Recommending Unanimously Preferred Items to Groups

Karim Benouaret
LIRIS – CNRS, Université Claude Bernard Lyon 1
Villeurbanne, France
karim.benouaret@liris.cnrs.fr

Kian-Lee Tan
National University of Singapore
Singapore
tankl@comp.nus.edu.sg

ABSTRACT

Due to the pervasiveness of group activities in people's daily life, group recommendation has attracted a massive research effort in both industry and academia. A fundamental challenge in group recommendation is how to aggregate the preferences of group members to select a set of items maximizing the overall satisfaction of the group; this is the focus of this paper. Specifically, we introduce a dual adjustment aggregation score, which measures the relevance of an item to a group. We then propose a recommendation scheme, termed k-dual adjustment unanimous skyline, that seeks to retrieve the k items with the highest score, while discarding items that are unanimously considered inappropriate. Furthermore, we design and develop algorithms for computing the k-dual adjustment unanimous skyline efficiently. Finally, we demonstrate both the retrieval effectiveness and the efficiency of our approach through an extensive experimental evaluation on real datasets.

1 INTRODUCTION

Nowadays, an increasing number of online businesses are using recommendation systems. Recommender systems represent a powerful opportunity for customers and service providers. Recommender systems make it easier for customers to find items that interest them. On the other hand, service providers can increase their profit. Current recommendation engine use cases include Amazon, recommending products; Netflix, recommending movies; and TripAdvisor, recommending tourist tips on restaurants, hotels, and more.

In many real-world settings, there is a need to recommend items for a group of users rather than a single user, e.g., department members going out to a festive dinner or friends planning to go to a movie. In addition, with the success of social networking, there is a recently emerging trend, where people form groups (most likely of large size) and participate together in activities. This calls for effective techniques for group recommendation.

Research on group recommendation discerns two main kinds of groups: persistent and occasional (or ephemeral) groups [37]. Persistent groups refer to groups with consistent structure and a rich historical information about the group-item interactions [21, 38], in this case, each group can be treated as a virtual user and personalized recommendation techniques are applied. In contrast, in occasional groups, only the user-item interactions are available [3, 4], and recommendations must be done on this basis, i.e., by aggregating the user-item interactions. Note that recently, hybrid approaches are proposed [11, 44, 45] to make use of both user-item and group-item interactions, specially to handle the case where only a poor set of group-item interactions is available. In this work, we focus on making recommendation to occasional groups.

Table 1: User-Restaurant Ratings

Restaurant	User			AVG Rating
	u_1	u_2	u_3	
ι_1	8	9	6	7.67
ι_2	10	7	5.5	7.50
ι_3	7.5	6.5	8	7.33
ι_4	8	8.75	6	7.58
ι_5	9	6	2	5.67
ι_6	6	5	3	4.67

The main challenge in developing an effective group recommender system is handling the conflicting preferences of the group members. Particularly, group members may have different ratings for a given item, either because they have different tastes or because ratings are subjective, and some users tend to give good ratings for most items, while others are harsh critics and strict, which current approaches fail to fully capture.

To better illustrate this issue, consider the following example. Suppose that three friends in Singapore are looking for a restaurant to arrange a dinner. Assume that the three friends are going to use a Web site (e.g., TripAdvisor, TheFork, etc.) to search and filter restaurants based on their preferences. Assume a list of available restaurants in Singapore, shown in Table 1. Each user (among the three friends) has some intrinsic rating for each of the restaurants, determined explicitly (by the user) or measured implicitly using personalized recommendation approaches such as collaborative filtering techniques [1].

To avoid users having to go through all the items, a typical group recommender system aggregates the individual preferences and returns the top-k items. Assume in our example that the recommendation engine employs the average aggregation strategy [4] and returns the top-2 restaurants. These supposed best restaurants are ι_1 and ι_4 (see Table 1). Comparing ι_1 and ι_4, observe that users u_1 and u_3 are indifferent, while user u_2 prefers item ι_1. That is, ι_1 is unanimously preferred over ι_4 and will be naturally selected by the group. In other words, including ι_4 in the result is not appropriate because it will not be chosen. It would therefore be interesting to replace ι_4 with another potentially relevant restaurant.

To do this, the natural option is to consider only the Pareto optimal set of items when producing the top-k answers. We refer to this set as the *unanimous skyline*, borrowing the terminology from the skyline operator [9], and it comprises all items that are not unanimously dominated. An item unanimously dominates another item, if the former has ratings as good as or better than the latter for all group members, and strictly better for at least one group member. In our example, restaurant ι_1 unanimously dominates restaurants ι_4 and ι_6. Likewise, restaurant ι_2 unanimously dominates restaurant ι_5. Restaurants ι_1, ι_2 and ι_3 are not unanimously dominated by any other item, therefore, they form the unanimous skyline.

Table 2: Utility Loss of Unanimous Skyline Restaurants

Restaurant	User			Total Loss
	u_1	u_2	u_3	
ι_1	20%	0%	25%	45%
ι_2	0%	22.22%	31.25%	53.47%
ι_3	25%	27.78%	0%	52.78%

Considering only the unanimous skyline restaurants, the top-2 query returns restaurants ι_1 and ι_2. However, observe that the users do not have the same rating scale. Users u_1 and u_2 tend to give good ratings and can achieve a utility of 10 and 9, respectively. In contrast, user u_3 is a harsh critic and can reach at best a utility of 8. Selecting ι_2 clearly favors user u_1 and deprives user u_3 of $^{(8-5.5)}/_8 = 31.25\%$ of this best utility. Table 2 shows the utility loss of each user as well as the total utility loss for the unanimous skyline restaurants. As can be seen, replacing restaurant ι_2 with restaurant ι_3 seems more interesting considering the users as a group regarding both the maximum utility loss per user and the total utility loss.

To deal with this shortcoming, the simplest strategy is to minimize either the maximum utility loss per user or the total utility loss. This interpretation, however, transposes to a kind of least misery aggregation strategy [4], whose main drawback is that unsatisfied users have a higher impact on the final decision than satisfied ones. Thus, in some settings, the recommendation engine would miss items that are expected to be highly liked by every group member except one [3]. It is thus important to set up a risk-benefit balance to preserve items that are highly liked by most of the group members while minimizing dissatisfaction among group members.

In this paper, we propose a dual adjustment aggregation. First, instead of considering the absolute rating of an item to determine its relevance to a given user, we introduce the notion of utility ratio, which measures how close/far this item is to the best rating of that user. This adjusts the individual utilities to address the fact that users rate differently. Second, for each item, we define a recommendation score, which adjusts its utility ratios so that items that are highly liked (resp. disliked) by most group members are preserved (resp. rejected). Then, to filter the candidate items, we propose a novel concept termed *k-dual adjustment unanimous skyline*, which comprises the k non-unanimously dominated items (i.e., belonging to the unanimous skyline) with the highest score.

The second challenge is how to compute the k-dual adjustment unanimous skyline efficiently. The computational complexity of the k-dual adjustment unanimous skyline is dominated by two factors: (1) the number of unanimous dominance checks, to test whether or not an item is unanimously dominated; and (2) the number of computed recommendation scores, to rank the unanimous skyline items.

A straightforward method to compute the k-dual adjustment unanimous skyline is to first compute the unanimous skyline using existing skyline computation algorithms, e.g., BNL [9], SFS [12, 13], SaLSa [5, 6], then compute the recommendation score of all items belonging to the unanimous skyline and return the top-k ones. However, this approach performs a large number of unanimous dominance checks, especially, in the presence of disagreements between users since for any item ι_i it is more likely there exists another item ι_j where ι_i and ι_j are better than each other over different users.

Another option is to first rank all items according to their recommendation scores and exploit the monotonicity nature of this score. The implication is that an item can only be unanimously dominated by items with better ranks. This is the idea behind the SFS algorithm [12, 13], to which we infuse a top-k layer to stop after retrieving the first k items that are not unanimously dominated. Even if this method reduces the number of unanimous dominance checks, it requires computing the score of all items and ranking them.

To optimize the extraction of the *k-dual adjustment unanimous skyline*, we go one step forward and propose a novel algorithm, which leverages effective pruning techniques minimizing both the number of unanimous dominance checks and the number of computed scores. To achieve this, the algorithm admits sorted lists that maintain the set of items in decreasing order of the ratings for each user. Using this scheme, the items that may unanimously dominate a given item are reduced to the part of unanimous skyline items located before it in a list. In addition, at each level, the algorithm creates a virtual item and uses it to compute an upper bound score for the items not examined, used as an early termination condition to avoid iterating through all items that are definitely not part of the result.

Furthermore, we perform a theoretical analysis to study the average computational complexity of the three algorithms.

Contributions. To recapitulate, the main contributions of this work can be summarized as follows.

- We propose a new group recommendation score, which takes into account the fact that users rate differently and preserves (resp. disfavors) items that are highly liked (resp. disliked) by most group members, using a dual adjustment aggregation;
- We introduce the notion of k-dual adjustment unanimous skyline, which consists of the k unanimously preferred items with the highest group recommendation score;
- We present two baseline methods by adapting prior work, and also propose a novel algorithm to efficiently compute the k-dual adjustment unanimous skyline;
- We perform a theoretical analysis to study the average computational complexity of the different algorithms;
- We conduct a comprehensive experimental study to evaluate our approach both in terms of retrieval effectiveness and efficiency, using real datasets.

The remainder of this paper is organized as follows. Section 2 formally defines the problem of k-dual adjustment unanimous skyline. Section 3 describes the k-dual adjustment unanimous skyline computation algorithms and their average time complexity. Section 4 presents our experimental study. Section 5 reviews the related work. Finally, Section 6 concludes the paper.

2 PROBLEM DESCRIPTION

In this section, we provide the basics used in this paper and formalize the notion of k-dual adjustment unanimous skyline. Table 3 lists and describes the frequently used symbols throughout this paper.

Let $\mathrm{I} = \{\iota_1, \iota_2, \ldots, \iota_n\}$ be a set of items, and $\mathrm{U} = \{u_1, u_2, \ldots, u_m\}$ be a set of users. For each item ι_i and each user u_α, we have a

Table 3: Notation

Symbol	Description
I, ι_i	set of items, a specific item
U, u_α	set of users, a specific user
n, m	number of items, number of users
$\iota_i.u_\alpha$	rating of u_α for ι_i
$\iota_i \succ \iota_j$	ι_i unanimously dominates ι_j
US	unanimous skyline
$t(u_\alpha)$	target (best rating) of u_α
$\tau(u_\alpha, \iota_i)$	utility ratio of ι_i to u_α
$s(\iota_i)$	score of ι_i to the group
k-DAUS	k-dual adjustment unanimous skyline
ℓ_α	sorted list of u_α
$\ell_\alpha[i]$	ith entry of ℓ_α

rating (relevance), which we denote by $\iota_i.u_\alpha$, that either u_α provided explicitly or obtained using personalized recommendation techniques; meaning that we have the full user-item interactions.

