challenge. The authors provide a detailed exploration of these concepts and introduce solutions to achieve p-fairness in rank aggregation. As several other papers on the topic of rank aggregation and proportionate fairness have been written, we will

discuss in the Related Works section how the approach presented by the authors provides a novel solution to this problem.

2. Related Works

Existing Fair Ranking Algorithms

2.1 Explanation of the main algorithm

To evaluate the effectiveness of their algorithm, the authors performed a set of extensive experiments against two state-of-the-art solutions to compare their performance for IPF and RAPF. Respectively, these algorithms are DetConstSort [1] and FairILP [2]. In their paper, Wei et al. highlight the shortcomings of these existing algorithms and demonstrate the advantages of their approaches in the given experimental scenarios.

2.2 Deterministic Constrained Sorting (DetConstSort)

2.2.1 DetConstSort ranking criteria

Geyik et al. present the Deterministic Constrained Sorting (DetConstSort) algorithm in their paper "Fairness-aware ranking in Search & recommendation systems with application to LinkedIn Talent Search" (2019). DetConstSort aims to satisfy three main criteria: (1) the produced ranked list r_i should deviate as little as possible from existing ranks, (2) no specific value of a protected attribute receives a disproportionate advantage, and (3) the produced ranked list satisfies a minimum representation for any given protected attribute value in a range. This last criterion is the main rule that drives DetConstSort and is defined with the following equation

$$\forall k \leq |\tau_r| \;\&\; \forall_{a_i} \in A,\ count_k(a_i) \geq \lfloor p_{a_i} \cdot k \rfloor .$$

2.2.2 DetConstSort algorithm

DetConstSort works by going over each entry in the list and keeping track of the number of items it has scanned with a counter. Next, it checks the list of protected attribute values and pushes back an entry into its tentative output list for each attribute value. For each entry it inserts, it moves it towards the back of the list until either the ranking of the older entry is greater than the ranking of the new entry or criterion (3) is no longer met. This fairness-aware algorithm creates a ranking that considers the value of protected attributes during the rank aggregation process.

2.2.3 Comparison to IPF

One issue with DetConstSort is that the rankings that it creates are not necessarily the most optimal rankings despite satisfying the three main conditions. *Example 3.1* in [3] demonstrates that a ranking generated by DetConstSort does not always satisfy both conditions of p-fairness and therefore is often not p-fair. The IPF algorithm developed by Wei et al. consistently satisfies p-fairness, thus ensuring that protected attribute values are equally represented while at the same time guaranteeing that any generated rankings make minimal changes to existing rankings.

2.3 Fair Integer Linear Programming (FairILP)

2.3.1 FairILP definitions [2]

$x \prec y$: denotes x being in a more optimal position in the ranking than y.

$$Rpar_{G_1}(\rho) = \sum_{x_i \in G_1} \sum_{x_j \in G_2} \frac{\mathrm{I}(x_i \prec_\rho x_j)}{|G_1||G_2|}$$

Rpar gives the advantage of G_1 over G_2 in list ρ.

Pairwise Statistical Parity: Given a ranking $\rho = [x_1 \prec x_2 \prec \ldots \prec x_n]$ of candidates belonging to mutually exclusive groups G_1, G_2, ρ satisfies pairwise statistical parity if the following condition is met:

$$RparG_1(\rho) = RparG_2(\rho)$$

In simple terms, *pairwise statistical parity* is achieved when either group has the same advantage as denoted by the Rpar score.

2.3.2 FairILP algorithm

The other algorithm used for comparison, FairILP, is presented by Kuhlman et al. in their paper "Rank Aggregation Algorithms for Fair Consensus" (2020). FairILP requires a binary protected attribute (e.g., *Male* or *Female*) and measures fairness using *pairwise statistical parity*. FairILP also adheres to three main constraints that each enforce strict ordering, transitivity, and parity. Adhering to these constraints creates a ranking that is Kemeny-optimal while also minimizing the Kendall-Tau distance and achieves pairwise statistical parity [2].

2.3.3 Comparison to RAPF

Wei et al. argue that FairILP is not p-fair. They demonstrate this by showing that FairILP may produce a ranking list that satisfies pairwise statistical parity while being identical to a ranking that is produced without any consideration to fairness at all. This is because pairwise statistical parity only checks pairs of items with protected attribute values that differ from each other without considering the final positions of the items in the aggregate ranking [3].

On the other hand, RAPF produces a ranking that is p-fair while also allowing for the processing of multi-valued protected attributes. Focusing on the AlgRAPF, this implementation of RAPF works by calling a subroutine AlgIPF to solve the IPF problem for each input ranking. From the fair rankings, it selects one that has the smallest Kemeny distance to the input rankings. This final selection is the aggregated p-fair ranking.

