Ensuring Fairness in Group Recommendations by Rank-Sensitive Balancing of Relevance

Mesut Kaya
m.Kaya@tudelft.nl
TU Delft
Delft, The Netherlands

Derek Bridge
derek.bridge@insight-centre.org
Insight Centre for Data Analytics,
University College Cork
Cork, Ireland

Nava Tintarev
n.tintarev@tudelft.nl
TU Delft
Delft, The Netherlands

ABSTRACT

For group recommendations, one objective is to recommend an ordered set of items, a top-N, to a group such that each individual recommendation is relevant for everyone. A common way to do this is to select items on which the group can agree, using so-called 'aggregation strategies'. One weakness of these aggregation strategies is that they select items independently of each other. They therefore cannot guarantee properties such as fairness, that apply to the set of recommendations as a whole.

In this paper, we give a definition of fairness that 'balances' the relevance of the recommended items across the group members in a rank-sensitive way. Informally, an ordered set of recommended items is considered fair to a group if the relevance of the items in the top-N is balanced across the group members for each prefix of the top-N. In other words, the first item in the top-N should, as far as possible, balance the interests of all group members; the first two items taken together must do the same; also the first three; and so on up to N. In this paper, we formalize this notion of rank-sensitive balance and provide a greedy algorithm (**GFAR**) for finding a top-N set of group recommendations that satisfies our definition.

We compare the performance of GFAR to five approaches from the literature on two datasets, one from each of the movie and music domains. We evaluate performance for 42 different configurations (two datasets, seven different group sizes, three different group types) and for ten evaluation metrics. We find that GFAR performs significantly better than all other algorithms around 43% of the time; in only 10% of cases are there algorithms that are significantly better than GFAR. Furthermore, GFAR performs particularly well in the most difficult cases, where groups are large and interests within the group diverge. We attribute GFAR's success both to its rank-sensitivity and its way of balancing relevance. Current methods do not define fairness in a rank-sensitive way (although some achieve a degree of rank-sensitivity through the use of greedy algorithms) and none define balance in the way that we do.

CCS CONCEPTS

• **Information systems → Recommender systems**.

RecSys '20, September 22–26, 2020, Virtual Event, Brazil

ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412232

KEYWORDS

group recommendations, fairness

ACM Reference Format:
Mesut Kaya, Derek Bridge, and Nava Tintarev. 2020. Ensuring Fairness in Group Recommendations by Rank-Sensitive Balancing of Relevance. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383313.3412232

1 INTRODUCTION

There are many scenarios where we want to recommend items to a *group* of people, rather than to an individual. For example, we may want to suggest a movie for a group of friends to watch together; or, during a car trip, we may want to play songs jointly to people who are sharing a ride.

Jameson & Smyth present three main approaches to group recommendations [5]. The first is to compute recommendations for each group member and then to merge the recommended items. In their second approach, the preferences (e.g. predicted ratings or rankings) of each group member for each candidate item are *aggregated*, often using strategies inspired by Social Choice Theory [8], to obtain the preferences of the group. The third approach is to build a group recommender model directly from the (un-aggregated) preferences of the members of the group.

By way of illustration, consider the second approach, which is the one most commonly reported in the literature. Let's assume that individual preferences are represented as ratings. The recommender predicts each group member's rating for each candidate item and then obtains the preferences of the group for each candidate item by aggregating these predicted ratings. For instance, in the *Average* (AVG) strategy, the predicted group rating for an item is the mean of the predicted ratings of that item by the group members, whereas in the strategy called *Least Misery* (LM) the predicted group rating is the minimum of the individual predicted ratings. For both AVG and LM, the N items with the highest aggregated predicted ratings are selected as recommendations to the group. In this paper, we will denote this ordered set of items as top-N_G, using the G to emphasize that these are recommendations to the group.

Our focus is on the *fairness* of top-N_G ordered sets of recommendations for groups. In this paper, fairness is a property of the top-N_G and not of any single recommendation within the top-N_G. It might be claimed, for example, that AVG pursues 'fairness' because it considers all group members' interests equally [9], or that LM pursues 'fairness' because it seeks to minimize the unhappiness of the least happy group member. However, aggregation strategies such as AVG and LM (and most other work on group recommendation) have a

Table 1: Example predicted relevance scores in the range $[0, 5]$, and aggregated scores for the least misery (LM) and average (AVG) strategies are given. Users denoted as $u_1 - u_3$ and candidate items as $i_1 - i_5$.

	u_1	u_2	u_3	AVG	LM
i_1	5.0	5.0	2.5	4.17	2.5
i_2	4.5	4.5	2.5	3.83	2.5
i_3	4.0	4.0	3.0	3.67	3.0
i_4	4.0	1.5	5.0	3.5	1.5
i_5	0.5	3.0	1.0	1.5	0.5

limitation: they select the items in a top-N_G *independently of each other*. Hence, while each item recommendation may be 'fair', it may still be the case that, across a *set* of N recommendations, one or more group members may be treated unfairly [15, 18]. For example, for a given group member $u \in G$, if the top-N_G seems consistently to put other group members' interests ahead of u's interests, then the top-N_G is not fair to u.

We can illustrate this problem using a simple example. Consider a group G with three members, $G = \{u_1, u_2, u_3\}$. Assume that there are five candidate items and the goal is to recommend a top-3 ordered set to the group. We run a recommender algorithm (e.g. matrix factorization) to predict relevance scores $s(u, i)$ denoting the relevance of candidate item i to group member u; see Table 1. We also assume that an item i is relevant to a user u if the predicted relevance score is greater than or equal to 4.

The AVG strategy computes the mean ratings for each candidate item (see the AVG column in Table 1) and recommends the three items with the highest mean predicted ratings, $\{i_1, i_2, i_3\}$. The LM strategy takes the minimum rating (LM column in the Table). Since i_1 and i_2 both have the same aggregated scores, its top-3 recommendations to the group are either $\{i_3, i_1, i_2\}$ or $\{i_3, i_2, i_1\}$, depending on its tie-breaking strategy. Both u_1 and u_2 find items that are relevant to them (where $s(u, i) \geq 4$) in the top-3 group recommendations for both the AVG and LM strategies. However, u_3 cannot find any relevant item for her in the top-3 recommendations for the group for either of the AVG or LM strategies. This is an example of the kind of unfairness that we address in this paper: even if each individual recommendation is found in a 'fair' way, because they are found independently of each other, the top-3 is unfair for user u_3.

Now let's consider a method that does consider the overall fairness of the top-N_G. In [3], Felfernig et al. present an algorithm, which they call FAI, that is based on ideas in [7]. FAI Aggregation takes each group member $u \in G$ *in turn* and inserts into the top-N_G the candidate item i (from those which have not yet been inserted into the top-N_G) for which $s(u, i)$ is highest. Then, for each group member, it chooses a second item, starting with the group member who chose the last item in the previous round. It repeats this until it has chosen N items. The idea behind this strategy is that every group member will find a comparable number of items that they will like within the top-N_G.

For the example (Table 1), let's assume that users take turns in the order u_1 then u_2 then u_3. For u_1, FAI will select i_1. u_2's first choice is also i_1, therefore FAI will select her second best choice, i_2. Finally, for u_3, i_4 will be selected. FAI will recommend the ordered set $\{i_1, i_2, i_4\}$ to the group. This top-N_G contains a relevant item for each user. This is an improvement compared to the sets generated by the AVG and LM strategies, where there was no relevant item for u_3. However, the ordered set $\{i_1, i_2, i_4\}$ is still unfair to u_3. While other members receive relevant items in the first two ranks, u_3 only receives a relevant item in the last rank.