Let us first recall the definition of the conventional skyline, adapted to our problem.

Definition 2.1 (Unanimous Dominance). Given two items ι_i and ι_j, we say that ι_i unanimously dominates ι_j, denoted by $\iota_i \succ \iota_j$, iff ι_i has better or equal ratings than ι_j for all users and there exists at least one user for which ι_i is preferred over ι_j. i.e., $\iota_i \succ \iota_j \Leftrightarrow \forall u_\alpha \in \mathrm{U} : \iota_i.u_\alpha \geq \iota_j.u_\alpha \wedge \exists u_\beta \in \mathrm{U} : \iota_i.u_\beta > \iota_j.u_\beta$.

Definition 2.2 (Unanimous Skyline). The unanimous skyline, denoted by US, comprises the set of items that are not unanimously dominated by any other item. i.e., $\mathrm{US} = \{\iota_i \in \mathrm{I} \mid \nexists \iota_j \in \mathrm{I} : \iota_j \succ \iota_i\}$.

Next, we introduce our ranking criterion based on a dual adjustment aggregation. The first adjusts the individual utilities to treat the group members equally, while the second adjusts the obtained values so that items that are highly liked (resp. disliked) by most group members are preserved (resp. disfavored).

Definition 2.3 (Target). The target of a given user u_α, denoted by $t(u_\alpha)$, is defined to be the best rating u_α can achieve. i.e., $t(u_\alpha) = \max_{i=1}^{n} \iota_i.u_\alpha$.

Definition 2.4 (Utility Ratio). Given an item ι_i and a user u_α, the utility ratio of ι_i to u_α, denoted by $\tau(\iota_i, u_\alpha)$, captures the ratio of satisfaction of u_α if ι_i is the item selected by the group. i.e., $\tau(\iota_i, u_\alpha) = \frac{\iota_i.u_\alpha}{t(u_\alpha)}$.

Definition 2.5 (Item Score). The score of a given item ι_i, denoted by $s(\iota_i)$, is an aggregation of the utility ratios of the group members w.r.t. ι_i as follows: $s(\iota_i) = \frac{1}{m}\sum_{\alpha=1}^{m} \tau(\iota_i, u_\alpha)^\lambda$, where $\lambda \geq 1$ is an emphasis parameter, penalizing low individual utility ratios and privileging high ones.

Definition 2.6 (k-dual adjustment unanimous skyline). The k-dual adjustment unanimous skyline, denoted by k-DAUS, is the set of k items belonging to the unanimous skyline with the highest recommendation scores. i.e., $k\text{-DAUS} = \{\mathrm{S} \subseteq \mathrm{US} \mid |\mathrm{S}| = \min(k, |\mathrm{US}|) \wedge \forall \iota_i \in \mathrm{S}, \nexists \iota_j \notin \mathrm{S} : s(\iota_j) > s(\iota_i)\}$.

Returning to our example, Table 4 shows the utility ratios and scores (setting $\lambda = 2$) for all items. Recall that the unanimous skyline comprises items ι_1, ι_2 and ι_3. Thus, a 2-DAUS query returns items ι_1 and ι_3, instead of items ι_1 and ι_2 (cf. Section 1). As can be seen, our score takes into account the fact that users may have manners of rating (i.e., some users are generous while others are harsh). This is formally expressed with the following property.

Table 4: Items' Scores

Item	Utility Ratio			Score
	u_1	u_2	u_3	
ι_1	0.8	1	0.75	0.734
ι_2	1	0.78	0.69	0.692
ι_3	0.75	0.72	1	0.695
ι_4	0.8	0.97	0.75	0.716
ι_5	0.9	0.67	0.25	0.439
ι_6	0.6	0.56	0.38	0.270

Property 2.1 (Users' Ratings Neutrality). *Users with different manners of rating, are equally important.*

Proof. It is apparent from the ranking scheme definition, since it does not consider the absolute rating, but considers for each user a relative relevance to her best rating. □

We now provide the formal definition for the k-dual adjustment unanimous skyline problem.

Problem statement. *Given a set of items* I*, a set of users* U *and a parameter k, compute the k-dual adjustment unanimous skyline.*

3 ALGORITHMS FOR k-DAUS

In this section, we show how to adapt prior algorithms to compute the k-dual adjustment unanimous skyline, and we then present our proposed algorithm. In addition, we perform a theoretical analysis to study the average time complexity of each algorithm.

3.1 Skyline First Algorithm

The k-dual adjustment unanimous skyline problem can be divided into two sequentially ordered computational problems: (1) computing the unanimous skyline and (2) computing the top-k items. We call this strategy: *Skyline First Algorithm* (SFA). SFA has the advantage of calculating the recommendation score of only the items belonging to the unanimous skyline.

Note that to compute the recommendation score of a given item, we need the rating of each user for this item as well as the target of each user. To avoid computing the target of each user each time we need to compute the recommendation score of a given item, it is important to first compute the target of all users and maintain them throughout the execution of the algorithm. We make the observation that the target of each user can be computed by iterating over the items in the unanimous skyline, instead of the entire dataset. This avoid iterating over all items after computing the unanimous skyline. This observation can be expressed formally using the following lemma.

Lemma 3.1. *For any user u_α, there exists an item ι_i in the unanimous skyline having the best rating of u_α. i.e., $\forall u_\alpha \in \mathrm{U}, \exists \iota_i \in \mathrm{US} : t(u_\alpha) = \iota_i.u_\alpha$.*

Proof. Assume a unanimously dominated item ι_j and suppose that $t(u_\alpha) = \iota_j.u_\alpha$. Since ι_j is unanimously dominated there exists an item ι_i such that $\forall u_\alpha \in \mathrm{U} : \iota_i.u_\alpha \geq \iota_j.u_\alpha$. That is $\iota_i.u_\alpha \geq t(u_\alpha)$. Meaning that either $\iota_i.u_\alpha = t(u_\alpha)$, in this case $\iota_i.u_\alpha$ is also a target for u_α or $\iota_i.u_\alpha > t(u_\alpha)$, which leads to a

Algorithm 1: Skyline First Algorithm (SFA)

```
Input: set of items I, set of users U, integer $k$
Output: $k$-dual adjustment unanimous skyline $k$-DAUS
1  US ← ∅, $k$-DAUS ← ∅
2  compute US of I            // using an existing algorithm
3  foreach $u_\alpha \in$ U do
4      compute $t(u_\alpha)$ in US                  // Lemma 3.1
5  foreach $\iota_i \in$ US do
6      compute $s(\iota_i)$
7      if $|k$-DAUS$| < k$ then
8          insert $\iota_i$ into $k$-DAUS
9      else
10         if $s(\iota_i) >$ worst score in $k$-DAUS then
11             remove the worst item from $k$-DAUS
12             insert $\iota_i$ into $k$-DAUS
13 return $k$-DAUS
```

contradiction since the target of a user is defined to be its best rating. □

The pseudocode of SFA is depicted in Algorithm 1. SFA maintains two sets US and k-DAUS, containing respectively the unanimous skyline and the set of intermediate k-dual adjustment skyline items sorted in non-ascending order of their score. Initially, both sets US and k-DAUS are empty (line 1). Then, SFA first computes the unanimous skyline US using an existing skyline computation algorithm, e.g., BNL [9], SFS [12, 13] or SaLSa [5, 6] (line 2). Afterwards, by Lemma 3.1 it computes the target $t(u_\alpha)$ of each user u_α using only the items in US (loop in line 3). After that, SFA iterates over the items in US (loop in line 5) for computing the score of each unanimous skyline item ι_i (line 6), and updates the k-DAUS set so that it contains the k best items with the largest score (lines 7–12). After examining all items, SFA returns k-DAUS (line 13).

3.2 Rank First Algorithm

Although SFA can indeed limit the number of computed scores, it needs to compute the entire unanimous skyline set, which requires a large number of unanimous dominance checks. The second algorithm we introduce, termed *Rank First Algorithm* (RFA) seeks to minimize the number of unanimous dominance checks. The main idea is to first rank all items according to their recommendation score and to exploit both the monotonicity nature of this recommendation score and the transitivity property of the unanimous dominance to avoid unnecessary comparisons (unanimous dominance checks).

Lemma 3.2 (Monotonicity). *Given two items ι_i and ι_j, if ι_i unanimously dominates ι_j then ι_i has higher recommendation score than ι_j. i.e., $\iota_i \succ \iota_j \Rightarrow s(\iota_i) > s(\iota_j)$.*

Proof. The recommendation scores of ι_i and ι_j can be written as: $s(\iota_i) = \frac{1}{m}\sum_{\alpha=1}^{m}(\frac{\iota_i.u_\alpha}{t(u_\alpha)})^\lambda$ and $s(\iota_j) = \frac{1}{m}\sum_{\alpha=1}^{m}(\frac{\iota_j.u_\alpha}{t(u_\alpha)})^\lambda$. Then, $s(\iota_i) - s(\iota_j) = \sum_{\alpha=1}^{m}\frac{(\iota_i.u_\alpha^\lambda - \iota_j.u_\alpha^\lambda)}{t(u_\alpha)\lambda}$. Since $\iota_i \succ \iota_j$, and the ratings and λ are positive real numbers, we have $\forall u_\alpha \in$ U : $\iota_i.u_\alpha^\lambda \geq \iota_j.u_\alpha^\lambda \land \exists u_\beta \in$ U : $\iota_i.u_\beta^\lambda > \iota_j.u_\beta^\lambda$. Thus, $s(\iota_i) - s(\iota_j) > 0$. Hence, $s(\iota_i) > s(\iota_j)$. □

Algorithm 2: Rank First Algorithm (RFA)

```
Input: set of items I, set of users U, integer $k$
Output: $k$-dual adjustment unanimous skyline $k$-DAUS
1  I$_s$ ← ∅, $k$-DAUS ← ∅
2  foreach $u_\alpha \in$ U do
3      compute $t(u_\alpha)$
4  foreach $\iota_i \in$ I do
5      compute $s(\iota_i)$
6      insert $\iota_i$ into I$_s$
7  foreach $\iota_i \in$ I$_s$ do
8      dominated ← false
9      foreach $\iota_j \in k$-DAUS do // Lemma 3.2 & Property 3.1
10         if $\iota_j \succ \iota_i$ then
11             dominated ← true
12             break
13     if ¬ dominated then
14         insert $\iota_i$ into $k$-DAUS
15         if $|k$-DAUS$| = k$ then
16             return $k$-DAUS
17 return $k$-DAUS
```

Lemma 3.2 helps reduce the number of unanimous dominance checks. In fact, given an item ι_i, searching for items that may unanimously dominate ι_i is reduced to the part of items with higher score than $s(\iota_i)$, i.e., those located before it after the ranking phase. This is the idea behind the SFS algorithm [12, 13], which we exploit with our scoring function.

