Chapter 6

A study on the role of uninterested items in group recommendations

This chapter is based on our journal article published in [69]. A recommender system recommends items that users might like from a large set of available items. Recommendation systems mainly fall into two categories: 1) content-based [47, 76, 90] and 2) collaborative filtering [43, 66, 133]. Content-based recommender systems work with implicit or explicit data provided by a user. A user profile is generated based on this available data. The user profile helps to recommend items similar to the items that the user had chosen earlier. Based on the content, a recommendation is made to the users. The collaborative filtering approach works on the paradigm - the wisdom of the crowd. Collaborative filtering [66] is used to match the user to the items based on the choices of other similar users and vice-versa. It calculates the similarities between users or items to rate an item. This approach relies on the fact that users who had provided similar preferences in the past are likely to act similarly in the future. Group recommender systems integrate individual user's preferences and recommend items in order to satisfy their shared taste. A group recommender system provides items that a group might like by merging individual choices [55, 77].

In practice, recommendations are generated by considering only common interest items, with the remaining items not playing a role in the recommendation. The items that are not of interest to a user based on her implicit or explicit feedback are called uninterested items. We hypothesize that these uninterested items may influence the overall recommendations, which can help us better interpret users' preferences. In this chapter, we also include the importance of uninterested items of the group members while

generating the recommendation list. We have performed experiments over automatically identified groups and random groups to check the efficacy of the proposed models. We conducted experiments on publicly available real-world datasets. We found that incorporating uninterested item in group recommendations play a positive role in group recommendations while considering order and flexibility in user preferences. We observed that the overall group satisfaction scores are better in an automatically identified group.

The uninterested items are equally important as interested items. In [39], the authors propose a COllaborative Full Feedback (COFFee) model that helps to alleviate rating elicitation problems. Authors consider user feedback as a categorical variable and model it with users and products in a triple way to provide meaningful recommendations. These tuples are decomposed using tensor factorization based on tucker decomposition, and it gives their corresponding relevance score. The model provides a granular view of all possible ratings, and the ranking of recommended items are easily suggestible. The model is helpful in Top-n recommendation where the system incorporates negative feedback. The authors also propose an evaluation metric (Discounted Cumulative Loss (DCL)) that helps to become sensitive to negative feedback. Later, in [10], authors propose a model to recommend products based on taking the explicit feedback for an item using conversation, like, features of the items that a user prefer. Additionally, when the users provide negative feedback for the suggested items, it becomes easy to accumulate detailed feedback on certain latent features. The model updates its strategies in making the relevant product recommendations.

In [123], the authors propose a model for a next-item recommendation. The model requires positive feedback on the selected items but ignores negative feedback on the unselected items. In e-commerce sites, users add some recommended items to the cart and discard few items. The negative items are those that were recommended by the sites but are not considered by the user. The model is helpful in sequential, next-basket and session-based recommendations. However, majority of traditional recommender systems do not consider negative feedback (a critical factor in the recommendation process) to improve the quality of recommender systems. In group recommendation, this work shows that a key issue is to aggregate preferences of different group members by incorporating uninterested items in the recommendation process.

The common list of interested items is considered as positive feedback. It requires

aggregating the individual preferences of the members of the group to generate final recommendations. Uninterested user preferences become more important when the group recommendation is designed mainly to avoid group members' dislike. Users give their choices of items that they do not prefer and are reluctant to have them present in the final recommendation. The uninterested user preferences may be directly on attributes of the dataset or metadata like movie genres, music length, type of food, etc. Since users do not consider some uninterested items during the recommendation process, these uninterested items may also play a significant role in influencing users' feedback behaviour [10, 26, 39, 96, 102, 123]. A group will have users with different tastes, and some users might dislike genres (in the case of movies) that others like. The role of uninterested items may be influential when a user dislikes only some of the items. We might not want to overlook the impact of the uninterested items in user preferences. Hence, it becomes interesting to incorporate the uninterested items in the process of decision making. The recommendation by aggregating users' preferences after considering uninterested items will impact the group's decision as a whole.

This chapter first captures the uninterested items from user preferences using the dataset and shows that introducing the uninterested items in group recommendations has a positive effect. We have performed experiments on automatically identified group (AIG) [15, 16] and random group (RG) to check the efficacy of the proposed models. Automatically identified groups are formed by incorporating swarm intelligence in K-means technique [134]; an optimization technique to cluster similar users and a community detection based algorithm [13]. This chapter investigates the importance of uninterested items in both a random group and an automatically identified group. We conducted extensive experiments over proposed models on MovieLens and Amazon-music datasets, which corroborate our claim.

Scope and limitations: The proposed work is useful when explicit user preferences are given and also when users wish to give their preferences of variable sizes. The proposed model plays a pivotal role in existing group recommender systems on various domains like movies, music, travel, food, etc. The proposed work is unsuitable when considering social phenomena like emotional contagion, conformity, etc., in making a recommendation because the proposed models do not consider these social factors as inputs. The proposed model can not alleviate the cold start problem where a system has no interactions his-

tory for new users or items. In this scenario, it is not easy to produce relevant group recommendations without asking the group members to provide initial feedback on some products.

The key contributions of this chapter are summarized in three-folds:

- This chapter proposes models for group recommendations by incorporating the importance of uninterested items captured using user preferences.
- We studied the effectiveness of proposed models based on a randomly formed group (random group) and an "automatically identified group" [15, 16].
- We adopted particle swarm optimisation [134], Louvain algorithm [13] and Gaussian mixture model [106] techniques to detect and compose "automatically identified groups" in group recommendations.

Outline: The rest of the chapter is organized as follows. Section 6.1 presents the proposed models and a brief overview of traditional clustering approaches. Section 6.2 presents results and discussions, and section 6.3 summarizes the conclusion of this chapter.

