International Journal of Grid and Distributed Computing
Vol. 9, No. 11 (2016), pp.261-272
http://dx.doi.org/10.14257/ijgdc.2016.9.11.22

Improving Recommendation Accuracy and Diversity through Cost-Awareness Probabilistic Spreading

[1]Guoyong Cai, [2]Dong Zhang, [3]Yumin Lin

Guangxi KeyLab of Trusted Software, Guilin University of Electro. Tech., PRC
ccgycai@guet.edu.cn mengjianzhizi@gmail.com ymlin@guet.edu.cn

Abstract

Recommender systems provide users with personalized suggestions for products. A key challenge is how to improve the diversity of recommendation results as much as possible, while maintaining reliably accurate suggestions. Although the bipartite graph based probabilistic spreading algorithm has its advantages of good accuracy and low computational complexity, its diversity is poor. In this paper, we introduce a cost-aware probabilistic spreading algorithm, and show how it can improve both recommendation accuracy and diversity by designing different spreading costs. Comparative experiments on widely used datasets confirm the effectiveness of the cost-aware probabilistic spreading approach in terms of accuracy, aggregate diversity and individual diversity of recommendation results. In addition, the time complexity of the proposed algorithm is also analyzed.

Keywords*: Recommender Systems; Diversity; Accuracy; Probabilistic Spreading*

1. Introduction

Recommender Systems provide users with personalized list of items they will like, based on past behaviors of users, content of items, and other information. Existing recommendation algorithms typically predict each user's ratings for unknown items, and recommend top N items with the highest predicted ratings. Accordingly, they optimize the accuracy of recommendations, such as the widely used content-based analysis [1], User(Item)-Based Collaborative Filtering [2-4] and Matrix Factorization [5-6].

However, higher recommendation accuracy does not always correspond to higher levels of recommendation quality [7-9]. In general, the length of final list recommended to user is far less (e.g. the length is 1, 5, 10) than the total number of items (millions such as Amazon.com [10]) in a system. On the one hand, those accuracy-oriented algorithms tend to find popular items with large amount of history data, but items with limited historical data are harder to be presented in the list [11, 22]. On the other hand, items in the recommendation list are too similar to what the user purchased in the past to capture the wide range of user preferences [7, 13], thus the usefulness of recommendation decreases. Improving the diversity of recommendation results can help solve these problems and offer benefits for both business holders and individual users.

Recommendation diversity can be classified into two aspects: aggregate diversity and individual diversity. The former measures the ability of a system for recommending all items, and the latter measures the diversity within the recommendation list of an individual user. Several studies explored different diversity measures from different aspects. [14] analyzed individual diversity and proposed a topic divarication method for decreasing the similarity between items within a recommendation list. [15-16] defined aggregate diversity as the total number of distinct items among all lists that have been recommended to all users, they also proposed a re-rank method and a maximization method for increasing aggregate diversity while maintaining acceptable loss of accuracy. [17] analyzed the importance of measuring the sales balance of items among aggregate

ISSN: 2005-4262 IJGDC

diversity, and proposed two list diversification approaches with inverted recommendation task. [18] developed several random walk based methods to help find long tail items related to users' past preferences. However, most existing algorithms improving recommendation diversity with the cost of accuracy loss. To the best of our knowledge, there is still no effort that analyzing and improving accuracy, aggregate diversity and individual diversity of recommendation results simultaneously under a unified framework.

The bipartite graph based probabilistic spreading algorithm, proposed by Zhou *et al*. [12], is practically important for its advantages of low complexity and high accuracy, but its diversity performance is poor. In this paper, we consider accuracy and diversity under a unified framework, and propose a cost-aware probabilistic spreading algorithm based on the network structure of basic probabilistic spreading algorithm. The proposed algorithm can largely improve accuracy, aggregate diversity and individual diversity of recommendation results simultaneously, at the same time, it outperforms some state-of-art widely used collaborative filtering approaches.

2. Probabilistic Spreading

In recommendation scenarios of probabilistic spreading, the input data contains a set of users *U*, a set of items *I*, and an adjacency matrix *A* where $a_{ui}=1$ if item $i \in I$ is rated by user $u \in U$ and otherwise $a_{ui}=0$.

