Group Recommendation: An evolution Approach based on Bayesian Networks

Wei Liu [1, 2] and Sheng Feng [1, 2]
1. Beijing University of Posts and Telecommunications, Beijing, China
2. Beijing Key Laboratory of Network System and Network Culture, Beijing, China
Email: twhlw@163.com, shengfeng2008@gmail.com

Daoli Huang
Ministry Third Research Institute of Ministry of Public Security, Shanghai, China
Email: huangdaoli@stars.org.cn

***Abstract*—Recommending to groups is even more complicated than recommending to individuals. Previous works has suggested that when generating recommendations to a group, it can achieve better result by learning information from other groups. Besides, recent research reports indicate that incorporating disagreement is critical to the effectiveness of group recommendation. Although the computation model build with Bayesian networks for group recommender system is very straightforward, the computation is rather complex (even though using approximate technology). In this paper, we will first present a Bayesian networks based evolution group recommendation model where groups can learn from each other. Then, we not only propose a new group recommendation computation framework, but also propose a new satisfaction measure model to refine the group recommendations. We evaluate the performance of our approach on the MovieLens dataset. Experiment results show that our Bayesian networks based evolution approach which is an ensemble of the above three sub-components outperforms the baseline one.**

***Index Terms*—Group Recommendation; Bayesian Networks; Group Satisfaction**

I. INTRODUCTION

A recommender system (RS) supports users to find information, products, or services (such as books, movies, music, digital products, Web sites, and TV programs, to name a few) by aggregating and analyzing suggestions from other users, reviews from various authorities, and user attributes. Collaborative filtering (CF) is known to be a successful recommendation technique. It makes recommendations to users based on other users' ratings on items, putting more weights on those from similar users (i.e., other users having similar personal attributes or product preferences). But to date, recommender systems have focused mainly on recommending items to individuals rather than groups of people intending to participate in a group activity. In recommendation domains such as shopping and asset investment, it is not a limitation because users in general behave individually and only their personal interests should be considered. In other domains such as movies, trips, book clubs, and restaurants, however, existing recommender systems have difficulty in aggregating individual users' tastes into a group's preference properly [1].

A group recommender system (GRS) is a recommender system aimed at generating a set of recommendations that will satisfy a group of users, with potentially competing interests. The challenges associated with this simple statement deal with: considering how to record and combine the preferences of many different users as they engage in simultaneous recommendation dialogs.

Some types of items that a system can recommend (e.g., restaurants and museum exhibits) tend to be used at least as often by groups as individuals, so addressing recommendations to individuals can actually be unnatural. Moreover, the evolution of computers away from the desktop PC makes it increasingly natural for systems to address groups as well as individuals: Wall displays, information kiosks, PDAs, and cell phones can be used easily by persons who are interacting with each other. And even with the traditional PC, users are being offered an increasing variety of ways to communicate with each other and perform tasks together. For these reasons, we can expect a continuing growth in the trend toward recommendation (and, more generally, adaptation) to groups of users.

In paper [2], Judith Masthoff gives a complete discuss about strategy for aggregating models of individual users to allow for group recommendation, what strategies have been used in existing systems, and what researcher have learned from experiments in this area.

In this paper, we will first present a Bayesian networks based evolution group recommendation model where groups can learn from each other. Then, we not only propose a new group recommendation computation framework, but also propose a new satisfaction measure

Manuscript received November 29, 2013; revised December 5, 2013; accepted December 6, 2013.
Copyright credit,
National Basic Research Program of China (No.2010CB734104)
Corresponding author: Wei Liu, twhlw@163.com

doi:10.4304/jmm.9.6.858-864

model to refine the group recommendations. In contrast to the traditional solutions, our satisfaction measure model uses the output of group recommendation model as input rather than use the system input. In short, our solution looks like a feedback model of modern control theory. To the best of our knowledge, it is an approach all of people before us never try. The above three sub-components compose our Bayesian networks-based evolution approach.

This paper is organized as follows: Section 2 overviews related works. Section 3 presents the proposed the evolution group model. The computing method for generating recommendations based on the new model is discussed in Section 4. Section 5 describes how to use the satisfaction model to refine the group recommendations. Performance evaluation is done in Section 6, where our algorithm is compared with the one proposed by Luis et al. [3]. Finally, in Section 7 we conclude this paper.

II. RELATED WORKS

CF makes recommendations based on item ratings by neighbors who are those having attributes or preferences similar to a user to whom recommendation is made. In general, CF systems make recommendation according to following three steps: user profile creation, neighbor formation and recommendation generation [4].

