Noname manuscript No.
(will be inserted by the editor)

Sequential Group Recommendations based on Satisfaction and Disagreement Scores

Maria Stratigi · Evaggelia Pitoura · Jyrki Nummenmaa · Kostas Stefanidis

Received: date / Accepted: date

Abstract Recently, group recommendations have gained much attention. Nevertheless, most approaches consider only one round of recommendations. However, in a real-life scenario, it is expected that the history of previous recommendations is exploited to tailor the recommendations towards meeting the needs of the group members. Such history should include not only which items the system suggested, but also the reaction of the members to these items. This work introduces the problem of sequential group recommendations, by exploiting the concept of satisfaction and disagreement. Satisfaction describes how well the group received the suggested items. Disagreement describes the satisfaction bias among the group members. We utilize these concepts in three new aggregation methods, SDAA, SIAA and Average+, designed to address the specific challenges introduced by sequential group recommendations. We experimentally show the effectiveness of our methods using big real datasets for both stable and ephemeral groups.

Keywords Group Recommendations · Sequential Recommendations · User Satisfaction · User Disagreement

Maria Stratigi *
Tampere University, Finland
Tel.: +358-50-4377440
E-mail: maria.stratigi@tuni.fi

Evaggelia Pitoura
University of Ioannina, Greece
Tel.: +30-26510-08811
E-mail: pitoura@cs.uoi.gr

Jyrki Nummenmaa
Tampere University, Finland
Tel.: +358-40-5277999
E-mail: jyrki.nummenmaa@tuni.fi

Kostas Stefanidis
Tampere University, Finland
Tel.: +358-50-4174121
E-mail: konstantinos.stefanidis@tuni.fi

1 Introduction

Recommendations have become a standard in many services that target a consumer. From listening to music to health information, recommender systems are employed to make the user experience better and smoother. One of the most widely used methods to produce a user recommendation is the Collaborative Filtering (CF) method. Here, the system assumes that if two users had previously interacted with a set of items in a similar way, then the patterns of one user can be used to predict the other's future interactions.

With the expansion of social media, another form of recommendations has emerged; namely the group recommendations [1, 21, 22]. It is often the case that a group of users asks a recommender system for suggestions instead of a single user. A typical instance of group recommendations is the following. A group of friends wants to see a movie. Each friend has his/her preferences for what kind of movie he/she would like to watch. The recommender system needs to address these preferences adequately and report a set of items that have a degree of relevance to each group member. There is extensive research on the selection process of the items. Two approaches are the most popular. The first is to create a pseudo user by combining each group member's data and then applying a standard recommendation method. The second and most used approach is to apply a recommendation method to each member individually and then aggregate the separate lists into one for the group.

One of the most common requirement that the aggregation phase of a group recommender system needs to fulfill is *fairness* towards all the members of the group [32–34, 19]. A rudimentary definition of fairness is to recommend to the group the item that has the best relevance. Intuitively, a way to find such an item is to use the *Average* method on all group members' preference scores for it. In this case, all group members are considered equals. However, this method has a big drawback. For example, consider the following scenario with a group that consists of three users. Two of them have similar preferences to each other, while the third does not. The average method is prone to ignore the opinion of the last user. Another option is the *Least Misery* approach. Here, the user with the minimum preference score will act as a veto to the rest of the group. This method has a drawback as well. It only takes the opinion of one group member under consideration while ignoring the others. Subsequently, in most cases, the system recommends an item that is acceptable for all members, but it will be an item with a somehow *low* preference score.

In many works, a hidden assumption is that each time the group interacts with the recommender system is distinct from the rest. In a real-life scenario, however, the system needs to keep a history of the group's interactions to tailor its suggestion accordingly. Returning to the previous example, where the group meets every week for a movie night, the system cannot suggest the same list of items each time, regardless of whether those items have the best relevance for the group. Internally, the system needs to log all the items suggested to the group and the members' reaction to the proposed items (either with explicit feedback from the users or with an implicit method). To illustrate this requirement of group history, in our running example, one user has completely different preferences to the rest of the members. So each time the group meets, the same user is dissatisfied with the recommender system's suggestions. To remedy that, we introduce the notion of

sequential group recommendations where the system considers the entire *sequence* of interactions for a group in order to produce its recommendations.

The concept of multiple rounds of recommendations allows us to make a conjecture. If a group member is dissatisfied with the proposed items at a specific iteration, he/she was either satisfied in the previous round or will be satisfied in the next. Subsequently, we exploit the notion of *satisfaction*, which estimates the degree of satisfaction in the recommendations for each group member, and *disagreement* which calculates the discrepancy among these satisfaction scores. We incorporate these satisfaction and disagreement scores during the group recommendation aggregation phase to ensure all members of a group are overall satisfied with the system's performance after several iterations.

Standard group recommendation approaches, such as Average and Least Misery, do not work very well when applied in a sequential scenario [35]. Average produce good satisfaction scores but mediocre disagreement scores, while Least Misery produces reasonable disagreement and poor satisfaction scores. In this work, we propose three new aggregation methods.

The Sequential Dynamic Adaptation Aggregation, SDAA, method draws its inspirations from the individual advantages of the Average and Least Misery methods, while mitigating their drawbacks. It considers the group as a whole and dynamically at each iteration, estimates a weight based on the group members' satisfaction scores, to combine the equality of the Average method and the inclusivity of the Least Misery method. In contrast, the Sequential Individual Adaptation Aggregation, SIAA, method is user-centric. It calculates a weight for each member based on his/her satisfaction and disagreement scores, and applies this weight on the group members' preference scores during the aggregation phase. Finally, we propose a heuristic method, named Average+. Contrary to the others, which examine each item individually, Average+ considers the entire list of data items. It builds upon the average method to take advantage of the good satisfaction scores, but considers more than the requested k items. It incrementally adds items to the group recommendation list that produce the least disagreement scores when examined together with the rest of the list.

In a nutshell, the contributions of our work are the following:

- We introduce the problem of sequential group recommendations that targets to offer results that maximize the overall group satisfaction, and minimize the variance between the users satisfaction scores.
- We exploit the concept of *satisfaction* and *disagreement* for proposing three new aggregation methods designed to address the challenges of sequential group recommendations. Satisfaction describes each group member's degree of satisfaction for the items recommended at each iteration, and disagreement describes the degree of bias a user and the group faces at each iteration.
- We experimentally evaluate our methods, using two big real datasets, MovieLens and GoodReads, and test the methods for different types of groups. Furthermore, we study the performance of stable groups, where the group members are the same for all iterations, and ephemeral groups, where at the end of each iteration, a member of the group leaves, and a new one enters it.

A preliminary version of this work appears in [35], where we introduced the SDAA method. In this article, we also propose the SIAA and Average+ methods. Furthermore, we evaluate the methods using a second real dataset, GoodReads,

in addition to the MovieLens dataset used previously. Finally, we consider stable and ephemeral groups. In stable groups, the members remain the same throughout all the system's iterations. In contrast, after each iteration in ephemeral groups, a member of the group leaves, and a new one enters it.

The rest of the paper is structured as follows: Section 2 presents basic concepts on recommender systems, and Section 3 introduces the problem of sequential group recommendations. Sections 4, 5, 6 describe the SDAA, SIAA, and Average+ aggregation methods, respectively. Section 7 presents our experimental setup, and Section 8 presents our evaluation results. Finally, Section 9 reports the related work, and Section 10 concludes this work with a summary of our contributions.

2 Basic Concepts

In this section, we introduce a simple group recommendation process. It is a stand-alone process and does not consider any past interactions that the group may have had with the system. We use the group recommendation approach to aggregate the individual group members' preference lists into one group preference list. We define a preference list to be the list we recommend to each member individually or the group.

Specifically, let U be a set of users and I be a set of items. Each user $u_i \in U$ has given a rating from 1 to 5 to an item $d_z \in I$ represented as $r(u_i, d_z)$. The subset of users that rated an item $d_z \in I$ is denoted by $U(d_z)$, while the subset of items rated by a user $u_i \in U$ is denoted by $I(u_i)$. Let's assume also a group of users G of size n, $G \subseteq U$. Having applied a single user recommendation algorithm to all group members in G and produced their preference lists $A_{u_1}, \ldots, A_{u_n}$, the next step is to aggregate these lists into one. After the aggregation phase, each item will have just one group preference score and the top k will be reported back to the group as the final recommendations.

Typically, two well established aggregation methods are used [2]. The first is the *Average* method. The basic concept of this approach is that all members are considered equals. So, the group preference of an item will be given be averaging its scores across all group members: $avgG(d_z, G) = \frac{\sum_{u_i \in G} p(u_i, d_z)}{|G|}$, where $p(u_i, d_z)$ gives us the preference score of d_z for user u_i (computed by a standard single user recommendation algorithm). The second is *least misery* aggregation method, where one member can act as a veto for the rest of the group. In this case, the group preference score of an item d_z is the minimum score assigned to that item in all group members preference lists: $minG(d_z, G) = min_{u_i \in G} p(u_i, d_z)$.