With AVG, LM and FAI in mind, let's ask what a fair top-N_G might look like. For a start, if possible, there should be at least one item in the top-N_G that is relevant to each group member. This is why, in the example, recommending $\{i_1, i_2, i_3\}$ (or a permutation of it) is unfair. This kind of fairness is not achieved, in general, by strategies that consider items independently, such as AVG and LM. The top-N_G will be even fairer if it seeks to balance, as far as possible, the relevance or utility of the items in the set across the users. In the example, this is another reason not to recommend set $\{i_1, i_2, i_3\}$: the total relevance to u_1 and u_2 (13.5 each) far exceeds the total relevance to u_3 (8.0), whereas recommending set $\{i_1, i_2, i_4\}$ has better balance (13.5, 11.0 and 10.0 for u_1, u_2 and u_3 respectively) as does $\{i_1, i_3, i_4\}$ (13.0, 10.5, 10.5 respectively). But this still treats the top-N_G as a set, ignoring the fact that it is in reality an ordered set.

To take into account the ordering within the top-N_G, we say that a top-N is fair to a group if the relevance of the items is balanced across the group members *for each prefix* of the top-N_G. In other words, the first item in the top-N should, as far as possible, balance the interests of all group members; the first two items taken together must do the same; also the first three; and so on up to N. Suppose we recommend $\{i_1, i_4, i_2\}$. There is not so much balance at rank 1 since i_1's relevance is 5.0, 5.0 and 2.5. An improvement is made for u_3 at rank 2 (at the expense of u_2), since the total relevances (for i_1 and i_4) are 9.0, 6.5 and 7.5. At rank 3, the total relevances (for i_1, i_4 and i_2) are 13.5, 11.0 and 10. Recommending $\{i_1, i_4, i_2\}$ is arguably better than recommending $\{i_1, i_2, i_4\}$, for example: for the latter, the total relevances are 5.0, 5.0, 2.5 at rank 1, then 9.5, 9.5, 5.0 at rank 2, which is still very unfair to u_3, the balance only being restored at rank 3 where the total relevances are 13.5, 11.0, 10.0. It is this rank-sensitive notion of fairness that we formalize in this paper. The formal definition is in Section 3. We call our approach Group Fairness-Aware Recommendation (**GFAR**). The definition leads naturally to a greedy algorithm for finding top-N group recommendations.

Note that, for simplicity, in the previous two paragraphs, we assumed that the total relevance of a top-N_G to a group member u is obtained by summing the $s(u, i)$ values for $i \in$ top-N_G. This simplifies the examples, but it is not what GFAR actually does. We postpone the details to Section 3.[1]

The main contributions of this paper are:

- We present, GFAR, a new, rank-sensitive definition of fairness for top-N group recommendations, based on balancing the relevance of items to group members for each prefix of the top-N. We give a greedy algorithm for finding top-N group recommendations according to the GFAR definition.

[1]These details also account for the fact that, for the example, GFAR will recommend $\{i_1, i_4, i_2\}$, rather than $\{i_1, i_4, i_3\}$.

- We run experiments to evaluate GFAR. We compare GFAR to five algorithms from the literature. We use two datasets, one from each of the movie and music domains. We use synthetic group of seven different sizes (from 2 to 8 members) and of three different types depending on the similarity between the group members' interests (*Similar*, *Divergent* and *Random*). Performance is measured using 10 fairness metrics that we have adapted from the literature.

We find that GFAR performs significantly better than all other algorithms around 43% of the time (179/420 cases); in only 10% of cases (43/420) are there algorithms that are significantly better than GFAR.

Our results also show that it is also more difficult to find fair top-N recommendations for groups in which the members' interests diverge, just as it is more difficult to reach a consensus in divergent groups [2, 14]. Additionally, it becomes more difficult to find fair recommendations as the group size grows.

In Section 2, we review relevant research on fairness in group recommenders. Section 3 presents our definition and algorithm, GFAR, in detail. Then, Section 4 describes our datasets, experimental methodology and metrics. Results are presented and analysed in Section 5. Finally, Section 6 concludes the paper with a discussion and ideas for future work.

2 RELATED WORK

Most commonly, group recommender research focuses on group satisfaction with each recommended item. Group recommender algorithms typically select the items in a top-N_G independently of each other and try to make sure that each item is 'good' (i.e. relevant) for everyone (e.g. on average) [2, 3, 5, 8]. But, in addition to FAI that we described in Section 1 [3], there is a small amount of fairly recent work that investigates fairness, treating it as a property of the set of items recommended to the group, the top-N_G [11, 12, 15, 18]. Unlike FAI, as we will see, this work does not use simple turn-taking; rather, it tries to balance the utility of the items in the top-N_G across the users.

Xiao et al. define fairness in terms of the utilities of the top-N_G to each group member [18]. The utility of the top-N_G to a group member u is simply the mean predicted relevance to u of each item in the top-N_G. Then Xiao et al. offer several alternative definitions of fairness, including: Least Misery Fairness, which is defined as the utility of the top-N_G to the group member whose utility is lowest; Variance Fairness, which is the complement of the variance of the utilities of the group members; and Min Max Ratio Fairness, which is the ratio of the lowest and highest utilities within the group. Xiao et al.'s objective function is a linear combination of mean utility and (one of the definitions of) fairness. They propose, for example, a greedy algorithm that incrementally inserts into the top-N_G the item that results in the largest value for the objective function. Let's use the phrase 'rank-sensitive' in the way we use it in GFAR, i.e. there must be a balance in the relevance or utility of the items across the group members for each prefix of the top-N_G. Then, we see that Xiao et al.'s definition is not rank-sensitive (since it treats the top-N_G as an unordered set). However, the use of a greedy algorithm does introduce a kind of rank-sensitivity into their approach.

Sacharidis defines the utility of a top-N_G to a member u of G as the similarity of the top-N_G to the top-N_u, i.e. u's top-N candidate items [12]. The fairness of a top-N_G he defines as the lowest member utility. He formulates the problem of finding a top-N_G using Pareto optimality. An item i dominates another i' if, for at least one member of the group $u \in G$, u ranks item i higher than item i' and, for each remaining member of the group, item i has equal or higher rank than i'. The Pareto Optimal items (PO) are the ones that are not dominated by any others. The N-level PO items are items that are dominated by at most $N-1$ other items. The top-N_G will be a subset of the N-level PO set. He presents probabilistic algorithms for finding this top-N_G. This approach to fairness in group recommendations is not rank-sensitive in the way that we defined it above.

Qi et al. [11] and Serbos et al. [15] address a different problem setting. They consider the recommendation of 'packages', such as vacation packages, to groups of users. A package differs from a top-N_G in that the group consumes *all* the items in the package, whereas a top-N_G comprises items that the group may choose between. Hence, the 'position' of the items within a package might imply, for example, the order of consumption (e.g. the order in which points-of-interest will be visited) whereas position in a top-N_G is a predicted preference ranking. Although the problem setting is different, their definitions of fairness are still of interest.