PolyLens [4] by Mark O'Connor et al, a recommender system for groups of users, had been designed to recommend items for groups of users, rather than for individuals. They found that users not only valued group recommendations, but were willing to yield some privacy to get the benefits of group recommendations. Users valued an extension to the group recommender system that enabled them to invite non-members to participate, via email.

George Popescu et al had modeled group recommender systems as a voting problem in facilitate music items recommendations. Their GroupFuns is a Facebook application which attempts to suggest a common set of music items to a group for a social event (e.g., party), with the aim of maximizing the satisfaction of all members in the group on the suggested items. Using a simple algorithm, probabilistic weighted sum, they had defined an incentive-compatible scheme in which scores are interpreted as probabilities. The static and the dynamic cases further contributed to measuring user preference for the deterministic case. This advances previous work carried on for understanding the voting mechanism as well as its dynamics and user choice. Users are free to state their preferences individually as well as modify them according to some group dynamics factor and intermediate common decision. In real-life examples the two cases presented are very frequently encountered and numerous applications stated in the beginning denote the need for adaptive group recommender systems. GroupFun is one of these systems designed for users to spend the least amount of time stating their preferences and be able to reach the common music playlist goal.

In paper [3], Luis et al proposed a collaborative Bayesian networks-based group recommender system (CBBGRS), where the group's rates are computed from past voting patterns of other users with similar tastes. A Bayesian network (BN) [5] [6] is a directed acyclic graph, where the nodes represent the variables from the problem we want to solve. In Figure 1, we give out the topology of CBBGRS. Each user variable U_a represent the probability distribution associated to its pattern of rating, i.e. information about the probability that U_a could vote with value i, $\Pr(U_a = i)$, with $i \in \{1, 2, \ldots, r\}$.

In a collaborative RS, the vote prediction for a given user depends on the votes of the people with similar tastes or preferences. In order to facilitate the presence of these relationships in the model, they included a new of set of nodes V to denote collaborative votes. There is one collaborative node for each user in the system, i.e. $V = \{V_1, V_2, \ldots, V_n\}$. These nodes will also be used to estimate the probability distributions of the user votes and they will therefore take their values in the same domain as U, i.e. $\{1, 2, \ldots, r\}$.

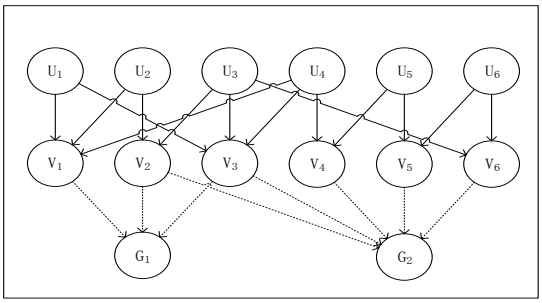

Figure 1. Collaborative Group Recommender system topology

Let's overview the approach in [3]. Note that *ev* and "evidence" mentioned below represent an item which a group may be interesting in.

1. First, they try to find *K* nearest neighborhoods for every member of the group according to the similarity measure such as Pearson's Correlation coefficient and then push those nearest neighborhoods to the $Pa(v_i)$ list of the member's respective collaborative node (*Pa* is the abbreviation of Parent). v_i is an instance of V_i, for example, $V_{i,j}$ is the j^{th} value of V_i.

2. Estimating the conditional probability distributions

$$\Pr(v_{i,j} \mid pa(V_i)) = \sum_{Y_k \in Pa(X_i)} w(y_{k,l}, v_{i,j}) \qquad (1)$$

and computing the $\Pr(U_i = s \mid ev)$, $1 \le s \le r$. Please refer to the original paper [3] to learn the meaning of $w(y_{k,l}, v_{i,j})$.

3. Evidence propagation from top to down in BN:

$$\Pr(v_{i,s} \mid ev) = \sum_{j=1}^{M_{V_i}} \sum_{k=1}^{L_{Y_j}} w(y_{j,k}, v_{i,s}) pr(y_{j,k} \mid ev) \qquad (2)$$

where the m_{V_i} is number of parents of V_i, Y_j is a node in $pa(V_i)$ and L_{Y_j} is the number of states that Y_j takes.