3 Sequential Group Recommendations

In this section, we introduce the notion of multiple rounds to the previous group recommendation process. Consequently, we do not consider each group query to the system as a stand alone process, but as a sequence of queries submitted to the system by the same group. We call each group query to the system an *iteration*. Each iteration has the main components of a standard group recommendation method: single user recommendations followed by the aggregation phase. Formally, let $\mathcal{GR}$ be a sequence of μ group recommendations $(Gr_1, \ldots, Gr_\mu)$. We use $p_j(u_i, d_z)$ for

Notation	Description	Notation	Description
G	Group.	$satO(u_i, \mathcal{GR})$	The u_i's overall satisfaction for the sequence.
$\mathcal{GR}$	Sequence of iterations	$groupSat(G, Gr_j, j)$	Group satisfaction for Gr_j
μ	Number of iterations in the sequence	$groupSatO(G, \mathcal{GR})$	Overall group satisfaction for the sequence
Gr_j	Group recommendation list at iteration j	$userDis(u_i, G, Gr_j, j)$	u_i's disagreement in Gr_j at iteration j
$p_j(u_i, d_z)$	u_i's preference score for item d_z at iteration j	$groupDis(G, Gr_j)$	Group disagreement for the Gr_j.
$A(u_i, j)$	u_i's individual recommendation list	$groupDisO(G, gr)$	Overall group disagreement for the sequence
$sat(u_i, Gr_j, j)$	u_i's satisfaction score for Gr_j at iteration j		

Table 1 The list of symbols and notations used in this paper.

the preference score of user u_i for item d_z at iteration j, $1 \leq j \leq \mu$. A single user recommendation algorithm estimates these scores.

In contrast to the previous one, this new model introduces the notion of multiple *iterations* or *sequence* of recommendations. We use these interactions to modify the recommender system's output in such a way that if a user had a low satisfaction score in a previous iteration, he/she will likely have a higher one during the current iteration.

3.1 Satisfaction Measure

To examine the effectiveness of our group recommender algorithm through a series of iterations, we need to define a measure that will help us elevate the problem from examining each iteration independently to examining a series of iterations. We introduce the notion of *satisfaction* that represents each group member's gratification for the recommended items after each iteration of the system. We define two variations of satisfaction: *single-user satisfaction* and *group satisfaction*.

For ease of reference, Table 1 holds all the symbols and notation used in the next sections.

3.1.1 Single User Satisfaction

First, we want to provide a formal measure of the degree of each user's satisfaction, $u_i \in G$, to the group recommendation Gr_j received at step j. We do so, by comparing the quality of recommendations that the user receives as a member of the group with the quality of the user's recommendations as an individual. We consider the user's individual recommendation list, as his/her ground truth because to the best of the system's knowledge, this list is the only information about the user.

Let $A(u_i, j)$ returns a list with the top-k items for user u_i, that is, the k items with the highest prediction scores for u_i. Our goal is to directly compare the user's satisfaction from the group recommendation list with that user's ideal case. Doing so gives us a clearer view of the user's satisfaction than the average method since we consider the top items for each user, not only the top items for the group. Formally:

$$sat(u_i, Gr_j, j) = \frac{\sum_{d_z \in Gr_j} p_j(u_i, d_z)}{\sum_{d_z \in A(u_i, j)} p_j(u_i, d_z)}. \quad (1)$$

In the nominator, we calculate the user's satisfaction based on the group recommendation list. For every item in Gr_j, we sum the score as they appear in each

user's top-k list. The denominator, calculates the ideal case for the user, by simply sum the scores of the k top items in the user's individual recommendation list. This way, we are able to normalize the user's satisfaction score. For example, if a user gives mainly low scores to items, then Equation 1 can compensate for it.

Note that we do not use the scores as they appear in the group list, but as they appear in the individual preference list of the user. Since the aggregation phase of the group recommendation process somewhat distorts the group members' individual opinions, we opt to take into consideration only the personal preference scores of each group member.

Still, Equation 1 remains *static*, in the sense that we calculate the satisfaction of a user for one iteration only. As stated before, what we want is to define a measure that considers the satisfaction scores from previous iterations of the system. Such a score will represent each group member's overall satisfaction with the entirety of the μ group recommendations.

Definition 1 (Overall Satisfaction) The overall satisfaction of user u_i with respect to a sequence $\mathcal{GR}$ of μ iterations is the average of the satisfaction scores after each iteration:

$$satO(u_i, \mathcal{GR}) = \frac{\sum_{j=1}^{\mu} sat(u_i, Gr_j, j)}{\mu}. \tag{2}$$

3.1.2 Group Satisfaction

Having defined each group member's satisfaction score, we can now define the satisfaction score of the entire group. Specifically, we define group G satisfaction concerning a group recommendation list Gr_j as the average of the satisfaction of the users in the group:

$$groupSat(G, Gr_j, j) = \frac{\sum_{u_i \in G} sat(u_i, Gr_j, j)}{|G|}. \tag{3}$$

Subsequently, we define the overall group satisfaction of a group G for a recommendation sequence $\mathcal{GR}$ of μ group recommendations $(Gr_1, \ldots, Gr_\mu)$, as:

$$groupSatO(G, \mathcal{GR}) = \frac{\sum_{u_i \in G} satO(u_i, \mathcal{GR})}{|G|}. \tag{4}$$

This measure indicates if the items we report to the group are acceptable to its members. Higher group satisfaction means that the group members are satisfied with the recommendations. However, since we average the members' satisfaction scores, a user's dissatisfaction can probably be lost in the computations. To counter this problem, we focus on the potential disagreements between the users in the group. For representing such *group disagreement* for just one instance of recommendations, we define the *groupDis* measure:

$$groupDis(G, Gr_j, j) = max_{u_i \in G} sat(u_i, Gr_j, j) - min_{u_i \in G} sat(u_i, Gr_j, j). \tag{5}$$

Subsequently, we define the overall group disagreement of G for the entire recommendation sequence as:

$$groupDisO(G, \mathcal{GR}) = max_{u_i \in G} satO(u_i, \mathcal{GR}) - min_{u_i \in G} satO(u_i, \mathcal{GR}). \tag{6}$$

Intuitively, we define the group's disagreement as the difference in the overall satisfaction scores between the most satisfied and the least satisfied member in the group. Ideally, we want this measure to take low values, indicating that the group members are all satisfied to the same degree. Higher *groupDis* values will demonstrate that at least one member of the group is biased against.

To further portray our system's behavior, we define the *user disagreement* of a user to be the difference between the satisfaction score of the most satisfied group member and his/her satisfaction score. Formally:

$$userDis(u_i, G, Gr_j, j) = max_{\forall_{u_l \in G}} sat(u_l, Gr_j, j) - sat(u_i, Gr_j, j). \quad (7)$$

This allows us to determine better if a group member is systematically being favored (he/she will have very low user disagreement scores) or is disregarded (he/she has very high user disagreement scores).

3.2 Problem Definition

Our sequential group recommender system needs to achieve two independent objectives that are critical to the success of the model. The first objective considers the group as an entity: we want to offer the best possible results for the group. The second objective considers the group members independently: we want to behave as fairly as possible towards all members.

Definition 2 (Sequential Group Recommendations) Given a group G, the sequential group recommender produces at the j^{th} iteration a list of k items Gr_j, $Gr_j \in \mathcal{GR}$, that:

1. Maximizes the overall group satisfaction, $groupSatO(G, \mathcal{GR})$, and
2. Minimizes the overall variance between users satisfaction scores: $groupDisO(G, \mathcal{GR})$.

To achieve the first objective, we target to maximize the *groupSatO* value. For that, we require the items with the highest preference scores for the group as a whole. Since *groupSatO* expresses the average satisfaction of the group members, and the dissatisfaction of just one member is easily lost, we also need to achieve the second objective, to minimize *groupDisO*, which represents the degree of dissatisfaction between the members of the group.

Depending on the similarity between the group members, these two objectives may be conflicting with each other. A simple example is a group of three members, two are highly similar to each other, and one is very dissimilar to them. To achieve high *groupSatO* values, we need to recommend items relevant to the two similar users to increase the average satisfaction of the group. On the other hand, by doing so, *groupDisO* will take high values since we do not address the needs of the third member. Overall, we need to recommend items that are not just good enough for all members, since that still returns low *groupSatO* values, but items that have high relevance to the group, without sacrificing the opinions of the minority.

4 Sequential Dynamic Adaptation Aggregation Method

Generally speaking, both the Average and the Least Misery aggregation methods have drawbacks when we consider them for sequential recommendations. Simultaneously, both have advantages (namely, equality for Average and inclusion of all opinions for Least Misery) that are assets for a recommender. Furthermore, it has been proven that to compute the best suggestions for a group that minimizes the gap between the least and highest satisfied group members is an NP-Hard problem [39]. We want to achieve that throughout the multiple iterations of the system, to be equally fair to all members of the group.

The first method we examine, capitalizes on the advantages of the Average and the Least Misery aggregation methods and proposes a novel aggregation method that is a weighted combination of them. We call the method *sequential dynamic adaptation aggregation method*, or shortly SDAA. The preference score of an item for a group in SDAA is computed as follows:

$$score(G, d_z, j) = (1 - \alpha_j) * avgG(G, d_z, j) + \alpha_j * leastG(G, d_z, j). \quad (8)$$

The function $avgG(G, d_z, j)$ returns the score of the item d_z as it is computed by the average aggregation method during iteration j, and function $leastG(G, d_z, j)$ returns the least satisfied user's score of d_z at iteration j. The variable α takes values from 0 to 1. If $\alpha = 0$, then the sequential hybrid aggregation method becomes average aggregation. At the same time, when $\alpha = 1$, it transforms to a modified least misery aggregation, and we only consider the preferences of the least satisfied member.

Intuitively, when $\alpha = 0$, our method satisfies the first part of the fair sequential group recommendation problem since the average aggregation considers the best options for the group as a whole. The benefits of $\alpha = 1$ can be seen for higher values of μ. At the first iterations, the group disagreement may remain high since it considers the users' overall satisfaction (Equation 2).

Given that our goal is to fulfill both the problem definition's objectives, we need to set the value of α between 0 and 1. Furthermore, we want this value to self-regulate, to describe the consensus of the group more effectively. Thus, we set the value of α dynamically in each iteration by subtracting the group members' minimum satisfaction score in the previous iteration from the maximum score.

$$\alpha_j = max_{u \in G} sat(u, Gr_{j-1}, j-1) - min_{u \in G} sat(u, Gr_{j-1}, j-1), \quad (9)$$

where $j > 1$. When $j = 1$ (the first iteration of the system), then $\alpha = 0$, and the aggregation method reverts to that of a classic average aggregation.