According to the description of [11], the recommendation process of probabilistic spreading (hereafter called ProbS) is based on a resource spreading process on a bipartite graph. ProbS firstly assigns all items an initial level of resource denoted by f_u (where $f^i{}_u$ is the resource possessed by item i). $f^i{}_u=1$ if item *i* is rated by user u and $f^i{}_u=0$ otherwise. Then resources are redistributed via a two steps transformation $f_u = f_u \cdot T$ (where *T* is the transition probability matrix whose element t_{ij} representing the probability of resource spreading between item *i* and *j*). Lastly items are sorted in descending order according to $\tilde{f}_u$, the top *N* items are recommended to the target user *u*. The probability of resource redistributed from item *i* to item *j* is calculated with formula (1), where k_i is the degree of node *i*.

$$t_{ij}=\frac{1}{k_i}\sum_{v\in U}\frac{a_{vi}a_{vj}}{k_v} \tag{1}$$

ProbS estimates scores of items based on resource spreading process on bipartite graph, it has the advantages of low complexity and high accuracy. However, the spreading process depends solely on the structure of graph, without discerning any differences between users and items, as well as the different tastes of users and the popularity of items, which could largely affect recommendation results. ProbS redistributes resource among all neighboring nodes in the same level, which leads to low diversity performance. Moveover, ProbS ignores the detailed value of item ratings, while this information is easy to obtain in many cases. Therefore, focusing on these problems, ProbS can be further improved for better recommendation results. In next section, we develop a Cost-Aware Probabilistic Spreading algorithm (hereafter called iProbS). We first introduce the theoretical detail of iProbS in Sections 3.1 and 3.2. The algorithmic implementation is then introduced in Section 3.3.

3. Cost-Aware Probabilistic Spreading

3.1 Model

Generally, the user-item relationship is represented by a bipartite graph $G=(V,E)$, where V represents the set of nodes and E represents the set of edges. V_U denotes the set of user nodes and V_I denotes the set of item nodes. m is the number of users and n is the number of items. There is an edge between node $u \in V_U$ and node $i \in V_I$ with weight $w_{ui} = w_{iu} = r_{ui}$ if user u rates item i, where r_{ui} is the rating of user u on item i.

iProbS assumes that the target user u has resource 1 in total, then redistributing the resource through a three steps probabilistic spreading. The amount of resources spreading from node x to node y is related to the resources of node x, spreading probability p_{xy} and spreading cost c_{xy}, which is computed as follows.

$$R(x) \cdot p_{xy} \cdot c_{xy} \tag{2}$$

Where $R(x)$ is the amount of resources node x has.

The resource spreading process is described by three steps: Firstly, resources are distributed among node u's all neighboring item nodes j (step 1 in Figure 1), then redistributed back to those nodes j's neighboring user nodes v (step 2 in Figure 1), lastly redistributed again among those nodes v's neighboring item nodes i (step 3 in Figure 1). Each step complies with Equation (2). Therefore, the estimated score of user u on items i, denoted as $\hat{r}_{ui}$, is the amount of resources spreading from user u to item i:

$$\hat{r}_{ui} = 1 \cdot \sum_{j \in V_I} \sum_{v \in V_U} p_{uj} p_{jv} p_{vi} \cdot c_{uj} c_{jv} c_{vi} \tag{3}$$

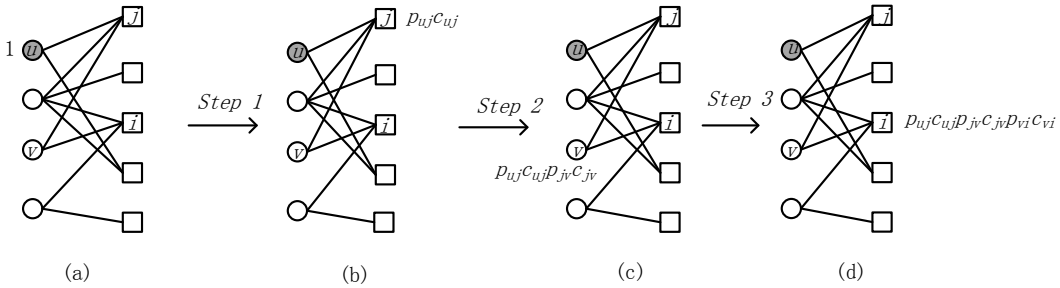


Figure 1. Resource Spreading Process of iProbS

The computational details of spreading probability and spreading cost will be introduced in section 3.2. An example of one-path resource spreading process of iProbS is shown in Figure 1(a)-(d). The final spreading result is the items receiving resources from all possible a three step spreading paths.