4. Computing:

$$\Pr(G_i = s \mid ev) = \sum_{Pa(G_i)} \Pr(G_i = s \mid Pa(G_i)) \Pr(Pa(G_i) \mid ev) \quad (3)$$

For example, if there is a group G_1 which has three members: U_1, U_2, U_3, $Pa(G_1)$ is the set of all possible instances of vector $\{V_1, V_2, V_3\}$. Luis et al took $\prod_{i=1}^{|Pa(G_i)|} \Pr(v_{i,j} \mid ev)$ as the approximation of $\Pr(Pa(G_i) \mid ev)$. The $\Pr(G_i = s \mid Pa(G_i))$ is determined by social value function, such as maximum social value function:

$$\Pr(G_i = s \mid Pa(G_i)) = \begin{cases} 1 \ if \ k = \max\{pa(G_i)\} \\ 0 \ otherwise \end{cases} \quad (4)$$

In paper [7], A. Y. Sihem et al proposed a formal semantics that accounts for both item relevance to a group and disagreements among group members. Their solution is from an intuitive observation: In general, group members may not always have the same tastes and a consensus score for each item needs to be carefully designed. There are two main aspects contributing to the consensus score. First, the score should reflect the member's preferred attitude to the item. The more group members prefer an item; the higher a score should be given to the group. Second, the score needs to reflect the level at which members disagree with each other. They call the first aspect group relevance and the second aspect group disagreement. They try to fit a consensus function to solve this problem. The consensus function, denoted by $F(G,i)$, combines the group relevance and the group disagreement of i for G into a single group recommendation score using the following formula:

$$F(G,i) = w_i \times rel(G,i) + w_2 \times (1 - dis(G,i)) \quad (5)$$

where $rel(G,i)$ is the relevance of an item i to a group G, $dis(G,i)$ is the disagreement of a group G over an item i, $w_1 + w_2 = 1$ and each specifies the relative importance of relevance and disagreement in the overall recommendation score.

III. EVOLUTION GROUP MODEL

In general, there are a lot of groups in a large social network system, such as Facebook, where every user usually joins several different groups. In paper [8], Baatarjav et al took University of North Texas (UNT) Facebook SN as a sample for research. There are 10 main group types, such as business, common interest, entertainment& arts, geography, music, etc. Six of them have over 500 groups, and four of them have range between 61 and 354 groups in each.

The work in [1] suggests that when generating recommendations to a group, we can get a good result by learning from other groups. In [1], their strategy for generating recommendations is first to aggregate user profiles into a group profile and then utilize a nearest-neighbor algorithm in identifying neighbor groups from whom recommendation sets are generated. Experiment results showed that their proposed system has consistently higher precision and individual members are more satisfied.

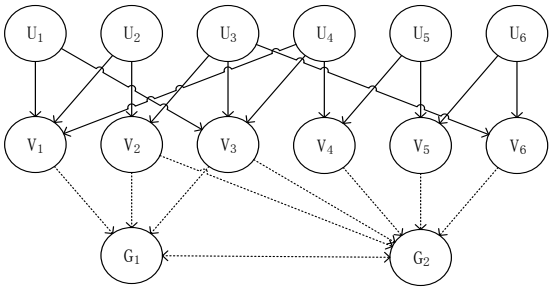

Figure 2. Collaborative group recommender system evolution topology

The research results above inspire us to develop a group recommender system which incorporates information from other groups when we compute recommendations for a special group with Bayesian networks model. Note that we are taking a strategy that groups learn from each other under the Bayesian networks model. For example, U_2, U_3 join both G_1 and G_2 in Figure. 1. When computing recommendations for G_1, we can learn something from G_2 (especially the recommendations of G_2) to achieve generating better recommendations for G_1. The topology of the Figure 1 should be updated as the system topology of the Figure 2.

Just like that there are many solutions for choosing *K*-nearest-neighborhoods in collaborative filtering RS, there are a lot of solutions for choosing *K*-nearest neighborhood groups, too. It can base on different group similarity measures or different value for *K* which can be set according to a threshold or a percentage). Here we simply present and compare two typical solutions: (1) choosing the top *K*-nearest neighborhood groups which has more common members with the given group; (2) using similarity measure (such as Pearson correlations or Spearman's rank correlations) to determine the top *K*-nearest neighborhood groups.