2.4 Multiple Attribute and Intersectional Group Fairness for Consensus Ranking

Though "Rank Aggregation with Proportionate Fairness" was published in June, a strongly related paper published just a month earlier went unnoticed by its authors. This paper, "MANI-Rank: Multiple Attribute and Intersectional Group Fairness for Consensus Ranking" [4], studies a problem which can be described as a more advanced version of the classical rank aggregation subject to the proportionate fairness problem tackled here. In their version, the problem allows for candidates to belong to multiple protected groups. It claims to be the first such paper to do so, citing that the vast majority of papers researching this topic focus entirely on allowing just one protected attribute with a binary value (only two categories). The ideas proposed in "Rank Aggregation with Proportional Fairness," of course, only allow for one protected attribute, but are one of the exceptional papers which allows that attribute to be non-binary.

Since the problem addressed in "Mani-Rank" is a more complicated version of this paper's, many of the algorithms proposed in it are applicable, but not directly translatable. For example, one of the authors' main concerns is the concept of intersectional group fairness, which is the idea of enforcing relative fairness in the subgroups of different protected attributes. As an

example, say that for a list of candidates there exist two binary protected attributes, gender (male or female) and age (older than 50 or younger than 50) If intersectional group fairness were not a concern, then the goal would be to ensure candidate rankings have a proportional amount of male vs female, and older vs younger employees. However, if intersectional group fairness were a concern, the demand to balance between the sub groups means that a valid ranking must have a proportional number of older male vs older female vs younger male vs younger female employees.

The experiments shown in Mani-Rank demonstrate that their algorithm does not scale nearly as well for the number of items, or candidates, being ranked. The competing algorithms demonstrated in Mani-Rank scale quadratically, with a highest plotted figure of more than 3000 seconds of runtime for just 100,000 candidates. They do, however, seem to scale competitively in terms of the amount of rankings per candidate. Common types of datasets used in real world problems may feature many more candidates than this test case though, meaning that non-trivial optimizations are essential in further solving this problem with reasonable computation time. "MANI-Rank: Multiple Attribute and Intersectional Group Fairness for Consensus Ranking" is relevant enough that the ideas it presents are worth considering, but due to the more complex nature of their problem, will not form the basis for comparative analysis.

3. Algorithms

3.1 Preliminary

Our Java implementations of the presented algorithms are intended to be strictly one-to-one implementations of the original Python source code, or description of algorithms from the authors. Given that some features and libraries available in Python are not present in Java, some portions of the code had to be rewritten to work in Java. However, these rewrites are limited to factors such as data structures and function calls (e.g., Python tuples, Python lists and the `itertools` library), and approximation techniques described in the paper. Modifications or improvements to the actual algorithms are outside of the scope of this project.

Additionally, our Java implementations are only for the new algorithms presented in the paper (GrBinaryIPF, ApproxMultivaluedIPF, AlgRAPF, and RandAlgRAPF). The existing algorithms (DetConstSort and FairILP) that the authors evaluate their algorithms against were originally going to be part of our implementation. However, we encountered issues that we were unable to overcome, such as necessary libraries needing to be paid for.

3.2 GrBinary IPF

GrBinary IPF, or Greedy Binary IPF, is as its name implies, a greedy solution to the individual proportional fairness problem in cases where the protected attribute exists in a binary state. As they prove, a greedy algorithm is capable of guaranteeing both p-fairness and minimal kendall-tau distance from the objective ranking with this algorithm. It starts by sorting the list of candidates into groups based on their protected attribute. Then, starting from the top of the ranking, it selects the best fit candidate which maintains the p-fairness of the final ordering, and places them next in the ranking. As the author proves, this solution gives the best possible P-fair answer in O(n) time. However, it is exclusively capable of doing so for cases where the protected attributes are binary.

3.3 ApproxMultiValuedIPF

In cases where the protected attribute is non-binary, a different algorithm is needed. ApproxMultiValuedIPF provides an approximate solution with reasonable runtime while

preserving p-fairness. The intuition of this algorithm is that through the definition of P-fairness, we can prove the final location of a candidate in a p-fair ranking will fall within some range. We produce a weighted bipartite graph where the existence of an edge represents the capability of a candidate in the position of a starting node ending up in the position of an ending node. The weight of the edge is based on the distance at which the final position deviates from the starting position. Through the traversal of this graph, we can produce an optimal solution. As the authors note, a minimum weight perfect matching, would accurately solve the problem, but would dominate the computational complexity for it, hence an approximation is used.