They define two alternative definitions of fairness: m-proportionality fairness and m-envy-free fairness [15]. A package exhibits m-proportionality fairness for group member u if u's predicted ratings for at least m ($m \geq 1$) items in the package are in the top Δ% of all of u's ratings. A package exhibits m-envy-free fairness for u if u's predicted ratings for at least m items in the package are in the top Δ% of ratings for that item for all members of the group. They use greedy algorithms to construct packages, item-by-item, to obtain high levels of either total m-proportionality or m-envy-free fairness across all members of the group. Note that, although this work focuses on the fairness of packages, it can be adapted to the fairness of top-N_G recommendations. For instance, for m-proportionality, we could define a top-N_G to be fair to group member u if at least m ($m \geq 1$) items in the top-N_G are in u's top-N_u. However, since a package is not ordered in the way a top-N_G is ordered, the resulting adaptation is not rank-sensitive in the way we defined it above. However, as with Xiao et al., the use of a greedy algorithm does introduce a kind of rank-sensitivity.

Illustrative comparison. For comparison we have implemented one algorithm from each of these three pieces of work: GreedyLM is Xiao et al.'s greedy algorithm using their Least Misery Fairness; XPO is one of the algorithms described by Sacharidis; and SPGreedy is the algorithm that Serbos et al. call the Single Proportionality Greedy Algorithm. Without spelling out the details of the calculations, we can show their top-N_G for the example given in the previous section (Table 1) and compare with what our algorithm, GFAR (which we will describe in Section 3), recommends, i.e. $\{i_1, i_4, i_2\}$. The simple example does not help us distinguish between GreedyLM and GFAR: they both recommend $\{i_1, i_4, i_2\}$. This is a good recommendation: there are relevant items in earlier ranks of the set for all users. XPO recommends $\{i_1, i_4, i_3\}$, and this too is plausible: all group members have at least one relevant item and, although XPO is not rank-sensitive in the same way as GFAR is rank-sensitive, it

so happens that in this example all group members find a relevant item in the earlier ranks of the recommendation set. SPGreedy recommends $\{i_1, i_2, i_3\}$ (the same as AVG): it fails to recommend any relevant items to u_3 and the item that is most relevant to u_3 is in the last rank. The results of our experiments using these algorithms are given in Section 5.

3 OUR APPROACH

In this section, we introduce **GFAR**, our definition of fairness for group recommendations and we give an algorithm for finding group recommendations that satisfy this definition. By way of notation, let I be the set of all items and U the set of all users. Let R be a $|U| \times |I|$ matrix, where $r_{ui} \in R$ is u's rating of i, or $r_{ui} = \perp$ if u has not rated i. We make no assumptions about the ratings: they can be explicit or implicit; they can be numeric, binary or unary. For each item i for which a user u has no rating, i.e. $\{i \in I : r_{ui} = \perp\}$, we assume we have an underlying recommender system (e.g. a matrix factorization algorithm) that can predict the relevance of that item to u, $s(u, i)$. We will write top-N_u for the ordered set of size N that we would recommend to user u, i.e. the N items whose $s(u, i)$ are highest. Let $G = \{u_1, \ldots, u_m\}$ be a group consisting of m users drawn from U. The ordered set of items of size N that we would recommend to group G is designated top-N_G.

In an ordered set, OS, we will write rank(i, OS) for the rank (position) of item i in OS and $OS[k]$ for the item whose rank in OS is k, where ranks will start from 1. For example, if top-N_G is $\{i_1, i_4, i_2\}$, then rank$(i_2$, top-$N_G)$ is 3, and top-$N_G[3]$ is i_2.

3.1 GFAR's definition of fairness

For a group member $u \in G$, let $p(\text{rel}\,|u, i)$ be the probability that item i is relevant to u. We estimate $p(\text{rel}\,|u, i)$ as:

$$p(\text{rel}\,|u, i) = \frac{\text{Borda-rel}(u, i)}{\sum_{j \in \text{top-}N_u} \text{Borda-rel}(u, j)} \tag{1}$$

Following Xiao et al. [18], we define Borda-rel$(u, i) = |\{j : \text{rank}(j, \text{top-}N_u) > \text{rank}(i, \text{top-}N_u), \forall_j \in \text{top-}N_u\}|$, where, from above, rank$(i, \text{top-}N_u)$ is the rank of item i in u's top-N candidate items, which are obtained using the $s(u, i)$ scores predicted by the underlying recommender algorithm.[2]

Let also $p(\neg\,\text{rel}\,|u, S)$ be the probability that none of the items in set S are relevant to user u. Then, we derive the probability that at least one item within S is relevant to u, $p(\text{rel}\,|u, S)$, as follows:

$$\begin{aligned} p(\text{rel}\,|u, S) &= 1 - p(\neg\,\text{rel}\,|u, S) \\ &= 1 - \prod_{i \in S} (1 - p(\text{rel}\,|u, i)) \end{aligned} \tag{2}$$

Now, from $p(\text{rel}\,|u, S)$ for each group member $u \in G$, we define $f(S)$ as the sum of each group member's probability of finding at least one relevant item within the set S:

$$f(S) = \sum_{u \in G} p(\text{rel}\,|u, S) = \sum_{u \in G} \left(1 - \prod_{i \in S} (1 - p(\text{rel}\,|u, i))\right) \tag{3}$$

[2] A more obvious definition is $p(\text{rel}\,|u, i) = s(u, i)/\sum_{j \in C} s(u, j)$, where $C \subseteq I$ are the candidate items. Compared to Eq. 1, this did not work well in our experiments. The probable explanation is that it relies too heavily on the actual $s(u, i)$ values, whereas Eq. 1 uses their ordering.

Eq. 3 shows how to 'balance' relevance across the group members for a set. It is not yet rank-sensitive. To make it rank-sensitive, we define the marginal gain in function f that arises when we add a new item to the set S, $f(i, S)$, as:

$$f(i, S) = f(S \cup \{i\}) - f(S) \tag{4}$$

Using Eq. 3 and Eq. 4, we can obtain the following:

$$f(i, S) = \sum_{u \in G} [p(\text{rel}\,|u, i) \prod_{j \in S} (1 - p(\text{rel}\,|u, j))] \tag{5}$$

Then, we can define an ordered set to be fair if there is balance in each prefix of the set. In other words, the first item in the set should, as far as possible, balance the interests of all group members; the first two items taken together must do the same; also the first three; and so on up to N:

$$\text{fair}(OS) = \sum_{k=1}^{|OS|} f(OS[k], \{i \in OS : \text{rank}(i, OS) < k\}) \tag{6}$$

Our definitions are, in fact, inspired by work on *intent-aware* Information Retrieval (IR) and recommendation [6, 13, 16] — work which aims to diversify a set of search results or recommendations. Faced with an ambiguous query, such as "apple", an IR system cannot know the user's intent, i.e. whether she intends to search for the corporation or the fruit. Informally, intent-aware methods in IR diversify the search results to ensure that they contain at least one document for each possible query interpretation. Analogously, in recommender systems, intent-aware methods diversify a set of recommendations to ensure that they cover each of the user's interests, as revealed by her profile [6, 16]. In GFAR, we adapted intent-aware diversification so that it can instead be used to generate fair recommendations to a group. Here, instead of trying to cover the different interpretations of a query or the different interests of an individual user, we try to cover the different interests of each user in the group.