3.2 Spreading Probability and Spreading Cost

For the spreading probability, p_{xy} represents the amount of resources starting from node x spreading to node y, it is only related to originator node x and computed with Equation (4). Here the resource is not evenly distributed like the one in Equation (1). We believe that different rating score means different relational strength between nodes, therefore rating scores are set to be a kind of weight when computing the spreading probability.

$$p_{xy} = \frac{w_{xy}}{\sum_y w_{xy}}, w_{xy} = \begin{cases} r_{xy}, x \in V_U \\ r_{yx}, x \in V_I \end{cases} \tag{4}$$

All spreading probability p_{uj}, p_{jv}, p_{vi} within Equation (3) are calculated according to Equation (4).

For the spreading cost, c_{xy} means the resistance incurred in the resource spreading process, it is only related to terminator node *y*. Different terminating nodes have different spreading resistance. To compute the spreading cost in each step respectively, we define spreading cost in three different ways.

If we ignore the spreading cost in the first spreading step (Figure 2(a) to (b)), the total resource received by item *j* is proportional to its degree in the whole recommendation process. That is to say, the total recommendation power assigned to item *j* is proportional to its degree, thus the impact of popular items (high degree items) is enhanced and is weakened for unpopular items. Although ProbS already has a good accuracy performance, decreasing the recommendation power of popular items could further improve the accuracy as well as the diversity. Motivated by this intuition, we take degree of items into consideration when calculating the spreading cost c_{uj} in the first spreading step. Therefore the spreading cost c_{uj} is defined in Equation (5), where k_j is the degree of node *j*, and $1_{r_{uj}\neq 0}$ means if $r_{uj} \neq 0$ then $1_{r_{uj}\neq 0}=1$, otherwise $1_{r_{uj}\neq 0}=0$.

$$c_{uj}=\frac{1}{k_j}, k_j=\sum_{u\in V_U} 1_{r_{uj}\neq 0} \tag{5}$$

Suppose there are two users, u_1 and u_2. User u_1 has a specific interest with small number of rating history, while user u_2 has a wide range of tastes with a large number of rating histories. Generally, to distinguish the specificity of items, the information shared by the specific user u_1 is likely to be more important than the ambiguous user u_2. The recommendation results based on u_1 may be more informative than u_2. Therefore, decreasing the recommendation power of ambiguous user and increasing the recommendation power of specific user could help find more accurate and personalized items. Generally, the ambiguity of a user is increasing when the user rates large amount of items, especially with equal ratings. Motivated by this intuition, we use user entropy [19] based on user ratings in Equation (6) to calculate the spreading cost c_{jv} in the second spreading step (Figure 3(b) to (c)).

$$c_{jv}=\frac{1}{E(v)}, E(v)=-\sum_{j\in V_I} p_{vj}\log p_{vj} \tag{6}$$

Consider an extreme case, a user has rated a hundred movies, each movie has been rated by a hundred users and these users have again rated a hundred movies respectively, and all users and movies are not overlapping. According to the spreading process of iProbS, nearly one million movies can receive resource, however, the length of recommendation list is often much shorter (e.g. 10) than this huge number. Although this extreme situation is almost impossible, the amount of items which can receive resource are still far greater than the length of recommending number, even most of items are irrelevant to the target user's tastes. Moreover, high degree items tend to get more resources, which leads to the same popular items are recommended to many users. Based on this intuition, we consider the concept of nearest neighbors. Items will not receive resources when it does not belong to the nearest neighbors of those items which have been rated by the target user. The spreading cost cvi in the third spreading step (Figure 2(c) to (d)) is calculated as:

$$c_{vi}=\left|KNN(j)\bigcap\{i\}\right| \tag{7}$$

Where *KNN(j)* represents the set of *K* nearest neighbor items of item *j*. We use cosine similarity to compute the similarity between items and choose most similar items into *KNN(j)*:

$$s_{ij}=\frac{\sum_{v\in V_U} r_{vi}r_{vj}}{\sqrt{\sum_{v\in V_U} r^2_{vi}}\sqrt{\sum_{v\in V_U} r^2_{vj}}} \tag{8}$$

3.3 Algorithmic Implementation

In the recommendation scenario for an individual user, iProbS can easily generate recommendation result through the spreading process described in Figure 2. Consider the whole recommendation process for all users, resources will repeatedly go through an edge multiple times. In order to avoid double counting, we also calculate the item-item transition matrix T, where

$$t_{ij} = \sum_{v \in V_U} p_{jv} p_{vi} c_{jv} c_{vi} \tag{9}$$

Then we assign items which target user has already rated with initial resources $f^{j}_{u} = p_{uj} c_{uj}$. Finally redistributing resources via transformation $\tilde{f}_u = f_u \cdot T$.