6.1 Models

6.1.1 Proposed models

Order and flexible-size preferences incorporating uninterested item-based model: Existing models do not consider the role of commonly uninterested items in producing group recommendations. In this chapter, we propose models incorporating uninterested items in group recommendations and adapt existing models for order and flexible-size preferences. In all the existing models, the recommendation is generated by considering the items of common interest before generating a final vector. Here, we propose to generate a recommendation vector by including the importance of uninterested items from user preferences. The proposed model is tested on the MovieLens (ml-100k) and Amazon-music datasets. The users' preferences along with metadata are available. The system converts each item into a genre binary vector and then accumulates all the genre vectors to produce the genre vector for a user. Now the information about users' genres interest

Algorithm 3: Procedure for computing dissatisfaction of an item

Input: **lis**, i, $urec$, **MG**[$item$]

Output: dissatisfaction of an item as $total\_sum/sizeof$(**lis**)

```
1 total_sum = 0;
2 for 0 ≤ i < len(lis) do
3     total_sum = total_sum+1-jaccard_similarity(urec[i], MG[item]) ;
4 end
```

Algorithm 4: Computing satisfaction of individual user in order or flexible-size preferences incorporating uninterested item-based model

Input: i, **lis**[**i**], **r**, **item**

Output: satisfaction score as t_{sum}

```
1 t_sum = 0;
2 for 0 ≤ p < len(r) do
3     item = r[p];
4     if (item in lis[i]);
5      then
6         idx = lis[i].index(item);
7         t_sum = t_sum + s[abs(p − idx)] + 1 − DISSATISFACTION(item);
8     end
9 end
```

is available. Taking negation of this gives the genre vector that the user is least interested in.

In this scenario, when we include an item into the final recommendation list, we also check its similarity with the uninterested vector of every user (every user has an uninterested vector). Our proposed models take number of users (n), total number of items (m), group budget (k) and user preference vector (lis) as inputs. After the preference acquisitions of users, these are taken as inputs into the proposed model to generate the final recommendation vector (r) of size k.

Example 1: Number of users (n): 3

Number of items (m): 7

Group budget (k): 5

Table 6.1: Movie-genre matrix (MG)

Movie	Genres				
	Horror	Action	Thriller	Drama	Sci-fi
movie1	0	1	0	0	1
movie2	1	0	1	0	0
movie3	0	1	1	1	0
movie4	0	1	0	1	0
movie5	0	0	0	0	1
movie6	1	1	0	0	1
movie7	0	1	0	1	0

The order preferences given by the users are as follow:

User1: {2, 5, 1 , 7, 3}

User2: {6, 4, 7, 1, 2}

User3: {5, 1, 4, 3, 2}

In the above example, each member of the group has unique preferences in order. As in a real-world scenario, a user provides ratings to distinct items. We have also assumed the same in the proposed work. In a recommendation application, mainly two types of information exist, viz., rating and item meta-data information. We assume that a movie can belong to different genres. Accordingly, we represent this auxiliary information as a matrix. Let a movie-genre matrix (Table 6.1) of dimension 7×5 where an entry '1' specifies that the movie belongs to the genre, and '0' means that it does not come under the genre. In an order preference model, the value of a particular cell of the user preference matrix (U) signifies the number of users who have opted for this order according to the given user preferences.

We accumulate all the movie-genre vectors and generate an uninterested genre vector for every user. We take a urec vector of a user of dimension 5, and it initializes with zeros. To generate the urec vector of user1, we look up the preferences for user1. In *Example 1*, user1 has preferred items {2,5,1,7,3}. We take the first preference of user1, and the corresponding genre vector is [1,0,1,0,0]. Then, we take the second preference of user1 and her corresponding genre vector, i.e., [0,0,0,0,1], to perform an element-wise sum operation. The updated urec vector is [1,0,1,0,1].

In the next iteration, we take the third preference of user1 and the corresponding genre vector to perform an element-wise sum operation in the last updated urec vector.

Table 6.2: Uninterested genre vector (urec) for the users

User1	0	1	0	0	0
User2	0	1	0	0	0
User3	0	1	0	0	0

Table 6.3: User preference matrix (U)

1	2	3	4	5	6	7
0	1	0	0	1	1	0
1	0	0	1	1	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
0	2	1	0	0	0	0

We repeat the procedure for all the select items of user1. After accumulating all the prefered items' corresponding genre vectors of user1, we get a urec vector of user1 with values [1,3,2,2,2]. There are five different genres associated with each item. Further, we normalise the generated urec vector for user1 by dividing it by value five. We get values between 0 to 1 (inclusive 1). So, the urec vector for user1 contains value [0.2, 0.6, 0.4, 0.4, 0.4]. We took a threshold value of 0.5 (we choose the threshold empirically), which maps these real numbers to 1 and 0. The value is greater than or equal to 0.5 maps to 1, while the value is less than the 0.5 maps to 0. Finally the generated urec vector for user1 is [0,1,0,0,0]. We follow the same procedure to evaluate the uninterested genre vector for other users. Table 6.2 depicts the disliking genre vector (urec) for the users, where '1' specifies that the user likes that genre and vice-versa.

Table 6.3 shows the user preference matrix of size 5×7, where columns correspond to the items (total users' preferred items) and rows represent positions (order in users' preferences). The user preference matrix is initialised to zero in the case of a flexible-size

Table 6.4: Score vector (s) when $k = 5$

Index	0	1	2	3	4
score	6	4.54	3.89	3.5	3.24

Table 6.5: User satisfaction matrix (S)

Position	Items						
	1	2	3	4	5	6	7
1	12.43	12.48	7.07	8.93	10.54	6.33	0.5
2	14.93	11.54	7.72	11.04	10.54	4.87	0.5
3	15.58	11.67	8.76	11.04	8.43	4.22	0.5
4	14.93	12.58	10.87	8.93	7.39	3.83	0.5
5	12.43	15.24	10.87	7.89	6.74	3.57	0.5

preference model. Each *item_id* has a unique value that lies between 1 and m. If a user has a preference vector of length p, i^{th} preference to the *item_id* j, the entry at user preference matrix ($U_{i,j}$) is incremented by a factor of $\frac{k}{p}$. In the flexible-size preference model, the user preference matrix (U) holds real numbers.

Once user preference matrix U is generated, we use score vector s (Table 6.4) of dimension k, to compute the user satisfaction matrix S of size $k \times m$, by using Equation 6.1. Score vector s has values based on user satisfaction with order [1] function. We took variable a and regularization factor c as 2 and 1, respectively. It determines value based on the index by taking the absolute difference between the position of the item i of the user preference vector and the recommendation vector's item position p. The lower the difference, the higher the value it contains.

$$S_{i,j} = \sum_{p=1}^{k} s_{|p-i|} \times U_{p,j} \tag{6.1}$$

Table 6.5 represents the user satisfaction matrix. It denotes the group satisfaction score of an item j when placed at position i in the recommendation list. Here, the calculation of the user satisfaction matrix is similar to an order [1], or flexible-size preference models [35]. Additionally, we have deducted the dissatisfaction score of an item to get the user satisfaction value. Our proposed group recommendation algorithms use the user satisfaction score matrix S to produce the recommendation vector. It contains the score (weights of the group members) according to the items in a recommendation list.

