Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 423–429

ISBN 978-83-60810-51-4

An Application of Bicriterion Shortest Paths to Collaborative Filtering

Federico Malucelli
Politecnico di Milano - DEI
Piazza Leonardo da Vinci, 32
Milan, Italy
Email: malucell@elet.polimi.it

Paolo Cremonesi
Politecnico di Milano - DEI
Piazza Leonardo da Vinci, 32
Milan, Italy
Email: paolo.cremonesi@polimi.it

Borzou Rostami
Politecnico di Milano - DEI
Piazza Leonardo da Vinci, 32
Milan, Italy
Email: rostami@elet.polimi.it

***Abstract*—Item-based collaborative filtering is one of most widely used and successful neighborhood-based collaborative recommendation approaches. The main idea of item-based algorithms is to compute predictions using the similarity between items. In such approaches, two items are similar if several users of the system have rated these items in a similar fashion. Traditional item-based collaborative filtering algorithms suffer from the lack of available ratings. When the rating data is sparse, as it happens in practice, many items without any rating in common are present. Thus similarity weights may be computed using only a small number of ratings and consequently the item-based approach will make predictions using incomplete data, resulting in biased recommendations. In this paper we present a two phase method to find the similarity between items. In the first phase a similarity matrix is found by using a traditional method. In the second phase we improve the similarity matrix by using a bicreterion path approach. This approach introduces additional similarity links by combining two or more existing links. The two criteria take into account on the one hand the distance between items on a suitable graph (min sum criterion), on the other hand the estimate of the information reliability (max min criterion). Experimental results on the Netflix and Movielens datasets showed that our approach is able to burst the accuracy of existing item-based algorithms and to outperform other algorithms.**

I. Introduction

A RECOMMENDER System (RS) filters a large amount of information to identify the items that are likely to be more interesting and attractive to a user. Recommendations are inferred on the basis of different user profile characteristics, in most cases including explicit ratings on a sample of suggested elements. Collaborative filtering (CF) recommends items on the basis of the ratings provided by groups of users [11]. There are two major approaches to collaborative filtering: (i) neighborhood models and (ii) dimensionality reduction models.

Neighborhood models base their prediction on the similarity relationships between either users or item. Algorithms based on the similarity between users predict a user's preference on an item based on the ratings that item has received from similar users. On the other hand, algorithms based on the similarity between items compute the user's preference for an item based on his/her own ratings on similar items. The latter is usually the preferred approach, as it usually performs better in terms of accuracy, while also being more scalable [19].

Item-based systems suffer from the lack of available ratings. When the rating data is sparse, it is possible to have items with few ratings in common; therefore, similarity weights may be computed using only a small number of ratings and consequently the item-based approach will make predictions using a very limited number of neighbors, resulting in a biased recommendation.

Dimensionality reduction is one of the common approaches used to overcome the problems of sparsity and scalability in CF. Decomposition of a user-rating matrix [4], [8], [23] and decomposition of a sparse similarity matrix [7] are essentially two ways in which dimensionality reduction can be used to improve recommender systems.

Graph-based approaches have been introduced to overcome the problems arising in neighborhood collaborative filtering due to sparsity. These approaches make use of a graph where nodes correspond to users, items or both, and edges represent the interactions or similarities between users and items. Recommendations are then induced by "transitive associations", that is suitable paths in the graph that have the role to reduce graph sparsity. The transitive associations can be used to recommend items in two different ways. In a first approach, the proximity of a user u to an item i in the graph is used directly to evaluate the rating of u for i [13], [25]. Following this idea, the items recommended to u by the system are those that are the closest to u in the graph. The second approach considers the proximity of two item nodes in the graph as a measure of similarity, and uses this similarity as the weights of a neighborhood-based recommendation method [6], [16].