Property 3.1 (Transitivity). *Given three items ι_i, ι_j and ι_k. If ι_i unanimously dominates ι_j and ι_j unanimously dominates ι_k then ι_i unanimously dominates ι_k. i.e., $\iota_i \succ \iota_j \land \iota_j \succ \iota_k \Rightarrow \iota_i \succ \iota_k$.*

Proof. It is apparent from Definition 2.1. □

With Property 3.1, we can avoid useless comparisons. In fact, if an item is unanimously dominated then it may be discarded as it is unnecessary for eliminating other items.

The pseudocode of RFA is depicted in Algorithm 2. The algorithm uses two sets I$_s$ and k-DAUS containing respectively the set of all items and the set of k-dual adjustment unanimous skyline items; in both sets, the items are maintained in non-ascending-order of their recommendation scores. Initially, both sets I$_s$ and k-DAUS are empty (line 1). Then, the algorithm computes the target $t(u_\alpha)$ of each user u_α (loop in line 2) and computes the score of each item ι_i in I to fill the set I$_s$ (loop in line 4). Afterwards, RFA iterates over the sorted items (loop in line 7). At each iteration, an item ι_i is examined to check whether it is (or not) unanimously dominated (loop in line 9). From Lemma 3.2 the items that may unanimously dominate ι_i is reduced to the part of items with larger score than $h_g(\iota_i)$, i.e., those located before ι_i in I$_s$ and combined with Property 3.1 it is sufficient to compare ι_i only against the items in k-DAUS. If ι_i is not unanimously dominated (line 13) then it is inserted into the set of the current k-dual adjustment unanimous skyline items k-DAUS (line 14). The algorithm terminates after retrieving the first k non-unanimously dominated items (line 15). If all items were examined and the size of k-DAUS is lower than k, meaning that the unanimous skyline comprises less than k items, then k-DAUS is returned (line 17).

ℓ_1	ℓ_2	ℓ_3
$\langle\iota_2, 10\rangle$	$\langle\iota_1, 9\rangle$	$\langle\iota_3, 8\rangle$
$\langle\iota_5, 9\rangle$	$\langle\iota_4, 8.75\rangle$	$\langle\iota_1, 6\rangle$
$\langle\iota_1, 8\rangle$	$\langle\iota_2, 7\rangle$	$\langle\iota_4, 6\rangle$
$\langle\iota_4, 8\rangle$	$\langle\iota_3, 6.5\rangle$	$\langle\iota_3, 5.5\rangle$
$\langle\iota_3, 7.5\rangle$	$\langle\iota_5, 6\rangle$	$\langle\iota_6, 3\rangle$
$\langle\iota_6, 6\rangle$	$\langle\iota_6, 5\rangle$	$\langle\iota_5, 2\rangle$

Figure 1: Sorted Lists Example

3.3 Sorted Lists-based Algorithm

Even if RFA reduces the number of unanimous dominance checks, it requires computing the score of all items and rank them. As our scoring function is monotone, it is well supported by the family of top-k threshold algorithms [15, 16], which aim to reduce the number of computed item scores; the added difficulty in our case is that the algorithm should compute also the unanimous skyline, which is more complex than computing the plain top-k set.

In the following, we present our new algorithm, termed Sorted Lists-based Algorithm (SLA), for efficiently computing the k-dual adjustment unanimous skyline. The key idea behind SLA is to establish a sorted list ℓ_α for each user u_α, where ℓ_α maintains the set of items in decreasing order of the ratings of u_α; in case of ties, by the average rating among users. Using this scheme, the items that may unanimously dominate a given item are reduced to the part of unanimous skyline items located before it in a list. In addition, at each level, the algorithm creates a virtual item and uses it to compute an upper bound score for the not examined items, used as an early termination condition to avoid iterating through all items that are definitely not part of the result. This helps reduce both the number of unanimous dominance checks and the number of computed recommendation scores, as we will see thereafter.

Figure 1 depicts the sorted lists established for our example. We denote by $\ell_\alpha[i]$ the i^{th} entry of sorted list ℓ_α. For instance, $\ell_1[1]$ stands for ι_2. Therefore $\ell_\alpha[i].u_\alpha$ stands for the rating of user u_α for item $\ell_\alpha[i]$. For instance $\ell_1[1].u_1 = 10$.

From the sorted lists, we make the following observations, which provide important insights to facilitate the development of our algorithm.

Observation 3.1. *The target of each user is equal to her rating for the first entry in her list. i.e.,* $\forall u_\alpha \in \mathrm{U} : t(u_\alpha) = \ell_\alpha[1].u_\alpha$.

Observation 3.2. *Given a sorted list* ℓ_α *and an item* $\ell_\alpha[i]$ *in* ℓ_α*, if* $\ell_\alpha[i]$ *is not unanimously dominated by any unanimous skyline item located before it in* ℓ_α *then it belongs to the unanimous skyline. i.e.,* $\nexists j \in [1, i[: \ell_\alpha[j] \in \mathrm{US} \wedge \ell_\alpha[j] \succ \ell_\alpha[i] \Rightarrow \ell_\alpha[i] \in \mathrm{US}$.

Observation 3.3. *Let* v_i *be the virtual item at index* i *having ratings* $\ell_1[i].u_1, \ell_2[i].u_2, \ldots, \ell_m[i].u_m$ *(e.g., the virtual item at index 3 in our sorted lists is built with the ratings 8, 7 and 6), then the score of* v_i *constitutes an upper bound of the recommendation scores of items located after index* i *in all sorted lists.*

The pseudocode of SLA is depicted in Algorithm 3. It operates on the previously described sorted lists and uses the following sets: US maintaining the set of all identified unanimous skyline items; $\mathrm{US}_1, \mathrm{US}_2, \ldots, \mathrm{US}_m$, maintaining the set of identified unanimous skyline items in the lists $\ell_1, \ell_2, \ldots, \ell_m$, respectively; and k-DAUS, maintaining the set of intermediate k-dual adjustment unanimous skyline items sorted in non-ascending order of their recommendation score. Initially, all sets are empty (line 1). Then,

Algorithm 3: Sorted Lists-based Algorithm (SLA)

```
Input: sorted lists $\ell_1, \ell_2, \ldots, \ell_m$, integer $k$
Output: $k$-dual adjustment unanimous skyline $k$-DAUS
1  US ← ∅, $\forall\alpha \in [1, m]$ : $\mathrm{US}_\alpha$ ← ∅, $k$-DAUS ← ∅
2  for $\alpha$ ← 1 to $m$ do
3      $t(u_\alpha) \leftarrow \ell_\alpha[1].u_\alpha$                    // Observation 3.1
4  foreach $i$ ← 1 to $n$ do
5      if |$k$-DAUS| = $k$ then
6          $v_i$ ← the virtual item at index $i$
7          compute $s(v_i)$
8          if $s(v_i) \leq$ worst score in $k$-DAUS then
9              return $k$-DAUS                             // Observation 3.3
10     for $\alpha$ ← 1 to $m$ do
11         $\iota_c \leftarrow \ell_\alpha[i]$
12         if $\iota_c$ was examined then
13             if $\iota_c \in$ US then
14                 insert $\iota_c$ into $\mathrm{US}_\alpha$
15         else
16             dominated ← false
17             foreach $\iota_j \in \mathrm{US}_\alpha$ do              // Observation 3.2
18                 if $\iota_j \succ \iota_c$ then
19                     dominated ← true
20                     break
21             if ¬ dominated then
22                 insert $\iota_c$ into US
23                 insert $\iota_c$ into $\mathrm{US}_\alpha$
24                 compute $s(\iota_c)$
25                 if |$k$-DAUS| < $k$ then
26                     insert $\iota_c$ into $k$-DAUS
27                 else
28                     if $s(\iota_c) >$ worst score in $k$-DAUS then
29                         remove the worst item from $k$-DAUS
30                         insert $\iota_c$ into $k$-DAUS
31 return $k$-DAUS
```

based on Observation 3.1, the algorithm sets the target $t(u_\alpha)$ of each user u_α to its rating for the first items in its list ℓ_α (loop in line 2). Afterwards, SLA starts iterating over the sorted lists in a round robin fashion (nested loops in lines 4 and 10) as follows. Let ι_c be the current item to examine, i.e., the i^{th} entry of sorted list ℓ_α (line 11). If ι_c was already examined in another sorted list (line 12) then two cases arise: ι_c was identified as a unanimous skyline item, in this case, it is inserted into US_α, i.e., the current unanimous skyline items of sorted list ℓ_α (lines 13 and 14); otherwise, i.e., ι_c is unanimously dominated, ι_c is ignored. If ι_c is not examined (line 15) then by Observation 3.2, the algorithm compares ι_c against the unanimous skyline items of sorted list ℓ_α so far retrieved (loop in line 17) to check whether it is (or not) unanimously dominated. If ι_c is not unanimously dominated (line 21) then it is inserted into both US and US_α sets (lines 22 and 23), and its score $s(\iota_c)$ is computed (line 24) in order to update the k-DAUS set so that it contains the k best items with the largest score (lines 25–30). When the size of k-DAUS reaches k items (line 5), the algorithm builds the virtual item v_i at each index i (line 6) and computes its score $s(v_i)$ (line 7). If $s(v_i)$ is lower than or equal to the worst score in k-DAUS then k-DAUS is returned since $s(v_i)$ constitute an upper bound of all not examined items by Observation 3.3. If all items were examined and the size of k-DAUS is lower than k, meaning that the unanimous skyline comprises fewer than k items, then k-DAUS is returned (line 31).

3.4 Theoretical Analysis

The performance of the presented algorithms mainly depends on the size of the unanimous skyline, whose expected value is [7, 17]: $\Theta(\ln(n)^{m-1}/(m-1)!)$. It is reasonable to assume that k is lower than this expected value. Hereafter, we calculate the average complexity of SFA, RFA and SLA, i.e., their complexity in the presence of $\ln(n)^{m-1}/(m-1)!$ unanimous skyline items.