3.2 GFAR's algorithm

Given a set of candidate items, $C \subseteq I$, the ideal top-N_G to recommend to group G is ordered set OS^*, which is the subset of the candidates, of size N, that has highest fairness, as defined in Eq. 6:

$$OS^* = \underset{OS \subseteq C, |OS| = N}{\arg\max} \text{fair}(OS) \tag{7}$$

Finding OS^* is intractable in general, since it requires considering all possible size N ordered subsets of the candidates.

A natural alternative is to find an approximation of OS^* using a greedy algorithm. The GFAR greedy algorithm starts with an empty set, $OS = \{\}$. At each iteration, it inserts into the ordered result set the item i^* from the remaining candidates (i.e. $C \setminus OS$) that gives the highest marginal gain:

$$i^* = \underset{i \in C \setminus OS}{\arg\max} f(i, OS) \tag{8}$$

3.3 GFAR example

We will illustrate GFAR using the example from Section 1. Based on the predicted relevance scores given in Table 1, we can obtain the top-N_u for each $u \in G$, i.e. each group member's top-3. From these, we can compute $p(\text{rel}\,|u, i)$ values using Eq. 1 (shown in the upper part of Table 2).

$p(\text{rel}\,|u_1, i_1) = 2/3, p(\text{rel}\,|u_1, i_2) = 1/3, p(\text{rel}\,|u_2, i_1) = 2/3, p(\text{rel}\,|u_2, i_2) = 1/3, p(\text{rel}\,|u_3, i_4) = 2/3, p(\text{rel}\,|u_3, i_3) = 1/3$

Iteration	OS	$f(i, OS)$ values
0	$\{\}$	$f(i_1, \{\}) = 4/3, f(i_2, \{\}) = 2/3, f(i_3, \{\}) = 1/3, f(i_4, \{\}) = 2/3$
1	$\{i_1\}$	$f(i_2, \{i_1\}) = 2/9, f(i_3, \{i_1\}) = 1/3, f(i_4, \{i_1\}) = 2/3$
2	$\{i_1, i_4\}$	$f(i_2, \{i_1, i_4\}) = 2/27, f(i_3, \{i_1, i_4\}) = 1/27$
3	$\{i_1, i_4, i_2\}$	–

Table 2: GFAR example. $p(\text{rel}\,|u, i)$ are based on Eq. 1. Only non-zero values are shown. At each iteration until $|OS| = 3$, we show OS and the non-zero $f(i, OS)$ values based on Eq. 5 for remaining candidate items. Finally, top-$N_G = \{i_1, i_4, i_2\}$ will be recommended.

The lower part of Table 2 shows values at each iteration. Initially (iteration 0), $OS = \{\}$. $f(i, \{\})$ simplifies to $\sum_{u \in G} p(\text{rel}\,|u, i)$, resulting in the selection of item i_1. At this point, $OS = \{i_1\}$, which is unfair to u_3. The algorithm will seek to 'redress the balance' in the next iteration (iteration 1).

In iteration 1, GFAR chooses the item that maximizes: $f(i, \{i_1\})$, which simplifies to $\sum_{u \in G} p(\text{rel}\,|u, i)(1 - p(\text{rel}\,|u, i_1))$, resulting in the selection of item i_4. Note that, the reason i_4 gives higher marginal gain than i_2, which is relevant to both u_1 and u_2, is that the already-selected item i_1 has a high relevance score to both u_1 and u_2, but has a zero relevance score to u_3. Now $OS = \{i_1, i_4\}$. The relevance scores of the items are now more balanced between the group members.

Finally, in iteration 2, the item that maximizes $f(i, \{i_1, i_4\}) = \sum_{u \in G} p(\text{rel}\,|u, i)(1 - p(\text{rel}\,|u, i_1))(1 - p(\text{rel}(u, i_4))$ will be selected. This reduces to $f(i, \{i_1, i_4\}) = \sum_{u \in G} p(\text{rel}\,|u, i)$, since $(1 - p(\text{rel}\,|u, i_1))(1 - p(\text{rel}\,|u, i_4))$ is equal for every group member. The item that gets selected is i_2. Now that $|OS| = 3$, the ordered set $\{i_1, i_4, i_2\}$ is recommended to the group. In this ordered set, all group members, u_1, u_2 and u_3, find relevant items, which was not the case with the AVG and LM aggregation strategies (Section 1). Arguably, GFAR's ordering of these items, $\{i_1, i_4, i_2\}$, is fairer than FAI's, $\{i_1, i_2, i_4\}$, since u_3 finds a relevant item at the second rank, rather than the third.

4 OFFLINE EXPERIMENTS

4.1 Experimental setup

4.1.1 Datasets. We use two datasets, one from each of the movie and music domains: the MovieLens 1M dataset[3] and the KGRec-music dataset[4]. The MovieLens dataset has ~1 million explicit, numeric ratings, $r_{ui} \in [1, 5]$, on ~3,700 movies by ~6,000 users. The KGRec-music dataset has ~750 thousand interactions (song downloads), giving us implicit unary ratings, on 8,640 songs by ~5,200 users.

4.1.2 Generation of synthetic group. Neither dataset is a group recommendation dataset. To run our offline experiments, we must create synthetic groups. We follow previous work on group recommendation [1, 12, 18]. We inquire whether fairness will be harder for larger groups, and for groups where users have divergent tastes. For this reason, for both datasets we create groups of sizes from $m = 2$ up to $m = 8$ inclusive, and we create three different types of group:

- *Random*: Members of *Random* groups are selected without replacement from U with uniform probability. Random groups loosely correspond to the real-life equivalent of groups that have unrelated members.
- *Similar*: Members of *Similar* groups are chosen to have similar tastes. We form these groups using a method based on previous work [1, 4]. We compute the similarities between pairs of users as the Pearson Correlation Coefficient (PCC) between their ratings. Since PCC lies between -1.0 and 1.0, it has been suggested [4] that PCC values of 0.1, 0.3 and 0.5 indicate small, medium and large effect sizes, respectively. To form a synthetic group, we randomly select a user from U and then greedily select at random further users but only drawing them from those who have a PCC greater than 0.3 (medium effect size in [4]) to any of the already-selected group members. This type of group is loosely equivalent to a group of people with similar tastes.
- *Divergent*: Members of *Divergent* groups are chosen to have less agreement between their tastes than in *Similar* groups. We create them in the same way that we create *Similar* groups but we greedily select users who have a PCC less than 0.1 (small effect size in [4]) to an already-selected group member.

Note that, while a user cannot appear more than once in a given group, they can be a member of multiple groups.

Seven different sizes of group and three different types of group gives 21 different scenarios. For each scenario, we create 1000 groups in each dataset. Figure 1 shows the distributions of the mean all-pairs similarity for these groups.