Let us consider a consensus function greedy aggregated method (GRAM) to recommend a group. The greedy aggregated method greedily chooses an item in each iteration

from the user satisfaction matrix S and adds that item into a recommendation vector. In the *Example 1*, item 2 at position 5 provides maximum group satisfaction from the user satisfaction matrix (TABLE 6.5) in the first iteration. Once we add this item to the recommendation list, all the entries correspond to the column (item) index and correspond to the row (position) index of the user satisfaction matrix holds values 0. These entries do not participate in further iterations. The item 1 is added in the position 3 of the recommendation list in the second iteration. We perform it iteratively and continues until a full 'k' recommended list of items are generated. The average group satisfaction is the ratio of the overall group satisfaction [1] to the number of users in a particular group. This method generates the output [5, 4, 1, 3, 2] with overall group satisfaction=64.59 and average group satisfaction=21.53.

Initially, the individual satisfaction score (weight) of each member of the group is considered as zero. Once an item is added to the recommendation list, the weight of members of the group gets updated according to the score vector (s) values (Table 6.4). The individual satisfaction of a user is calculated using Algorithm 4. The group budget is k, $lis[i]$ is the preference vector of a particular user i, r is a recommendation vector of length k and $DISSATISFACTION(item)$ is the dissatisfaction score of an item using *DISSATISFACTION* procedure (Algorithm 3). Accordingly, we update the weights of the group users in further iterations until we get the recommendation list of size k. The overall group satisfaction is calculated by combining the individual satisfaction of all users of the array lis. The length of the array lis is equal to the total number of users.

6.1.2 Automatically identified group

The automatically identified group [15,16] is a group that is automatically detected for the available resources by considering user preferences. For example, users who give similar ratings to the same items come under the same cluster. Since users show higher similarity to each other by providing similar preferences in this kind of group, partitions of similar objects (users) reside in the same cluster. We have formed groups using particle swarm optimization [134], louvain algorithm [13] and Gaussian mixture model [106] techniques. K-means algorithm was used to detect automatically identified groups in group recommendations [15, 16]. When introducing uninterested items in group recommendations, it is good to see the recommendations using an automatically identified group. It becomes

essential to automatically detect such groups from the dataset to get a better-composed group than using existing clustering approaches. We adopted an optimization technique, community detection algorithm and probabilistic approaches to form potential clusters so that members of the groups are maximally satisfied. Uninterested items might increase over group satisfaction score in this case.

Louvain clustering

Louvain clustering algorithm [13] is a modularity based community detection approach. The objective of this algorithm is to optimize the modularity gain using a greedy optimization principle. This algorithm is capable of clustering a vast network in linear time. Firstly, we assign nodes to their community. The introduced modularity function helps to calculate the modularity gain of nodes. The nodes and edges are interconnected based on the modularity gain values of nodes. We repeatedly estimate and update the modularity gain of vertices in the subgraph in every iteration until it converges. Louvain algorithm is a level-wise optimisation method to explore different communities. A particular community corresponds to a cluster.

Gaussian mixture model clustering

Gaussian mixture model (GMM) [106] is an unsupervised learning algorithm. GMM is a flexible probabilistic approach to the clustering problem. It provides a probabilistic model of data. Unlike K-means, the GMM model also helps to group unlabeled data. GMM involves a mixture of multiple Gaussian distributions. K-means handles only circular shapes, while the GMM model handles different shapes like elliptical as well. K-means associates data with cluster centres. In GMM, the variance is taken into account and weights are associated with each Gaussian distribution, but K-means does not consider this. Each cluster is associated with a different Gaussian distribution. Each data point is generated by any of the distributions with the corresponding probability.

Particle swarm optimization based K-means clustering (PSOKmeans)

We applied optimization in the K-means technique to find clusters of points in the dataset that share some common characteristics. Particle swarm optimization technique [134] is inspired by the social behaviour of birds flow or fish school like events. This technique

helps to solve an optimization problem. Each particle has a set of values for the parameters and the initial velocity. We calculate the fitness of the particles by feeding these values in the cost functions. In multiple iterations, we calculate the fitness for each particle to obtain the best fitness value (global best) of the swarm. We calculate the particle velocity by updating the initial velocity and using the local best of the particle and global best of the swarm. Accordingly, we update the particle position until we reach the optimal solution.

Using the techniques mentioned above, the obtained groups use different consensus functions to define models where we have incorporated uninterested items from user preferences. The proposed models based on the automatically identified groups and randomly composed groups get an improvement over our previous models [1, 35].

Modified least misery method incorporating uninterested item (MLMMIUI)

In the modified least misery method incorporating uninterested items, the minimum satisfaction score among all the users is maximized. Initially, the default satisfaction value of every user is marked as 0. It operates in "k" iterations wherein the first iteration, the item that gives the highest group satisfaction is selected and added to the recommendation list. User satisfaction values are updated in the remaining $(k-1)$ iterations before selecting the least misery user. Then the least misery user is assigned, and the item with the highest satisfaction from her preference vector is added to the recommendation list.

Modified least misery method with priority incorporating uninterested item (MLMMPIUI)

Modified Least Misery Method with Priority Incorporating Uninterested Item (MLMMPIUI) is an extension of the Modified Least Misery Method with Priority (MLMMP) that considers uninterested items from user preferences. This method examines similarities among users' interests to resolve the ties in the least misery values. A priority of a user is calculated based on the overlap similarity [1] or Jaccard Similarity [35]. Example2: Suppose there are *n(=7)* users in a group and *m(=8)* items with the group budget of *k(=5)*. Each user preferences are given as follows:

User1: {2,6,7}

User2: {2,3,4,6,7}

User3: {3,8}

User4: {1,2,6,8}

User5: {5,7}

User6: {1,3,4,5,8}

User7: {7}

In the overlap similarity procedure, each user preferences are compared to other users' preferences based on the position of items. The weights are added according to the score vector (Table 6.4). In Example 2, to compute priority score for User1, we calculate the similarity of preference vector lis_1 (correspond to User1's preference vector, i.e, [2 6 7]) to the vectors lis_2,lis_3.....lis_7 pairwise by using either Jaccard Similarity or overlap similarity method. We get the priority[1]=31.29 after summing up all those values. Priority values for all users are given in Table 6.6.

Table 6.6: Priority values for the users in case of modified least misery with priority(MLMMP)

User	1	2	3	4	5	6	7
Priority	31.29	46.24	16.47	32.59	16.08	34.08	11.67

In [1], we form a group based on the similarity in the weights of the users. In the proposed work, we should also calculate the dissatisfaction score that is to be deducted from the group's preference vectors to obtain the users' priority.