Theorem 3.3. *SFA has an average case complexity of $O(mn)$.*

Proof. The average computational cost of SFA is the sum of three stages. The first is computing the unanimous skyline, which takes $O(mn)$ in the average case [18, 19]. The second is computing the targets. Since the expected size of the unanimous skyline is $\Theta(\ln(n)^{m-1}/(m-1)!)$. This step takes $O(m\ln(n)^{m-1}/(m-1)!)$. The third is to iterate over the unanimous skyline for computing the scores and updating the k-dual adjustment unanimous skyline. This stage takes $O((m+log(k))\ln(n)^{m-1}/(m-1)!)$. $O(mn)$ dominates $O(m\ln(n)^{m-1}/(m-1)!)$. Moreover, since $k \leq \ln(n)^{m-1}/(m-1)!$, $O(mn)$ dominates $O(log(k)\ln(n)^{m-1}/(m-1)!)$. Hence, the average computational complexity of SFA is $O(mn)$. □

Theorem 3.4. *RFA has an average case time of $O(mn+nlog(n))$.*

Proof. The computational cost of RFA is the sum of three stages. The first is computing the targets, which takes $O(mn)$. The second is computing scores and sorting the items, which takes $O((m+log(n))n)$. The last is to iterate over the items for performing unanimous dominance checks and updating the k-dual adjustment unanimous skyline. Since, the items are already sorted according to their scores, an update costs $O(1)$. There are k updates. Thus, updating the k-dual adjustment unanimous skyline costs $O(k)$. The number of comparisons phase can reach $O(mn)$ [19]. Thus, the computational complexity of RFA is $\mathrm{O}(mn+nlog(n))$. □

Theorem 3.5. *SLA has an average case complexity of $O(mn)$.*

Proof. SLA first computes the targets, which takes $O(m)$. Then it iterates over the sorted lists. The iterations involve the following parts: computing the score of the virtual items, which takes $O(mn)$; performing the unanimous dominance checks, which takes $O(mn)$ in the average case [19]; computing the scores and updating the k-dual adjustment unanimous skyline, which takes $O((m+log(k))\ln(n)^{m-1}/(m-1)!)$. Using eliminations similar to the proof of the Theorem 3.3, the time complexity of SLA is $O(mn)$. □

As can be seen, on average, RFA is dominated by both SFA and SLA, which have the same complexity. However, in practice, SLA is more than an order of magnitude faster than SFA as it employs various optimization strategies for reducing the search space (see Section 4).

Table 5: Datasets

Dataset	# Items	# Users	# Ratings	Density
MovieLens HR	10,109	2,113	855,598	0.04006
MovieLens 1M	3,706	6,040	1,000,209	0.04468
Personality	35,196	1,820	1,020,429	0.01593
Magazine	72,098	2,428	88,318	0.00050

4 EXPERIMENTAL EVALUATION

In this section, we present an extensive experimental study of our approach. Specifically, we evaluate our approach from two major angles. First, we investigate the benefits resulting from the use of our proposed ranking criterion in terms of retrieval effectiveness. Second, we consider the performance of the proposed algorithms. Technically speaking, our experiments aim to answer the following research questions.

- **RQ1**: How does the k-DAUS perform as compared to existing occasional group recommendation approaches?
- **RQ2**: How does the k-DAUS perform when varying the problem parameters?
- **RQ3**: How do SFA, RFA and SLA algorithms scale with the problem parameters?

In business, the results of our evaluation will help companies that want to build a group recommender system to choose the appropriate recommendation strategy. To provide accurate recommendations to their customers, is it better to design the recommender system with k-DAUS or with another group recommendation method? In case k-DAUS is deemed to be the appropriate strategy, with which algorithm k-DAUS should be implemented? SFA, RFA or SLA?

4.1 Evaluation Setup

Datasets. Four real-world datasets are used in our experiments. The first two datasets are MovieLens HetRec 2011 [1] and MovieLens 1M [2], which are two different versions of the well-known MovieLens datasets [20]. MovieLens HR (HetRec 2011), contains 855,598 ratings, ranging from 0.5 to 5 stars, of 10,109 movies rated by 2,113 users. MovieLens 1M contains 1,000,209 ratings of 3,706 movies by 6,040 users; ratings vary from 1 to 5 stars. The third dataset is Personality 2018 [3] [30], containing 1,020,429 ratings, ranging from 0.5 to 5 stars, of 35,196 movies by 1,820 users. The last dataset is Magazine (Subscriptions), from Amazon Review Data 2018 [4] [31]. This dataset contains 88,318 ratings, ranging from 1 to 5, of 72,098 magazines by 2,428 users. Table 5 summarizes the main characteristics of these datasets (Density = #Ratings/(#Items × #Users)).

For the first three datasets (movies), a real-life example of a group is sport team members who meet to watch a movie. For the Magazine dataset, a concrete example of a group is residents of a building who want to subscribe to a magazine.

For the construction of the groups, we consider two main factors: *group size* and *group affinity*, i.e., how similar are group members in their ratings. In particular, we generate groups of sizes 10, 15, 20, 25 and 30. As for affinity, to cover a large number

[1]https://grouplens.org/datasets/hetrec-2011/
[2]https://grouplens.org/datasets/movielens/1m/
[3]https://grouplens.org/datasets/personality-2018/
[4]https://nijianmo.github.io/amazon/index.html

of real-life scenarios, we generate three kinds of groups: *similar*, *random* and *diverse*. In order to generate these groups, we compute the similarities between users using the Pearson correlation coefficient. A similar group is created by randomly selecting a user and then progressively adding the most similar user to the group. A diverse group is generated in the same way, but by adding the less similar user to the group at each iteration. Random groups are formed by picking users uniformly. Similar to previous work [3, 4, 25, 35, 36, 39, 41] we adopt collaborative filtering [1] to fill in the missing ratings in the datasets. In particular, we use the Singular Value decomposition (SVD) matrix factorization technique. To focus only on how group recommendation methods aggregate individual ratings, we assume that the estimated ratings are correct. This means that the effectiveness values obtained are lower than those expected in business. For the retrieval effectiveness evaluation, we naturally compute the set of unanimously preferred items for each group (since unanimously dominated items are inappropriate, as argued in Section 1) and set the ground truth of each group member to her k favorite items among them. Meaning that, for a given user, ideally, the system retrieves her top-k items.

Compared Methods. To answer **RQ1** and **RQ2**, we compare the retrieval effectiveness of our approach, i.e., k-dual adjustment unanimous skyline (k-DAUS), with eleven state-of-the-art group recommendation methods. **AVG** and **MIN** apply respectively the average and the minimum score aggregation strategies [4]. **MUL** and **MAX** employ respectively the multiplicative and the maximum score aggregation strategies [28]. **SDP** subtracts from the average aggregation the standard deviation as a penalty [23]. **CRD** is the consensus method from [3] combining the group relevance and the group disagreement. **GRF** corresponds to the group rating fairness of [35, 36]. **SPG** and **EFG** implement respectively the single proportionality and envy-freeness greedy algorithms [41]. **GRV** implements the greedy variance algorithm [25]. **XPO** corresponds to the x-level Pareto Optimal aggregation approach [39].

It is worth noting that for a fair comparison, for all approaches we consider only the unanimously preferred items when retrieving the top-k recommendations. Also, note that the choice of algorithm for computing the k-DAUS is not important for effectiveness, i.e., **RQ1** and **RQ2**, as the three algorithms, i.e., SFA, RFA and SLA, deliver the same result. Indeed, this is the purpose of **RQ3**.

To answer **RQ3**, we investigate the performance, in terms of the amount of time required to produce the k-dual adjustment unanimous skyline, of five algorithms: **SFA-BNL**, **SFA-SFS** and **SFA-SaLSa** are the implementation of SFA (Algorithm 1) using respectively BNL [9], SFS [12, 13] and SaLSa [5, 6] algorithms for computing the unanimous skyline; **RFA** (Algorithm 2); and **SLA** (Algorithm 3). Notice that for both **SFA-SFS** and **SFA-SaLSa**, we do not consider the preprocessing time.

Effectiveness Metrics. We evaluate the retrieval effectiveness of the group recommendation methods with three widely used metrics: **Precision**, **Average Precision** and Normalized Discounted Cumulative Gain (**NDCG**). Each metric quantifies the relevance of a recommended list rec to each user's ground truth list rel_u. Precision indicates the proportion of relevant recommended items from the total number of recommended items. Then, the precision w.r.t. a user u is $P = \frac{|rel_u \cap rec|}{k}$. The average precision w.r.t. a user u measures the average of precision values calculated after each relevant item for u is retrieved in the recommended list, i.e., $AP = \sum_{r=1}^{k} \frac{P@r \cdot rec[r]}{k}$, where $P@r$ is the precision at rank r and $rec[r]$ is an indicator equaling 1 if the recommended item at rank r belongs to rel_u, and 0 otherwise. NDCG measures how well a method can rank the best relevant items higher. Similar to [25, 39], we use the Borda semantics and set the relevance of an item at rank r, which we denote by $rel_u[i]$, in the ground truth rel_u of a user u to $k - r + 1$. The discounted cumulative gain is defined as: $DCG = \sum_{r=1}^{k} \frac{rel_u[r]}{log(r+1)}$. The ideal discounted cumulative gain (IDCG) is defined to be the DCG achieved when the relevant items are ranked as in the ground truth. Then, the Normalized Discounted Cumulative Gain is calculated as the ratio of DCG over IDCG, i.e., $NDCG = \frac{DCG}{IDCG}$. We average the metric values across group members to obtain relevance to a group. For all three metrics, higher values indicate better recommendations.

Table 6: Parameters and Examined Values

Parameter	Examined Values	Default
Group size (m)	10, 15, 20, 25, 30	20
# recommendations (k)	10, 15, 20, 25, 30	20
Emphasis factor (λ)	2, 4, 6, 8, 10	6

Implementation and Parameter Setting. All group recommendation methods and k-DAUS algorithms are implemented in python 3. We employ the SVD algorithm from Surprise library [5] [22] for generating the missing ratings. We investigate different problem settings. The involved parameters and their examined values are summarized in Table 6. Due to space constraints, we choose MovieLens HR as a primary dataset since it has a good compromise between the number of items (to evaluate the scalability of the algorithms) and density. Higher density implies a more accurate estimation of individual ratings by the collaborative filtering algorithm. In all experiments, we investigate the effect of one parameter, while we set the remaining ones to their default values. For the remaining datasets, we report only the results of the default execution. All experiments were conducted on a 2.9 GHz Quad-Core Intel Core i7 processor with 16 GB 2133 MHz LPDDR3 memory, running macOS Catalina. Reported values are averages of 100 generated groups.

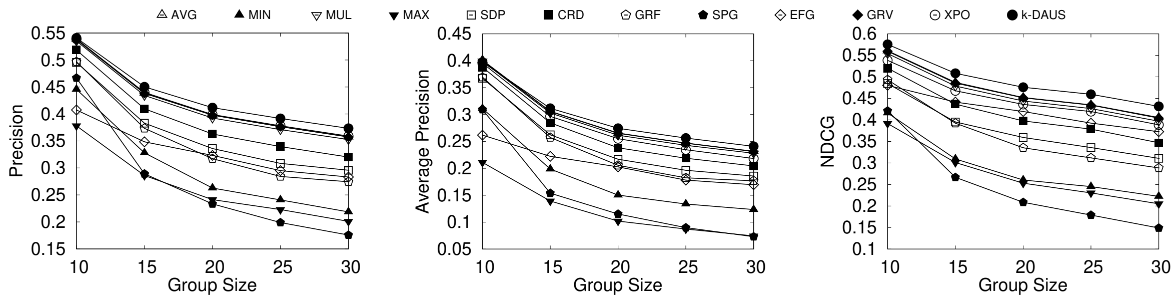
4.2 Retrieval Effectiveness (RQ1 & RQ2)

4.2.1 Results on Movielens HR. We study the effect of the problem parameters on the retrieval effectiveness of the group recommendation methods using Movielens HR.