Algorithm 1 describes the algorithmic implementation of iProbS, where U is the set of users, I is the set of items, $U(i)$ is the set of users who have rated item i, $I(u)$ is the set of items which are rated by user u, A is the user-item rating matrix, S is the item-item similarity matrix, R is the score estimation results whose element $\hat{r}_{ui}$ represents the estimated score of user u on item i, N is the length of recommendation list, L is the recommendation lists for all users.

Algorithm 1 Cost-Aware Probabilistic Spreading

Input: A;
Output: L;
1: Construct user-item bipartite graph G according to A;
2: Compute similarity matrix S according to A;
3: Compute item-item transition matrix T according to G and S:
4: FOR each i in I:
5: FOR each v in $U(i)$:
6: FOR each j in $I(v)$:
7: $t_{ij} += p_{iv} p_{vj} c_{iv} c_{vj}$
8: END FOR
9: END FOR
10: END FOR
11: Estimate score of unrated items for all users:
12: FOR each u in U:
13: FOR each i in I:
14: FOR each j in $I(u)$:
15: $\hat{r}_{ui} += p_{uj} c_{uj} t_{ji}$
16: END FOR
17: END FOR
18: END FOR
19: Select N items with max score from $R(u)$ into $L(u)$ for every user u.

4. Experiments

In order to test the effectiveness of the proposed Cost-Aware Probabilistic Spreading approach, we carry out several experiments on two widely used datasets: MovieLens 1M and Netflix Prize [20]. Each dataset includes ratings, ranging from 1 to 5, of users on movies. The MovieLens dataset is in its original form. In the case of Netflix dataset, we randomly select a subset from the huge original dataset. Table 1 summarizes the basic properties of each dataset.

For each dataset, we randomly select 20% of the ratings as test set, the rest as training set. Firstly, we apply five different recommendation algorithms to estimate unknown ratings for all users in test set, namely User-Based Collaborative Filtering (UB) [2], User-

Based nearest neighbor Collaborative Filtering (UBKNN) [21], Matrix Factorization (SVD++) [5], Probabilistic Spreading (ProbS) [12], and Cost-Aware Probabilistic Spreading (iProbS). The neighborhood number of UBKNN is initialized as 50. For the Matrix Factorization, we use SVD++ proposed by [5] and initialize the number of factor as 50. For the iProbS, one can adjust neighborhood number K to adjust the trade-offs between accuracy and diversity, here we initialize K as 50 in MovieLens while 5 in Netflix. Secondly, recommending items for every user according to different length of recommendation list (ranging from 1 to 50) based on the estimated score of items. Finally, we use one accuracy metric and three diversity metrics to evaluate performance of the recommendation results.

Table 1. Properties of Datasets

Dataset	Users	Movies	Ratings	Sparsity
MovieLens	6040	3706	1000209	4.5%
Netflix	8000	5000	680283	1.7%

4.1 Recommendation Performance Metrics

Precision. Precision measures the proportion of relevant test items of target user included in his recommendation list, denoted by Precision:

$$Precision = \sum_{u \in U_T} \frac{\sum_{i \in L_u} 1_{r_{ui} \neq 0}}{N \cdot |U_T|} \tag{10}$$

Where U_T denotes the set of user in test set, L_u denotes the recommendation list for user u, r_{ui} is rating of user u on item i in test set.