By having this dynamically calculated α, we counteract the average and least misery's drawbacks. Intuitively, suppose the group members are equally satisfied at the last round. In that case, α takes low values, and the aggregation will closely follow that of an average, where everyone is treated as an equal. On the other hand, if one group member is extremely unsatisfied in a specific iteration, α takes a high value and promotes that member's preferences on the next iteration. Formally:

Property 1 Let u be a user in a group G, such that,
$sat(u, Gr_{j-1}, j-1) < sat(u_i, Gr_{j-1}, j-1)$, $\forall u_i \in G \backslash \{u\}$. Then, $\exists u_y \in G \backslash \{u\}$, such that, $sat(u_y, Gr_j, j) < sat(u, Gr_j, j)$.

Algorithm 1: SDAA Aggregation Algorithm

Data: Group G, iteration j, top k
Result: Group Recommendation List: Gr_j

1 $Gl \leftarrow \cup_{u_i \in G} A(u_i, j)$;
2 **if** *j=1* **then**
3 $\quad \alpha_j = 0$;
4 **else**
5 $\quad \alpha_j = maxSat(G, j-1) - minSat(G, j-1)$;
6 **end**
7 **for** *item* d_z *in* Gl **do**
8 $\quad score(G, d_z, j) = (1 - \alpha_j) * avgScore(G, d_z, j) + \alpha_j * leastScore(G, d_z, j)$;
9 $\quad Gr_j \leftarrow Gr_j \cup score(G, d_z, j)$;
10 **end**
11 $sort(Gr_j)$;
12 $report(top(Gr_j, k))$;

Proof For the purpose of contradiction, assume that $sat(u, Gr_j, j) < sat(u_y, Gr_j, j)$, $\forall u_y \in G \backslash \{u\}$. Then, this means that u is the least satisfied user at iteration j, which in turn means that α takes values not leading towards a least misery approach at this iteration, meaning that u was not the least satisfied user at iteration $j-1$, which is a contradiction.

The SDAA method is described in Algorithm 1. The input to the algorithm is the group G, the iteration j, and the size k of the group recommendation list Gr_j, which we report to the group after each iteration is finished. n Line 1 we construct a set, Gl, that contains all the items that appear in the group members individuals recommendation lists. In Lines 2–6, we calculate the α; if it is the first iteration, meaning $j = 1$ (Line 2), then $\alpha = 0$, otherwise, we use Equation 9 (Line 5) to calculate it. In Lines 7–10, we perform the SDAA method. For all the items in Gl, we calculate the item score using Equation 8 (Line 8) and insert them in the group list Gr_j (Line 9). After we have examined all the items in Gl, we sort the items in the group list (Line 11), and finally, we report the top-k of them to the group (Line 12). The complexity of our algorithm is $\mathcal{O}(n \log n)$ due to the time complexity of the sorting function, where $n = |Gl|$.

5 Sequential Individual Adaptation Aggregation Method

The goal of the SDAA method is to calculate a weight score in order to combine the average and least misery methods. This weight was the same for all group members, and even though it was calculated using all members' satisfaction scores, we did not directly address each individually. Thus a member's interest can be ignored. As a result of this observation, we propose a new aggregation method that concentrates on each group member individually. We accomplish that by assign each member a *weight* score. This weight is based on each user's statistics, namely the satisfaction and disagreement scores, to calibrate the system for each user independently.

Specifically, during the group recommender's aggregation phase, we exploit the concept of overall user satisfaction and user disagreement. Each group member is assigned an individual weight, which is applied to an item's relevance score given by the recommender system (Equation 10). The final score of that item for the

group is the summation of its weighted relevance score for all group members.

$$score(G, d_z, j) = \sum_{u_i \in G} w_{u_i,j} * p_j(u_i, d_z). \quad (10)$$

Taking into consideration the particulars of our problem, we pay attention to satisfaction and disagreement values. Since we want them to be applied to each group member individually, we do not use the group variant of these measures, but instead, we focus on the individual ones (Equations 2 and 7).

The satisfaction score of a user offers a way to balance the recommendations for all members of a group through a series of recommendation iterations. These members that were not satisfied in the previous iterations are promoted for the next ones. A straightforward approach to accomplish this is to utilize the overall satisfaction of a user. We want the users with a low overall satisfaction score to have a higher weight compared to those with a high overall satisfaction score. More specifically, let assume that the group is in the j-th iteration. We calculate the overall satisfaction score for all $j-1$ iterations (Equation 2). Subsequently, the weight that is assigned for each member at iteration j is:

$$w_{u_i,j} = 1 - satO(u_i, \mathcal{GR}_{j-1}). \quad (11)$$

where $\mathcal{GR}_{j-1}$ are all previous recommendations at the $j-1$ iterations of the system.

The users with high overall satisfaction are assigned a lower weight, while the users with lower satisfaction scores are assigned a higher one. If the group members are satisfied to the same degree, then they are assigned similar weights. Since we compute a user's overall satisfaction score, we ensure that a user that is systematically biased against will have the highest weight.

Such an aggregation method is based only on the user's overall satisfaction score, which is his/her average satisfaction scores. As with any Average approach, it makes the aggregation method slow to react in extreme cases. Suppose a user was highly satisfied in the first μ iterations and then extremely dissatisfied in the rest. In that case, the method will require some iterations to assign him/her a higher weight to promote him/her. The same is also true in the reverse case, where a user was dissatisfied and then became highly satisfied. This leads to an aggregation method that promotes him/her even though he/she is satisfied during some iterations.

A way to counter this drawback, is to shorten the iterations needed for a change in user's satisfaction to be reflected in the user's assigned weight. To achieve this, we introduce the *sequential individual adaptation aggregation method*, or SIAA, that considers not only the overall satisfaction of a user, but also the user disagreement of the previous iteration.

$$w_{u_i,j} = (1-b) * (1 - satO(u_i, \mathcal{GR}_{j-1})) + b * userDis(u_i, G, Gr_{j-1}, j-1), \quad (12)$$

where b is the weight we use to balance the overall satisfaction and user disagreement scores. This is a constant variable and we found experimentally its best value. Given that we are in the j-th iteration of the system, $\mathcal{GR}_{j-1}$ expresses the recommendations of all the previous $j-1$ iterations. Finally, Gr_{j-1} includes the recommendations of the immediately preceding iteration.

Algorithm 2: SIAA Aggregation Algorithm

Data: Group G, iteration j, top ks, b
Result: Group Recommendation List: Gr_j
1 $Gl \leftarrow \cup_{u_i \in G} A(u_i, j)$;
2 **for** *item* d_z *in* Gl **do**
3 $score(G, d_z, j) = 0$;
4 **for** u_i *in* G **do**
5 $w_{u_i,j} = (1-b) * (1 - satO(u_i, \mathcal{GR}_{j-1})) + b * userDis(u_i, G, Gr_{j-1})$;
6 $score(G, d_z, j) = score(G, d_z, j) + w_{u_i,j} * p_j(u_i, d_z)$
7 **end**
8 $Gr_j \leftarrow Gr_j \cup score(G, d_z, j)$;
9 **end**
10 $sort(Gr_j)$;
11 $report(top(Gr_j, k))$;

To generalize, if a user was severely dissatisfied in the previous iteration and is overall dissatisfied, then the weight is high. If the user was dissatisfied in the previous round but overall has a high satisfaction score, then the weight is significantly lower. This is in accordance with the previous method that disregards any disagreements. However, the added variable of the user disagreement influences the next iteration. Specifically, suppose the user was considerably biased against in the current iteration (has a low satisfaction score while another member has a high one). In that case, he/she will be promoted faster than the plain satisfaction weighted method. Furthermore, if a user is the most satisfied in the current iteration, then his/her weight will be lowered to promote alternative group members.

The SIAA method is described in Algorithm 2. The output and Line 1 are the same as Algorithm 1. Additionally, we have the same input as 1 but with the added variable b, which is used to calculate the weight score (Line 5). This calculated weight is then applied to group members' preference score that the recommender system has predicted for each item d_z, denoted as $p_j(u_i, d_z)$, $u_i \in G$, $d_z \in Gl$. The final score of the item is the weighted sum for all users (Line 6). Finally, as in Algorithm 1, we sort the group recommendation list Gr_j and report to the group the top k items (Lines 10-11). The complexity of our algorithm is $\mathcal{O}(n \log n)$, due to the time complexity of the sorting function, where $n = |Gl|$.

6 Average+ Aggregation Method

All the previous aggregation methods examine how good a *single* item is for the group and add it in the group recommendation list, without taking into account the items already present there. Since we examine the users' satisfaction based on the entire list, adding to the list each item individually may not produce the best suggestions. Based on this observation, we propose the Average+ aggregation method that exploits the Average method that offers the best group satisfaction [35]. Specifically, Average concentrates on finding the items that will satisfy the majority of the group. At the same time, Average also has high group disagreement scores because it is prone to ignore the outlier of a group. If one group member is dissimilar to the rest of the group, due to the Average method, he/she is almost always dissatisfied. This observation drives us to propose a new aggregation

method, which not only secures the advantages of the Average method but, at the same time, mitigates the problems that the drawback mentioned above creates.

This aggregation method has two phases. In the first phase, we capitalize on the advantage of the average method, and, in the second phase, we counter its drawback. The first phase is straight-forward. The average aggregation method does all the work for us. Since it is the method with the best group satisfaction scores, we use a standard average aggregation method for the group. However, we do not take the last step of a group recommender process: a group recommendation system produces a long list of potential items but it only presents to the group the top ones with the highest predicted score. Instead of only keeping that short list, we keep a significantly longer one that contains the items with the highest scores.