Algorithms 3,4 and 5 collaboratively help to calculate the priority of a user. A user's higher similarity gives her top priority in the group. In this method, the member of the group with the highest priority is considered as the least satisfied user for the current state, and the corresponding item that provides the highest group satisfaction is added in the recommendation vector (r_1). After appending an item j at the position i in the recommendation list, the corresponding row and column of the user satisfaction matrix (S) are reset to 0. We enforce this constraint to avoid conflict that arises in the item selection of the next iteration. This procedure works in multiple iterations and continues till the recommended list of items of group budget k is generated.

Algorithm 5: Procedure for computing priority in MLMMPIUI or MMPMP

Input: **lis**, j, l, $\mathbf{r}_1$

Output: priority of user j as sum

```
1 sum = 0;
2 r_1 = sizeof(lis_j);
3 for 0 ≤ l < len(lis) do
4     if (l ≠ j);
5      then
6         sum = sum+COMPUTE_SATISFACTION(l, lis[l], r_1) ;
7     end
8 end
```

Modified greedy aggregation method incorporating uninterested item (MGRAMIUI)

This algorithm uses a greedy approach to select an item for a recommendation. It operates in 'k' iteration. In each iteration, it selects an item j greedily from the user satisfaction matrix(S) for the i^{th} position. Like MLMMPIUI, after selecting the best item for a particular position in every iteration, this method places the minimum value of user satisfaction score matrix, i.e., 0 at the corresponding i^{th} row and j^{th} column of S to avoid conflicts. MGRAMIUI model reduces the computation time and increases total group satisfaction. The average time complexity to produce recommendations using this method is $\mathcal{O}(k^2m + kn)$.

Modified hungarian aggregated method incorporating uninterested item (MHAMIUI)

Modified Hungarian Aggregated Method Incorporating Uninterested Item (MHAMIUI) is an extension of the modified hungarian aggregated method. The approach considers aggregated voting as its base when containing uninterested items from user preferences. In an order preference model, to obtain maximum optimal cost by using the Hungarian method, firstly, each element of the user satisfaction matrix is updated by $(max - S_{i,j})$ to get a resultant matrix. The minimum cost *Hungarian* procedure [1] uses resultant matrix as input to determine an appropriate position i of an item j. The *Hungarian* procedure generates the resultant matrix of dimension $m \times m$ in $\mathcal{O}(m^3)$. In the resultant matrix, the first k rows contain appended items in the recommendation vector. It pro-

vides a better group satisfaction than other consensus function-based approaches, while smaller group sizes and items are taken. The average time complexity to generate group recommendations using this approach is $\mathcal{O}(m^3 + k^2n + kn)$.

Modified most pleasure method with priority (MMPMP)

Modified Most Pleasure Method with Priority (MMPMP) is an add-on of Most Pleasure Method with Priority (MPMP) [35] incorporating uninterested items of user preferences. This group recommendation technique generates the recommendation list based on the choices of the most satisfied user. According to this algorithm, this paradigm looks for the most satisfied user at every epoch based on the current recommendation list and appends an item to the recommendation list. In each iteration, an item and a position are selected from the user satisfaction score matrix (U), so that adding a product in the recommendation vector maximizes group satisfaction. If an item is already selected for a particular position, the corresponding row and column of the user satisfaction score matrix are excluded in the subsequent iterations for considerations. In case of a tie, to choose the most satisfied user, the priority of a user is computed based on a similar calculation as applied in the modified least misery method with priority. The notion of working principle of the modified most pleasure method with priority is opposite to the least misery method with priority [77].

Either taking into account the least misery with priority or the most satisfied with priority incorporating uninterested item-based model, the average time complexity to generate group recommendations using both the approaches are $\mathcal{O}(k^3n + k^2n^2)$.

6.2 Results and discussions

6.2.1 Experimental setup and data-preprocessing

We choose two real-world public datasets- MovieLens (ml-100k)[1] and Amazon-music (Digital Music) [2]. Each movie and music has the corresponding genre vector as meta-data in MovieLens and Amazon-music datasets.

[1]https://grouplens.org/datasets/movielens/

[2]https://jmcauley.ucsd.edu/data/amazon/

MovieLens-100k: The ml-100k dataset is a stable benchmark dataset containing 943 different users (userid) and 1682 different items (movieid) over 100000 ratings. Five is the best score on the rating scale of 1-5. During data cleaning, it was observed that each user rated at least 20 movies. In ml-100k, we dropped timestamp attributes from the dataset during the data preprocessing task as it does not play any role in the experiments. Only three attributes out of four were chosen (i.e., UserID, MovieID, and Rating). The preference vector of the user contains movies having a rating higher than 0 (on a scale of 1-5) given by the user. The order of the items (movies) is the same as they appear in the dataset. In the ml-100k dataset, the item genre vector is explicitly available. We combine all the items and generate a genre vector for the user.

Amazon-music (Digital Music): The digital music dataset has reviews and metadata information from amazon. The digital music dataset has 5148 distinct reviewers and 2582 distinct asin (a unique number assigned to each product (music)). The rating scale (named as an attribute "overall") is 1-5. The total number of reviewers and asin interactions are 51796. An asin is the vector of dimension 461. The order of music remains the same as it appears in the dataset.

6.2.2 Results and analysis

We have conducted extensive experiments over automatically identified groups (AIG) [15, 16] and random group (RG) using different consensus functions to measure the efficacy of the proposed models. Automatically identified groups have been formed using a probabilistic model, an optimization technique and a community detection based algorithm (Louvain algorithm [13]). Random groups have been formed on an ad-hoc basis from the given dataset, and users might be dissimilar to each other. We randomly took a single set of users from this dataset to experiment. We compute the overall group satisfaction of each group. We might not get satisfactory results in the case of a random group. In contrast, considering uninterested items in an automatically identified group increases the overall group satisfaction scores because group members opt for similar preferences in an order. In this case, incorporating the role of the uninterested items in the model is essential to justify our claim. In this kind of study, the generation of the automatically identified group using optimization techniques and probabilistic models is desired. We have adopted these techniques to compose the groups and conducted the

experiments. The cluster constructed by these procedures is less coarse-grained than the existing clustering approaches to identify an automatically identified group.

In the proposed models, we claim that the generated recommendation vectors contain some items dissimilar to the generated recommendation vector of order or flexible-size preference models due to the inclusion of the role of uninterested items. The proposed models produce overall group satisfaction or average group satisfaction score, either equal or slightly more significant than baseline methods [1, 35]. To corroborate our claim, we have performed experiments on MovieLens and Amazon-music datasets. We took values of a and regularization factor c as 2 and 1, respectively, to generate the score vector. The overall group satisfactions have also been computed for both models [1, 35] to investigate the performance of proposed models. We use different consensus functions on order and flexible-size incorporating uninterested item-based models to generate the recommendation list for a random group (RG) and automatically detected group (AIG). The resultant outputs determine that a user of a group is maximally satisfied with the recommended item-set. The obtained results are good enough to improve the quality of group recommendations. We have conducted experiments on both types of groups to see the effect.