Assuming we need to determine a nearest-neighbor group for G_j, the former solution can be represented as:

$$G_{result} = \arg\max_{G_i} |U_{G_i} \cap U_{G_j}| \quad (6)$$

where $|U_{G_i} \cap U_{G_j}|$ is the size of intersection of members of G_i and members of G_j. For solution (2), we use the Pearson correlation coefficient of modified:

$$sim(G_a, G_b) = \frac{|I(G_a) \cap I(G_b)|}{|I(G_b)|} \times abs\left(\frac{\sum_j (r_{a,j} - \overline{r_a})(r_{b,j} - \overline{r_b})}{\sqrt{\sum_j (r_{a,j} - \overline{r_a})^2 (r_{b,j} - \overline{r_b})^2}}\right) \quad (7)$$

Solution (1) has two advantages including less computation and transparence to the group recommendation history. When applying solution (2) to a new system or a system which has very little recommendation history, it almost has no reliability. In our experiment, we just use the solution (1).

IV. GENERATING THE RECOMMENDATIONS

The Bayesian networks-based group recommendation model has high computation complexity. Even though using approximate technology, the computation complexity of $\Pr(G_a \mid ev)$ still is $O(r^m)$, where r is the highest rate and m is the size of a group. In short, the computation complexity grows exponentially as the group size grows. So in this section, we propose a new framework to simplify the group recommendation computation with insurance that the evolution group model can run well in this framework.

A. A New Framework to Simplify the Computation

Let's focus on the formula (3) and take $\Pr(G_i = s \mid Pa(G_i))$ and $\Pr(Pa(G_i) \mid ev)$ as coefficient and weight respectively. As we use $\prod_{i=1}^{|Pa(G_i)|} \Pr(v_{i,j} \mid ev)$ to approximate the real $\Pr(Pa(G_i) \mid ev)$, so the weight is a product of all $\Pr(v_{i,j} \mid ev)$, $1 \le i \le |Pa(G_i)|$. Although the approach of multiplying together all the $\Pr(v_{i,j} \mid ev)$ is an effective social value function [8], when $|Pa(G_i)|$ is large, the computation process of $\Pr(G_i = s \mid ev)$ will be very complex. So we proposed algorithm 1 to manage the application of social value function and simplify the group recommendation computation. To reduce computing time, we just use single item evidence as input to all of our algorithms.

Algorithm1:
//a novel approach to compute group recommendations
//every group will run an instance of this algorithm with //Group information as hidden input
// *GroupPredRecList* and *CandRecList* are global //variables of a group
// *GroupPredRecList* ← { }, *CandRecList* ← { };
Phase I (prepare):
Input: *ev* which has not been rated or predicted

1. Every member (denoted by bold point) locates at circle initially and divides the circle equally. The radius of the circle is *R*.
2. Compute all $\Pr(x_{i,s} \mid ev)$; such as $\Pr(x_{1,1} \mid ev)$, $\Pr(x_{1,2} \mid ev)$, …, $\Pr(x_{a,1} \mid ev)$.
3. For every rate *s*, ($1 \le s \le r$), we make every member move toward the center of circle along the radius according to the value of $\Pr(x_{i,s} \mid ev) * R$. For *s = 2*, the member u_1 moves according to the value of $\Pr(x_{1,2} \mid ev) * R$ and the member u_2 moves according to the value of $\Pr(x_{2,2} \mid ev) * R$. So for every rate *s*, there exists a circle with radius R_s which cover all member points fitly. After going over *r* rates, we get *r* circles.
4. $R_{result} = \arg\min_S(R_S)$, ($1 \le s \le r$); add the (*ev, result,* R_{result}) to *GroupPredRecList*. Note that *GroupPredRecList* is used to store the above temporary recommendations which may be washed out in Phase II and real recommendations will be stored in *CandRecList*.

In algorithm 1, the *3th* step of Phase I is the place where we can apply different social value functions and use different rules to determine the radius *R*s for a given rate *s*. The example rule is Local Least Misery. For a given rate *s*, if we find an *R*s circle which only covers the member with the largest $\Pr(x_{i,s} \mid ev)$, this is the case applying Local Most Pleasure rule. Note that it's local, not global. Actually, this approach will produce error, but it can be compensated with satisfaction measure model. For example, given rate $s = 3$, in the process of computing its $Rs = 3$, algorithm 1 will only take the effect of all $\Pr(x_{i,3} \mid ev)$ into account or in other way, algorithm 1 only sees the values including $\Pr(x_{1,3} \mid ev)$, $\Pr(x_{2,3} \mid ev)$, $\Pr(x_{3,3} \mid ev)$ and so on , and will not see the values including $\Pr(x_{1,3-1} \mid ev)$, $\Pr(x_{1,s+1} \mid ev)$ and so on. In standard Bayesian networks, we need to consider the impact of $\Pr(x_{1,s-1} \mid ev)$, $\Pr(x_{1,s+1} \mid ev)$ and so on. (This is another reason why the original algorithm [3] has high complexity). The idea of our algorithm is that for a given rate *s=3*, if the power of all $\Pr(x_{i,s=3} \mid ev)$ temporarily outperforms the power of all $\Pr(x_{i,s} \mid ev)$ *(s|1≤s≤r && s!=3)*, it reveals the potential of rate *s=3*. But if it doesn't pass the test of algorithm3, it will fail at last.