After experimentation, the size of the list that provides the best results is $5k$, where k is the number of items we propose to the group. We denote this list as $AvgList_j^G$ for group G at iteration j. We do not want to take a longer list since the items further down typically do not have good enough group relevance scores (meaning that the items are not relevant to most of the group), and hence are not good enough for us to use to take advantage of the performance of the average aggregation method.

Having found the most relevant items for the group and subsequently have secured a high group satisfaction, we next want to minimize the group disagreement. We first exploit a simple idea. For each item in the $AvgList_j^G$ list, we calculate the group disagreement score they would have generated if they were the only item proposed to the group. Meaning that Gr_j consists of only that item (Equation 13). We return to the group the k ones with the lowest group disagreement.

$$Gr_j = Gr_j \cup \{d_z \mid min_{\forall d_z \in AvgList_j^G}(groupDis(G, d_z, j)) \ \vee d_z \notin Gr_j\}. \quad (13)$$

This allows us not only to narrow our search to items that are relevant to the group, but at the same time ensure that they have the lowest group disagreement possible.

This is a simple idea because it examined each item individually. However, the system returns to the group a list of items, and we calculate the various satisfaction and disagreement scores based on the entire list. We consider the list as a whole, not each item independently. For example, a user may not be interested in one item but be excited for another. These two items can balance each other out and offer better group satisfaction and disagreement scores. Therefore, we propose a more sophisticated method.

To find the best k items to propose to the group based on the group satisfaction and group disagreement scores they generate is a hard problem. It has been proven that to compute the best suggestions for a group that minimizes the gap between the least and highest satisfied group members is an NP-Hard problem [39].

Thus, we propose an incremental heuristic method. This method consists of two steps. In the first step, we include in the group recommendation list Gr_j the item with the highest predicted score in $AvgList_j^G$. The second step of the method is recursive. At each step, we examine one by one all the items in the $AvgList_j^G$. We include in the list the item that its inclusion generates the lowest group disagreement score (Equation 14). We perform this step until we have included k items in the group recommendation list.

$$Gr_j = Gr_j \cup \{d_z \mid min_{\forall d_z \in AvgList_j^G}(groupDis(G, Gr_j \cup d_z, j)) \vee d_z \notin Gr_j\}. \quad (14)$$

Algorithm 3: Average+ Aggregation Algorithm

Data: Group G, iteration j, top k, $AvgList_j^G$
Result: Group Recommendation List: Gr_j
1 $Gr_j \leftarrow AvgList_j^G.getFirst()$;
2 **while** $|Gr_j| < k$ **do**
3 select an item $d_z \in AvgList_j^G \backslash Gr_j$ that minimizes: $groupDis(G, Gr_j \cup d_z, j)$;
4 Add item d_z to Gr_j: $Gr_j = Gr_j \cup d_z$;
5 **end**
6 $report(top(Gr_j, k))$;

By incrementally filling the group recommendation list, we can examine all the items and the effects they collectively have for the group.

We describe this process in Algorithm 3. As input, we have the group G, the iteration j of the system, the number k of the reported items, and the list $AvgList_j^G$, which contains the $5k$ items with the highest predicted scores after applying the average aggregation method. The output of the algorithm is the group recommendation list Gr_j. In Line 1, we insert in the group recommendation list Gr_j the item with the highest predicted score given by the average aggregation. The recursion starts in Line 2 and stops when we have included in the Gr_j the requested k items. In Line 3 we iterate over all the items in the $AvgList_j^G$ list and select the item d_z that produces the minimum group disagreement score and it is not already included in the group recommendation list Gr_j. In Line 4 we insert the selected item to the list. Finally, we report to the group the recommendation list. The complexity of this algorithm is $\mathcal{O}(n^2)$ where $n = |Gr_j|$. We perform the inner loop n times, over $5n$ items.

7 Experimental Setup

7.1 Datasets

For the experimental evaluation of our sequential group recommender system, we do not have access to any dataset that contain ratings for groups. We instead artificially created groups using two real datasets, namely the 20M MovieLens Dataset [12], and the 15M book reviews from GoodReads [38]. MovieLens contains 20M ratings across 27,3K movies given by 138,5K users between 01.1995 and 03.2015. The GoodReads dataset contains around 15M ratings from 465K users for about 2M books. To simulate multiple iterations of suggestions, we split the datasets into chunks. Each chunk is added to the system, representing new information, and along with the already existing data in the system, is used for locating the suggestions for the next iteration.

MovieLens Dataset. Initially, we divide the dataset into two parts of roughly the same size. The first part that contains the ratings given between 01.1994 and 12.2003, is the starting dataset of the recommender. This gives us an initial dataset that consists of 8.381.255 ratings, 73.519 users and 6.382 movies. We initiate the system with this starting dataset to avoid any issues emanating from the cold

start problem[1]. The second part of the dataset is further split into chunks based on timestamps. We create 22 chunks, where each one includes information for a period of 6 months. During the first iteration, the system will have access only to the initial dataset. When that iteration ends, the system will be enhanced with one additional chunk, and after each subsequent iteration, the system will be given access to the next chunk.

GoodReads Dataset. For GoodReads, we altered the splitting process due to the difference in the ratings' distribution over time. The distribution is skewed towards the latter years, with the starting years having very few ratings. The starting data for MovieLens corresponded to around 40% of the entire dataset. We take a comparable size for GoodReads, which amounts to around 6.296.000 ratings, and split the rest into equal-sized chunks that contain 674.565 additional ratings each. It is worth noting that GoodReads is far more sparse than MovieLens. Subsequently, this is going to have an adverse effect on the quality of the recommendations.

7.2 Group Formation

The experimental procedure of [35] was focused on stable groups. This means that the group members do not change between iterations. We extend the experiments for this work to also include ephemeral groups. We now allow group members to leave the group and new ones to enter it. The users that have left a group are allowed to participate in the group again.

We examine our recommender on three types of groups based on the similarity shared between the members. We calculate the similarity between the group members, employing the starting dataset, using Pearson Correlation [29], which takes values from -1 to 1. Higher values imply a higher similarity between the users, while negative values indicate dissimilarity:

$$s(u_i, u_l) = \frac{\sum_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})(r(u_l, d_z) - \bar{r}_{u_l})}{\sqrt{\sum_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})^2} \sqrt{\sum_{d_z \in X} (r(u_l, d_z) - \bar{r}_{u_l})^2}}, \text{ where } X = I(u_i) \cap I(u_l) \text{ and } \bar{r}_{u_i} \text{ is the mean of the ratings in } I(u_i), \text{ i.e., the mean of } u_i\text{'s ratings.}$$

We consider two users to be highly similar to each other if they share an above 0.5 similarity score, while dissimilar when they have -0.5 or lower similarity score. The types of groups we are considering are the following:

4 similar − 1 dissimilar (4+1): The four members of the group share a high similarity score, while the dissimilar one shares a low similarity score with the rest of the group members.

3 similar − 2 similar (3+2): We divide the group into two subgroups. The members of each subgroup are similar to each other, while at the same time, the subgroups are dissimilar to one another, i.e., all members of one subgroup are dissimilar to all members of the other subgroup.

5 dissimilar: All members of the group are dissimilar with each other.

[1] The cold start problem in collaborative filtering appears when there is not enough information about a user and we are unable to find similar other users to him/her. This problem is beyond the scope of this research, and to overcome it, we initialize our system with a large enough dataset.

With these group formats, we want to simulate three different real-life scenarios. First, when a new person enters an established group, and his/her interests may very well be dissimilar to the rest of the group. For example, consider a working environment in which a new colleague joins an existing project team. Second, when two different groups join together for a single activity, such as two working groups that need to collaborate on a new project. The final group type simulates the case when random people join together for an activity, like in a business lunch or a tour offered by a travel agency.

We will also examine the performance of our aggregation methods for ephemeral groups. The initial members of the group will be the same as the stable ones. To simulate the ephemeral groups' dynamic characteristics, we randomly remove a group member after each iteration and randomly include a new user in the group so as the group size to remain the same. The new member selected is such that the type of the group does not change. For example, if we have a 4+1 group, and the one dissimilar user is removed, then the new member has to be dissimilar to the four remaining group members.

8 Experiments

8.1 Experimental Procedure

In our experiments, we examine the behavior of the three types of groups. For each type, we consider 100 different groups with five members. For all the groups in the set, we calculate the average of $groupSatO(G, \mathcal{GR})$ and $groupDisO(G, \mathcal{GR})$ at the end of each iteration. In the ephemeral groups, at each iteration, we focus only on the users that are currently members of the group. Additionally, to examine the overall performance of the aggregation methods, we utilize the harmonic mean of $groupSatO$ and $groupDisO$, namely their F-score. Considering the input functions that F-score needs, we use $1 - groupDisO$ to simulate the group agreement. $F\text{-}score = 2\frac{groupSatO*(1-groupDisO)}{groupSatO+(1-groupDisO)}$.

To further analyze the methods' quality, we use the Normalized Discounted Cumulative Gain (NDCG), where we want the best possible items for a user to appear with the highest ranking possible in the group recommendations: $NDCG(u_i, G, j) = \frac{DCG(u_i,G,j)}{IDCG(u_i,G)}$, with $DCG(u_i, G, j) = \sum_{d_z \in Gr_j} \frac{|d_z \cap A(u_i,j)|}{log_2(r(d_z,Gr_j)+1)}$, where $r(d_z.Gr_j)$ is the rank of item d_z in the group recommendation list Gr_j. IDCG is the best possible outcome for user u_i. We supplement the results from NDCG, with the Discounted First Hit (DFH) metric that counts if a user has seen an item that is highly relevant to him/her in the early ranks of the group recommendation list: $DFH(u_i, Gr_j) = \frac{1}{log_2(fh+1)}$, where fh is the rank of the first item that appears both in the group recommendation list and the individual preference list of the user. Since NDCG and DFH refer to one user, we apply them to the group by computing them for each member and then taking the average over them. We do this at the end of each iteration.