Based on the above discussion and in order to overcome sparsity in CF, we present an optimization approach in item-based CF which is based on the item graph [24]. We define a weighted graph where nodes correspond to items and arcs are similarity link between items. For each arc a real numbers is assigned representing the reliability of the arc. In order to find a new similarity link between two items with unknown similarity in the item graph, first we formulate the problem as a bicriterion path optimization problem [21]. By applying an efficient polynomial algorithm [10] for bicriterion path optimization we find a subset of "efficient" paths in the graph as a best candidate set of paths between these two nodes.

Eventually we use the best path in the candidate set to assign the similarity weight to the new link.

The rest of the paper is organized as follows: In section 2 we review some related works which has been done so far to improve the similarity in collaborative filtering. Section 3 describes some collaborative algorithms considered in our study, to provide the needed technical background for the following sections. In section 4, we first formulate our problem as an optimization problem then we present an efficient algorithm to find the bicriterion path problem in networks and apply this algorithm to our problem. The experimental results will be provided in section 5. In section 6 the conclusion and discussion will be presented.

II. Related Work

Computation of the similarity weights is one of the most critical aspects of building a neighborhood-based recommender system. The similarity weights play a double role in the neighborhood-based recommendation methods: 1) they allow for the selection of trusted neighbors whose ratings are used in the prediction, and 2) they provide the means to give more or less importance to these neighbors in the prediction. Item based recommendation algorithm contains two main phases, the model building phase and the prediction phase. In the model building phase, the similarities between each pair of items are computed and for each particular item i, the algorithm will store its k most similar items and their similarity values with i. According to this fact that similarity weights can have a significant impact on both accuracy and performance of collaborative filtering, a lot of research has been devoted to improve the similarity in the literature.

The basic idea in similarity computation between two items i and j is to first consider the users who have rated both of the items and then apply a similarity technique to determine the similarity weight. There are a number of different ways to computing the similarity between items [19]: Cosine-based similarity, correlation-based similarity and adjusted-cosine similarity are three important methods. In a sparse dataset the possibility of existing items with just a few rating in common is high. Therefore, the item-based approach will make predictions using a very limited number of neighbors, resulting in biased recommendation.

The decomposition of the similarity matrix is one of the way to overcome the problem of sparsity which was used to recommend jokes in the Eigentaste system [7]. Moreover in the literature there are some attempts that create the similarity paths in the graph-based model to improve the similarity weights. In path-based similarity, the distance between two nodes of the graph is evaluated as a function of the number of paths connecting the two nodes, as well as the length of these paths. A recommendation approach that computes the similarity between two users based on their shortest distance in a graph is the one described in [1]. In this method, the data is modeled as a directed graph whose nodes are users, and in which edges are determined based on the notions of horting and predictability. The number of paths between a user and an item in a bipartite graph can also be used to evaluate their compatibility [13]. This method of computing distances between nodes in a graph is known as the Katz measure [14]. Another direction in collaborative filtering research combines user-based and item-based approaches. For example, [26] clusters the user data and applies intra-cluster smoothing to reduce sparsity. Also in [9] a procedure for computing similarities between elements of a database has been presented which is based on a Markov-chain model of random walk through a graph representation of the database. The presented similarity measures can be used in order to compare items belonging to database tables that are not necessarily directly connected.

The general framework of our work, like other graph-based models, is based on finding one or more paths between two items. However, in our method the similarity between items is found by considering not only the distance between two items or the length of the path joining them but also by taking to account the "reliability" of the path which connects them.

III. Collaborative filtering

Collaborative filtering recommends items on the basis of the ratings provided by groups of users. The main input to collaborative algorithms is the *user rating matrix*, where each element r_{ui} is user u's rating on item i (missing ratings are set to zero). There are two major approaches to collaborative filtering: (i) neighborhood models and (ii) latent factor models.