Effect of m. In this experiment, we study the effectiveness of the group recommendation methods w.r.t. the group size m. Particularly, we vary the group size m from 10 up to 30, asking for the top-20 recommendations, and measure the precision, average precision and NDCG. Figure 2, Figure 3 and Figure 4 show the results of this experiment for similar, random and diverse groups, respectively.

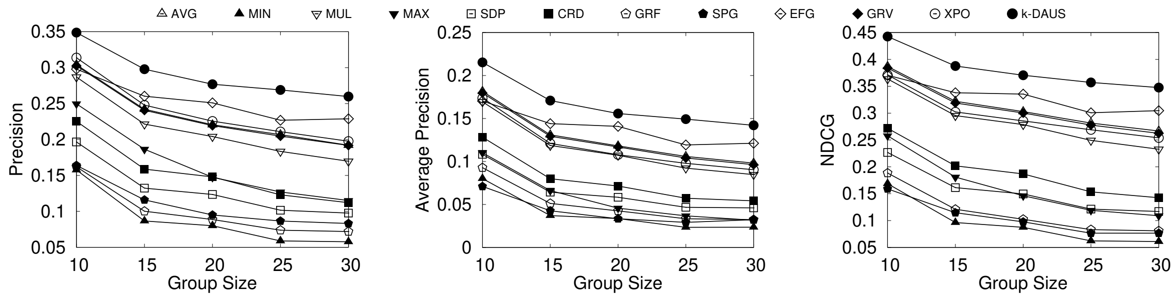
A general observation is that the effectiveness of all methods deteriorates with the increase of m. This is because increasing the number of users in a group affects the difficulty of reaching a consensus among group members. Another observation is that

[5] http://surpriselib.com/



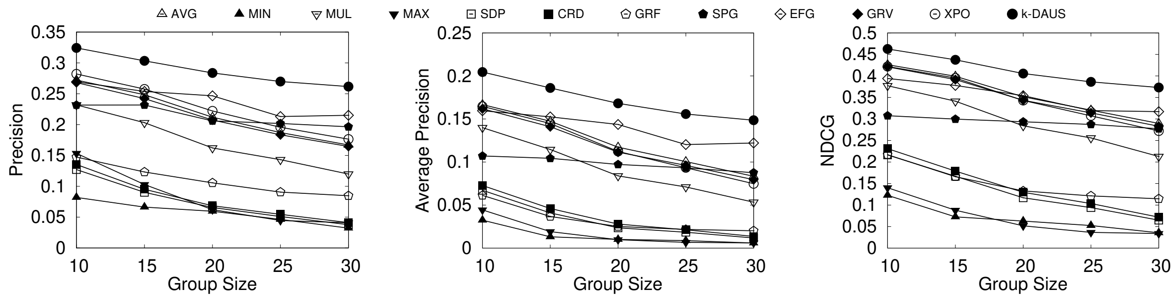
(a) Precision (b) Average Precision (c) NDCG

Figure 2: Effectiveness vs. Group size (m) for Similar Groups; MovieLens HR



(a) Precision (b) Average Precision (c) NDCG

Figure 3: Effectiveness vs. Group size (m) for Random Groups; MovieLens HR



(a) Precision (b) Average Precision (c) NDCG

Figure 4: Effectiveness vs. Group size (m) for Diverse Groups; MovieLens HR

all methods achieve higher (resp. lower) effectiveness for similar (resp. diverse) groups since it is easier (resp. more difficult) to find items that satisfy all group members.

In comparison, the results show that k-DAUS achieves better retrieval effectiveness than all methods, however, there is no clear winner among the other methods. On the other hand, observe that the difference between k-DAUS and the other methods is more pronounced for random and diverse groups since the problem becomes more difficult. This shows the superiority of our approach in handling conflicting user preferences. This can be interpreted as follows: the items that are highly liked by all group members will be selected by almost all approaches. The problem arises for the other selected items. Approaches carrying only on fairness (e.g., MIN, SPF, EFG) miss items that are highly liked by all group members except one (or a few) since unsatisfied users have a higher impact on the final decision. The remaining approaches include items supposedly moderately good for all group members. The problem is that in practice, an average rating (e.g., 3 stars in a 1–5 stars rating system) is not considered relevant for an individual user. Thus, these items are not in the ground truth of any group member. Consequently, they do not satisfy any group member. In contrast, k-DAUS preserves items that are highly liked by most group members. Therefore, most group members find items that are in their ground truth. Also, as k-DAUS disfavors items disliked by most group members, it does not include items that are only liked by a few members.

Effect of k. In the next experiment, we investigate the effectiveness of the group recommendation methods w.r.t. k. In particular, we fix the group size to $m = 20$ and vary the number of requested recommendation k from 10 up to 30 and measure the precision, average precision and NDCG. The results of this experiment for similar, random and diverse groups are shown in Figure 5, Figure 6 and Figure 7, respectively.

Observe in Figure 5, Figure 6 that the effectiveness of the group recommendation methods increases with higher k for similar and random groups. The reason is that in this kind of groups, the preferences of group members are similar or not

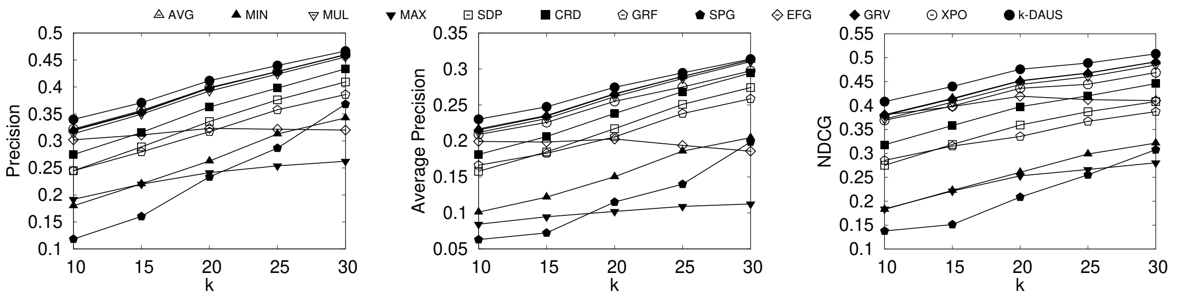

(a) Precision (b) Average Precision (c) NDCG

Figure 5: Effectiveness vs. k for Similar Groups; MovieLens HR

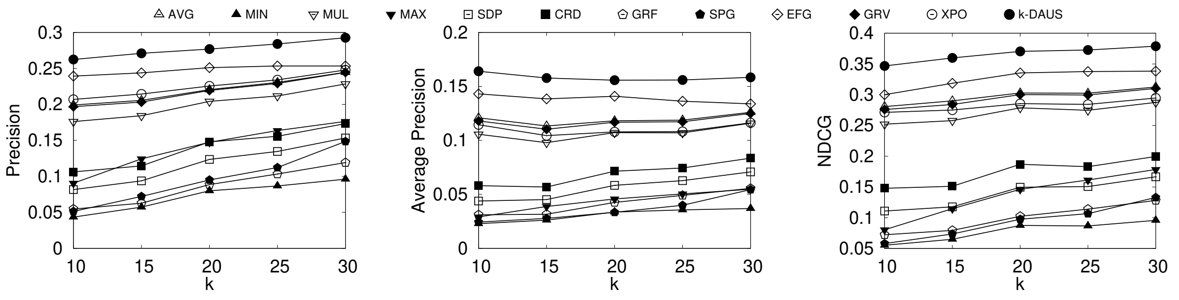

(a) Precision (b) Average Precision (c) NDCG

Figure 6: Effectiveness vs. k for Random Groups; MovieLens HR

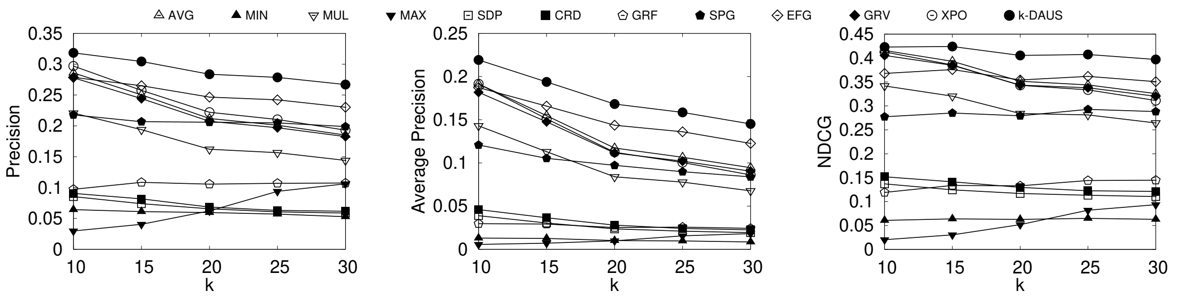

(a) Precision (b) Average Precision (c) NDCG

Figure 7: Effectiveness vs. k for Diverse Groups; MovieLens HR

very conflicting at worst (for random groups), thus, increasing the number of requested recommendations increases the chance to retrieve items that satisfy all group members. In contrast, for diverse groups, the effectiveness of the methods decreases with the increase of k. In this case, the preferences of the group members are conflicting, and thus it is difficult to find more items liked by all group members. Roughly speaking, it is easier to find 10 items liked by the group than to find 30 items liked by the group. On the other hand, similar to the previous experiment, the results show that all methods achieve higher effectiveness for similar groups.

Further, notice that k-DAUS constantly outperforms all methods. Similar to the previous experiment, there is no clear winner among the other methods, and the difference between k-DAUS and the other methods is more important for random and diverse groups for the same reasons.

Effect of λ. This experiment compares the effectiveness of k-DAUS under different choices of the emphasis parameter λ (Definition 2.5). Particularly, we fix the group size to $m = 20$ and ask

Table 7: Effectiveness of k-DAUS vs. λ; MovieLens HR (%)

λ	Similar			Random			Diverse		
	P	**AP**	**NDCG**	**P**	**AP**	**NDCG**	**P**	**AP**	**NDCG**
2	40.17	26.88	45.87	23.76	12.90	32.52	26	15.01	40.16
4	40.79	27.31	46.85	26.41	14.65	35.71	**28.60**	**16.94**	**41.49**
6	41.16	**27.45**	47.58	27.69	15.58	37.05	28.37	16.82	40.55
8	**41.28**	27.41	47.95	28.4	16.05	37.6	28.1	16.56	39.70
10	41.16	27.22	**48.06**	**28.65**	**16.22**	**37.74**	27.72	16.33	39.11

for the top-20 recommendations while varying λ from 2 up to 10; results are depicted in Table 7.