We find similarities between users by utilizing the Pearson Correlation. We consider two users as similar if the similarity is greater than a threshold and if they have rated more than 5 identical items. Due to the difference in each dataset's semantics, the similarity threshold we set for the MovieLens dataset is 0.8, and for the GoodReads dataset is 0.7. To predict preference scores for a

user, we use the Weighted Sum of Others Ratings [36], and take into account only the top 100 most similar users (denoted as P_{u_i}) to him/her: $p(u_i, d_z) = \bar{r}_{u_i} + \frac{\sum_{u_l \in (P_{u_i} \cap U(d_z))} s(u_i,u_l)(r(u_l,d_z) - \bar{r}_{u_l})}{\sum_{u_l \in (P_{u_i} \cap U(d_z))} |s(u_i,u_l)|}$. We recommend to the group the 10 items with the highest group preference score that are not previously recommended.

For the ephemeral groups, we simultaneously perform all aggregation methods with the same group. Since the members' exit and entry is done randomly, this ensures that the experiment is as fair as possible for all different aggregation methods.

We consider six different aggregation methods.

- **Avg** is the classic Average aggregation method.
- **RP80 [1]** combines group relevance and group disagreement scores. As group relevance it utilizes the average method and as a group disagreement method it uses the Average Pair-wise Disagreement: $dis(d_z, G) = \frac{2}{|G|(|G|-1)} \sum(|p(u_i, d_z) - p(u_l, d_z)|)$, where $u_i, u_l \in G$, and $u_i \neq u_l$. The Average Pair-wise Disagreement reflects the degree of consensus in the relevance scores for d_z among group members. The final score for item d_z for the group G is: $\mathcal{F}(d_z, G) = (1 - w) * avg(d_z, G) + w * dis(d_z, G)$. We choose to set variable w to 0.8, since it performs better based on the experiments presented in [1].
- **Par [39]** generates a group recommendation list by incrementally adding to it items that have the best combination of Social Welfare (SW) and Fairness (F) scores. The Social Welfare of an item is the average satisfaction of all members for that item. As Fairness it employs the variance over the member's satisfaction. The final score of an item for the group is: $\mathcal{F}(d_z, G) = \lambda * SW(d_z, G) + (1 - \lambda) * F(d_z, G)$, where $\lambda = 0.8$ according to [39].
- **SDAA** is the Sequential Dynamic Adaptation Aggregation method proposed in this work.
- **SIAA** is the Sequential Individual Adaptation Aggregation method proposed in this work.
- **Avg+** is the Average+ Aggregation method proposed in this work.

8.2 Experimental Results

In Figures 1, 2 and 3, we show the group satisfaction and group disagreement scores for the varying group types for 15 iterations of the system. We want to achieve high group satisfaction and low group disagreement scores. In Tables 2 to 4, we depict their corresponding F-scores. In Tables 5 and 6 we present the NDCG and DFH scores for each aggregation method, for both stable and ephemeral groups, in the MovieLens and GoodReads datasets. We calculate the average scores for all iterations of the system for each aggregation method.

8.2.1 Stable Groups

For the stable groups, we see from the overall performance of the methods (F-scores) that the SDAA method outperforms the classic Average method for all group types. SDAA, SIAA, and Average+ are the best three performing methods for all group types for both datasets.

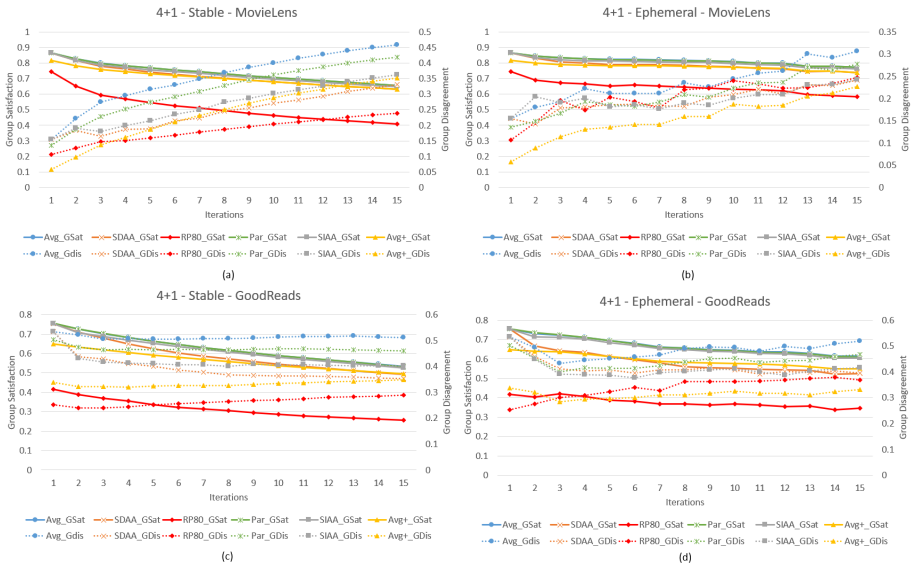

Fig. 1 Group Satisfaction and Disagreement for all aggregation methods for 15 iterations for stable and ephemeral groups, using MovieLens and GoodReads.

4+1 Groups / MovieLens Dataset												
	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
1	0.855	0.855	0.812	*0.865*	0.855	**0.875**	0.855	0.855	0.812	*0.865*	0.855	**0.875**
5	0.724	**0.774**	0.660	0.746	*0.771*	0.770	0.805	0.802	0.716	*0.817*	0.815	**0.821**
10	0.649	**0.704**	0.585	0.671	*0.697*	0.695	0.781	0.783	0.690	0.791	**0.798**	*0.792*
15	0.594	**0.650**	0.532	0.616	*0.643*	0.640	0.732	0.748	0.657	0.746	**0.760**	*0.756*
4+1 Groups / GoodReads Dataset												
	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
1	0.576	0.576	0.535	*0.599*	0.576	**0.655**	0.576	0.576	0.535	*0.599*	0.576	**0.655**
5	0.567	0.612	0.465	0.592	*0.619*	**0.631**	0.613	0.602	0.493	0.635	*0.646*	**0.655**
10	0.533	0.587	0.412	0.559	*0.588*	**0.595**	0.567	0.572	0.466	0.591	*0.611*	**0.622**
15	0.510	*0.559*	0.378	0.536	*0.559*	**0.562**	0.539	0.561	0.446	0.569	*0.593*	**0.602**

Table 2 F-score for all aggregation methods for both stable and ephemeral groups. With **bold** is the best value and with *italics* the second best.

In the MovieLens dataset, for the more homogeneous group types (4+1 and 3+2), SDAA performs slightly better in the latter iterations, with our new proposed methods SIAA and Average+ being a close second. The differences in performance between SDAA, SIAA, and Average+ are very low for all group types. This especially apparent in the latter iterations, where the differences in the scores are around 2%.

For the GoodReads dataset, Average+ is the most effective method for all group types. Average+ offers the best group disagreement scores after the RP80 method. In contrast, in the MovieLens dataset, SDAA performs better in group disagreement than Average+. At the same time, for both datasets, the group satisfaction scores for both methods are comparable, with SDAA clearly outperforming Average+ in the early iterations. This discrepancy in the group disagreement values' performance allows Average+ to be constantly better than the rest of the methods in the GoodReads dataset. The discrepancy of performance between the

two datasets is because the GoodReads dataset is more sparse than the MovieLens dataset. This negatively affects the single recommender system and reduces the number of items that are equally good for all members. Subsequently, SDAA tends to use the Least Misery approach, while Average+ considers the entire group recommendation list and overcomes the sparse obstacle.

In more detail, we can see from Figures 1a, 1c, 2a, 2c, 3a and 3c that depict the stables groups, that Average and Par offer best group satisfaction scores but the worst group disagreement scores. The high group disagreement scores culminate in the Average's low overall performance. The Par method suffers from the same problem as Average. This is understandable since the Par method focuses more on the group satisfaction rather than disagreement. In the opposite case, the best disagreement scores are achieved by RP80, which mainly focuses on finding items with low disagreement, but at the same time, it also produces the worst group satisfaction scores. SIAA achieves the same group satisfaction as the Average but also produce lower group disagreement scores. Thus, out-performing Average.

For Average+, we can observe lower group satisfaction scores. This is expected since we build the group recommendation list based on an expanded list provided by Average aggregation. We do not limit ourselves on the items that are going to provide the best group satisfaction score (those are the top k returned by Average) but also consider items that may offer lower satisfaction but considerable better group disagreement scores. Consequently, the satisfaction scores for Average+ are lower than the scores of the Average method.

8.2.2 Ephemeral Groups

As we can observe in the F-score tables, the aggregation methods' performance vastly differs in the ephemeral groups' scenario. Overall, Average+ and SIAA are the best for all cases, with the difference between them to be around 8% in the early iterations and dropping to 1% in the latter ones.

We can observe a discrepancy on the performance of the SDAA method. The constant change of members in the group has an adverse effect on the SDAA method, with its performance dropping drastically. The addition of a new unknown member at each iteration forces the SDAA to utilize higher α values. The new member has no previous satisfaction score, which in turn makes the value of α to be the highest possible at each iteration (Equation 9). SDAA does not perform well with high α values [35]. With new members repeatably being added to the group, SDAA cannot keep an optimal performance and is outperformed by the classic Average method.