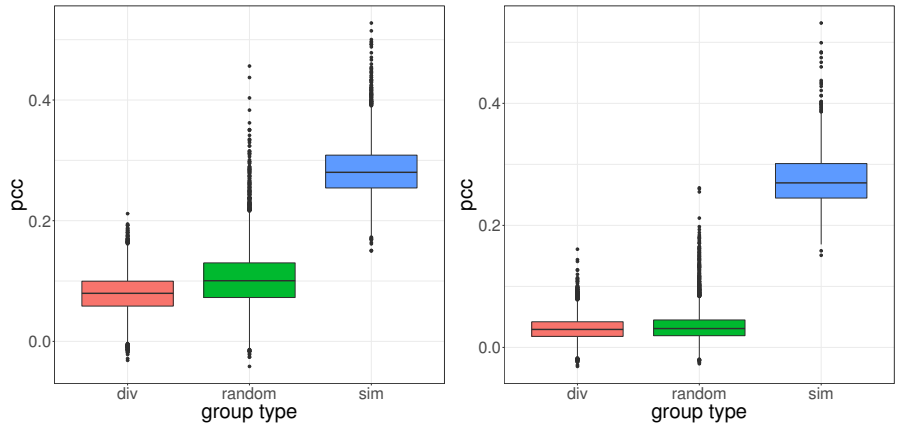
In Figure 1, we make two observations. First, for both datasets, the mean all-pairs similarity for *Divergent* is similar to that of *Random* groups. Second, the KGRec-Music dataset differs from the MovieLens dataset in that its *Divergent* and *Random* groups have lower mean all-pairs similarity than they do in the MovieLens dataset.

4.1.3 Approaches to compare. We compare the performance of GFAR to the following approaches from the literature that we described in Section 1:[5]

- AVG Score Aggregation [1, 8].

[3] http://grouplens.org/datasets/movielens/
[4] https://www.upf.edu/web/mtg/kgrec

[5] We also tried the LM Score Aggregation strategy [8]. Since it always performed worse than all other approaches, we do not include these results.



(a) MovieLens **(b) KGRec-Music**

Figure 1: Mean all-pairs similarity of the groups used in the experiments.

- FAI Aggregation Strategy [3].

We also compared with the following algorithms, which are drawn from the papers we reviewed in Section 2:[6]

- GreedyLM, which is Xiao et al.'s greedy algorithm using their Least Misery Fairness [18].
- XPO, as defined by Sacharidis [12].
- SPGreedy, which is the algorithm that Serbos et al. call the Single Proportionality Greedy Algorithm [15].

4.1.4 Underlying recommendation algorithm. All of the approaches to fairness that we compare require that we predict relevance scores $s(u, i)$ and, in some cases, generate top-N_u recommendations for the individual users in the groups. For this, we need an underlying recommender algorithm. To enable comparability with previous work, e.g. [12, 18], we use a form of matrix factorization (MF). We use a fast, accurate ALS-based MF algorithm that works for both explicit and implicit feedback datasets [10].[7]

4.1.5 Dataset splits. In our experiments, we randomly partition the ratings into training, validation and test sets such that 60% of each user's ratings are in the training set, 20% of them are in the validation set and 20% are in the test set. Results are averaged over five runs with different random splits.

4.1.6 Setting hyper-parameter values for MF. We find values for the hyper-parameters of the underlying MF algorithm by training it on the training sets and selecting the hyper-parameter values that optimize nDCG@N (see Section 4.2.4) on the validation sets. Specifically, for each user u, for all items i which are not rated by u in the training set, a score $s(u, i)$ is computed by MF; we select an ordered set of the top-N_u recommendations, $N = 20$; then the resulting recommendations are evaluated in terms of nDCG on the validation set. We select hyper-parameter values that give the highest mean nDCG across the users. Computing nDCG requires that we say which items are relevant to the user. For MovieLens, items in the validation set are considered relevant if $r_{ui} \geq 4$; for KGRec-music, all interactions in the validation set are considered relevant, since it is is an implicit dataset.

Our MF algorithm has two hyper-parameters: d, the number of latent factors; and α, the confidence level factor. For MovieLens, we find $d = 30$ and $\alpha = 1.0$. For KGRec-Music, $d = 230$ and $\alpha = 1.0$.

4.1.7 Generating group recommendations. Now, using the hyper-parameter values from above we train the MF on the union of the training and validation sets, which we will refer to as R^{train}. Once trained, we can obtain $s(u, i)$ for all users $u \in U$ and items $i \in I$. Specifically, if $r_{ui} \notin R^{\text{train}}$ (an unseen item), we use the MF model to predict $s(u, i)$. But, if $r_{ui} \in R^{\text{train}}$ (a seen item), then, following [18], we set $s(u, i) = 0$ to discourage items seen by an individual from being recommended to that individual again in a group recommendation. Once we have scores, $s(u, i)$, it is possible to compute individual top-N_u and $p(rel|u, i)$ for those algorithms that need them.

Then, for each group, we use each algorithm to generate a top-N_G ordered set of recommendations, with $N = 20$ (the same value used in [12, 18]). Finally, we evaluate the performance of the top-N_G using the metrics given below.

4.2 Evaluation metrics

We compare the top-N_G for each group and each algorithm with the group members' optimal rankings, top-N_u for each $u \in G$, which we obtain from the ratings in the test set [1].

Let I_u^{test} be the set of items in the test set of a user u. We will refer to items in I_u^{test} that are relevant for user u as u's ground-truth, gt_u. For MovieLens, $gt_u = \{r_{ui} \geq 4 : \forall_i \in I_u^{\text{test}}\}$; for KGRec-Music, $gt_u = \{r_{ui} \neq \perp : \forall_i \in I_u^{\text{test}}\}$.

Below, we describe and motivate the metrics we use to evaluate the top-N_G recommendations for the group G.

[6] We also tried two other algorithms: GreedyVar and EFGreedy. GreedyVar is Xiao et al.'s greedy algorithm using their Variance Fairness. It always performed worse than GreedyLM, so we choose not to include its results. EFGreedy is one of Serbos et al.'s algorithms, this time using their envy-freeness definition of fairness. It performed worse than SPGreedy except in a few configurations (such as some small *Similar* groups) so, again, we omit its results.

[7] We use its RankSys implementation: https://github.com/RankSys/RankSys

*4.2.1 Zero-recall (*zRecall*).* zRecall is a fairness metric. It measures the fraction of group members for whom *no relevant item was retrieved in the top-N_G*.

$$\text{zRecall}(G) = \frac{|\{u \in G : \text{Recall}\,@N(u) = 0\}|}{|G|} \tag{9}$$

where Recall $@N(u, G)$ is formally defined below. We expect to see lower scores for better performing algorithms: if we would like every group member to find at least one relevant item, then fair recommendations would have a zRecall score close to 0.

This metric is an adaptation of $rec_0^{\mathcal{U}}$, which is a metric used in recommending to shared accounts [17]. $rec_0^{\mathcal{U}}$ measures the fraction of users who share an account who do not get any relevant recommendation. It is also related to m-proportionality [15]. Since 1-proportionality is the fraction of group members for whom at least one relevant item appears in the top-N_G, zRecall is the same as (1 - *1-proportionality*).

4.2.2 Recall $@N$. This metric, Recall $@N$, and the others that we describe in the remainder of this section (Discounted First Hit and Normalized Discounted Cumulative Gain), do not directly give a score for a group. Rather they all give a score for a member of a group. We will first describe and motivate them, and only in Section 4.2.5 will we explain how we have converted them into group metrics.