Aggregate diversity. For the aggregate diversity, Adomavicius *et al*. [15] proposed a popular metric to measure diversity in a recommender system, by using the total number of distinct items among all lists that have been recommended to all users, denoted here as *Aggr*:

$$Aggr = \left| \bigcup_{u \in U_T} L_u \right| \tag{11}$$

Clearly, higher *Aggr* values represent higher aggregate diversity. *Aggr* is intuitive and effective for measuring aggregate diversity, but it treats items that have been recommended many times and items recommended just one time the same. From business perspective, using a metric to measure the imbalance in the number of times each item is recommended would be more suitable [17]. *Gini coefficient* can further measure the overall balance in the sales of goods beyond *Aggr*, denoted as *Gini*:

$$Gini = \frac{2}{n-1} \sum_{i=1}^{n} (n+1-i) \frac{n(i)}{N \cdot |U_T|} \tag{12}$$

Where n is the total number of items, $n(i)$ denotes the number of users who get recommended item i. Along with the prior work [15, 17], we also use the rescaled *Gini coefficient* for more intuitive that higher *Gini* values represent higher balance and higher aggregate diversity.

Individual diversity. For the individual diversity, one of the best known metric to measure diversity on individual list recommended to a user is from Ziegler *et al*. [14]. They define the individual diversity as similarity of all item pairs in the recommendation list, denoted as *ILS*:

$$ILS = \frac{1}{|U_T|} \sum_{u \in U_T} \frac{1}{N(N-1)} \sum_{i,j \in L_u; i \neq j} s_{ij} \tag{13}$$

Where s_{ij} is the similarity between item *i* and *j*. Here we compute it using cosine similarity based on rating history of items. Higher *ILS* values denote low individual diversity.

4.2 Numerical Results

According to four different evaluating metrics, the results are described as four parts: Table 2, Figures 2(a) and (b), Figures 2(c) and (d), Figures 2(e) and (f).

Table 2 shows the aggregate diversity (*Aggr*) performance of Algorithms on two datasets with different recommending number. The numerical values in the table is the corresponding *Aggr* values. It is apparent that the iProbS has significant aggregate diversity performance, and, outperforms ProbS as well as other three popular collaborative filtering algorithms. For instance, in the MovieLens dataset, the *Aggr* value of iProbS is 2373 when recommending 10 items (N=10), while ProbS is 109. The best algorithm in the rest three algorithms is UBKNN with *Aggr* value 587, which is far less than *Aggr* value of iProbS. That is to say, when recommending 10 items to every user, iProbS can recommending 64% of all the movies (3706 in MovieLens), while the rest algorithms UB, UBKNN, SVD++, ProbS just 1.9%, 15.8%, 13.3%, 2.9% respectively. Algorithms get similar results in Netflix dataset .

Table 2. Aggr with Different Recommending Number

Dataset	Method	N=1	N=10	N=20	N=30	N=40	N=50
MovieLens	UB	25	71	125	165	199	241
	UBKNN	190	587	843	1044	1252	1448
	SVD++	120	494	765	967	1127	1286
	ProbS	38	109	181	258	319	371
	iProbS	**764**	**2373**	**2916**	**3174**	**3313**	**3424**
Netflix	UB	64	144	210	269	322	370
	UBKNN	210	745	1200	1600	1953	2215
	SVD++	46	237	393	503	611	701
	ProbS	83	285	405	541	635	721
	iProbS	**1870**	**3789**	**4004**	**4074**	**4105**	**4119**