A general observation is that there is not an optimal value of λ. More precisely, for similar and random groups, higher values lead to better effectiveness, while for diverse groups better results are achieved with average values. Since in diverse groups it is more likely that a given item ι is rated very good by some users and very bad by others. Thus, with higher values of λ the value of $\tau(\iota_i, u_\alpha)^\lambda$ (see Definition 2.5), for each user u_α for which ι is bad

Table 8: Effectiveness on MovieLens 1M (%)

Method	Similar			Random			Diverse		
	P	AP	NDCG	P	AP	NDCG	P	AP	NDCG
AVG	57.94	46.55	62.49	32.25	19.73	39.68	27.90	15.81	40.09
MIN	48.84	35.60	49.90	19.62	10.25	23.01	5.56	1.64	6.31
MUL	57.74	46.38	62.17	31.25	19.08	38.47	22.13	11.18	31.56
MAX	32.87	15.88	29.56	16.26	5.18	14.7	5.59	0.95	4.27
SDP	55.36	44.22	58.82	25.73	15.10	31.08	11.42	4.32	15.22
CRD	56.62	45.42	60.45	27.55	16.52	33.63	10.59	4.07	14.27
GRF	51.59	40.34	55.08	23.36	12.69	27.38	26.62	13.77	34.30
SPG	53.92	39.23	50.87	27.7	14.73	30.25	26.1	14.22	32.93
EFG	49.98	36.16	56.29	31.08	17.89	38.29	29.9	17.81	39.17
GRV	57.90	46.51	62.45	32.19	19.67	39.58	27.04	14.99	38.86
XPO	58.06	43.73	57.23	32.52	18.35	37.58	28.57	15.16	38.55
k-DAUS	**58.26**	**46.62**	**63.26**	**35.09**	**21.59**	**42.94**	**32.98**	**20.36**	**43.70**

Table 9: Effectiveness on Personality (%)

Method	Similar			Random			Diverse		
	P	AP	NDCG	P	AP	NDCG	P	AP	NDCG
AVG	55.46	45.99	62.24	27.54	18.53	32.96	14.52	5.66	21.14
MIN	44.54	35.05	50.32	17.24	10.8	20.44	2.79	0.81	3.92
MUL	55.12	45.71	61.84	26.18	17.86	31.48	10	3.55	14.79
MAX	28.27	13.73	23.49	11.12	2.73	9.44	2.48	0.26	1.68
SDP	51.16	42.21	57.51	21.81	15.64	26.6	5.81	2.14	9.36
CRD	52.82	43.78	59.3	22.69	16.18	27.56	5.28	1.99	8.67
GRF	48.60	39.87	55.94	20.32	15.08	25.24	7.74	2.46	10.82
SPG	43.05	26.31	37.97	17.71	7.84	15.97	10.18	3.07	13.42
EFG	46.87	33.53	53.76	25.32	13.68	31.74	19.22	10.44	25.31
GRV	55.4	45.94	62.2	27.49	18.47	32.89	14.47	5.52	20.92
XPO	55.55	41.90	55.58	27.81	16.51	30.57	15.41	5.48	20.88
k-DAUS	**56.28**	**46.41**	**63.45**	**31.26**	**20.12**	**37.40**	**22.16**	**11.65**	**29.87**

Table 10: Effectiveness on Magazine (%)

Method	Similar			Random			Diverse		
	P	AP	NDCG	P	AP	NDCG	P	AP	NDCG
AVG	91.29	89.74	89.20	73.34	67.26	73.12	19.05	8.54	29.21
MIN	88.54	85.61	84.30	62.24	51.23	59.18	7.6	2.79	11.72
MUL	91.26	89.72	89.2	73.28	67.16	73.04	18.3	8.18	28.45
MAX	85.74	80.87	78.95	55.02	39.36	46.69	0.9	0.14	0.52
SDP	90.85	89.21	88.69	71.69	65.05	71.22	16.18	6.24	23.75
CRD	91.08	89.52	89.04	72.64	66.37	72.36	16.78	6.79	25.14
RGF	90.92	89.35	88.83	68.71	61.85	68.79	17.65	7.6	26.44
SPG	85.52	80.04	78.55	66.38	53.54	59.42	17.75	8.35	26.84
EFG	89.08	86.74	86.84	66.96	59.55	68.25	18.78	9.54	28.58
GRV	91.27	89.73	89.20	73.33	67.23	73.10	18.78	8.42	28.97
XPO	91.3	88.81	84.03	73.33	65.16	68.98	19.08	8.36	28.24
k-DAUS	**91.32**	**89.76**	**89.22**	**73.43**	**67.41**	**73.28**	**19.8**	**9.68**	**31.10**

tends to 0. That is, the preferences of users that rated bad ι are neglected and therefore the selection rule becomes dictatorial.

4.2.2 Results on other Datasets. We now investigate the effectiveness of the group recommendations methods for the default setting on MovieLens 1M, Personality and Magazine datasets; results are shown in Table 8, Table 9 and Table 10, respectively.

An important observation is that the best retrieval effectiveness is attained by k-DAUS for all datasets, all kinds of groups and all metrics. Also, similar to the previous experiments, the margin is more pronounced for random and diverse groups. The second position (underlined) is shared between, AVG and XPO for similar and random groups, while EFG offers a good compromise between datasets for diverse groups.

The results show also, similar to the previous experiments, that all methods achieve higher (resp. lower) effectiveness for similar (resp. diverse) groups.

In addition, observe that the methods attained high effectiveness on the Magazine dataset for similar and random groups. After analyzing the dataset, we found that it contains more than 60% of 5 stars and almost 15% of 4 stars. Thus, it is more likely to find items satisfying all group members.

4.3 Scalability (RQ3)

4.3.1 Results on MovieLens HR. We measure the execution time for all k-DAUS algorithms using MovieLens HR. The results varying m, k and λ are shown in Figure 8, Figure 9 and Figure 10, respectively.

As expected, observe in Figure 8 that the performance of all algorithms deteriorates with the increase of m. This is because, on the one hand, the cost of unanimous dominance checks increases, and on the other hand, increasing the number of users leads to less unanimously dominated items. Therefore, fewer items can be quickly eliminated.

As shown in Figure 9 the execution time of the algorithms increases very slightly with higher k, as more items need to be retrieved. The difference is negligible for all SFA variants and RFA since the time cost of these algorithms is dominated by computing the skyline and the items' scores, respectively.

Figure 10 shows that λ does not have any effect on all SFA variants and RFA. However, the execution time of SLA decreases with higher λ. This is because higher values of λ penalize more low ratings, thus it is faster to reach the termination condition.

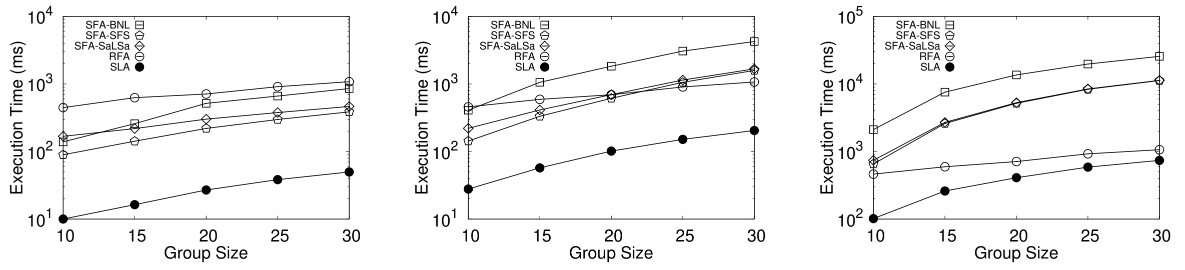
Overall, the results indicate that SLA is consistently faster than all SFA variants and RFA. This demonstrates that our pruning techniques are effective. Among SFA variants SFA-BNL is the least efficient since SFS and SaLSa skyline algorithms perform on preprocessed data to perform less dominance checks. In addition, note that all SFA variants outperform RFA for similar groups since in this case there are fewer unanimous skyline items and the time cost of algorithms is dominated by the number of computed scores. However, for diverse groups, there are a large number of unanimous skyline items and the time cost is dominated by the skyline computation. This is why, in this case, RFA outperforms all SFA variants. For random groups, which is a case in between, SFA-SFS, SFA-SaLSa and RFA are as good as each other.

In summary, SLA is the better choice to implement the k-dual adjustment unanimous skyline.

4.3.2 Results on other Datasets. Table 11 shows the execution time of the k-DAUS algorithms on MovieLens 1M, Personality and Magazine datasets, for the default setting. As in the previous experiments, all SFA variants outperform RFA for similar groups while RFA outperforms all SFA variants for diverse groups.

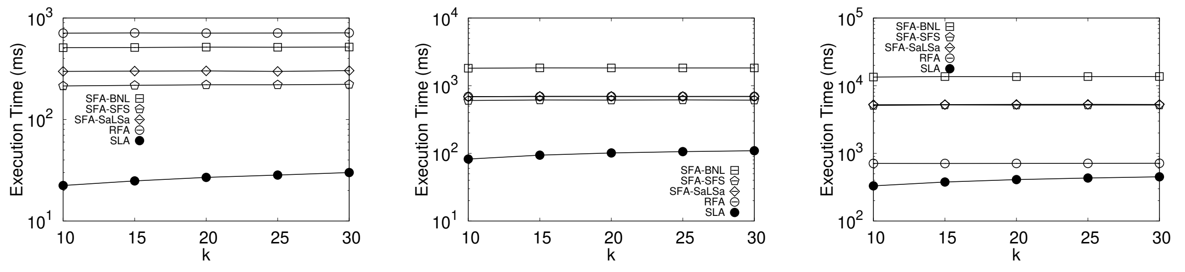
In addition, SLA performs faster than all SFA variants and RFA, except for diverse groups on the Magazine dataset. This is due to the distribution of this dataset. In fact, after examining the execution of RFA, we observe that after the ranking phase, the k first items belong to the unanimous skyline, thus the number of unanimous dominance checks is minimal. Formally, RFA performs $1 + 2 + \cdots + k - 1 = {}^{(k-1)k}/_{2}$ unanimous dominance checks. This is the best case for this algorithm, assuming $|\mathrm{US}| \geq k$. In other words, the complexity of RFA becomes $O(mk^2 + nlog(n))$ (instead of $O(mn+nlog(n))$). Even if this complexity is dominated by that of SLA in the average case, i.e., $O(mn)$, the complexity is computed in the limit of n, and n is not big enough so that $O(mn)$ will be better than $O(mk^2 + nlog(n))$.