A. Neighborhood models

Neighborhood models base their prediction on the similarity relationships between either users or items. Algorithms based on the similarity between users predict a user's preference on an item based on the ratings that item has received from similar users. On the other hand, algorithms based on the similarity between items compute the user's preference for an item based on his/her own ratings on similar items. The latter is usually the preferred approach (e.g., [19]), as it usually performs better in terms of accuracy, while also being more scalable. Both of these advantages are due to the fact that the number of items is typically smaller than the number of users. Another advantage of the latter algorithms is that the reason why a specific recommendation was made to a user can be explained in terms of the items previously rated by him/her. In addition, basing the model on items (rather than on users) allows a seamless handling of users and ratings that are new to the model.

The similarity s_{ij} between item i and item j is measured as the tendency of users to rate items i and j similarly. It is typically based either on the cosine, the adjusted cosine, or

(more commonly) the Pearson correlation coefficient [19]

$$s_{ij} = \begin{cases} \dfrac{\sum_u r_{ui} r_{uj}}{\sqrt{\sum_u r_{ui}^2}\sqrt{\sum_u r_{uj}^2}} & \text{Cosine} \\ \dfrac{\sum_u (r_{ui}-\bar{r}_u)(r_{uj}-\bar{r}_u)}{\sqrt{\sum_u (r_{ui}-\bar{r}_u)^2}\sqrt{\sum_u (r_{uj}-\bar{r}_u)^2}} & \text{Adjusted cosine} \\ \dfrac{\sum_u (r_{ui}-\bar{r}_i)(r_{uj}-\bar{r}_j)}{\sqrt{\sum_u (r_{ui}-\bar{r}_i)^2}\sqrt{\sum_u (r_{uj}-\bar{r}_j)^2}} & \text{Pearson correlation} \end{cases}$$

where $\bar{r}_u$ is the average rating of the u-th user and $\bar{r}_i$ (respectively, $\bar{r}_j$) is the average rating of the i-th (respectively, j-th) item. Summations in the cosine similarity are computed over all the users. On the contrary, summations in both the adjusted cosine and the Pearson similarities are computed only on users who have rated both items i and j – the common raters – and the similarity is set to zero for pairs of items with no common raters. In the typical case of a very sparse dataset, it is likely that some pairs of items will have a poor support (i.e., a small number of common raters), leading to a non-reliable similarity measure. This is why, if n_{ij} denotes the number of common raters and s_{ij} the similarity between item i and item j, we can define the shrunk similarity d_{ij} as the coefficient

$$d_{ij} = \frac{n_{ij}}{n_{ij} + \lambda} s_{ij} \tag{1}$$

where λ is a shrinking factor. A good value for λ is 100 [15].

Neighborhood models are further enhanced by means of a kNN (k-nearest-neighborhood) approach: when predicting a rating $\hat{r}_{ui}$ for user u on item i, only the k items rated by u that are the most similar to i are considered. The kNN approach discards the items that are poorly correlated to the target item, thus decreasing noise for improving the quality of recommendations. We denote the set of k items rated by user u, and most similar to i, as $\mathcal{D}^k(u;i)$. We have focused our attention on two item-based neighborhood algorithms, i.e., Non-Normalized Cosine Neighborhood and Direct Relations.

- *Non-Normalized Cosine Neighborhood (NNCosNgbr)* predicts the rating $\hat{r}_{ui}$ for user u on item i as the weighted average of the ratings of similar items.
  Before computing the weighted average, we normalize the ratings by removing different biases which mask the more fundamental relations between items. The bias associated with the rating of user u to item i is denoted by b_{ui} and it is subtracted from rating r_{ui}. Such biases include item-effects, which represent the fact that certain items tend to receive higher ratings than others, and user-effects, which represent the tendency of certain users to rate higher than others. For instance, a simple formulation for the bias could be

  $$b_{ui} = \bar{r} + (\bar{r}_u - \bar{r}) + (\bar{r}_i - \bar{r}) \tag{2}$$

  where $\bar{r}$ is the average of all the ratings in the user rating matrix, $\bar{r}_u$ is the average rating of the u-th user and $\bar{r}_i$ is the average rating of the i-th item. By removing the bias effects, the rating estimation is