The two most popular strategies for score aggregation are the Aggregated Voting (AV) [77], and the Least Misery (LM) [77]. We have used the extensions of these two popular heuristic aggregation strategies in our experiments to check the efficacy of the produced models. The variants of consensus functions use to generate the group score, which is reflected by the interest and preferences of all the group members. We evaluated the recommendation quality by calculating the average group satisfaction for the composed groups. Algorithm 4 helps to determine the average group satisfaction. Users' satisfaction is calculated for recommendation by considering uninterested items from the user preferences in both models. This model also recommends the item for a group by assuming flexibility in user preferences considering the order in the user's preference list and producing recommendations without considering the order. We have evaluated the user satisfaction score on the recommendation vector generated using COMPUTE_SATISFACTION procedure (Algorithm 4). In the computation of individual user satisfaction, the dissatisfaction score of items also requires.

To evaluate the efficacy of the proposed models, the obtained overall group sat-

isfaction of the proposed models compare with the estimated group satisfaction using generated recommendation vectors of order [1] or flexible-size preference model [35]. The proposed models create a recommendation vector that produces overall group satisfaction. The order or flexible-size preference model is used to conduct a comparative study that generates a recommendation vector and provides total group satisfaction based on the proposed model's COMPUTE_SATISFACTION procedure (Algorithm 4). Our objective is to study the inclusion of the role of uninterested items in the proposed models. The generated recommendation vector produces more overall group satisfaction than the generated recommendation vector of the order preference or flexible-size preferences model using the COMPUTE_SATISFACTION procedure.

We measured the performance of proposed models by varying values of Group Budget (k), Group Size (n), and No. of items (m). We have conducted experiments to see the effect in overall group satisfaction or average group satisfaction results for random and automatically identified groups.

Varying group budget (k):

It is evident that an increased value of group budget k provides a greater overall group satisfaction score or average group satisfaction for different group sizes. Figures 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7 and Figures 6.8, 6.9 show this characteristics. Figures 6.10 and 6.11 also depict the same characteristic for a random group and an automatically identified group that have the same cluster size by varying k. It shows that when the group budget (k) increases, the total group satisfaction score increases. Figure 6.12 show that average group satisfaction increases when increasing the value of k in two different group categories.

Varying the number of users (n):

Increasing the number of users in a group introducing uninterested items from user preferences shows the same characteristic in overall group satisfaction or average group satisfaction as fixed or flexible-size preference model produces. The total group satisfaction maximizes by increasing the number of users in a group. Figure 6.9 shows the same characteristics by varying the group size. Figures 6.10, 6.11 and 6.12 show this characteristics when algorithm incorporates importance of uninterested items.

Table 6.7: Total group satisfaction of a random group of cluster size(n=50, m=1682) using GRAM and MGRAM on ml-100k dataset without order preferences

Group Budget(k)	GRAM		MGRAM	
	Uninterested items	Without order	Uninterested items	Without order
5	86.6	78	83.2	72.4
10	347.7	271.3	286.3	218.5
15	851.9	612.9	701.1	523.9
20	1648.6	1160.7	1358.3	995.8
25	2298.7	1668.6	1814.6	1337
30	3156.7	2276.4	2658.6	1977.5

Varying the number of items (m):

The overall group satisfaction increases with the increasing number of items (m). LMM, LMMP, GRAM, HAM consider the order in user preferences. Accordingly, MLMM, MPMP, MGRAM and MHAM consider flexibility in user preferences. Including uninterested items in these models provides better group satisfaction when the number of items increases. Figures 6.1and 6.10 depict the same characteristics. Since the generated recommendation vectors are not identical in most proposed models, the produced overall group satisfaction values are not the same. It has also been observed from the experiments that the flexibility in user preferences does not perform so outstanding compare to fixed-size preferences when we include uninterested items from user preferences. The outcome of fixed-size preferences incorporating uninterested items based model is overwhelming in this case.

Figures 6.10 and 6.11 depicts that Hungarian aggregated based models outperform in the case of smaller group sizes and lesser number of items. Here, we have provided comparisons over different consensus functions for automatically identified and random groups. Both formed groups have the same cluster size. We took the number of users (n) and the number of items (m) as 55 and 200, respectively. These results also infer that applying the role of uninterested items in an 'automatically identified group' is preferable than a randomly formed group.

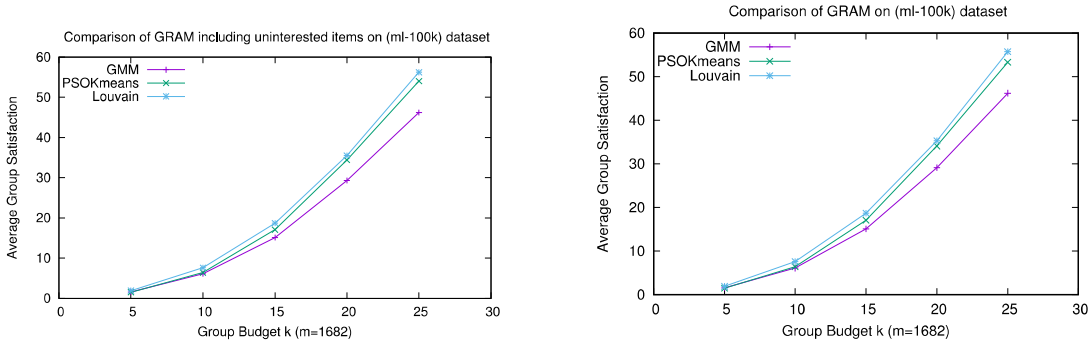


Figure 6.1: Average group satisfaction of cluster size ($55 \leq clusterSize \leq 60$) of an automatically identified group on ml-100k dataset using GRAMIUI and GRAM