For $u \in G$, Recall $@N(u, G)$ measures the proportion of u's relevant test set items that are in the top-N_G:

$$\text{Recall}\,@N(u, G) = \frac{|\text{top-}N_G \cap gt_u|}{|gt_u|} \tag{10}$$

*4.2.3 Discounted First Hit (*DFH $@N$*).* This metric measures whether *a group member finds an item that is relevant in the earlier ranks of the top-N_G*. Hence, unlike Recall$@N$, this metric is rank-sensitive. For a given user $u \in G$, DFH is defined as:

$$\text{DFH}\,@N(u, G) = \frac{1}{log_2(\text{fhr} + 1)} \tag{11}$$

where fhr is the rank of the first hit in the top-N_G, i.e. the rank in top-N_G of the first item that is in gt_u. If there is no such hit, then DFH is zero for user u.

*4.2.4 Normalized Discounted Cumulative Gain (*nDCG $@N$*).* This metric measures the extent to which *a group member finds relevant items in the earlier ranks of the top-N_G*. It is defined as:

$$\text{nDCG}\,@N(u, G) = \frac{\text{DCG}\,@N(u, G)}{\text{IDCG}\,@N(u, G)} \tag{12}$$

where $\text{DCG}\,@N(u, G) = \sum_{k=1}^{N} \frac{|\{\text{top-}N_G[k]\} \cap gt_u|}{\log(k+1)}$, and IDCG$@N$ is the maximum possible *DCG*$@N$.

Like DFH, nDCG is sensitive to the rank of items. Unlike DFH, it takes into account *all* of the items in top-N_G that are relevant to the user.

4.2.5 Aggregated Recall, DFH *and* nDCG *metrics.* As we have seen, Recall $@N(u, G)$, DFH $@N(u, G)$ and nDCG $@N(u, G)$ are specific to a given user u in a group G. We need to aggregate these metrics across all members of a group. We aggregate these user-specific metrics to give group-level metrics in three different ways:

- Mean (mean): The mean value of the metric over the group members, e.g. $\text{Recall}\,@N(G)(\text{mean}) = \sum_{u \in G} Recall@N(u, G)/|G|$.
- Minimum (min): The minimum value of the metric over the group members, which is inspired by [12, 18].
- Min-Max Ratio (minmax): The ratio of the minimum value to the maximum value of the metric over the group members, inspired by [18].

5 RESULTS

The results are reported as the mean of each metric (e.g., zRecall) over all groups G.

5.1 Results for zRecall

Figures 2 and 3 show zRecall results for MovieLens and KGRec-Music respectively. Note that smaller values of zRecall are better.

Comparing algorithms. For both datasets and for all three group types (*Random*, *Similar* and *Divergent*), all of the fairness-aware algorithms perform better than the AVG system. For both datasets, the worst performing fairness-aware algorithms are SPGreedy and GreedyLM. Of these two algorithms, for larger groups, $|G| \geq 6$, SPGreedy performs better than GreedyLM but, for smaller groups, it is GreedyLM that performs better. The best-performing algorithms are GFAR and FAI, and sometimes XPO. In the 42 configurations (two datasets, seven group sizes, three group types), mostly GFAR is the top-performing algorithm (34 out of 42 configurations).

Comparing group types. For both datasets and all three types of groups, as group size grows zRecall worsens (increases). This is because, as the group size grows, it becomes more difficult to generate fair and good recommendations to the groups. For both datasets and across the group sizes, *Divergent* groups have worse zRecall than do *Random* groups, which are in turn worse than *Similar* groups. These results suggest that ensuring fairness is likely to be more difficult in groups where preferences differ.

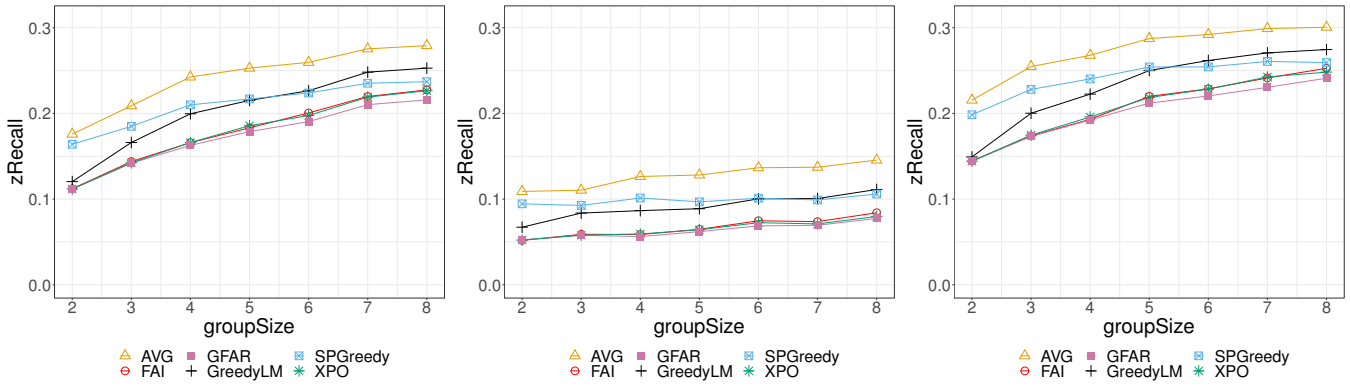
Comparing datasets. Compared to MovieLens, zRecall values are better (lower) for KGRec-Music for all configurations. It appears to be easier to generate fairer recommendations for the music dataset. This may be because KGRec-Music has a higher mean number of relevant test items per user (28.9), compared to MovieLens (19.1). This may be due to its use of implicit ratings. This needs to be confirmed by comparing more datasets.

5.2 Results for the other metrics

Due to space limitations, we only show the results for the other metrics for group size $m = 8$. We choose this because it is an extreme case for which it is more difficult to generate fair group recommendations. These results are in Tables 3 and 4. Note that larger values of these metrics can be interpreted as 'more fair'. Although we do not show the results for other group sizes, the results tend to follow similar trends.[8]

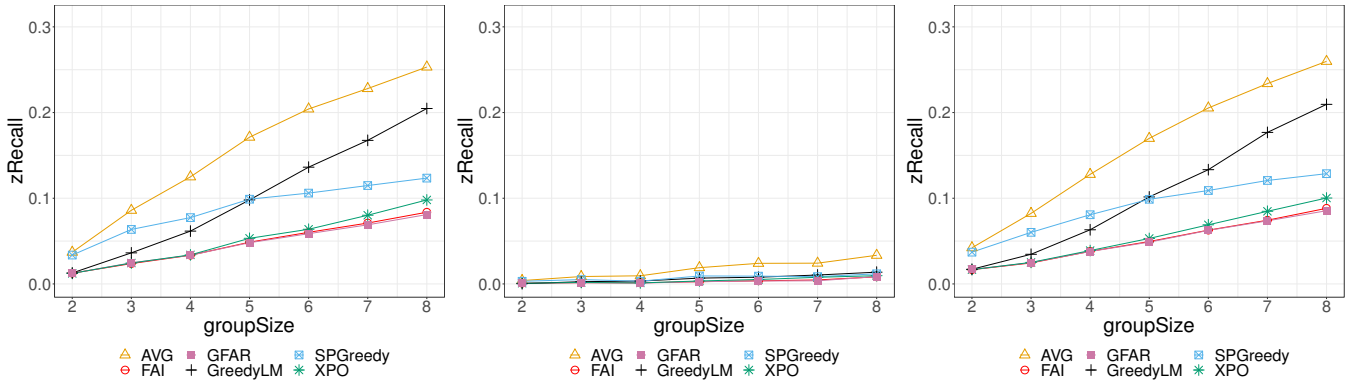
Comparing algorithms. For MovieLens, there are no metrics for which another method *significantly* outperforms GFAR. GFAR has the highest Recall(mean), nDCG(mean), nDCG(min), DFH(mean) and DFH(min). In these cases, with the exception of DFH(min) for

[8] Omitted results and the code used to run all of the experiments are publicly available: https://github.com/mesutkaya/recsys2020



(a) *Random* groups **(b) *Similar* groups** **(c) *Divergent* groups**

Figure 2: MovieLens dataset: zRecall @20 **against group size, for different group types.**



(a) *Random* groups **(b) *Similar* groups** **(c) *Divergent* groups**

Figure 3: KGRec-Music dataset: zRecall @20 **against group size, for different group types.**

the *Similar* groups, the differences are statistically significantly better than the most competitive other approaches. For Recall(min), Recall(minmax), nDCG(minmax) and DFH(minmax), mostly it is again GFAR that performs the best; however, these differences are not statistically significantly better than those for the other algorithms, with the exception of Recall(min) and Recall(minmax) for *Random* groups, which are significant.

