

Iterative Voting under Uncertainty for Group Recommender Systems

Lihi Naamani-Dery, Meir Kalech, Lior Rokach, Bracha Shapira
Department of Information Systems Engineering and Deutsche Telekom Laboratories,
Ben-Gurion University, Israel
{ln,kalech,liorrk,bshapira}@bgu.ac.il

ABSTRACT

Group Recommendation Systems (GRS) aim at recommending items that are relevant for the joint interest of a group of users. Voting mechanisms assume that users rate all items in order to identify an item that suits the preferences of all group members. This assumption is not feasible in sparse rating scenarios which are common in the recommender systems domain. In this paper we examine an application of voting theory to GRS. We propose a method to accurately determine the winning item while using a minimal set of the group members ratings, assuming that the recommender system has probabilistic knowledge about the distribution of users' ratings of items in the system. Since computing the optimal minimal set of ratings is computationally intractable, we propose two heuristic algorithms that proceed iteratively that aiming at minimizing the number of required ratings, until identifying a "winning item". Experiments with the Netflix data show that the proposed algorithms reduce the required number of ratings for identifying the "winning item" by more than 50%.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms

Algorithms, Human Factors, Experimentation

1. INTRODUCTION

Group Recommender Systems (GRS) provide recommendations for groups of users and are applicable for domains where a group of people participate in a single activity. This paper addresses the recommendation of one item, a definite "winning item" (e.g., a TV show) while assuming that the preferences for the items are not known in advance. The necessary preferences are acquired during the recommendation process.

It is impractical to ask all members of the group for their preferences for all items. Some studies [3] deal with this challenge by computing the probabilities of the user preferences on candidate items and thus predict a probable winning item. We, on the other hand, assume that the distribution of the preferences of the users

over the items is known and thus we compute the minimal set of required references and query the users in order to identify a definite winning item. Unlike critique based GRS system [8], we ask the users to provide ratings of items rather than interactively critiquing critique on preferred features of the items. We propose to use a voting mechanism as it allows users to rank possible items and choose a winning item that reflects their joint preferences. Specifically, we focus on Range voting in which users are asked to assign a rating within a specified range for the items. The ratings for each item are summed, and the item with the highest score is the winner. Range voting is relevant to many existing recommender systems applications where users are asked to rate items within a specified range (e.g., Netflix). The voting mechanism discussed in this paper reduces the number of user ratings required for reaching an accurate and definite recommendation.

In voting theory, it is possible to determine a winner by specifically requesting users for certain preferences rather than for their whole set of preferences. A key question is what partial information is essential for determining a winner. To cope with this challenge we assume in this paper that the recommender system has probabilistic knowledge about the distribution of the group members preferences for the candidate items. For instance, in the case of friends wishing to watch a TV show together, the distribution can be inferred by rankings of these TV shows by similar users using collaborative filtering methods as illustrated in Section 6.

Computing the optimal minimal set of queries that are required to determine the winner is computationally intractable due to the combinatorial space of queries orders. Thus we propose two heuristic approaches to address this challenge. Both approaches proceed greedily and iteratively by querying a selected user about its rating for of a specific selected items.

2. RELATED WORK

Hazon et al. [4] assume predefined probability distribution of the ratings. They show theoretical bounds for the ability to calculate the probability of an outcome. However, while they focus on calculating the winning probability for each item, we focus on finding the winner using a minimal amount of queries.

Konczak et al. [6] address the case of partial information, where users do not set the preferences for all the items. They show how to compute the sets of **possible** winners and **necessary** winners. These sets determine which items no longer have a chance of winning and which are certain to win. We adopt their approach to propose a systematic preference aggregation protocol in which the users do not need to send their entire set of preferences. Walsh [9] surveys the computational complexity of possible and necessary winners in various voting rules. Walsh studies this problem by examining weighted or unweighted ratings and bounded or unbounded items.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

RecSys2010, September 26–30, 2010, Barcelona, Spain.
Copyright 2010 ACM 978-1-60558-906-0/10/09 ...\$10.00.