As can be seen, SLA is more than one order of magnitude faster than the other algorithms, except for one case where RFA is better. However, RFA performs very badly on similar and random groups. This confirms that SLA is the better choice to implement the k-dual adjustment unanimous skyline.



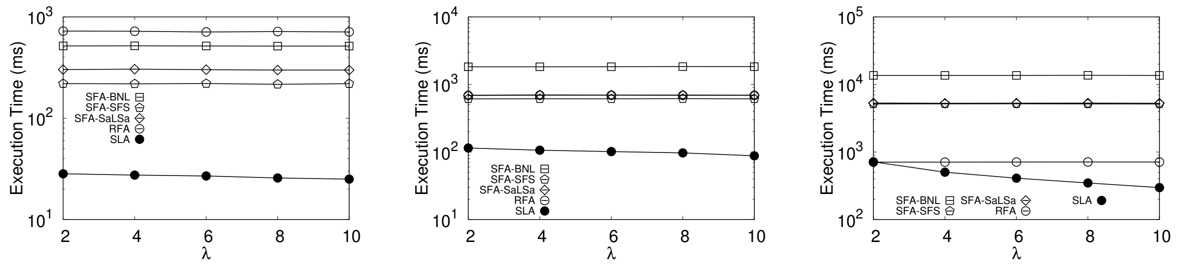
(a) Similar (b) Random (c) Diverse

Figure 8: Execution Time vs. Group size (m); MovieLens HR



(a) Similar (b) Random (c) Diverse

Figure 9: Execution Time vs. k; MovieLens HR



(a) Similar (b) Random (c) Diverse

Figure 10: Execution Time vs. λ; MovieLens HR

Table 11: Execution Time (ms) on other Datasets

Algorithm	MovieLens 1M			Personality			Magazine		
	Sim	Rnd	Div	Sim	Rnd	Div	Sim	Rnd	Div
SFA-BNL	115	437	5666	1426	4646	71038	1056	1665	299515
SFA-SFS	70	160	1773	591	774	11979	1127	1247	138340
SFA-SaLSa	99	187	1808	864	1052	12316	1618	1746	140621
RFA	266	276	291	2690	2686	2696	5047	5174	**5498**
SLA	**13**	**41**	**271**	**35**	**92**	**900**	**613**	**529**	14234

5 RELATED WORK

In this section, we review relevant work in the areas of: (1) group recommendation; (2) skyline query computation; and (3) top-k threshold algorithms.

Group Recommendation. Group recommendation studies can be divided into two main categories based on the groups types: recommendation to persistent groups and occasional (ephemeral) groups [37]. In the former case, there is a rich historical information about group-item interactions and the group can be treated as a virtual user, then personalized recommendation techniques can be adapted to make recommendations [21, 38]. In the latter case, there is no historical information about the group-item interactions, and recommendations must be computed by aggregating the user-item interactions [3, 4]. Note that recently, hybrid approaches are proposed [11, 44, 45] to make use of both user-item and group-item interactions, specially to handle the case where only a poor set group-item interactions is available. In this work, we focus on recommendation to occasional groups, where we assume the availability of user-item interaction only.

Making recommendations to occasional groups is much more challenging since there is no group-item interactions. Existing occasional group recommendation methods fall into two aggregation paradigms [3]: profile aggregation and score aggregation. Profile aggregation approaches first aggregate the profiles of group members into a single profile, then make recommendations based on the generated profile; see e.g., [8, 29, 40, 46]. Score aggregation strategies first predict the missing user-item interactions (i.e., ratings) using e.g., collaborative filtering techniques [1], then aggregate ratings across group members to derive group scores. Compared with preference aggregation, score aggregation offers better flexibility and more opportunities for improvement [3], and thus receives more research attention. Popular score aggregation strategies include the average [4, 28], offering an overall group desirability; the minimum, a.k.a. least misery [4, 28], not strongly displeasing any user; the multiplication, is in between; and the maximum, a.k.a. most pleasure [28], ensuring the greatest pleasure among group members. Under the multiplication and least misery aggregation rules, unsatisfied users have a higher impact on the final decision than satisfied ones. It is the reverse under the most pleasure strategy.

There is line of work that seeks to ensure fairness in group recommendations, so that group members have small disagreement on the recommended items. [23] combines the average aggregation with the standard deviation as a penalty that reflects the amount of variation of individual ratings. Similarly, [3] introduces the concept of consensus, which achieves a balance between group relevance, calculated using either the average or the least misery rules, and group disagreement computed either with the average pairwise disagreement or the variance. [25] defines the individual utility of a user to a list of items as the normalized sum of ratings of all items belonging to the list, then introduces the notion of social welfare as the overall utility of all group members to the list, which is combined with a fairness measure. Four fairness semantics are considered: namely, least misery, variance, Jain's fairness and Min-Max Ratio. The main difference between this method and the two previous works is that the utility is determined by how a list of items, instead of a single item, is relevant to a user.

[39] considers the N-level Pareto optimal set of items, which comprises the items that are dominated by at most N-1 other items, and selects top-N among them by considering a large number of ways in which a group may reach a decision. More specifically, this procedure generates a large number of random weight vectors, each representing a different aggregation strategy, and counts how many times each item is ranked within the top-N, then items are ranked based on their counts. A variant that chooses the smallest $x \in [1, N]$ such that there are at least N items in the x-level Pareto optimal set is also studied.

Some recent work focus on recommending packages, i.e., a set of items rather than a ranked list of items. This is practical in some cases, e.g., suggesting points of interest (museum, park, restaurant, etc.) to a group of tourists. [35, 36] proposes probabilistic models that captures the relevance of items/packages to a group. The notion of fairness is also investigated. More precisely, a package is fair for a group member if it comprises at least one item ranked in the top-$\Delta\%$ of the member ratings on all items, then the fairness of a package is the proportion of members for which it is fair. The probabilistic relevance and the fairness are aggregated to identify a package that is both attractive and fair. [41] considers two alternative definitions of fairness. The first in an extension of the previously discussed fairness measure, called m-proportionality, which is the proportion of group members for which at least m items in the package are in their top-$\Delta\%$ items. The second is m-envy-freeness, which is the proportion of group members for which there is at least m items in the package among the top-$\Delta\%$ ratings of all group members, in this package. This works can be adapted to the case of single items recommendation by setting the size of the package to 1.

Different from these prior works, our approach takes into account the fact that the ratings are subjective and users rate in a different manner. In addition, it preserves (disfavors) items that are highly liked (disliked) by most group members. Further, except for XPO [39], the produced recommendations include unanimously dominated – and thus less relevant – items.

Skyline Query Computation. The concept of skyline query was introduced by Börzsönyi et al. into the database community in [9], where several algorithms were proposed. The most well-known method is the Block Nested Loop (BNL), which iterates over the dataset, comparing each tuple against every other, and reports those are not dominated. Sort First Skyline (SFS) [12, 13] improve BNL by presorting the dataset according to a monotone aggregation function, reducing the number of dominance checks. Sort and Limit Skyline algorithm (SaLSa) [5, 6] infuses a stopping condition into SFS for early termination. Other algorithms operate on precomputed indexes on the dataset, see, e.g., Bitmap [42], Index [42], a B+ tree-based algorithm, Nearest Neighbor (NN) [24] and Branch and Bound Skyline (BBS) [33, 34] are R-tree bases-algorithms. However, index structures suffer from dependency to data dimensionality, and their performance becomes increasingly degraded with the increase of the dimensionality. Even if these algorithms can be adapted to solve our problem, they are very time-consuming as shown in Section 4. Our algorithm employs effective pruning techniques to compute the k-dual adjustment unanimous skyline efficiently.

Top-k Threshold Algorithms. The family of threshold algorithms, which were originally proposed in a multimedia context [15, 16], operates on sorted index lists and aims to retrieve the top-k set according to a monotone scoring function without scanning the whole dataset. The main idea is to maintain the worst score among the top-k results retrieved so far, and the best possible score for all unseen objects, which serves as a threshold for stopping the algorithm when no unseen object can exceed the score of the currently k^{th} object. Since then, threshold algorithms were adapted in many contexts, e.g., relational databases [2, 14, 32], web-accessible databases [10, 27], XML [26, 43] and so on. In this work, we draw inspiration from this family of algorithms to procure a termination condition for our SLA algorithm.

6 CONCLUSION

In this paper, we studied the problem of recommending items to occasional groups. We first introduced a novel group recommendation score, which measures the relevance of an item to a group by taking into account the fact that users rate differently, using a dual adjustment aggregation. Our score has the particularity of preserving (resp. disfavoring) items that are highly liked (resp. disliked) by most group members. We then proposed the notion of k-dual adjustment unanimous skyline, which comprises the k unanimously preferred items with the highest group recommendation score. For computing the k-dual adjustment unanimous skyline, we presented two baseline methods by adapting prior work, and also proposed a more efficient algorithm based on effective pruning strategies. We performed a theoretical analysis to study the average computational complexity of the different algorithms. Our experimental results on real-world datasets showed that our recommendation scheme is superior to state-of-the-art group recommendation methods. We also investigated the scalability of the k-dual adjustment unanimous skyline algorithms, varying the problem parameters. We saw that our algorithm is more efficient than the adapted ones.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 17, 6 (2005), 734–749. https://doi.org/10.1109/TKDE.2005.99

[2] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. 2007. Best Position Algorithms for Top-k Queries. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Christoph Koch, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Florescu, Chee Yong Chan, Venkatesh Ganti, Carl-Christian Kanne, Wolfgang Klas, and Erich J. Neuhold (Eds.). ACM, Vienna, Austria, 495–506. http://www.vldb.org/conf/2007/papers/research/p495-akbarinia.pdf

[3] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawla, Gautam Das, and Cong Yu. 2009. Group Recommendation: Semantics and Efficiency. *Proceedings of the VLDB Endowment (PVLDB)* 2, 1 (2009), 754–765. https://doi.org/10.14778/1687627.1687713

[4] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. 2010. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker (Eds.). ACM, Barcelona, Spain, 119–126. https://doi.org/10.1145/1864708.1864733

[5] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. 2006. SaLSa: computing the skyline without scanning the whole sky. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, Philip S. Yu, Vassilis J. Tsotras, Edward A. Fox, and Bing Liu (Eds.). ACM, Arlington, Virginia, USA, 405–414. https://doi.org/10.1145/1183614.1183674

[6] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. 2008. Efficient sort-based skyline evaluation. *ACM Transactions on Database Systems (TODS)* 33, 4 (2008), 31:1–31:49. https://doi.org/10.1145/1412331.1412343