On KGRec-Music, the GFAR results for Recall(mean), Recall(min), nDCG(mean), and DFH(mean) are statistically significantly better than the most competitive other approaches. However, FAI achieves significantly higher values for nDCG(minmax). FAI also achieves significantly better results for Recall(minmax) for *Similar* groups (only).

Comparing group types. For both datasets, *Divergent* groups have the worst (smallest) values for the mean, min and min-max versions of Recall, nDCG and DFH; *Similar* groups have the largest values. This supports what we found with zRecall: ensuring fairness is more difficult in groups where preferences differ.

Comparing datasets. Across these metrics, results for KGRec-Music are generally higher than those for MovieLens (the exceptions are Recall(mean) and nDCG(mean)). Again, for the same reason given earlier, we find that it seems easier to generate fair recommendations in the dataset that has a higher mean number of relevant items per user.

We can summarize the results that we have shown in Sections 5.1 and 5.2, and also the results that we did not show (for group sizes $m = 2 \ldots 7$), as follows. Overall, GFAR performs significantly better than all other algorithms for ~43% of the cases (179/420); in ~29% of cases (122/420), GFAR has the highest value for the metric but the difference between it and its closest competitor is not statistically significant; in ~18% of cases (76/420), another algorithm has a higher value for the metric than GFAR but the difference is not statistically significant; in only ~10% of the cases (43/420) are there algorithms that are significantly better than GFAR. We attribute GFAR's success to both its notion of rank-sensitivity and its way of balancing relevance. Other definitions of fairness are not rank-sensitive in the same way (although some approaches do achieve a degree of rank-sensitivity through the use of greedy algorithms) and none define balance in the way that we do.

Table 3: MovieLens dataset: Results when $m = 8$. For each group type, the best result for each metric is highlighted in bold. If the best result is achieved by GFAR, the second best is marked with ⋆; otherwise GFAR is marked with ⋆. The metrics for the best result are also marked with ι if the results are statistically significant with respect to the approach marked with ⋆. (Paired t-test with $p < 0.05$, with Bonferroni correction)

	zRecall	Recall			nDCG			DFH		
		mean	min	minmax	mean	min	minmax	mean	min	minmax
Random groups										
AVG	0.2791	0.1186	0.0032	0.01	0.1269	0.0023	0.0067	0.3896	0.0144	0.0157
FAI	0.2274	0.1282	0.0056	0.0163	0.1174	0.0051	0.0171⋆	0.3598	0.0288⋆	0.0342⋆
XPO	0.2265⋆	0.1346⋆	0.0057⋆	0.0166⋆	0.1274	0.0052⋆	0.0161	0.3783	0.0277	0.0323
GreedyLM	0.2528	0.1222	0.0049	0.0153	0.127	0.0038	0.0115	0.3974	0.0212	0.0235
SPGreedy	0.237	0.1283	0.0056	0.017	0.1314⋆	0.0045	0.0132	0.3993⋆	0.0257	0.0275
GFAR	**0.2158**$^\iota$	**0.138**$^\iota$	**0.0065**$^\iota$	**0.0187**$^\iota$	**0.1369**$^\iota$	**0.0061**$^\iota$	**0.0175**	**0.4059**$^\iota$	**0.0328**$^\iota$	**0.0361**
Similar groups										
AVG	0.1455	0.134	0.0179	0.0776	0.1524	0.0166	0.0494	0.485	0.0978	0.1043
FAI	0.0841	0.1606	0.029	0.1046	0.167	0.0318	**0.1003**	0.4763	0.1553	0.1708
XPO	0.0797⋆	0.1716⋆	**0.0318**	0.1062	0.1808⋆	0.0347⋆	0.0972	0.5057	0.1612⋆	**0.1716**
GreedyLM	0.1112	0.1472	0.0269	**0.1087**	0.1623	0.0255	0.0764	0.4995	0.1307	0.1393
SPGreedy	0.106	0.1487	0.0256	0.1024	0.1717	0.027	0.0788	0.5142⋆	0.138	0.143
GFAR	**0.0775**	**0.1742**$^\iota$	0.0315⋆	0.1042⋆	**0.1908**$^\iota$	**0.0362**$^\iota$	0.0962⋆	**0.5317**$^\iota$	**0.1649**	0.1711⋆
Divergent groups										
AVG	0.3006	0.118	0.0025	0.0074	0.1217	0.0019	0.0058	0.3716	0.0113	0.0129
FAI	0.2526	0.127	0.0046	0.0125	0.1127	0.004	0.0133⋆	0.3456	0.023⋆	0.0283⋆
XPO	0.2482⋆	0.1336⋆	0.0051⋆	0.0136⋆	0.1215	0.0041⋆	0.0126	0.3616	0.0226	0.0266
GreedyLM	0.2745	0.1214	0.004	0.012	0.1217	0.0031	0.0093	0.3794	0.0178	0.0199
SPGreedy	0.2593	0.1283	0.0047	0.0131	0.1263⋆	0.0037	0.0113	0.3804⋆	0.021	0.023
GFAR	**0.2411**$^\iota$	**0.1362**$^\iota$	**0.0056**	**0.0148**	**0.1299**$^\iota$	**0.0047**$^\iota$	**0.0136**	**0.3848**$^\iota$	**0.0258**$^\iota$	**0.0288**