Algorithm1:
// GroupPredRecList ← { }, CandRecList ← { };
Phase II (generate recommendation):
Input: *k*

1. *TopPredRecList* ← { }
2. Sort(*GroupPredRecList*) by *result* or (*result,* R_{result}*);*
3. Consume top *k ev* with highest result from *GroupPredRecList* and push them into *TopPredRecList*.
4. Pick out (*ev, result,* R_{result}*)* from *TopPredRecList* one by one to consult every group member to learn whether they are satisfactory with it. (Satisfaction measure model will be presented in Algorithm3 of section 5). If all feel satisfaction, we add (*ev, result,* R_{result}) to *CandRecList*. If there is someone don't feel satisfaction with the level of disagreement Dis_{ev}, the member will move toward the outside of the circle along the radius according to the value of Dis_{ev}. At this time, system needs to re-compute the R_s for the *ev*.
5. Pick out *ev* from *CandRecList* randomly to recommend to the group.
6. Return *ev*

B. Learning from Other Groups

Let's assume that group G_1 and G_2 have some common users which were denoted by $U_{G1} \cap U_{G2}$ and G_2 had reached consensus for choosing item *j*. Now we can learn that the users $U_{G1} \cap U_{G2}$ in G_1 also like item *j* and the possibility of successfully recommending item *j* to G_1 increase. What we need to do in the following is to observe the satisfactory level of remain members for item *j*. This kind of learning is very helpful for the case where users tend to consume a given item more than one time. If

users enjoy some novel things, we can pick out an item from the *CandRecList* of G_2. All in all, learning from other groups make us stand on a top start point. Algorithm 2 presents the approach in which groups can learn from each other. Note that algorithm 2 and algorithm 1 can run concurrently.

Algorithm2:
//assuming we want to generate recommendation for G_p
Input: G_p
1. Find *K*-nearest neighborhood groups of G_p.
2. Find *W* ever recommendations in every relevant group; package every recommendation and its rate into a pair and add result to the $GiftMap_i$, where *i* is the identity of a relevant group.
3. For *i* : *K*

For every item *j* in $GiftMap_i${
If item *j* had not been cooked yet
Sat(G_p, *j*) = Satisfaction(G_p, *j*);
}

4. Find Q pairs with highest Sat(G_p, *j*) over all $GiftMap_i$ and add them to the *CandRecList of* G_p

V. USING THE SATISFACTION MODEL TO REFINE THE GROUP RECOMMENDATIONS

To the best of our knowledge, traditional satisfaction measure models are always embedded in the process of computing recommendations. But this approach doesn't fit the Bayesian networks-based RS well, because Bayesian networks are born with complexity of probability computation. In this section, we propose a satisfaction measure model based on feedback which is a power technique in control theory. Feedback is a technology which uses the system output to control or adjust the system input. The *TopPredRecList* of algorithm 1 include the temporary recommendations which may be washed out in Phase II. Note that before generating those temporary recommendations, we don't involve with any satisfaction measure. We package those recommendations and their rates as pairs and deliver them as feedback to group members. According to the satisfactory level of group members to the feedback, those temporary recommendations should be processed further. For example, if all members of a group agree on the rate = 4 for a given item 1, the item 1 wins and lives in *CandRecList*; if some members of the group disagree on the rate = 4, we need further process for the item1.

There are two challenges, including: (1) how to compute the satisfaction level of member for a given item; (2) if some members disagree on the rate, how to perform further process.

A. Define the Satisfaction Measure Model

Before computing the satisfactory level, we need a reference system. For example, level 3 represents quiet mood (baseline). Level 5 represents very satisfaction mood and level 1 represents very dissatisfaction. But in GRS, the basic problem of measuring satisfactory level is that we are predicting whether a user likes a novel item. The user never learns something about this new item and expresses mood on it. If a user A just expresses that he doesn't like this item, now asked to explore whether user A likes this item, we are almost sure that user A aren't interesting in this item. Based on the above observation, algorithm 3 uses effective collaborative filtering algorithm (such as SVD) or the ensemble of several collaborative filtering predictors [9] to predict the rate of user A for a given item; then we compares the predicted result with the predicted rate generated by GRS to measure the satisfactory level. Algorithm 3 presents the detail of computing the satisfactory level of a member for a given item.