We may conclude that previous work in voting theory has addressed voting in systems where users send partial information, but most existing studies and algorithms do not aim at reducing ratings. Furthermore, while most papers in this field propose theoretical analysis of solving a voting problem, we apply it to a realistic domain and evaluate our algorithms empirically. As for related work in the GRS domain, most systems assume knowledge of the users' preferences on the items [7] while some studies suggest effective methods to acquire them [5]. To our knowledge, only de Campos [3] is the deals with uncertainties in GRS and estimates the user preferences using Bayesian Networks. HeThey then computes an estimated recommended item while we compute a definite winner, i.e. a recommended item that certainly suits the preferences of the group. McCarthy et al. [8] is the only known study that assumes that the users' preferences are not known. They apply the critique approach using case-based reasoning to acquire user preferences on desired items features, until they can infer the winning item that fits the set of desired features values. Such a system requires analysis and maintenance of items features which is not always feasible.

3. PROBLEM FORMULATION

Let us formalize a voting system with a rating distribution; a distribution of the users' preferences over the items. Let us refer to the item set as $C = \{c_1, c_2, \dots, c_m\}$, and to the users as $V = \{v_1, v_2, \dots, v_n\}$. In Range voting, the users assign values to the items from a predefined domain $D = \{d_{min}, \dots, d_{max}\}$, where d_{min} stands for the lowest value and d_{max} denotes the highest value. User v_i 's preferences are represented by a rating function $value^i : C \rightarrow D$ that assigns a value $d_i \in D$ to every item $c_j \in C$.

In our model we assume that the recommender system knows the probability distribution of the users' preference over domain D for each item. Formally:

DEFINITION 1 (RATING DISTRIBUTION). c_i^j is a discrete random variable that represents the rating of user v_i over the item c_j . c_i^j is distributed according to some Rating Distribution VD_j^i , such that $VD_j^i[d_g] \equiv Pr(c_i^j = d_g)$. The set $\mathcal{VD} = \{VD_1^1, VD_2^1, \dots, VD_m^1, VD_1^2, VD_2^2, \dots, VD_m^2, \dots, VD_m^n\}$ is the set of all the rating distributions of the users' rating for the items.

Since we use Range voting, we can assume independence between the probability distributions of the user's preferences for the items. While the independence assumption is a naive one, it can be used for approximating the actual probability. Note that the precise probability value is not required if we can still sort the queries correctly according to their informativeness. In closely related domains, such as machine learning, a similar naive assumption is known to provide accurate classification although the independence assumption is not always true.

In the example presented in Table 1, we show the rating distribution of three users for two items over a domain of $\{1, 2, 3\}$. E.g., the probability that v_1 will assign rating 1 to item c_1 is 0.2.

rating	v_1		v_2		v_3	
	c_1	c_2	c_1	c_2	c_1	c_2
1	0.2	0.2	0.4	0.5	0.3	0.7
2	0.2	0.2	0.3	0.2	0.3	0.1
3	0.6	0.6	0.3	0.3	0.4	0.2

Table 1: Rating distribution of the users in set $V = \{v_1, v_2, v_3\}$.

Based on the rating distribution settings, we can determine the winner with some probability. To set a definite winner, the system must disambiguate the accurate users' rating over the items. For this, we explicitly query the user. q_i^j represents the query of user v_i for its rating for item c_j . We assume that the user is sincere regarding her response.

A query has a cost denoted by the function $cost : q \rightarrow \mathbb{R}$. Throughout this paper we assume that the cost is equal for all queries. Fortunately, it may not be necessary to query the users about the whole set of items in order to determine the winner. It is possible to conclude who the winner is from partial information about the users' ratings [6, 9], although in the worst case each user should submit all its preferences [2]. To guarantee a winner with minimum cost, we proceed in rounds. In each round one user needs to rate one item. Following that, we determine the next query. Thus the total cost is minimized.

To determine whether a partial set of user ratings suffices for setting the winner, we use the necessary winner set [6]. The **necessary winners** set contains the items that win in **all** complete extensions of the partial preference relations among the items. It is sufficient to generate only the extensions that provide the most optimistic (possible maximum aggregated rating) and pessimistic (possible minimum aggregated rating) scenarios. Then, a *necessary winner* is an item whose minimum aggregated rating is greater than the maximum aggregated rating of all the others.