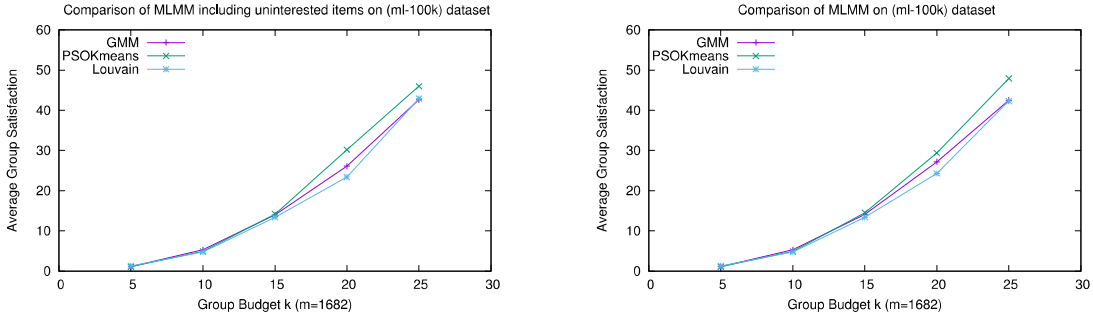


Figure 6.2: Average group satisfaction of cluster size ($55 \leq clusterSize \leq 60$) of an automatically identified group on ml-100k dataset using MLMMIUI and MLMM

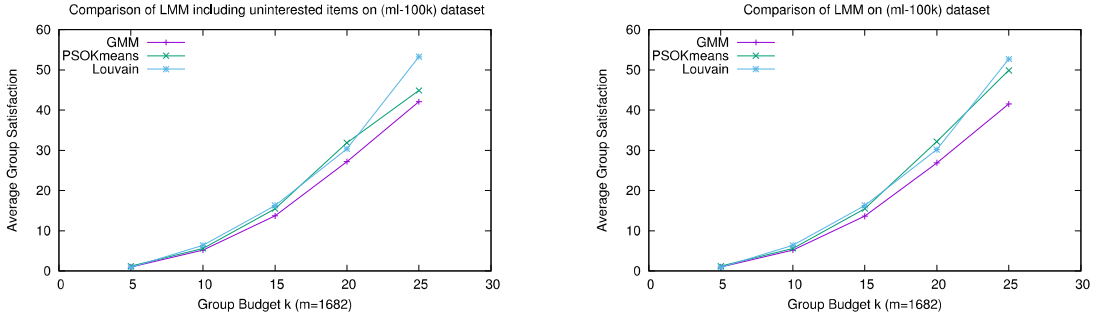


Figure 6.3: Average group satisfaction of cluster size ($55 \leq clusterSize \leq 60$) of an automatically identified group on ml-100k dataset using LMMIUI and LMM

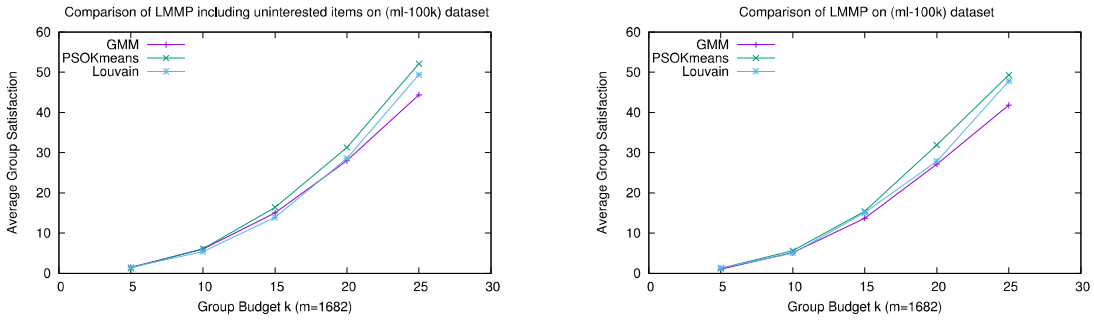


Figure 6.4: Average group satisfaction of cluster size ($55 \leq clusterSize \leq 60$) of an automatically identified group on ml-100k dataset using LMMPIUI and LMMP

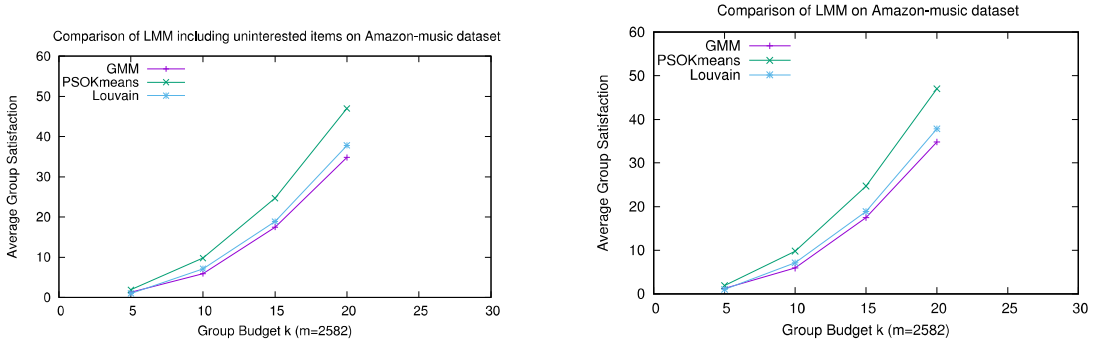


Figure 6.5: Average group satisfaction of cluster size ($50 \leq clusterSize \leq 55$) of an automatically identified group on Amazon-music dataset using LMMIUI and LMM

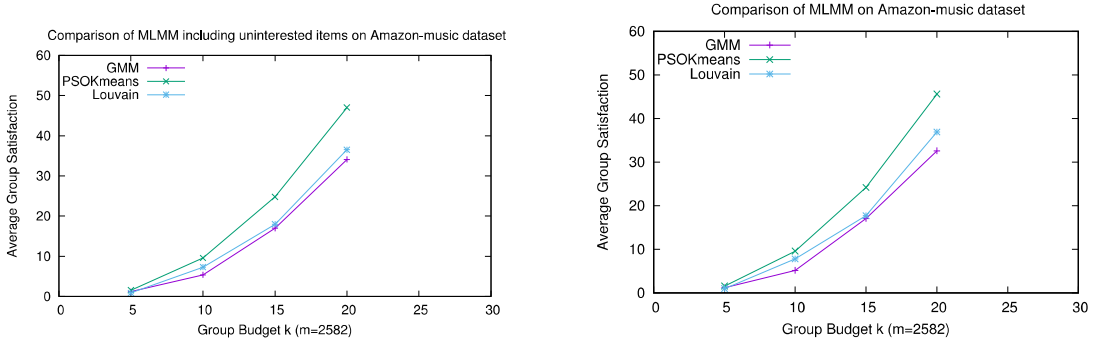


Figure 6.6: Average group satisfaction of cluster size ($50 \leq clusterSize \leq 55$) of an automatically identified group on Amazon-music dataset using MLMMIUI and MLMM