B. Deal with the Disagreement of Member for a Given Item

If user *u* disagrees on the predicted rate *s* of group recommendation in *TopPredRecList* (after running algorithm 3, the return value is greater than zero), we make the user *u* move toward the outside of the circle along the radius according to the return value of algorithm 3. Then system needs to re-compute the *Rs* for the *ev* and execute the remain of algorithm 1. This process repeats till all members reach a consensus to a rate *s*.

If user *u* agrees on the predicted rate *s* of group recommendation in *TopPredRecList* and the return value of algorithm 3 is less than zero, we make user *u* move toward the center of circle along the radius according to return value of algorithm 3. Now system also needs to re-compute the *Rs* for the *ev* and execute the remain of algorithm 1. This process repeats till all members reach a consensus to a rate *s*. If we want to relax the condition, we can just take the case where return value of algorithm 3 is less than zero as the case where return value of algorithm 3 is equal to zero.

If algorithm 3 returns values which are equal or less than zero for all members of the group for a given item, it indicates that the group has reached a consensus.

VI. EXPERIMENTS

In this section, we first present the experiment methodology and experimental settings, and then analyze the experiment results.

First, we present the measures to evaluate the accuracy of the system, including MAE and Recall. The system generates predicted ratings $\hat{r}_{gi}$ for a test set T of group-item pairs *(g, i)* for which the true ratings r_{gi} are known. Typically, r_{gi} are known because they are hidden in an offline experiment, or because they were obtained through a user study or online experiment. The set of groups in the system will be denoted by $\mathcal{T}$. *T(g)* is the subset of test items that a group g found relevant. *L(g)* is a list of items which are recommendations.

$$MAE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(g,i) \in \mathcal{T}} \left| \hat{r}_{g,i} - r_{g,i} \right|} \qquad (8)$$

Percentage of success (*%S*):

$$\mathrm{Re}\,call(L) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \frac{|L(g) - T(G)|}{|T(g)|} \qquad (9)$$

MovieLens dataset was used to train and test our algorithm. MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota. This data set consists of 100,000 ratings (1-5) from 943 users on 1682 movies. Each user has rated at least 20 movies. The data sets u1.base and u1.test through u5.base and u5.test are 80%/20% splits of the MovieLens data into training and test data. Each of u1, ..., u5 have disjoint test sets; this is helpful for 5 fold cross validation.

The collaborative group recommendation model in Figure. 1 is learned from training sets. In particular, for each collaborative node V_i we look for the 10 most similar users (if they exist) using Pearson similarity measure. It should be noted that all the conditional probability distributions stores in U and V nodes have been estimated from the training set using the method in [3].

We use the following approach to create some groups from MovieLens dataset. We set every user as group admin and try to look for K-nearest-neighbors for every user. If we can't find any nearest neighbor for the admin, the admin will cancel the action of creating a group. If there exist some nearest neighbors, we will add the most relevant nearest neighbors to the new group. We create some groups having at most five members rather than at most four members in [1], because we expect one group can learn something from other groups. The group test sets can be obtained from the MovieLens test datasets [3].

The baseline algorithm is simple, but is the basement of our algorithms. The group rate in the baseline algorithm is obtained by merging, with the MIN, MAX or AVG criteria, the rates that individually the collaborative component proposes for each group member, i.e. using the results at nodes V_i. Besides that, we compare our evolution approach with the approach in [3] which is represented by "Group layer" in TABLE I. Due to the lack of space, there are some experimental settings we don't describe in detail, which can be found in [3]. Note that we just use most probable (MP) case to determine the final rate for a given item for all MIN, MAX or AVG criteria.

In this simulation, the collaborative filtering algorithm used in our algorithm 3 is SVD [10], which can achieve performance of RMSE 0.8657 by using iterate technology. To reduce the computing time, we just use single item evidence as input to all of our algorithms.

TABLE I presents the average results obtained after repeating the experiment with each training and test set under old computation framework. From TABLE I, we can see that group learning solution (Group learn) does improve the performance of GRS in terms Recall or MAE in the MAX or AVG gate case. The recommendation solution in which groups can learn from each other under the Bayesian networks model does better than the baseline approach which outperforms the original approach of [3] in MP case [11-17].