In Range voting, the pessimistic value (possible minimum) of a item is the lowest bound of the range. The optimistic value (possible maximum) is the upper bound. To formalize, let $O^i = \{\langle c_p, d_p \rangle_1^i, \dots, \langle c_r, d_r \rangle_r^i\}$ be the partial rating values given by user v_i as response to the center queries, where $\langle c_j, d_j \rangle^i$ stands for $value^i(c_j) = d_j$ as response to query q_j^i . $\mathcal{O}^A = \{O^1, \dots, O^n\}$ is a set of O^i sets. The function $pmax^A(c_j, \mathcal{O}^A)$ computes the possible maximum of item c_j , based on the preference values of the users:

DEFINITION 2. [Range voting Possible Maximum]
 $pmax^A(c_j, \mathcal{O}^A) = \sum_i pmax^i(c_j, O^i)$, where

$$pmax^i(c_j, O^i) = \begin{cases} d & \text{if } \exists d : \langle c_j, d \rangle \in O^i \\ d_{max} & \text{otherwise} \end{cases}$$

Similarly, we define a function of the possible minimum of an item:

DEFINITION 3. [Range voting Possible Minimum]
 $pmin^A(c_j, \mathcal{O}^A) = \sum_i pmin^i(c_j, O^i)$, where

$$pmin^i(c_j, O^i) = \begin{cases} d & \text{if } \exists d : \langle c_j, d \rangle \in O^i \\ d_{min} & \text{otherwise} \end{cases}$$

Now we can define the necessary winner. For this definition we define the set $C_t \subseteq C$ which contains the items that have been sent by any user after t querying rounds: $C_t = \{c | \langle c, d \rangle_j^i \in O^i, O^i \in \mathcal{O}^A\}$. A *necessary winner* is an item whose minimum aggregated rating is greater than the maximum aggregated rating of all the others:

DEFINITION 4. [Necessary Winner]
 $C_n = \{c_i | pmin^A(c_i, \mathcal{O}^A) > pmax^A(c_j, \mathcal{O}^A) \forall c_j \in C_t \setminus \{c_i\}\}$

Given a set of users V , a set of items C and probability distribution of users' rating for the items \mathcal{VD} , our goal is to determine a querying policy which minimizes the number of ratings that should be provided by the users in order to determine the winner.

This challenge can be represented as a Markovian decision processes (MDP). An MDP model is a tuple of states, actions, transition function from state to state by an action, and a reward function. In our problem, the states are the possible combinations of the users' ratings for the items. Every user can assign $|D|$ possible values to item c_i . If a user has not been assigned any value yet, the current value of the item is unknown. Thus, the combination space is $(|D| + 1)^{nm}$, where n is the number of users and m is the number of items. The actions are the possible queries, where a query is sent to a specific user about a single item's value. Thus the queries space is nm . The transition function between two states is affected by the probability distribution of the ratings of the item about which the user has made a query. The reward is the negative query costs. The policy is to determine which query to choose on each iteration.

Dynamic programming methods as Value Iteration or Policy Iteration [1] can compute the optimal queries vector by finding the optimal Policy. . However, these methods grow polynomially in the number of states and actions. In our case the state space itself is exponential and dynamic programming is not suitable for such large settings. Thus, we present in the next sections heuristic approaches that compute the next query greedily.

4. DYNAMIC INFORMATION GAIN

The information gain approach uses a greedy calculation in order to select the best query. In this approach we calculate the information gain of each query among the $m \times n$ possible queries, and choose the query that maximizes the gain. The information gain of a specific query is the difference between the prior probability distribution of the items to win and the posterior distribution of the items given the possible responses to the query. To calculate the distribution of the items to win, we define a winning item as a discrete random variable:

DEFINITION 5 (WINNING ITEM). A Winning Item WC is a discrete random variable over the item set C , where $Pr(WC = c_i)$ is the probability of item c_i to win.

The steps for the calculation of the information gain of each query:

1. Compute the probability of each item to win (WC).
2. Calculate the entropy of WC ($H(WC)$).
3. For each possible query q_i^j :
 - a. Calculate the entropy of WC given q_i^j ($H(WC|q_i^j)$).
 - b. Calculate the information gain achieved by q_i^j .
4. Select the query that maximizes the information gain.