[7] Jon Louis Bentley, H. T. Kung, Mario Schkolnick, and Clark D. Thompson. 1978. On the Average Number of Maxima in a Set of Vectors and Applications. *Journal of the ACM (JACM)* 25, 4 (1978), 536–543. https://doi.org/10.1145/322092.322095

[8] Shlomo Berkovsky and Jill Freyne. 2010. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker (Eds.). ACM, Barcelona, Spain, 111–118. https://doi.org/10.1145/1864708.1864732

[9] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The Skyline Operator. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Dimitrios Georgakopoulos and Alexander Buchmann (Eds.). IEEE Computer Society, Heidelberg, Germany, 421–430. https://doi.org/10.1109/ICDE.2001.914855

[10] Nicolas Bruno, Luis Gravano, and Amélie Marian. 2002. Evaluating Top-k Queries over Web-Accessible Databases. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Rakesh Agrawal and Klaus R. Dittrich (Eds.). IEEE Computer Society, San Jose, CA, USA, 369–380. https://doi.org/10.1109/ICDE.2002.994751

[11] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive Group Recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, Ann Arbor, MI, USA, 645–654. https://doi.org/10.1145/3209978.3209998

[12] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. 2003. Skyline with Presorting. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman (Eds.). IEEE Computer Society, Bangalore, India, 717–719. https://doi.org/10.1109/ICDE.2003.1260846

[13] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. 2005. Skyline with Presorting: Theory and Optimizations. In *Proceedings of the International IIS: IIPWM'05 Conference on Intelligent Information Processing and Web Mining (Advances in Soft Computing, Vol. 31)*, Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, and Krzysztof Trojanowski (Eds.). Springer, Gdansk, Poland, 595–604. https://doi.org/10.1007/3-540-32392-9_72

[14] Gautam Das, Dimitrios Gunopulos, Nick Koudas, and Dimitris Tsirogiannis. 2006. Answering Top-k Queries Using Views. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim (Eds.). ACM, Seoul, Korea, 451–462. http://dl.acm.org/citation.cfm?id=1164167

[15] Ronald Fagin. 1999. Combining Fuzzy Information from Multiple Systems. *Journal of Computer and System Sciences (JCSS)* 58, 1 (1999), 83–99. https://doi.org/10.1006/jcss.1998.1600

[16] Ronald Fagin, Amnon Lotem, and Moni Naor. 2003. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences (JCSS)* 66, 4 (2003), 614–656. https://doi.org/10.1016/S0022-0000(03)00026-6

[17] Parke Godfrey. 2004. Skyline Cardinality for Relational Processing. In *Proceedings of the Third International Symposium on Foundations of Information and Knowledge Systems (FoIKS) (Lecture Notes in Computer Science, Vol. 2942)*, Dietmar Seipel and Jose Maria Turull Torres (Eds.). Springer, Wilhelminenberg Castle, Austria, 78–97. https://doi.org/10.1007/978-3-540-24627-5_7

[18] Parke Godfrey, Ryan Shipley, and Jarek Gryz. 2005. Maximal Vector Computation in Large Data Sets. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi (Eds.). ACM, Trondheim, Norway, 229–240. http://www.vldb.org/archives/website/2005/program/paper/tue/p229-godfrey.pdf

[19] Parke Godfrey, Ryan Shipley, and Jarek Gryz. 2007. Algorithms and analyses for maximal vector computation. *The International Journal on Very Large Data Bases (VLDBJ)* 16, 1 (2007), 5–28. https://doi.org/10.1007/s00778-006-0029-7

[20] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19:1–19:19. https://doi.org/10.1145/2827872

[21] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Wei Cao. 2014. Deep Modeling of Group Preferences for Group-Based Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Carla E. Brodley and Peter Stone (Eds.). AAAI Press, Québec City, Québec, Canada, 1861–1867. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8458

[22] Nicolas Hug. 2017. Surprise, a Python library for recommender systems. http://surpriselib.com.

[23] Anthony Jameson and Barry Smyth. 2007. Recommendation to Groups. In *The Adaptive Web, Methods and Strategies of Web Personalization*, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Lecture Notes in Computer Science, Vol. 4321. Springer, Verlag Berlin Heidelberg, 596–627. https://doi.org/10.1007/978-3-540-72079-9_20

[24] Donald Kossmann, Frank Ramsak, and Steffen Rost. 2002. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, Hong Kong, 275–286. https://doi.org/10.1016/B978-155860869-6/50032-9

[25] Xiao Lin, Min Zhang, Yongfeng Zhang, Zhaoquan Gu, Yiqun Liu, and Shaoping Ma. 2017. Fairness-Aware Group Recommendation with Pareto-Efficiency. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, Paolo Cremonesi, Francesco Ricci, Shlomo Berkovsky, and Alexander Tuzhilin (Eds.). ACM, Como, Italy, 107–115. https://doi.org/10.1145/3109859.3109887

[26] Amélie Marian, Sihem Amer-Yahia, Nick Koudas, and Divesh Srivastava. 2005. Adaptive Processing of Top-K Queries in XML. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Karl Aberer, Michael J. Franklin, and Shojiro Nishio (Eds.). IEEE Computer Society, Tokyo, Japan, 162–173. https://doi.org/10.1109/ICDE.2005.18

[27] Amélie Marian, Nicolas Bruno, and Luis Gravano. 2004. Evaluating top-k queries over web-accessible databases. *ACM Transactions on Database Systems (TODS)* 29, 2 (2004), 319–362. https://doi.org/10.1145/1005566.1005569

[28] Judith Masthoff. 2015. Group Recommender Systems: Aggregation, Satisfaction and Group Attributes. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). Springer, Science+Business Media New York, 743–776. https://doi.org/10.1007/978-1-4899-7637-6_22

[29] Joseph F. McCarthy and Theodore D. Anagnost. 2000. MUSICFX: an arbiter of group preferences for computer supported collaborative workouts. In *Proceeding on the ACM 2000 Conference on Computer Supported Cooperative Work (CSCW)*, Wendy A. Kellogg and Steve Whittaker (Eds.). ACM, Philadelphia, PA, USA, 348. https://doi.org/10.1145/358916.361976

[30] Tien T. Nguyen, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan. 2018. User Personality and User Satisfaction with Recommender Systems. *Information Systems Frontiers* 20, 6 (2018), 1173–1189. https://doi.org/10.1007/s10796-017-9782-y

[31] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, 188–197. https://doi.org/10.18653/v1/D19-1018

[32] HweeHwa Pang, Xuhua Ding, and Baihua Zheng. 2010. Efficient processing of exact top-k queries over disk-resident sorted lists. *The International Journal on Very Large Data Bases (VLDBJ)* 19, 3 (2010), 437–456. https://doi.org/10.1007/s00778-009-0174-x

[33] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2003. An Optimal and Progressive Algorithm for Skyline Queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, Alon Y. Halevy, Zachary G. Ives, and AnHai Doan (Eds.). ACM, San Diego, California, USA, 467–478. https://doi.org/10.1145/872757.872814

[34] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2005. Progressive skyline computation in database systems. *ACM Transactions on Database Systems (TODS)* 30, 1 (2005), 41–82. https://doi.org/10.1145/1061318.1061320

[35] Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2016. Recommending Packages to Groups. In *IEEE International Conference on Data Mining (ICDM)*, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu (Eds.). IEEE Computer Society, Barcelona, Spain, 449–458. https://doi.org/10.1109/ICDM.2016.0056

[36] Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2018. Recommending packages with validity constraints to groups of users. *Knowledge and Information Systems (KAIS)* 54, 2 (2018), 345–374. https://doi.org/10.1007/s10115-017-1082-9

[37] Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca. 2016. Recommending New Items to Ephemeral Groups Using Contextual User Influence. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, Shilad Sen,

Werner Geyer, Jill Freyne, and Pablo Castells (Eds.). ACM, Boston, MA, USA, 285–292. https://doi.org/10.1145/2959100.2959137

[38] Inbal Ronen, Ido Guy, Elad Kravi, and Maya Barnea. 2014. Recommending social media content to community owners. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Shlomo Geva, Andrew Trotman, Peter Bruza, Charles L. A. Clarke, and Kalervo Järvelin (Eds.). ACM, Gold Coast , QLD, Australia, 243–252. https://doi.org/10.1145/2600428.2609596

[39] Dimitris Sacharidis. 2019. Top-N group recommendations with fairness. In *Proceedings of the ACM/SIGAPP Symposium on Applied Computing, (SAC)*, Chih-Cheng Hung and George A. Papadopoulos (Eds.). ACM, Limassol, Cyprus, 1663–1670. https://doi.org/10.1145/3297280.3297442

[40] Christophe Senot, Dimitre Kostadinov, Makram Bouzid, Jérôme Picault, Armen Aghasaryan, and Cédric Bernier. 2010. Analysis of Strategies for Building Group Profiles. In *Proceedings of the International Conference On User Modeling, Adaptation And Personalization (UMAP) (Lecture Notes in Computer Science, Vol. 6075)*, Paul De Bra, Alfred Kobsa, and David N. Chin (Eds.). Springer, Big Island, HI, USA, 40–51. https://doi.org/10.1007/978-3-642-13470-8_6

[41] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2017. Fairness in Package-to-Group Recommendations. In *Proceedings of the International World Wide Web Conference (WWW)*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, Perth, Australia, 371–379. https://doi.org/10.1145/3038912.3052612

[42] Kian-Lee Tan, Pin-Kwang Eng, and Beng Chin Ooi. 2001. Efficient Progressive Skyline Computation. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass (Eds.). Morgan Kaufmann, Roma, Italy, 301–310. http://www.vldb.org/conf/2001/P301.pdf

[43] Martin Theobald, H. Bast, Debapriyo Majumdar, Ralf Schenkel, and Gerhard Weikum. 2008. TopX: efficient and versatile top- k query processing for semistructured data. *The International Journal on Very Large Data Bases (VLDBJ)* 17, 1 (2008), 81–115. https://doi.org/10.1007/s00778-007-0072-z

[44] Lucas Vinh Tran, Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, Gao Cong, and Xiaoli Li. 2019. Interact and Decide: Medley of Sub-Attention Networks for Effective Group Recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (Eds.). ACM, Paris, France, 255–264. https://doi.org/10.1145/3331184.3331251

[45] Hongzhi Yin, Qinyong Wang, Kai Zheng, Zhixu Li, Jiali Yang, and Xiaofang Zhou. 2019. Social Influence-Based Group Representation Learning for Group Recommendation. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. IEEE, Macao, China, 566–577. https://doi.org/10.1109/ICDE.2019.00057

[46] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. 2006. TV Program Recommendation for Multiple Viewers Based on user Profile Merging. *User Modeling and User-Adapted Interaction* 16, 1 (2006), 63–82. https://doi.org/10.1007/s11257-006-9005-6