To validate the efficiency of the new computation framework, we perform another experiment with algorithm baseline, group learning and the combination of group learning and satisfaction evaluation. From TABLE II, we can see that the combination of group learning algorithm and satisfaction evaluation method can achieve a good score and the satisfaction evaluation method can really refine the recommendations.

TABLE I. AVERAGE EXPERIMENTAL RESULTS WITH OLD COMPUTATION FRAMEWORK

Old comp_ framework	Baseline %S MAE		Group layer %S MAE		Group learn %S MAE	
MAX	54.20	0.562	53.27	0.582	54.71	0.551
MIN	42.14	0.741	35.81	0.971	36.98	0.905
AVG	50.95	0.547	50.28	0.545	52.10	0.523
AVG_Time	31.20min		45.00min		49.50min	

These exists an obvious difference between TABLE I and TABLE II which is the time of running an algorithm with average gate (AVG_Time). The time of running an algorithm with max or min gate is almost the same as average gate. Due to the lack of space, we do not include them here. (The simulation was running in a desktop PC with AMD dual core 2.1G HZ processor and 2G memory).

TABLE II. AVERAGE EXPERIMENTAL RESULTS WITH NEW COMPUTATION FRAMEWORK

New comp_ framework	Baseline %S MAE		Group learn %S MAE		Group len+sat %S MAE	
MAX	54.81	0.512	54.92	0.572	54.15	0.481
MIN	50.35	0.741	51.41	0.637	52.52	0.585
AVG	52.10	0.547	54.80	0.505	55.91	0.492
AVG_Time	23.50min		25.10min		27.80min	

By analyzing the MAE results for each test set under new computation framework in Figure. 3, we can see that MIN_BL perform the worst, and the last second one is MIN_GLS, because they focus too much on the least misery of every member of the group.

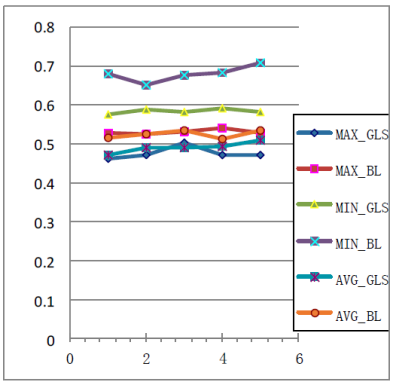

Figure 3. MAE result for each test set with new computation framework

VII. CONCLUSIONS

In this paper, we propose a Bayesian networks-based evolution approach which is composed of a Bayesian networks-based evolution group recommendation model, a new group recommendation computation framework and a new satisfaction measure model. In contrast to the traditional solutions, our satisfaction measure model use

the output of group recommendation model as input rather than use the system input. New group recommendation computation framework cooperated with satisfaction evaluation give an alternative way to the computation framework of Bayesian networks model. We further evaluate the performance of our approach on the MovieLens 100K dataset. From the experiment results, we can see that our algorithms not only make progress in term of predict accuracy and percentage of success (recall), but also have better industrial practice value in term of time. Our approach demonstrates that the application of our new satisfaction measure model under Bayesian networks model can refine the group recommendation well.

VIII. FUTURE WORK

Future work will consider best ways for allowing group members to interactively achieve common outcomes that they are willing to consume. By studying user interaction in a group recommender system we will be able to match group dynamics with taste preferences and group satisfaction for a set of events. Fairness is another study point worth investigating in the evaluation. Furthermore we plan to deduct what inspirational process produces user motivation for deciding upon a specific item list. The explanations future provided by algorithms based on Bayesian Networks will offer more information transparency and increasing group awareness [18-21].

ACKNOWLEDGMENT

This work was financially supported by National Basic Research Program of China (No.2010CB734104), NSFC Project (30970904/30400137), NNFC and the CASF Project (60672181), Key Lab of Information Network Security, Ministry of Public Security (2010). Besides, Sheng Feng would like to express my thanks to Dr. Qing Tan and Prof. Fei Gao for allowing me to combine my research project with the one for the Game Theory course. In this way I could benefit from the theoretical implications of aspects related to computation game theory and my practical work on group recommender in social websites.

REFERENCES

[1] J. K. Kim, H. K. Kim, H. Y. Oh, Y. U. Ryu, "A group recommender system for online communities," *International Journal of Information Management*, 30(3), pp. 212-219. Elsevier Ltd.

[2] F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, "Recommender Systems Handbook," *Media*, 54, pp. 73-105. Springer US. Retrieved from http://www. springerlink. com/index/10. 1007/978-0-387-85820-3.