Our implementation represents the bipartite graph through a specialized adjacency matrix where the row of the matrix represents the starting position of the candidate and the column represents the final positions. A weight in the matrix, as described, implies that it is possible for such a change in position to occur. The candidates are then organized based on how many options they have for a final p-fair position, and assigned the final position based on their lowest potential weight impact, completing our approximation.

3.4 AlgRAPF

AlgRAPF is an implementation for solving the Rank Aggregate Proportional Fairness Problem. It operates by first generating a p-fair ranking for each of an independent set of rankings through use of an IPF solving subroutine. It then finds which of the generated P-fair rankings is closest to the objective truth. To determine this distance, the sum of all kendall-tau distances between a p-fair adjusted ranking and the set of objective rankings is aggregated together and compared.

In our solution, the IPF subroutine of choice is GrBinaryIPF, but this decision could easily be swapped for ApproxMultiValuedIPF if the data tested on were of a non-binary protected attribute. The final average distance from the individual rankings of the closest p-fair solution discovered is reported with a time complexity of $O(m^2n \log n)$

3.5 RandAlgRAPF

This algorithm is a modified version of AlgRAPF which relies on the assumption that each member of our set of rankings is likely already quite similar, meaning that their aggregation can be approximated. Because of this assumption, the distance between the p-fair adjustment for each individual ranking would also be assumed to be quite low. The benefit of this is that instead of computing a p-fair score for every ranking in the set, and having to find the total distance between it and all objective rankings, we can instead compute the p-fair adjustment of just one ranking chosen at random, and report it as the aggregate.

This drastically reduces the time complexity of the algorithm to only the cost of the IPF subroutine implemented, meaning that it's implementation using GrBinaryIPF has complexity of just O(n). The utility of this assumption is entirely dependent on the data set in question. As the authors point out, if the rankings are totally different, and not comparable. Then this approximation will likely produce untrue results. In reality however, most real world objective rankings would be quite similar.

4. Evaluation

To evaluate the proposed algorithms listed above, we analyzed each algorithm, emphasizing the quality and scalability of the solutions for IPF and RAPF while comparing these to the state-of-the-art alternatives to stress fairness in rank aggregation. Our work involves utilizing the Linprog server, which has 56 CPUs, 2.9 terabytes of storage, and 251 GB of RAM. The server runs CentOS Linux 7 (Core) and employs Intel(R) Xeon(R) CPUs. We use Java dependencies provided by Linprog for implementing Rank Aggregation with Proportionate Fairness. Our implementation is based on Java 17.0.2 LTS. We conduct experiments using two real-world datasets: Fantasy football (NFL players' rankings) and MovieLens (movie ratings).

Throughout the duration of the evaluation, we mainly measured Kenall-Tau distance, Kemeny distance, proportion of p-fairness, and approximation factors. Overall, we observed that p-fairness impacts Kendall-Tau and Kemeny distance by improving fairness. Additionally, the previous state of the solutions does not satisfy fairness criteria compared to the algorithms introduced. While the paper's solutions manifest scalability to significantly larger datasets.

5. Conclusion

All in all, there are numerous real-world applications that must be considered in which ensuring the proportionate representation of different groups can be applied. Specifically, our paper initiated the Rank Aggregation with Proportionate Fairness (RAPF) which briefs over the p-fairness per binary and multi-valued protected attributes. Further, our approach involves addressing the IPF for binary-protected attributes through a greedy technique, handling multi-valued protected attributes using both ExactMultiValuedIPF and ApproxMultiValuedIPF and ultimately implementing RAPF to generate a group ranking that prioritizes proportionate representation for each value of the protected attribute. In summary, contributions are made that rethink existing state-of-the-art solutions. As a formalized problem, optimization of the IPF subroutine beyond the current state of the Art is both necessary and possible. Furthermore, the expansion of IPF into the RAPF problem has very little research performed for a problem very much in need of solving. It is critical and necessary to extend these to all aspects of decision-making processes to retrieve more "fair" results.

6. Distribution of Work

Mark Mori	• ApproxMultiValuedIPF algorithm • Documentation
Chris Pierre-Paul	• DetConstSort algorithm • GRBinaryIPF algorithm • Documentation
Brian Poblete	• AlgRAPF algorithm • RandAlgRAPF algorithm • Documentation

Throughout the project, we all kept each other accountable and would all try to chip in equivalently to ensure the splitting of the workload. For example, when implementing the algorithms, Chris took on the "DetConstSort", and "GrBinaryIPF" algorithms to make up for Brian and Mark covering me in other areas. Specifically, we split the documentation throughout each stage evenly except in Stage 4.

7. Artifacts

[1] GitHub Repository - https://github.com/Brian-Pob/rankAggregation