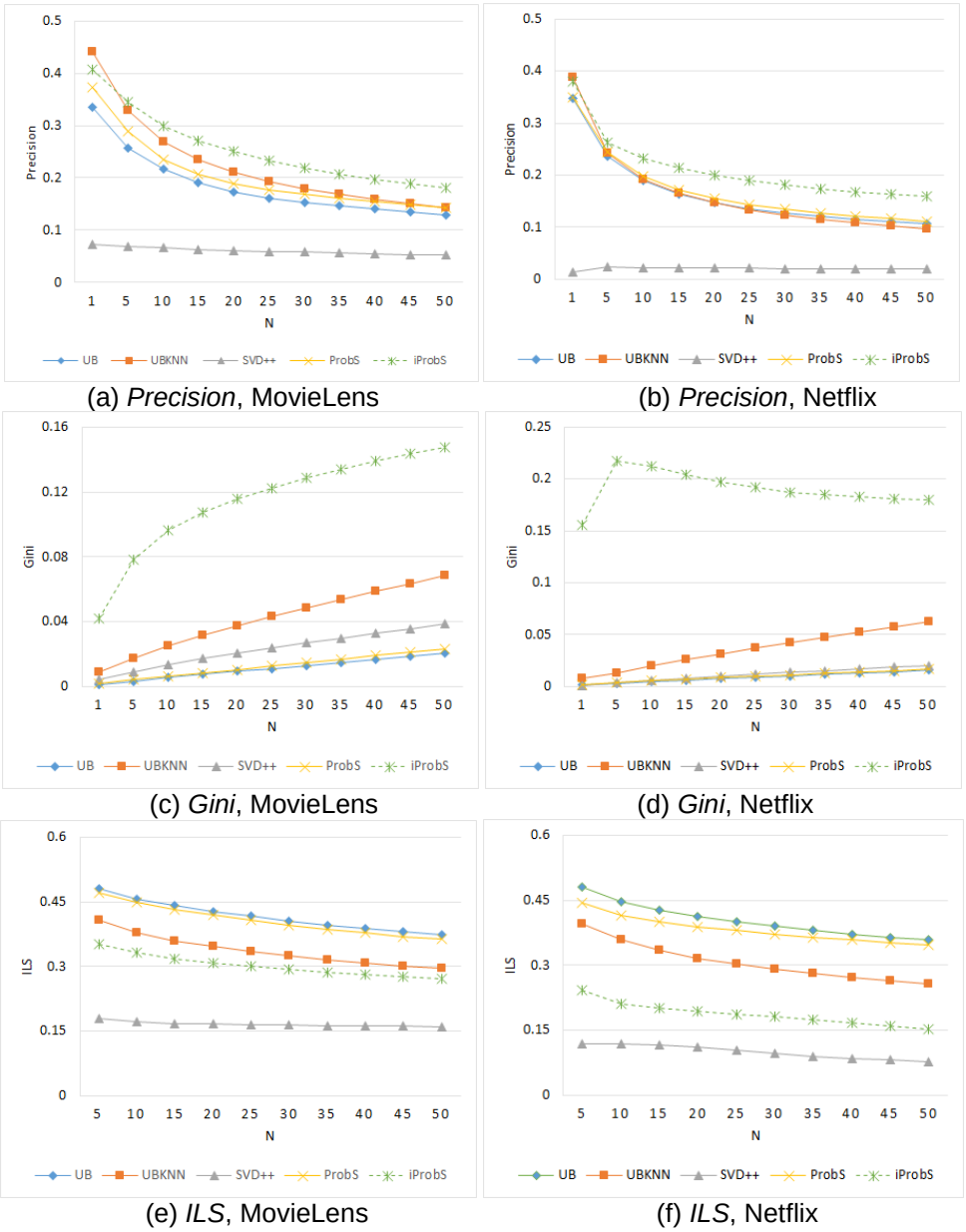
Figures 2(a) and (b) show the Accuracy (*Precision*) performance of the algorithms. The horizontal values denote recommending number and the vertical values denote *Precision* value. We can clearly observe that Precision values decreasing with the recommending number increasing. When the recommending number is greater than 5, the best performing algorithm in terms of *Precision* is still the iProbS. The results confirm that the proposed iProbS approach can obtain better accuracy performance than other algorithms, while significantly improving the diversity of recommendation. A noticeable result is SVD++, its Precision performance is much worse than other four algorithms.

Figures 2(c) and (d) show the aggregate diversity (*Gini*) performance of the algorithms. Once again, the proposed iProbS shows surprisingly good results on both two datasets, for the curves of iProbS is far above the curves of other algorithms. Consistent with the *Aggr* in Table 2, the best performing results in the rest four algorithms is the UBKNN, the worse algorithm is UB.

Figures 2(e) and (f) show the individual diversity (*ILS*) performance of the algorithms. We can observe that *ILS* value slowly decreasing with the recommending number increasing. According to the discussion in Section 4.1 that higher *ILS* values denote lower individual diversity, therefore the best performing algorithm in terms of individual diversity is SVD++, the worst algorithm is UB. Note that while the SVD++ outperforms iProbS in terms of *ILS* value, the accuracy performance of SVD++ is far lower than the average level of other algorithms. Recommendation with too low accuracy is

meaningless. In the rest four algorithms, the best *ILS* curve is still iProbS. It confirms that iProbS can obtain great individual diversity performance while maintaining better accuracy recommendation results.

In conclusion, over both MovieLens and Netflix datasets, the proposed Cost-Aware Probabilistic Spreading approach is consistently the top performer in terms of accuracy (*Precision*), aggregate diversity (*Aggr* and *Gini*), except for comparison with SVD++ on individual diversity (*ILS*). The reason is that iProbS combines multiple factors for computing item score. First, the recommendation power of popular items are decreased, while the recommendation power of specific users are increased, thus can help find more diversified, accurate and personalized items. iProbS also distributes resources to items which belong to nearest neighbors of those items that have been rated by target user, thus it can exclude interference of irrelevant items.