[3] M. d. C. Luis, M. F. a. -L. Juan, F. H. Juan, A. R. -M, "Miguel. Group Recommending: A methodological Approach based on Bayesian Networks," *2007 IEEE 23rd International Conference on Data Engineering Workshop*, pp. 835-844. IEEE.

[4] B. Sarwar, "Sparsity, Scalability, and Distribution in Recommender Systems," *Ph. D. Thesis Proposal, Computer Science and Engineering Dept, University of Minnesota. July, 1999.*

[5] E. Alpaydin, "Introduction to machine learning," *ISBN-10: 0-262-01211-1, chapter 3. The MIT Press,* 2009.

[6] R. E. Neapolitan, "Learning Bayesian Networks," *ISBN: 9780130125347, chapter 3. Prentice Hall,* 2003.

[7] A. Y. Sihem, S. B. Roy, A. Chawla, G. Das, C. Yu, "Group Recommendation: Semantics and Efficiency," *Group, 2(1), pp. 754–765. Retrieved from http://portal. acm. org/citation. cfm?id=1687627. 1687713,* (2009).

[8] E. A. Baatarjav, S. Phithakkitnukoon, R. Dantu, "Group Recommender system for Facebook," *Work, pp. 211-219. Springer. Retrieved from http://www. springerlink. com/index/g004720x1k36087p. pdf,* (2008).

[9] X. L. Bao, L. Bergman, R. Thompson, "Stacking Recommendation Engines with Additional Meta-features," *Proceedings of the third ACM conference on Recommender systems RecSys 09* (p. 109). ACM Press.

[10] C. C. Ma, "A Guide to Singular Value Decomposition for Collaborative Filtering, csientuedutw," (1), pp. 1-14. *Retrieved from http://www. csie. ntu. edu. tw/~r95007/thesis/svdnetflix/report/report. pdf*

[11] CONITZER, V., 2006. Computing slater rankings using similarities among candidates. *In Proceeding AAAI'06 Proceedings of the 21st national conference on Artificial intelligence*. Vol. 1

[12] Crossen, A., Budzik, J., aAnd Hammond, K., J., 2002. Flytrap: Intelligent Group Music Recommendation. *In Proceedings of the 7th international conference on Intelligent user interfaces, Intelligent Information Laboratory, Northwestern University, USA*

[13] Endriss, U., Maudet, N., Sadri, F., and Toni, F., 2006. Negotiating socially optimal allocations of resources. *In Journal of Artificial Intelligence Research*, Vol. 25, pp. 315–348

[14] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997):'GroupLens: Applying Collaborative Filtering to Usenet News', *Communications of the ACM*, vol. 40, no. 3, March 1997, pp. 77-87.

[15] McCarthy, J., & Anagnost, T. (1998): 'MusicFX: An arbiter of group preferences for computer supported collaborative workouts', *in Proceedings of the ACM 1998 Conference on CSCW, Seattle, WA,* 1998, pp. 363-372.

[16] Mitchell, A., Posner, I., & Baecker, R. (1995): 'Learning to Write Together Using Groupware', *in Conference Proceedings on Human Factors in Computing Systems, Denver, CO,* 1995, pp. 288-295.

[17] Neuwirth, C. M., Kaufer, D. S., & Chandhok, R. (1994): 'Computer Support for Distributed Collaborative Writing: Defining Parameters of Interaction', *in Proceedings of the Conference on CSCW, Chapel Hill, NC,* 1994, pp. 145-152.

[18] Schafer, J. B., Konstan, J., & Riedl, J. (1999): 'Recommender Systems in E-Commerce', *in Proceedings of the First ACM Conference on Electronic Commerce, Denver, CO,* 1999, pp. 158-166.

[19] Shardanand, U., & Maes, P. (1995): 'Social In formation Filtering: Algorithms for Automating "Word of Mouth"', *in Conference Proceedings on Human Factors in Computing Systems, Denver, CO,* 1995, pp. 210-217.

[20] Smith, R. B., Hixon, R., & Horan, B. (1998): 'Supporting Flexible Roles in a Shared Space', *in Proceedings of the ACM 1998 Conference on CSCW, Seattle, WA,* 1998, pp. 197-206.

[21] Ungar, L. H., & Foster, D. P. (1998): 'Clustering Methods for Collaborative Filtering', *in AAAI Workshop on Recommendation Systems, Menlo Park, CA,* 1998.