To compute $Pr(WC = c_i)$, we sum the probabilities of the cases in which the aggregated rating of c_i is greater than the aggregated rating of the other items. Recall that $Pr(c_i = s)$ is the probability that the aggregated rating of c_i is equal to s . Due to the independence of probabilities, the probability of c_i to win with a aggregated rating s is the product of the probability that the aggregated rating of c_i is s and the aggregated rating of the other items is at most $s - 1$. To calculate $Pr(c_i = s)$, we use a dynamic programming algorithm. $Pr(c_i = s) = \sum_{k=d_{min}}^{d_{max}} Pr(\text{rating of } c_i \text{ from users } v_1 \dots v_{n-1} = s - k) * Pr(c_n^i = k)$.

The probability that $c_1 = s$ ($s \in \{n * d_{min}, \dots, n * d_{max}\}$) is presented in the first three rows of Table 2 (based on the VD presented in Table 1). For instance, to calculate $Pr(c_1 = 6)$ based on the ratings of all the three users (sixth column in line one, 0.236), we use the probabilities that were computed in columns 3–5 in the second line. These are the probabilities that $c_1 = s$ for $s \in \{3, 4, 5\}$ based on the users v_1, v_2 : $0.14 * Pr(c_3^1 = 3) + 0.36 * Pr(c_3^1 = 2) + 0.24 * Pr(c_3^1 = 1) = 0.236$.

	users	s=3	s=4	s=5	s=6	s=7	s=8	s=9
	v_1, v_2, v_3	0.024	0.066	0.182	0.236	0.27	0.15	0.072
$Pr(c_1=s)$	v_1, v_2	0.14	0.36	0.24	0.18	0	0	0
	v_1	0.6	0	0	0	0	0	0
$Pr(c_1 \leq s)$		0.024	0.09	0.272	0.508	0.778	0.928	1
$Pr(c_1 = s) \wedge Pr(c_2 \leq s)$		0.001	0.011	0.089	0.162	0.245	0.144	0.072

Table 2: Calculating the probability of c_1 to win.

Based on Table 2, it is easy to calculate $Pr(c_1 \leq s)$ by aggregating the results of the first row, as presented in the fourth row. The probability that a certain item c_i is a winner with a certain aggregated rating s is: $Pr(c_i \text{ is the winner with rating } s) = Pr(c_i = s \text{ and } \forall j \neq i c_j < s) = Pr(c_i = s) \cdot \prod_{j \neq i} Pr(c_j < s)$, where $Pr(c_j < s) = \sum_{k=n * d_{min}}^s Pr(c_j = k)$. The probability that item c_1 is a winner

with a certain aggregated rating s is presented in the last row of the table. For instance, the probability that c_1 is the winner with a aggregated rating of 5, is the probability that its aggregated rating is 5 and the aggregated rating of c_2 is at most 5: $0.182 * 0.492 = 0.089$. Finally, the probability that a certain item c_i is a winner: $Pr(c_i \text{ is the winner}) = \sum_{s=n * d_{min}}^{n * d_{max}} Pr(c_i \text{ is the winner with ratings } s)$. To calculate the probability of c_1 to win in Table 2, we aggregate its probability to win over the possible ratings of s (last line in table 2). In our example $Pr(WC = c_1) = 0.727$. Ties are broken according to the item positions according to an increasing order of all items.

To calculate the information gain (IG) of a certain query, we calculate the entropy reduction of WC that is achieved by that query. Mathematically, $IG(WC|c_j^i \text{ is queried}) = H(WC) - \sum_{g=d_{min}}^{d_{max}} H(WC|c_j^i = d_g) Pr(c_j^i = d_g)$, where $H(WC)$ is the entropy function $-\sum_{j=1}^m p(WC = c_j) \log(WC = c_j)$. The reduced side in the equation ($H(WC|c_j^i = d_g)$) represents the entropy of WC given the possible values by querying user v_i about item c_j . The calculation of $WC|c_j^i = d_g$ is processed exactly as described above based on the VD where $Pr(c_j^i = d_g) = 1$ and $\forall k \neq g Pr(c_j^i = d_k) = 0$. In our example, $WC = c_1|c_1^1 = 1$ is 0.996, $WC = c_1|c_1^1 = 2$ is 0.920 and $WC = c_1|c_1^1 = 3$ is 0.646.