(a) *Precision*, MovieLens (b) *Precision*, Netflix

(c) *Gini*, MovieLens (d) *Gini*, Netflix

(e) *ILS*, MovieLens (f) *ILS*, Netflix

Figure 2. Precision, Gini Coefficient, ILS with Different Recommending Number

4.3 Complexity Analysis

If we denote k_u and k_o the average degree of users and items in the bipartite graph, K the neighborhood number, n_R the average number of items whose estimated score are larger than zero, the computational complexity of ProbS is $O(nk_ok_u + mnk_u + mn_R \log n_R)$, where the first term accounts for the calculation of transition matrix of items (see Eq. (1)), the second term accounts for the calculation of the estimation score, and the third term accounts for selecting top N items using Quick Sort approach. Substituting the equation $mk_u = nk_o$, we are left with $O(mk^2{}_u + mnk_u + mn_R \log n_R)$. Clearly, $k^2{}_u << nk_u$, thus the resulting complexity form of ProbS is $O(mnk_u + mn_R \log n_R)$. The computational complexity for iProbS is $O(nk_ok_u + mk^2{}_uK + mnk_u + mn_R \log n_R)$, where the first term accounts for the calculation of similarity between items (step 2 in Algorithm 1), the second term accounts for the calculation of transition matrix of items (step 3-10 in Algorithm 2), the third term accounts for the calculation of the estimation score (step 11-18 in Algorithm 2), and the last term accounts for selecting top N items using Quick Sort (step 19 in Algorithm 2). Analogously, substituting the equation $nk_o = mk_u$, the resulting complexity of iProbS is $O(mk^2{}_uK + mnk_u + mn_R \log n_R)$.

Table 3 shows the running time of different algorithms on the MovieLens dataset with N=10, where 0 denotes the algorithm doesn't need corresponding computation step. We can observe that UBKNN runs much fast than other algorithms, followed by ProbS, and SVD++ runs slowest. iProbS runs slightly slower than ProbS. According to the preceding analysis, iProbS runs slower than ProbS when computing transition matrix. However, iProbS consider the concept of nearest neighbors, lead to n_R in iProbS is far less than in ProbS, therefore, iProbS is runs faster than ProbS when selecting top N items. Hence, according to the analysis of computational complexity and experiment demonstrate that the improved approach iProbS can obtain more accurate and diversified recommendation results than ProbS, while maintaining the advantage of low computational complexity.

Table 3. Running Time of Algorithms on MovieLens Dataset with *N=10*

Algorithms	UB	UBKNN	SVD++	ProbS	iProbS
Calculation of similarity matrix (s)	3.4	3.4	0	0	2.08
Calculation of transition matrix (s)	0	0	0	2.46	8.38
Calculation of estimated score (s)	23.85	8.24	1092.82	21.49	21.41
Selection top *N* items (s)	6.32	0.94	7.256	6.71	0.79
Total time (s)	33.57	12.67	1100.01	30.66	32.66

5. Conclusion

Both accuracy and diversity are important properties for measuring the quality of recommendation results. In this paper, we consider accuracy and diversity simultaneously, and introduced an improved probabilistic spreading algorithm, namely cost aware probabilistic spreading. In the proposed approach, we further define the resource spreading process as spreading probability and spreading cost, and consider three different factors for computing the spreading cost. Several experiments on two real world dataset confirm that the improved probabilistic spreading algorithm can effectively improve the accuracy, aggregate diversity and individual diversity, at the same time, outperforms the widely used collaborative filtering algorithms. Finally, the computational complexity analysis and experiments verify the proposed approach maintain the advantage of low complexity of basic probabilistic spreading algorithm. As future work,

we will deeper study on how to further control the trade-offs between accuracy and diversity.

Acknowledgements

This paper is supported by China National Science Foundation (No. 61540053,61562014), Guangxi Key Lab of Trusted Software (kx 201503),and Innovation Team of Guilin University of Electronic Technology.