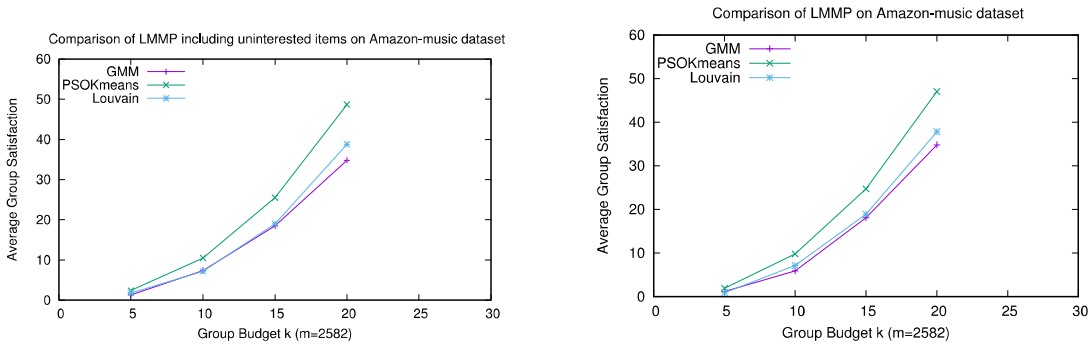


Figure 6.7: Average group satisfaction of cluster size ($50 \leq clusterSize \leq 55$) of an automatically identified group on Amazon-music dataset using LMMPIUI and LMMP

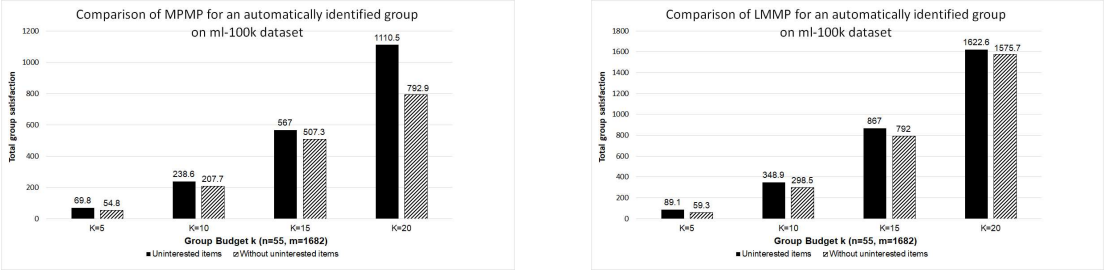
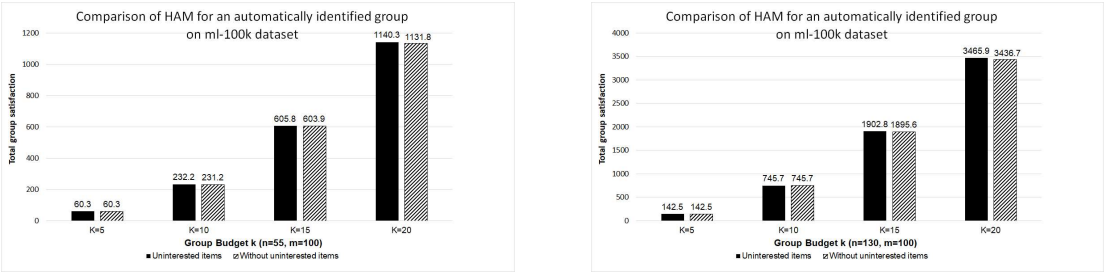
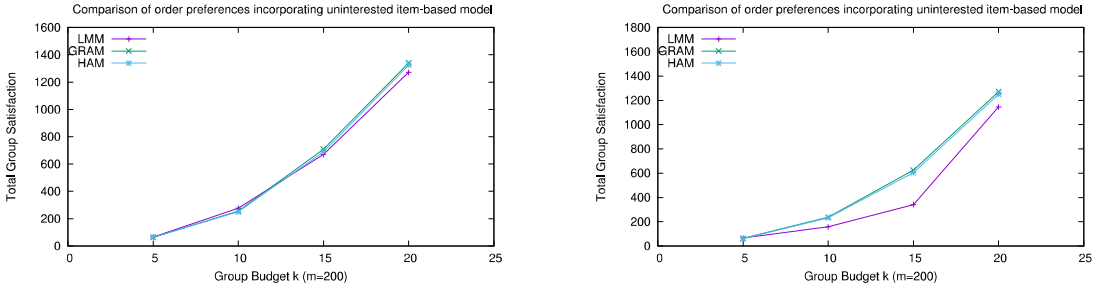


Figure 6.8: Total group satisfaction of an automatically identified group of cluster size (n=55 and m=1682) using MPMP and LMMP on ml-100k dataset.



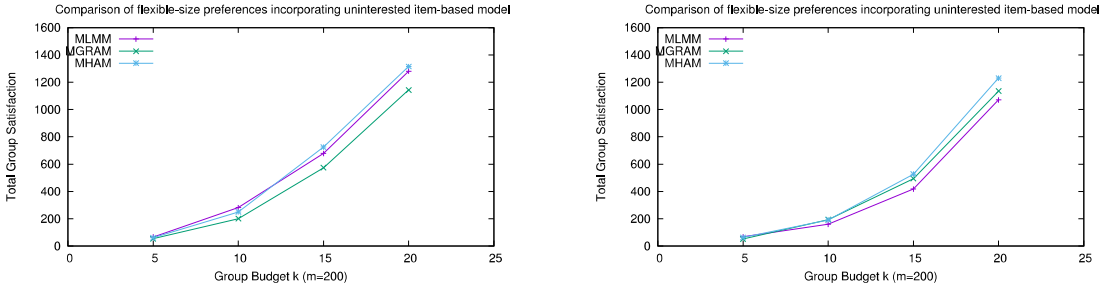
(a) (b)

Figure 6.9: Total group satisfaction of an automatically identified group of cluster size a)(n=55) and b)(n=130) using HAM on ml-100k dataset.



(a) (b)

Figure 6.10: Total group satisfaction using different consensus functions on ml-100k dataset for order preferences incorporating uninterested item-based model (n=55 and m=200) using a) Automatically identified group and b) Random group



(a) (b)

Figure 6.11: Total group satisfaction using different consensus functions on ml-100k dataset for flexible-size preferences incorporating uninterested item-based model (n=55 and m=200) using a) Automatically identified group and b) Random group

Effect on results by considering automatically identified group and random group:

The average group satisfaction increases in both the types of the groups by increasing the values of k, n and m. Figures 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 6.9, 6.10 and 6.11 show these characteristics. In most of the cases, PSOKmeans clustering approach outperforms among all the three clustering approaches when we took variants of aggregated voting and least misery method consensus functions.