To calculate the weighted average of the entropy we multiply the entropy by the probability of the random variable $Pr(c_j^i = d_g)$ (the reduced side in the equation). $\arg \max_{j,i} IG(WC|c_j^i \text{ is queried})$. In our example, querying user v_2 about c_2 generates the maximum information gain. To conclude, the algorithm iterates until a necessary winner is found.

The complexity of this algorithm is affected by the dynamic programming algorithm that computes the probability that $c_j = s$ ($\forall s \in \{n * d_{min}, \dots, n * d_{max}\}$). We calculate this probability for all items (m) and ratings ($n|D|$) for every users (n). This is done by going through the possible ratings $\sum_{k=d_{min}}^{d_{max}} Pr(\text{rating of } c_i \text{ from users } v_1 \dots v_{n-1} = s - k) * Pr(c_n^i = k)$ ($|D|$). This dynamic algorithm is implemented for every possible query of the users over the items ($mn|D|$). Thus the worst case complexity is $O(m^2 n^3 |D|^3)$.

5. HIGHEST EXPECTED MAXIMUM

The highest expected maximum heuristic is based on the exploration vs. exploitation tradeoff. As mentioned above, a necessary winner is an item whose possible minimum is greater than the possible maximum of the other items.

We propose a heuristic which chooses its next query by considering the **item that has the possible maximum and the user that is expected to maximize the rating** of that item. Using this approach, we encourage a breadth exploration over the items since the less information we have about an item's rating the higher possible maximum it has. In addition, we exploit the preferences revealed in order: (1) to refrain from querying about items that have been proved as impossible winners (since their possible maximum is less than a minimum of another item); and (2) to further examine an item, that has the highest possible maximum, and might be a necessary winner.

Once the target item of the query is chosen we further choose the user that is to be queried about that item. Here again we choose the user that is expected to maximize the item's rating by computing the **expected rating** using the rating distribution of that item. In particular, the expected rating of c_i^j based on the rating distribution VD_i^j is: $ES(VD_i^j) = \sum_{g=d_{min}}^{d_{max}} Pr(c_i^j = d_g) \cdot d_g$. For item c_j we choose the user that maximizes the expected rating: $\arg \max_i ES(VD_i^j)$. This process is repeated until a necessary winner is found.

The complexity of this algorithm is polynomial in the number of

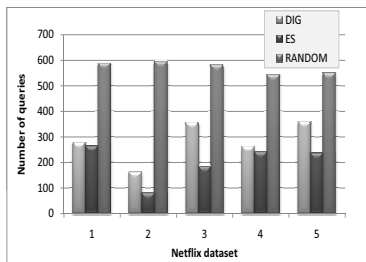


Figure 1: Number of users & items are fixed to 30.

users, items and domain size. In order to select the item that is to be queried, we compute the possible maximum of each item which is $O(nm)$, to select which user to query we compute the expected rating of the users about the specific item which is $O(n|D|)$. Thus, the total complexity is $O(n(m + |D|))$.

6. EXPERIMENTAL EVALUATION

We compared the proposed methods in terms of the number of queries required for finding the necessary winner.

We evaluated the performance of the (1) Dynamic Information Gain algorithm (*DIG*) and the (2) Expected rating algorithm (*ES*). The baseline for measuring the effectiveness of the proposed methods is a random procedure (*RANDOM*), in which the next query is randomly selected. We used the *Netflix dataset* to simulate real-world cases. Netflix provides a training data set of over 100 million ratings on a 1-5 scale as given by 480,000 users.

We extracted 5 different dense rating sub-matrices from Netflix. Each sub-matrix contains 40 users (the group size) and 40 items for which all actual ratings are known. The selected movie of the group is the one that obtains the maximum sum of ratings. Our goal is to find this movie with a minimum number of queries.

In the Netflix domain, the actual ratings are known but the rating distribution is unknown. Since we are interested in evaluating our algorithm using the rating distribution, we estimated the rating distribution of each user over an item by using simple user-to-user collaborative filtering with Pearson correlation. Collaborative filtering is commonly used to estimate a predicted rating by weighting the ratings of similar users (neighbours). In our case instead of weighting the ratings into a single predicted rating, we accumulate the neighbours' weights which fall into each of the rating intervals: $(-\infty, 1.5)$, $[1.5, 2.5)$, $[2.5, 3.5)$, $[3.5, 4.5)$, $[4.5, \infty)$. We normalize the vector to obtain a proper probability distribution.