References

[1] M. J. Pazzani and D Billsus, Editor, Content-based recommendation systems. Springer Berlin Heidelberg, Berlin **(2007)**
[2] G. Adomavicius and A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions[J]. Knowledge and Data Engineering. 6, 17 **(2005)**
[3] M. Deshpande and G. Karypis, Item-based top-n recommendation algorithms[J]. ACM Transactions on Information Systems. 1, 22 **(2004)**
[4] Y. Shi, M. Larson and A. Hanjalic, Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges[J]. ACM Computing Surveys. 1, 47 **(2014)**
[5] Y. Koren, Editors. Factorization meets the neighborhood: a multifaceted collaborative filtering model. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge discovery and data mining, **(2008)** August 24-27; Las Vegas, USA
[6] Y. Koren, R. Bell and C. Volinsky, Matrix Factorization Techniques for Recommender Systems[J]. IEEE Computer. 8, 42 (2009)
[7] S. M. McNee, J. Riedl and J. A. Konstan, Editors. Being accurate is not enough: how accuracy metrics have hurt recommender systems. Proceedings of the CHI'06 Conference on Human Factors in Computing Systems, **(2006)** April 24-27; Montréal, Canada
[8] P. Cremonesi, F. Garzotto, S. Negro, A. V. Papadopoulos and R. Turrin, Editor, Looking for "good" recommendations: A comparative evaluation of recommender systems, Springer Berlin Heidelberg, Berlin **(2011)**
[9] D. Fleder and K. Hosanagar, Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity[J]. Management science. 5, 55 (**2009)**
[10] G. Linden, B. Smith and J. York, Amazon. com recommendations: Item-to-item collaborative filtering[J]. Internet Computing. 1, 7 (2003)
[11] Y. J. Park and A. Tuzhilin, Editors. The long tail of recommender systems and how to leverage it. Proceedings of the 2th ACM conference on Recommender systems, **(2008)** October 11-18; Lausanne, Switzerland
[12] T. Zhou, Z. Kuscsik, J. G. Liu, M. Medo, J. R. Wakeling and Y. C. Zhang, Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of the National Academy of Sciences. 10, 107 **(2010)**
[13] P. Adamopoulos and A. Tuzhilin, Editors. On over-specialization and concentration bias of recommendations: Probabilistic neighborhood selection in collaborative filtering systems. Proceedings of the 8th ACM Conference on Recommender systems, **(2014)** October 6-10; Silicon Valley, USA
[14] C. N. Ziegler, S. M. Mcnee, J. A. Konstan and G. Lausen, Editors. Improving recommendation lists through topic diversification. Proceedings of the 14th International Conference on World Wide Web, **(2005)** May 10-14; Chiba, Japan
[15] G. Adomavicius and Y. O. Kwon, Improving aggregate recommendation diversity using ranking-based techniques[J]. Knowledge and Data Engineering. 5, 24 (2012)
[16] G. Adomavicius and Y. Kwon, Editors. Maximizing aggregate recommendation diversity: A graph-theoretic approach. Proceedings of the 1st International Workshop on Novelty and Diversity in Recommender Systems, **(2011)** October 23; Chicago, USA
[17] S. Vargas and P. Castells, Editors. Improving sales diversity by recommending users to items. Proceedings of the 8th ACM Conference on Recommender systems, **(2014)** October 6-10; Silicon Valley, USA
[18] H. Yin, B. Cui, J. Li, J. Yao and C Chen,Challenging the long tail recommendation. Proceedings of the VLDB Endowment. 9, 5 **(2012)**
[19] C. E. Shannon, Prediction and entropy of printed English[J]. Bell system technical journal. 1, 30 **(1951)**
[20] J. Bennett and S. Lanning, Editors. The netflix prize. Proceedings of KDD Cup and Workshop, **(2007)**, Aug 12; California, USA
[21] L. Lü, M. Medo, C. H. Yeung, Y. C. Zhang, Z. K. Zhang and T. Zhou, Recommender systems[J]. Physics Reports. 1, 519 **(2012)**

Authors

Guoyong Cai. Male, PhD, Professor, School of computer and information security, Guilin University of Electronic Technology. Research direction: Social computing, Cloud computing.

Dong Zhang. Male, MD candidate, School of computer and information security, Guilin University of Electronic Technology. Research direction: Social computing, Data mining.

Yumin Lin. Male, PhD, Associate Professor, School of computer and information security, Guilin University of Electronic Technology. Research direction: Machine Learning, Social computing.