Figures 6.10, 6.11 and 6.12 show that an automatically identified group provides a better group satisfaction in comparison to randomly composed groups. We take different clusters with a similar size for the random and automatically identified groups to investigate this behaviour. It shows that in the case of an automatically identified group, the proposed model provides a better result, but the impact of the random group is not so effective. We conducted extensive experiments, and the obtained results are the outcome of a single run. The proposed models' time complexity also shows that greedy based models provide the recommendation in a short period and are computationally efficient than other proposed models. The computed group satisfaction scores using the Hungarian based model is overwhelming. In Table 6.7, the term without order preferences signifies that generated recommendation vector does not consider the order of user preferences in the generated recommendation vector. The system applies the aggregation strategies to the user preferences of the group to produce the recommendation vector for members of a group. The recommendation steps involve sorting the items in increasing order at the end and insert those items in a recommendation vector. The model calculates the overall group satisfaction based on the produced recommendation vector. It is worth noting that the generated recommendation in the order user preferences holds a higher score than the un-order user preferences.

It can be seen from Table 6.7 that the produced overall group satisfaction without order preferences by incorporating uninterested item-based model is minimum in comparison to the order preferences including uninterested item-based model. It is also a point to note from the experiment that the proposed model see the effect of uninterested items correctly when the model considers the entire dataset items so that at least members of an automatic group who gave the preferences would be equal to the group budget.

We have performed an analysis using uninterested items and without using them.

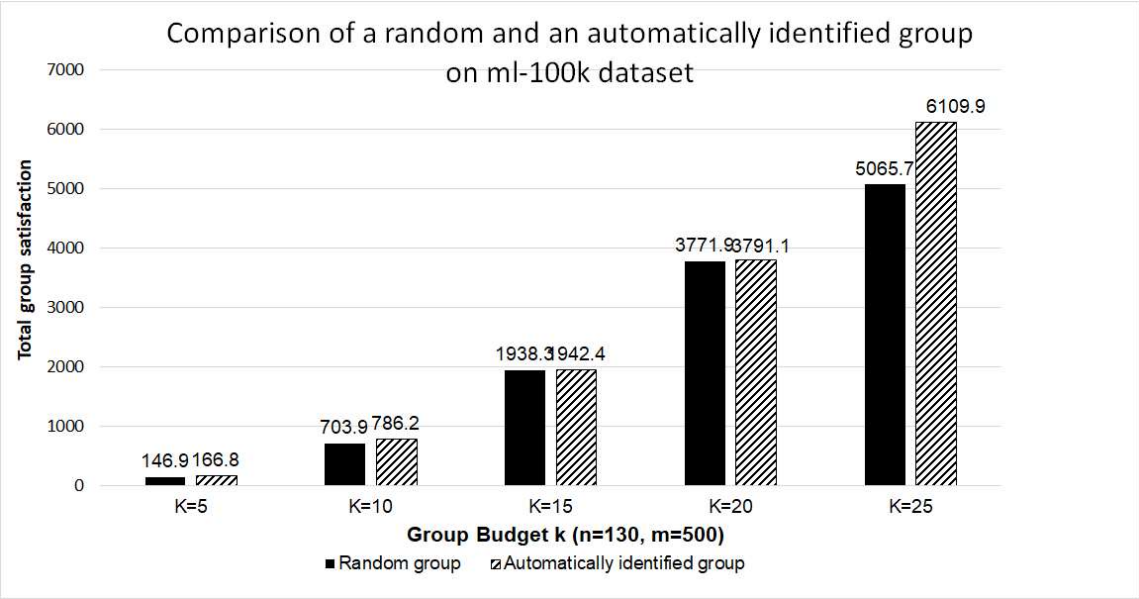


Figure 6.12: Comparison of a random group and an automatically identified group using GRAM incorporating uninterested item-based model (GRAMIUI)

Those users who provide similar preferences to the same items form an automatically identified group. When we take an automatically identified group, the generated recommendation vectors using uninterested items and without them are different. One or more than one produced item is dissimilar in the generated recommendation vector when we include uninterested items. Furthermore, the overall group satisfaction increases in the proposed model. Figures 6.2,6.3, 6.4, 6.5, 6.6 and 6.7 show improvements in the results as the system considers an automatically identified group. While in the case of a random group, most group members have different preferences, and composed group members do not have the preferences in the same order in most cases. So, the overall group satisfaction score is not better than the produced overall group satisfaction score of an automatically identified group. Figure 6.12 shows this characteristics. This study concludes that the group's composition is equally important as the inclusion of uninterested items in group recommendations. The system can only observe the effect of this crucial factor by introducing some novel methods to auto-detect the less coarse-grained cluster or by generating a random group using random samples of the dataset. The overall group satisfaction is better in an automatically identified group compared to a random group.

6.3 Summary

Since social and group activities have evolved in all walks of life, considering only the common list of interested preferences is not the only factor in determining the satisfaction of the group members. Uninterested items can also be taken as a key factor to filter the choices more accurately. This chapter examines the group recommendation by incorporating uninterested items which affect the positive feedback behaviour, such that the satisfaction of each user of the group is maximized. Due to the non-availability of a dataset with negative feedback, we formulate this problem by obtaining uninterested items with the help of other metadata information. We propose models by considering the order in user preferences and also flexibility in user preferences. We adopted the similar consensus functions of our previous works. The proposed algorithm for calculating user priority when group members have flexibility in user preferences has increased the overall group satisfaction scores. The algorithm works to alleviate the problem of breaking the tie when choosing the least satisfied user in each iteration.

The experimental results concluded that in aggregating the users' preferences, uninterested items are also essential for reaching the consensus. The proposed models update the list of recommendations by including the uninterested items. We introduce models in both automatically identified groups and random groups. We employed a probabilistic clustering approach, an optimized enhanced clustering approach and a community detection based approach to compose and auto-detect the group (sub-section 6.1.2). While forming a random group, the system considers random samples of the dataset. We conducted extensive experiments on two real-world datasets that belong to domains of music and movie. The proposed models are also suitable for working on different fields like food, cloth, travel package, etc., where users have provided some preferences on the products/tour-itinerary destinations. We observed that the inclusion of uninterested items in group recommendations is an important aspect. The results depict that including this crucial factor in user preferences impacts the quality of group recommendations positively. The study concludes that group members are maximally satisfied with the recommended item-set in an automatically identified group.