  $$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in \mathcal{D}^k(u;i)} d_{ij} (r_{uj} - b_{uj})}{\sum_{j \in \mathcal{D}^k(u;i)} d_{ij}} \tag{3}$$

  where d_{ij} is computed as (1), and s_{ij} is measured as the Pearson correlation coefficient.
  Notice that the denominator in (3) forces the predicted rating values to fall within a defined range, e.g., $[1 \dots 5]$ for a typical star-rating system. However, for a top-N recommendation task, exact rating values are not necessary. We simply want to rank items by their appeal to the user. In such a case, we can simplify the formula by removing the denominator. A consequential benefit of this is that items with many similar neighbors, that is with a high value of $\sum_{j \in \mathcal{D}^k(u;i)} d_{ij}$, which means in turn that we have a high confidence in the recommendation, have higher rankings. Therefore, we propose to rank items with the following rating estimation

  $$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathcal{D}^k(u;i)} d_{ij} (r_{uj} - b_{uj}) \tag{4}$$

  Here $\hat{r}_{ui}$ does not represent a proper rating, but is rather a value we can use to rank the items according to user u's taste. We should note that similar non-normalized neighborhood rules have been mentioned by others [15], [5].
- *Direct Relations (DR)*. An alternative and simple way of computing the similarity between pair of items i and j in (4) is to count the number of users that rated both items, without any normalization factor

  $$d_{ij} = \#\text{ users rating both items} \tag{5}$$

  According to [2] this metric emphasizes the similarity between popular items.

B. Dimensionality reduction models

Recently, several recommender algorithms based on dimensionality reduction have been proposed. Some of the most successful realizations of dimensionality reduction models are based on *matrix factorization*. In its basic form, matrix factorization characterizes items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. These methods have become popular since they combine predictive accuracy with good scalability.

In dimensionality reduction models, each item i is associated with a vector $\mathbf{q}_i \in R^f$, and each user u is associated with a vector $\mathbf{p}_u \in R^f$, where f is the number of latent factors. For a given item i, the elements of $\mathbf{q}_i$ measure the extent to which the item possesses those factors, either positively or negatively. For a given user u, the elements of $\mathbf{p}_u$ measure the extent to which the user is interested in items that have high values for the corresponding factors, again, either positively or negatively. The resulting dot product, $\mathbf{q}_i^T \mathbf{p}_u$, captures the

interaction between user u and item i, i.e., the user's overall interest in the item's characteristics. This approximates user u's rating of item i, leading to the estimate $r_{ui} = \mathbf{q}_i^T \mathbf{p}_u$. The major challenge is to compute the mapping of each item and user to factor vectors $\mathbf{q}_i$, $\mathbf{p}_u$. After the recommender system completes this mapping, it can easily estimate the rating a user will give to any item.

Dimensionality reduction models based on *matrix factorization* informally known as Singular Value Decomposition (SVD) models. Since conventional SVD is undefined in the presence of missing values, which translate to unknown user ratings, several alternative solutions have been proposed. Earlier works fill the missing ratings with baseline estimations (e.g., average user/item rating [20]). This however leads to a very large and dense user rating matrix, whose factorization might be computationally infeasible. More recent works learn the values from the known ratings through a suitable objective function which minimizes the prediction error (e.g., RMSE). The proposed objective functions are usually regularized in order to avoid over-fitting [18]. Typically, gradient descent is applied to minimize the objective function. In this work we have considered PureSVD techniques treats missing ratings as zeros and performs a traditional SVD.