In the early iterations, Par has comparable results as SIAA and Average+, but in latter iterations the difference becomes greater. This is because in the latter iterations, with a high probability the most relevant items have already been suggested to the group (and cannot be suggested again, per our experimental scenario), so the old members are disadvantaged. Their items were suggested in the previous iterations and the interests of the new members are more likely to be recommended. Additionally, the Par method only considers the current iteration, without taking into account if a member was not satisfied in any of the previous iterations. This culminates to the high disagreement scores, observed in the ephemeral group scenario.

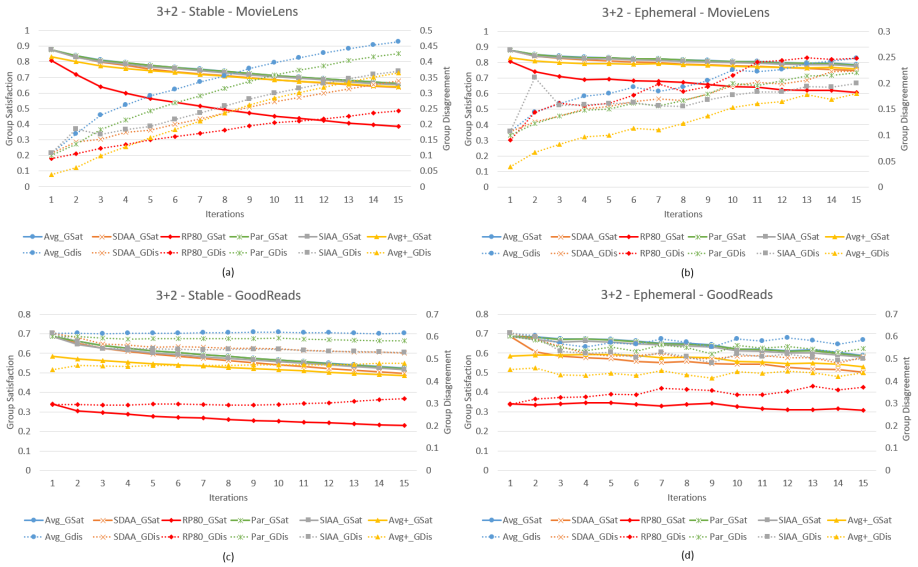

Fig. 2 Group Satisfaction and Disagreement for all aggregation methods for 15 iterations for stable and ephemeral groups, using MovieLens and GoodReads.

3+2 Groups / MovieLens Dataset												
	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
1	0.885	0.885	0.855	*0.889*	0.886	**0.891**	0.885	0.885	0.855	*0.889*	0.886	**0.891**
5	0.743	0.784	0.678	0.767	*0.787*	**0.790**	0.826	0.827	0.759	*0.840*	0.831	**0.844**
10	0.654	**0.706**	0.575	0.676	0.702	*0.700*	0.790	0.791	0.708	0.803	**0.810**	*0.809*
15	0.592	**0.648**	0.510	0.614	*0.641*	0.637	0.768	0.762	0.673	*0.782*	**0.787**	**0.787**
3+2 Groups / GoodReads Dataset												
	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
1	0.493	0.493	0.458	*0.506*	0.493	**0.565**	0.493	0.493	0.458	*0.506*	0.493	**0.565**
5	0.473	0.510	0.398	0.490	*0.519*	**0.537**	0.522	0.526	0.454	0.537	*0.548*	**0.578**
10	0.455	0.493	0.371	0.473	*0.502*	**0.521**	0.495	0.515	0.436	0.516	*0.538*	**0.558**
15	0.444	0.486	0.343	0.466	*0.493*	**0.502**	0.487	0.498	0.413	0.512	*0.537*	**0.546**

Table 3 F-score for all aggregation methods for both stable and ephemeral groups. With **bold** is the best value and with *italics* the second best.

5 Dissimilar Groups / MovieLens Dataset												
	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
1	0.882	0.882	0.846	*0.891*	0.882	**0.893**	0.882	0.882	0.846	*0.891*	0.882	**0.893**
5	0.745	0.774	0.641	0.766	*0.782*	**0.789**	0.824	0.829	0.751	*0.837*	0.831	**0.843**
10	0.653	0.695	0.522	0.674	*0.696*	**0.700**	0.793	0.795	0.699	0.806	*0.808*	**0.815**
15	0.591	**0.637**	0.466	0.611	0.633	*0.636*	0.759	0.752	0.643	0.774	*0.781*	**0.784**
5 Dissimilar Groups / GoodReads Dataset												
	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
1	0.755	0.755	0.698	*0.773*	0.755	**0.793**	0.755	0.755	0.698	*0.773*	0.755	**0.793**
5	0.639	0.684	0.526	0.660	*0.687*	**0.696**	0.682	0.690	0.561	0.699	*0.714*	**0.716**
10	0.580	0.627	0.445	0.603	*0.632*	**0.637**	0.605	0.613	0.487	0.629	*0.653*	**0.655**
15	0.546	0.589	0.401	0.568	*0.592*	**0.595**	0.566	0.574	0.445	0.589	*0.608*	**0.615**

Table 4 F-score for all aggregation methods for both stable and ephemeral groups. With **bold** is the best value and with *italics* the second best.

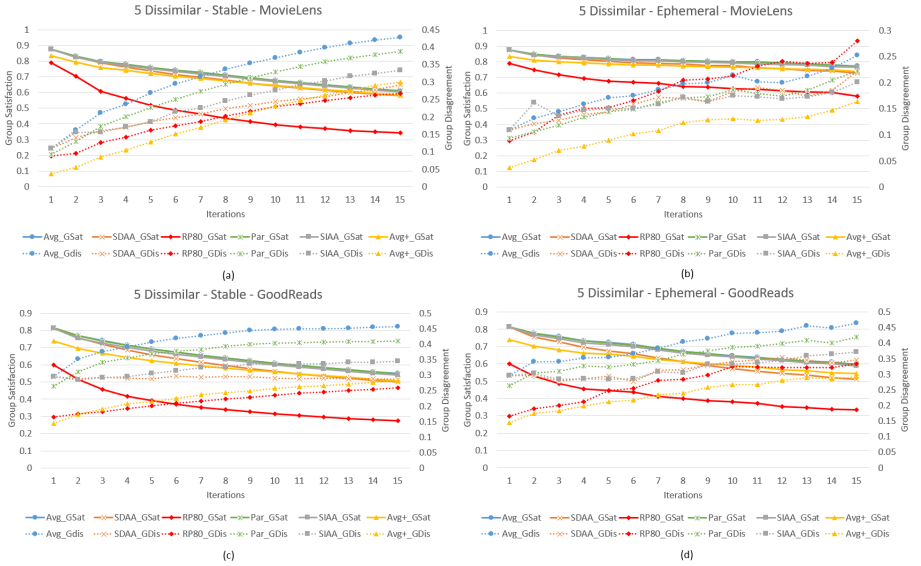

Fig. 3 Group Satisfaction and Disagreement for all aggregation methods for 15 iterations for stable and ephemeral groups, using MovieLens and GoodReads.

In Figures 1b, 1d, 2b, 2d, 3b and 3d, we examine the performance of the methods in more detail. As in the case with the stable groups, the best group satisfaction scores are achieved by the Average and Par methods and the best group disagreement scores by Average+. In contrast to the stable groups scenario, Average+ now achieves better disagreement scores than RP80.

Although in the early iterations, Average+ has lower satisfaction scores than the rest of the methods except RP80, in the latter ones, it manages to outperform SDAA. However, it is considerably below both Average, Par and SIAA. SIAA has maintained exemplary group satisfaction scores throughout all the scenarios, but with ephemeral groups, it can now achieve good disagreement scores. This is due to the user-focused concept of SIAA. It can cope better with the added new group member than a group-focused method like SDAA.

8.2.3 Stable Versus Ephemeral Groups

In general, all aggregation methods perform better overall for ephemeral groups rather than the stable ones. This can be seen in the latter iterations when comparing the ephemeral and the stable groups scenarios. This is expected, since our group recommender system does not suggest items to the group that it has already suggested before. In the stable groups, the best items for each member were probably suggested in the early iterations. Now that the members interchange, the system is able to suggest to the new members items that are the best for them.

This can be further corroborated by the satisfaction scores shown in the figures. For the ephemeral groups, we can observe that the group satisfaction scores tend to remain high throughout the iterations, without the notable slope noted in the stable groups. Furthermore, in correlation to the high satisfaction scores,

the disagreement scores are again lower for the ephemeral groups than the stable ones. The new interests that the added members represent in the sequential group recommendation help the system maintain high performance throughout many iterations.

We can also observe that the stable groups' disagreement scores tend to increase uniformly as the satisfaction scores drop. This again represents that the group recommender system achieves increasingly lower quality suggestions for the group over the passage of the iterations. Meanwhile, for the ephemeral groups, we can see high and low spikes in the disagreement scores. These spikes represent the new group members and how their inclusion slows down the overall disagreement scores' deterioration.

8.2.4 Methods Quality

In Tables 5 and 6, we show the NDCG and DFH scores for each aggregation method, for the various test conditions. For both stable and ephemeral groups and both datasets, SDAA is the best performing, as far as NDCG scores, while SIAA is the second best. Average+ has low NDCG scores since we build the group recommendation list with items that can have lower relevance to the group. It is able to report back items that are relevant (it generates good satisfaction scores), but most of them are not the best for each user; thus, it has lower NDCG scores. Once again, we observe a difference in the quality of the methods when they are applied to stable and ephemeral groups. These results supplement the results from before, where the methods perform better in the ephemeral group scenario.