Table 4: KGRec-Music dataset: Results when $m = 8$. For each group type, the best result for each metric is highlighted in bold. If the best result is achieved by GFAR, the second best is marked with ⋆; otherwise GFAR is marked with ⋆. The metrics for the best result are also marked with ι if the results are statistically significant with respect to the approach marked with ⋆. (Paired t-test with $p < 0.05$, with Bonferroni correction)

	zRecall	Recall			nDCG			DFH		
		mean	min	minmax	mean	min	minmax	mean	min	minmax
Random groups										
AVG	0.2531	0.0561	0.0025	0.0195	0.0919	0.0032	0.014	0.375	0.0207	0.0236
FAI	0.0837⋆	0.0713	0.0177⋆	0.1364⋆	0.102	0.0231⋆	**0.1164**$^\iota$	0.4296	0.1361⋆	**0.1522**
XPO	0.098	0.0726⋆	0.0147	0.1058	0.1064⋆	0.0176	0.078	0.4203	0.1087	0.1248
GreedyLM	0.2047	0.0579	0.0057	0.0454	0.0933	0.007	0.0308	0.3884	0.045	0.0504
SPGreedy	0.1235	0.0644	0.0107	0.0822	0.1011	0.0151	0.0706	0.4435⋆	0.0945	0.1
GFAR	**0.0809**$^\iota$	**0.0743**$^\iota$	**0.0189**$^\iota$	**0.1387**	**0.1133**$^\iota$	**0.0234**	0.1016⋆	**0.4615**$^\iota$	**0.1383**	0.15⋆
Similar groups										
AVG	0.0334	0.1093	0.0353	0.1791	0.1979	0.0544	0.1473	0.5951	0.227	0.2365
FAI	**0.0083**	0.1088	0.049	**0.278**$^\iota$	0.1776	0.0758	**0.2537**$^\iota$	0.5447	0.2892	**0.3011**
XPO	0.0096	0.1268⋆	0.052⋆	0.2472	0.2185⋆	0.0823⋆	0.2223	0.6074	0.2847	0.2933
GreedyLM	0.0137	0.1158	0.0505	0.265	0.2054	0.0786	0.2215	0.6083⋆	0.2865	0.2995
SPGreedy	0.0118	0.1125	0.0451	0.2337	0.1967	0.0735	0.2156	0.6078	0.2898⋆	0.2942
GFAR	0.0086⋆	**0.1277**$^\iota$	**0.0546**$^\iota$	0.2615⋆	**0.2255**$^\iota$	**0.0873**$^\iota$	0.2317⋆	**0.6358**$^\iota$	**0.2936**	0.2976⋆
Divergent groups										
AVG	0.2598	0.0553	0.0022	0.0167	0.0901	0.0027	0.0118	0.3695	0.0178	0.02
FAI	0.0882⋆	0.0701	0.0164⋆	0.1279⋆	0.1004	0.0213⋆	**0.1071**$^\iota$	0.4257	0.1282⋆	**0.1424**
XPO	0.1003	0.0717⋆	0.0142	0.1031	0.1052⋆	0.017	0.0773	0.4182	0.1056	0.1228
GreedyLM	0.2096	0.0571	0.0058	0.0465	0.0914	0.007	0.0313	0.3832	0.0459	0.0524
SPGreedy	0.1287	0.0632	0.01	0.0774	0.0994	0.0141	0.0671	0.4398⋆	0.0895	0.0954
GFAR	**0.0854**	**0.073**$^\iota$	**0.0173**$^\iota$	**0.129**	**0.1117**$^\iota$	**0.0217**	0.0951⋆	**0.4591**$^\iota$	**0.13**	0.142⋆

6 CONCLUSIONS AND FUTURE WORK

We present a novel definition of fairness for group recommendations, inspired by intent-aware recommender systems, called Group Fairness Aware Recommendations (GFAR). GFAR is rank-sensitive in the sense that it defines a top-N as fair if the relevance of the items to the group members is 'balanced' across the group members for each prefix of the top-N. A greedy algorithm for finding the top-N group recommendations follows naturally from the GFAR definition of fairness. We have compared GFAR against five other group recommendation approaches from the literature. In experiments using synthetic groups generated for two datasets, across a variety of metrics, the results show that GFAR performs best (significantly better) in ∼43% (179 out of 420) of cases.

In the future, we would like to apply GFAR to new domains, especially to better understand its relative performance on explicit and implicit ratings. We are also planning to conduct user trials to test GFAR's effectiveness for real groups. Besides this, we are interested in further investigating variants of the GFAR definition such as allowing user-specific weights. These weights can be used to achieve a form of *positive discrimination* to favour certain group members. For example, when recommending to families it may be useful if the children have greater weight than the parents. Alternatively, weights might be learned to improve GFAR's performance on goal-specific metrics.

ACKNOWLEDGMENTS

This research is affiliated with the Delft Data Science consortium. This work of the second author has been conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2, which is co-funded under the European Regional Development Fund.

REFERENCES

[1] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. 2010. Group recommendations with rank aggregation and collaborative filtering. In *Procs. of the 4th ACM conference on Recommender Systems*. 119–126.

[2] Lucas Augusto Montalvão Costa Carvalho and Hendrik Teixeira Macedo. 2013. Users' satisfaction in recommendation systems for groups: an approach based on noncooperative games. In *Procs. of the 22nd International Conference on World Wide Web*. 951–958.

[3] Alexander Felfernig, Ludovico Boratto, Martin Stettinger, and Marko Tkalčič. 2018. *Group recommender systems: An introduction.* Springer.

[4] Daniel Herzog and Wolfgang Wörndl. 2019. A User Study on Groups Interacting with Tourist Trip Recommender Systems in Public Spaces. In *Procs. of the 27th ACM Conference on User Modeling, Adaptation and Personalization*. 130–138.

[5] Anthony Jameson and Barry Smyth. 2007. Recommendation to Groups. In *The Adaptive Web*. Springer, 596–627.

[6] Mesut Kaya and Derek Bridge. 2019. Subprofile-aware diversification of recommendations. *User Modeling and User-Adapted Interaction* 29, 3 (2019), 661–700.

[7] Judith Masthoff. 2004. Group modeling: Selecting a sequence of television items to suit a group of viewers. In *Personalized Digital Television*. Springer, 93–141.

[8] Judith Masthoff. 2011. Group recommender systems: Combining individual models. In *Recommender Systems Handbook*. Springer, 677–702.

[9] Thuy Ngoc Nguyen, Francesco Ricci, Amra Delic, and Derek Bridge. 2019. Conflict resolution in group decision making: insights from a simulation study. *User Modeling and User-Adapted Interaction* (2019), 1–47.

[10] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. 2010. Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In *Procs. of the 4th ACM conference on Recommender Systems*. 71–78.

[11] Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2016. Recommending packages to groups. In *Procs of the 16th IEEE International Conference on Data Mining*. 449–458.

[12] Dimitris Sacharidis. 2019. Top-N group recommendations with fairness. In *Procs. of the 34th ACM/SIGAPP Symposium on Applied Computing*. 1663–1670.

[13] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. 2010. Exploiting query reformulations for web search result diversification. In *Procs. of the 19th International Conference on World Wide Web*. 881–890.

[14] Young-Duk Seo, Young-Gab Kim, Euijong Lee, Kwang-Soo Seol, and Doo-Kwon Baik. 2018. An enhanced aggregation method considering deviations for a group recommendation. *Expert Systems with Applications* 93 (2018), 299–312.

[15] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2017. Fairness in package-to-group recommendations. In *Procs. of the 26th International Conference on World Wide Web*. 371–379.

[16] Saúl Vargas, Pablo Castells, and David Vallet. 2012. Explicit relevance models in intent-oriented information retrieval diversification. In *Procs. of the 35th international ACM SIGIR conference on Research and development in information retrieval*. 75–84.

[17] Koen Verstrepen and Bart Goethals. 2015. Top-n recommendation for shared accounts. In *Procs. of the 9th ACM Conference on Recommender Systems*. 59–66.

[18] Lin Xiao, Zhang Min, Zhang Yongfeng, Gu Zhaoquan, Liu Yiqun, and Ma Shaoping. 2017. Fairness-aware group recommendation with pareto-efficiency. In *Procs. of the 11th ACM Conference on Recommender Systems*. 107–115.