Figure 1 presents the results obtained for 5 different sub-matrices. The results indicate a clear superiority of ES and DIG on Random. On average, ES requires only 32% of 625 maximum queries to find the winning item and DIG requires 45%. While a random selection of queries requires 91%. From the perspective of interrupting the user, users were required for an average of only 8 items in ES and 11.3 in DIG rather than an average of 22.8 requests in the maximum scenario. This result fits the requirement of minimizing the number of requests from the perspective of the human interface. The differences were found to be statistically significant with $F(2,8)=53.48$ and $p < 0.05$.

Figures 2 and 3 present the effect of the number of items/users where the number of users/items is fixed at 20 on the number of queries required. The results indicate that there is a clear correlation between the number of queries and the number of items or users. In all cases, the *ES* heuristic requires the least number of queries in all cases and the random method requires the largest number of queries.

The differences were found to be statistically significant with

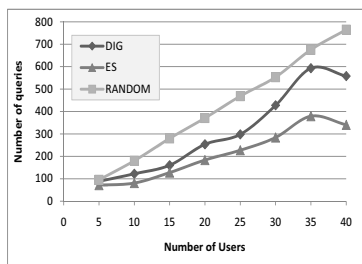


Figure 2: Number of users varies.

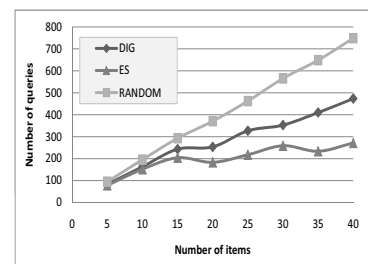


Figure 3: Number of items is varies.

$F(2,14)=13.61$ and $p < 0.05$ for the number of items and $F(2,14)=20.85$ and $p < 0.05$ for the number of users.

7. SUMMARY AND FUTURE WORK

We presented incremental query-based algorithms to reduce the number of queries required to find the winner item when the probability distribution of the users' rates is known in advance or can be estimated. The DIG algorithm chooses the query that maximizes the information gain based on the entropy of the probability of the items to win. The ES algorithm selects the item that has the highest possible maximum rating and queries the user that is expected to maximize the rating of that item. We showed that both methods significantly outperform a random selection of queries. Specifically, DIG reduced the necessary queries although its computational time complexity was found to be worse. ES was found to be a much more useful approach with negligible additional computational effort. In the future we plan to further develop more anytime algorithms that do not necessarily guarantee the winner but guarantee the most likely item under bounded resources.

8. REFERENCES

- [1] BELLMAN, R. Dynamic programming treatment of the travelling salesman problem. *J. ACM* 9, 1 (1962), 61–63.
- [2] CONITZER, V., AND SANDHOLM, T. Communication complexity of common voting rules. In *Proc. 6th ACM Conference on Electronic Commerce* (2005), pp. 78–87.
- [3] DE CAMPOS, L., FERNÁNDEZ-LUNA, J., HUETE, J., AND RUEDA-MORALES, M. Managing uncertainty in group recommending processes. *User Modeling and User-Adapted Interaction* 19, 3 (2009), 207–242.
- [4] HAZON, N., AUMANN, Y., KRAUS, S., AND WOOLDRIDGE, M. Evaluation of election outcomes under uncertainty. In *AAMAS-08* (2008), pp. 959–966.
- [5] JAMESON, A. More than the sum of its members: challenges for group recommender systems. In *Proc. Working conference on Advanced visual interfaces* (2004), ACM, pp. 48–54.
- [6] KONCZAK, K., AND LANG, J. Voting procedures with incomplete preferences. In *Proc. of Multidisciplinary IJCAI'05 Workshop on Advances in Preference Handling* (2005).
- [7] MASTHOFF, J. Group recommender systems: combining individual models. In *Recommender Systems Handbook* (2010).
- [8] MCCARTHY, K., SALAMÓ, M., COYLE, L., MCGINTY, L., SMYTH, B., AND NIXON, P. Group recommender systems: a critiquing based approach. In *Proc. 11th international conference on Intelligent user interfaces* (2006), p. 269.
- [9] WALSH, T. Uncertainty in preference elicitation and aggregation. In *AAAI'07* (2007), pp. 3–8.