In contrast to NDCG scores, Average+ has the best scores for DFH. This means that Average+ succeeds in proposing at least one item in the group recommendation highly relevant to each member. However, the rest of the list is composed of unsatisfactory items (in terms of relevance), hence the low NDCG scores. On the other hand, the DFH scores for SDAA are relatively low. This indicates that SDAA, although able to propose many items relevant to the members, high NDCG scores, cannot suggest the best items for these users.

8.2.5 Fluctuating Group Size

In this final test case, we want to observe the performance of each aggregation method, when the size of the group is not fixed. We examine the 4+1 group format. We want to keep this group format for the duration of the experiment i.e., for all 15 iterations. Since the group size is not fixed, we require that at each iteration 20% of the group members be dissimilar to the rest. For example, when the group size is 7, then 6 members should be similar to each other and 1 is dissimilar to them (20% of 7 is 1.4, which gives us 1 dissimilar user when rounding down). If in the next iteration we add another user to the group, then that user will have to be dissimilar to the 6 others (20% of 8 is 1.6, which gives us 2 dissimilar users when rounding up).

With equal probability we enact one of the three scenarios:

- **Nothing:** No new members will enter or exit the group.
- **Insert:** Insert a new member to the group.
- **Remove:** Remove a member from the group.

MovieLens Dataset / Stable Groups												
	NDCG						DFH					
	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
4+1	0.042	**0.044**	0.020	0.042	*0.043*	0.034	0.894	0.861	0.803	*0.906*	0.889	**0.933**
3+2	0.042	**0.044**	0.020	*0.043*	*0.043*	0.035	0.905	0.867	0.809	*0.918*	0.898	**0.939**
5 Dis	0.041	**0.042**	0.017	*0.042*	0.041	0.034	0.892	0.838	0.780	*0.909*	0.885	**0.934**
GoodReads Dataset / Stable Groups												
	NDCG						DFH					
	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
4+1	0.055	**0.062**	0.014	*0.057*	*0.057*	0.045	0.829	0.759	0.696	*0.847*	0.826	**0.887**
3+2	0.053	**0.057**	0.013	*0.055*	*0.055*	0.042	0.784	0.742	0.607	*0.796*	0.782	**0.829**
5 Dis	0.057	**0.062**	0.018	*0.058*	*0.058*	0.047	0.861	0.777	0.766	*0.877*	0.859	**0.919**

Table 5 NDCG and DFH scores for the stable groups. With **bold** is the best value and with *italics* the second best.

MovieLens Dataset / Ephemeral Groups												
	NDCG						DFH					
	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
4+1	0.059	**0.062**	0.033	0.059	*0.060*	0.043	0.972	0.951	0.927	*0.976*	0.967	**0.981**
3+2	0.055	**0.058**	0.032	0.055	*0.056*	0.039	0.980	0.957	0.943	*0.985*	0.975	**0.986**
5 Dis	0.052	**0.055**	0.030	0.052	*0.053*	0.038	0.977	0.956	0.926	*0.983*	0.976	**0.986**
GoodReads Dataset / Ephemeral Groups												
	NDCG						DFH					
	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
4+1	0.069	**0.080**	0.025	*0.070*	*0.070*	0.053	0.898	0.828	0.769	*0.908*	0.895	**0.928**
3+2	0.066	**0.078**	0.023	0.066	*0.067*	0.050	0.846	0.782	0.681	*0.854*	0.842	**0.875**
5 Dis	0.063	**0.072**	0.025	0.047	*0.065*	0.050	0.907	0.842	0.820	*0.915*	0.902	**0.939**

Table 6 NDCG and DFH scores for the ephemeral groups. With **bold** is the best value and with *italics* the second best.

	MovieLens	GoodReads
Nothing	36.58%	35.45%
Insert	34.99%	35.98%
Remove	28.43%	28.57%

Table 7 The percentage of time each scenario was enacted for all 100 groups.

In Table 7, we show the number of times each scenario was enacted for all 100 groups in the two datasets. In Figure 4, we show the group satisfaction and disagreement scores for the 15 iterations, and in Table 8 the average values after 15 iterations for the F-Score, NDCG and DFH.

We can observe the same trend as the previous results in Figures 1b and 1d. Meaning that the Average and Par methods offer the best group satisfaction scores and simultaneously the worst group disagreement scores. SIAA performs comparable good in satisfaction to Average and Par but has better group disagreement scores and finally Average+ has noticeable lower satisfaction scores but far better disagreement scores.

This is further corroborated by the values in Table 8, where both SIAA and Average+ have the best F-Score. The high disagreement scores that Par produces lower its overall performance. SDAA identifies more relevant items to the group in terms of quantity, as indicated by the highest NDCG score, but under-performs in terms of quality with low DFH scores.

The effect of the fluctuating group size can be observed in the lower group satisfaction scores and the higher group disagreement scores. If we again compare these results to Figures 1b and 1d, we can see that all methods perform slightly worse when the size of the group is not fixed. This is an expected result. As it

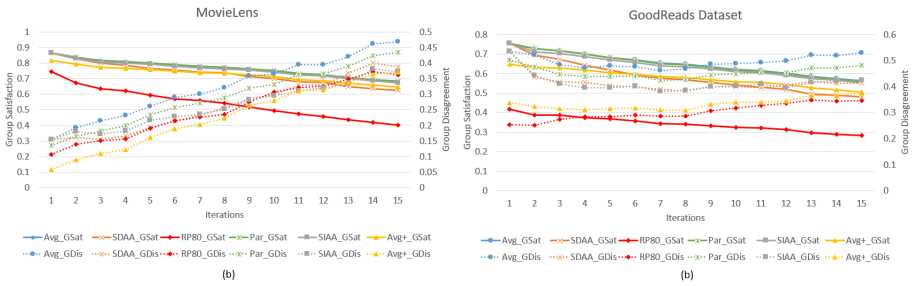

Fig. 4 Group Satisfaction and Disagreement for all aggregation methods for 15 iterations for fluctuated group size, using MovieLens and GoodReads.

	MovieLens			GoodReads		
	F-Score	NDCG	DFH	F-Score	NDCG	DFH
Avg	0.719	*0.052*	0.963	0.567	0.062	0.915
SDAA	0.734	**0.055**	0.920	0.579	**0.069**	0.827
RP80	0.629	0.025	0.801	0.459	0.020	0.710
Par	*0.736*	*0.052*	*0.975*	0.591	*0.064*	*0.929*
SIAA	**0.749**	*0.052*	0.963	*0.608*	0.063	0.914
Avg+	**0.749**	0.039	**1.000**	**0.618**	0.050	**0.969**

Table 8 Average scores after 15 iterations. With **bold** is the best value and with *italics* the second best.

has been shown in the literature, the size of the group has an adverse effect in the performance of a group recommender system. The more members a group has the more difficult is for the system to find items that are relevant to all members. In Table 7, we show that in most cases we add a member to the group, which results in groups bigger than 5; the size used in the previous experiments.

9 Related Work

Group Recommendations. One of the most popular methodology to achieve group recommendations is to apply a standard recommendation algorithm to each group member individually and aggregate their lists into one. During the aggregation phase, many criteria can be taken into account. [41] proposes a group recommendation model that considers the influence that each member has on the group's final choice, stating that a member has more influence on the group if he/she is more knowledgeable about the recommended items. In our work, we propose that if a group member is more dissatisfied than the rest, then that member will have more influence in the group decision. [6] learns the aggregation strategy from data, which is based on the recent developments of attention network and neural collaborative filtering (NCF), while we dynamically change our proposed aggregation method based on the satisfaction of the group members. [40] also utilizes an attention mechanism as well as a Bipartite Graph Embedding Model (BGEM) to learn the influence that each member has on the group decision. [31] exploits the social interactions between the group members, via a preference-oriented social network, in order to make group decisions without complete knowledge of the group members' preferences. In our work, we require the full knowledge of the

group members' preferences. [37] learns the aggregation strategy based on the interactions between group members. To simulate how a group consensus is reached, they model these interactions as multiple voting processes, and proposed a stacked social self-attention network to learn the voting scheme of group members. [25] offers a novel approach to producing recommendations for a large group of people by dividing the big group into different interest subgroups. For each subgroup, they find a potential candidate set of media-user pairs and finally aggregate the CF produced recommendation lists for each pair.

Recently, there is also some research work on ephemeral groups. [28] determines the preference of a group that meets for the first time by combining the group members' individual preferences based on their contextual influence, where the contextual influence represents the ability of a member to guide the decisions of the group. [18] proposes a probabilistic model, namely the personal impact topic (PIT) model. An individual member's preference is modeled as a distinct distribution over the system's topics. Each topic is expressed as a distinct distribution over all the items known to the system. We explicitly use the users' profiles (ratings) to provide the group with a group list.

Fairness in Group Recommendations. Some of the first approaches to achieving fairness [24, 23] in group recommendations, exploit ideas from voting theory. For example, [20] assumes that the recommender has probabilistic knowledge about the distribution of users' ratings, and through voting theory recommends to the group a "winning" item. From a different perspective, [9] proposes a group recommendation method by allowing a group member to comment on the choices of the rest of the group. This allows each user to get new recommendations similar to the proposals made by other group members and to communicate the rationale behind their own counter-proposals.

More recently, [32] proposes two definition of fairness: *fairness proportionality* and *envy-freeness*. In the former, the user u considers the list of recommended items fair for him/her, if there are at least m items that the user likes. In the latter, u considers the package fair, if there are at least m items for which the user does not feel envious. [39] presents yet another definition of fairness. It defines a utility score for each group member based on how relevant are the recommended items to them. Then, it models fairness as a proximity of how balanced the utilities of users are when group recommendations are given. [30] defines a user's utility to be the similarity between the user's individual and the group recommendation lists. They construct the group recommendation list by considering sets of N-level Pareto optimal items. [15] proposes the notion of rank-sensitive balance in order to achieve fairness during the aggregation phase. All these works, for achieving fairness, consider one instance of group recommendations and do not take into account the sequential group recommendation problem, as we do in our work.

Sequential Recommendations. In general, there are three categories of sequential recommenders, and they are divided based on how many past user interactions they consider: *Last-N interactions-based recommendations, Session-based recommendations* and *Session-aware recommendations* [26]. In the first approach, only the last N user actions are considered [7, 16, 17]. This is because the system has logged a huge amount of historical data for the user, with many of them be duplicates, which do not offer relevant information to the system. In session-based recommendations, only the last interaction of the user with the system is used. They are typically found in news [8] and advertisement [13] recommenders. In

the last category, we have information about both the last interaction of the user with the system, as well as the history of the user. These recommenders are often implemented in e-commerce or for app suggestions [11, 14, 27]. In our work, we use the last method to approach sequential group recommendations. [10] proposes another session-aware system for music recommendations. It uses a neural network architecture that models users' preferences as a sequence of embeddings, one for each session, suggesting that the user's recent selections and the session-level contextual variables (such as time and device used) are enough to predict the tracks a user will listen to. [3] presents a multi-round recommender system using Variational Autoencoders (VAEs) and tries to achieve fairness during multiple rounds by introducing randomness in the regular operation of VAE, while [5, 4] penalize scores given to items according to historical popularity for mitigating the bias and promoting diversity in the results. The above works have been done for single user recommenders, and to our knowledge, our work is the first one in sequential group recommendations that handles the notion of fairness.

This article extends [35], where we first presented SDAA. We now examine two more aggregation methods, SIAA and Average+, designed to deal with sequential group recommendations. Instead of focusing only on the group, like with SDAA, now with SIAA, we focus more on the user side. We also examine a heuristic method, Average+, where we try to find the best possible items to offer to the group based on the satisfaction and disagreement score they generate. In addition, we also consider the ephemeral group scenario, where at the end of each iteration, a group member leaves the group, and a new one enters the group.

10 Conclusions

In this paper, we introduce the notion of sequential group recommendations. We treat the single user recommender system as a black-box, and focus on aggregating the individual group members' recommendation lists into a group list. We propose three new methods for aggregating group members' recommendation lists. Specifically, SDAA focuses on the group as a whole, and dynamically, based on the degree of users' satisfaction, balances the advantages of the average and least misery methods. With SIAA, we concentrate on singular members, focus on their needs, and assign personal weights. Finally, we propose the Average+ method that takes advantage of the satisfaction scores that a classic average aggregation can produce and mitigates the high disagreement scores by incrementally building the group list with the best possible item. These aggregation methods focus on the sequential aspect of the group recommendation process and how we can achieve fairness for all group members in multiple recommendation rounds. We evaluate the proposed methods using two large real datasets, MovieLens and GoodReads, and test the methods for three different group types for both stable and ephemeral groups. We experimentally show the effectiveness of our methods. SDAA outperforms in the latter iterations for the stable group scenario. However, in the ephemeral groups, SDAA cannot perform optimally primarily due to the users' transitory attitude. In contrast, SIAA and Average+ are the best for the ephemeral groups and among the best in the stable groups' scenario.

References

1. Amer-Yahia, S., Roy, S.B., Chawlat, A., Das, G., Yu, C.: Group recommendation: Semantics and efficiency. PVLDB **2**(1), 754–765 (2009)
2. Baltrunas, L., Makcinskas, T., Ricci, F.: Group recommendations with rank aggregation and collaborative filtering. In: RecSys (2010)
3. Borges, R., Stefanidis, K.: Enhancing long term fairness in recommendations with variational autoencoders. In: MEDES (2019)
4. Borges, R., Stefanidis, K.: On measuring popularity bias in collaborative filtering data. In: Proceedings of the Workshops of the EDBT/ICDT 2020 Joint Conference, Copenhagen, Denmark, March 30, 2020, vol. 2578 (2020)
5. Borges, R., Stefanidis, K.: On mitigating popularity bias in recommendations via variational autoencoders. In: SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021, pp. 1383–1389. ACM (2021)
6. Cao, D., He, X., Miao, L., An, Y., Yang, C., Hong, R.: Attentive group recommendation. In: SIGIR (2018)
7. Cheng, C., Yang, H., Lyu, M.R., King, I.: Where you like to go next: Successive point-of-interest recommendation. In: International Joint Conference on Artificial Intelligence (2013)
8. Garcin, F., Dimitrakakis, C., Faltings, B.: Personalized news recommendation with context trees. CoRR **abs/1303.0665** (2013)
9. Guzzi, F., Ricci, F., Burke, R.: Interactive multi-party critiquing for group recommendation. In: RecSys (2011)
10. Hansen, C., Hansen, C., Maystre, L., Mehrotra, R., Brost, B., Tomasi, F., Lalmas, M.: Contextual and sequential user embeddings for large-scale music recommendation. In: RecSys (2020)
11. Hariri, N., Mobasher, B., Burke, R.: Context-aware music recommendation based on latenttopic sequential patterns. In: RecSys (2012)
12. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst. **5**(4), 19:1–19:19 (2015)
13. Hidasi, B., Quadrana, M., Karatzoglou, A., Tikk, D.: Parallel recurrent neural network architectures for feature-rich session-based recommendations. In: RecSys (2016)
14. Jannach, D., Lerche, L., Jugovac, M.: Adaptation and evaluation of recommendations for short-term shopping goals. In: RecSys (2015)
15. Kaya, M., Bridge, D., Tintarev, N.: Ensuring fairness in group recommendations by rank-sensitive balancing of relevance. In: RecSys (2020)
16. Lian, D., Zheng, V.W., Xie, X.: Collaborative filtering meets next check-in location prediction. In: WWW (2013)
17. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: A recurrent model with spatial and temporal contexts. In: AAAI Conference on Artificial Intelligence (2016)
18. Liu, X., Tian, Y., Ye, M., Lee, W.C.: Exploring personal impact for group recommendation. In: RecSys (2012)
19. Machado, L., Stefanidis, K.: Fair team recommendations for multidisciplinary projects. In: WI (2019)
20. Naamani Dery, L., Kalech, M., Rokach, L., Shapira, B.: Iterative voting under uncertainty for group recommender systems. In: RecSys (2010)
21. Ntoutsi, E., Stefanidis, K., Nørvåg, K., Kriegel, H.: Fast group recommendations by applying user clustering. In: ER (2012)
22. Ntoutsi, E., Stefanidis, K., Rausch, K., Kriegel, H.: "strength lies in differences": Diversifying friends for recommendations through subspace clustering. In: CIKM (2014)
23. Pitoura, E., Koutrika, G., Stefanidis, K.: Fairness in rankings and recommenders. In: Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020, pp. 651–654 (2020)
24. Pitoura, E., Stefanidis, K., Koutrika, G.: Fairness in rankings and recommendations: An overview. CoRR **abs/2104.05994** (2021). URL https://arxiv.org/abs/2104.05994
25. Qin, D., Zhou, X., Chen, L., Huang, G., Zhang, Y.: Dynamic connection-based social group recommendation. IEEE TKDE (2018)
26. Quadrana, M., Cremonesi, P., Jannach, D.: Sequence-aware recommender systems. ACM Comput. Surv. **51**(4), 66:1–66:36 (2018)

27. Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: Personalizing session-based recommendations with hierarchical recurrent neural networks. In: RecSys (2017)
28. Quintarelli, E., Rabosio, E., Tanca, L.: Recommending new items to ephemeral groups using contextual user influence. In: RecSys (2016)
29. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: CSCW (1994)
30. Sacharidis, D.: Top-n group recommendations with fairness. SAC '19. Association for Computing Machinery, New York, NY, USA (2019). DOI 10.1145/3297280.3297442. URL https://doi.org/10.1145/3297280.3297442
31. Salehi-Abari, A., Boutilier, C.: Preference-oriented social networks: Group recommendation and inference. RecSys '15. Association for Computing Machinery, New York, NY, USA (2015). DOI 10.1145/2792838.2800190. URL https://doi.org/10.1145/2792838.2800190
32. Serbos, D., Qi, S., Mamoulis, N., Pitoura, E., Tsaparas, P.: Fairness in package-to-group recommendations. In: WWW (2017)
33. Stratigi, M., Kondylakis, H., Stefanidis, K.: Fairness in group recommendations in the health domain. In: ICDE (2017)
34. Stratigi, M., Kondylakis, H., Stefanidis, K.: Fairgrecs: Fair group recommendations by exploiting personal health information. In: DEXA (2018)
35. Stratigi, M., Nummenmaa, J., Pitoura, E., Stefanidis, K.: Fair sequential group recommendations. In: ACM SAC (2020)
36. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence **2009** (2009)
37. Vinh Tran, L., Nguyen Pham, T.A., Tay, Y., Liu, Y., Cong, G., Li, X.: Interact and decide: Medley of sub-attention networks for effective group recommendation. SIGIR'19. Association for Computing Machinery, New York, NY, USA (2019). DOI 10.1145/3331184.3331251. URL https://doi.org/10.1145/3331184.3331251
38. Wan, M., Misra, R., Nakashole, N., McAuley, J.: Fine-grained spoiler detection from large-scale review corpora (2019)
39. Xiao, L., Min, Z., Yongfeng, Z., Zhaoquan, G., Yiqun, L., Shaoping, M.: Fairness-aware group recommendation with pareto-efficiency. In: RecSys (2017)
40. Yin, H., Wang, Q., Zheng, K., Li, Z., Yang, J., Zhou, X.: Social influence-based group representation learning for group recommendation. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 566–577 (2019). DOI 10.1109/ICDE.2019.00057
41. Yuan, Q., Cong, G., Lin, C.Y.: Com: A generative model for group recommendation. In: KDD (2014)