- *PureSVD.* PureSVD is a recently proposed latent factor algorithm [4]. Its rating estimation rule is based on conventional SVD, where unknown ratings are treated as zeros. In terms of predictive power, choosing zero is not very important, and we have received similar results with higher values. What is important is that the conventional SVD decomposition of the user rating matrix becomes feasible, since all matrix entries are now non-missing and it can be performed using highly-optimized tools for conventional SVD on sparse matrices. The user rating matrix $\mathbf{R}$ is estimated by the factorization [2]

$$\hat{\mathbf{R}} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{Q}^{\mathrm{T}} \tag{6}$$

  where $\mathbf{U} \in \Re^{n \times f}$ and $\mathbf{Q} \in \Re^{m \times f}$ are two orthonormal matrices representing, respectively, the left and right singular vectors associated to the top-f singular values of $\mathbf{R}$ with the highest magnitude. The top-f singular values are stored in the diagonal matrix $\mathbf{\Sigma} \in \Re^{f \times f}$. As detailed in [4], once the user rating matrix has been decomposed, the prediction rule for PureSVD can be written as

$$\hat{r}_{ui} = \mathbf{r}_u \cdot \mathbf{Q} \cdot \mathbf{q}_i{}^{\mathrm{T}} \tag{7}$$

  where $\mathbf{r}_u$ denotes user u's vector of ratings (where unknown ratings are filled with zeros), and $\mathbf{q}_i$ represents the i-th column of $\mathbf{Q}$. Note that, similarly to (3), $\hat{r}_{ui}$ is not a proper normalized rating, but can be used to rank items according to user u's interests.

IV. Bicriterion path optimization

In this section we first describe the items and relationship between them by means of a weighted graph, then we formulate the problem of finding the similarity between items with unknown relationship as a bicriterion path problem. Suppose that $M = \{1, 2, ..., m\}$ is the set of users, $V = \{1, 2, ..., n\}$ is the set items and $R \in \Re^{m \times n}$ is the user-rating matrix where each entry a_{hk} gives the rating that user h gave to item$k$, if any. Moreover, suppose that we are given the item similarity matrix $S \in \Re^{n \times n}$. For instance, S can be obtained by Cosine similarity method. Based on similarity matrix we define a weighted graph G with vertex set V and arc set $E = \{(i, j) \in V \times V : s_{ij} > 0\}$. Each arc $(i, j) \in E$ is weighted by the similarity of two items.

Now consider two non adjacent nodes a and b in the graph G that is two items with unknown similarity. Our objective is to look for a similarity weight between items a and b so as to improve the quality of the recommender system. This objective can be translated as introducing one arc between nodes a and b in the graph G. A natural way to find a new connection between two nodes in a graph is to find a path which connects these two nodes. Selecting the "best" path among all possible paths between these two nodes is very important.

Let $P = (a, i, j, ..., b)$ denote a path from node a to node b, and $\prod_{ab}$ be the set of all such paths in G. In order to find the best path(s) between two items a and b with unknown similarity weight in G, let us first define the concept of the "reliability" of a path as follow:

Definition 4.1: the reliability of a path is defined by the lowest reliability arc in the path. and

$$\textit{reliability}(P) = \min_{(i,j) \in P} s_{ij}$$

This implies that finding a path with the maximum reliability between two items a and b, corresponds to finding a path whose arc of minimum weight s_{ij} is maximum among all paths $P \in \prod_{ab}$, giving rise to the following bottleneck path problem:

$$\max_{P \in \prod_{ab}} \quad \text{reliability}(P) \tag{8}$$

One of the most critical issues with the previous problem is that the maximum reliability path might be too long in terms of arcs. Although in our formalization paths are only weighted by the value of the arc of minimum reliability, in practice it also makes sense to require that the paths should be short in terms of the number of "hops" in the path. The realization of this idea of this idea yields the following optimization problem:

$$\min_{P \in \prod_{ab}} \quad |P| \tag{9}$$

Where $|P|$ denotes the cardinality of P.

Definition 4.2: A path would be selected as a "best" path if it satisfies in the two following criteria:

Criterion 1: Selected path must have the maximum reliability in $\prod_{ab}$.

Criterion 2: Selected path must include the minimum number of "hops".

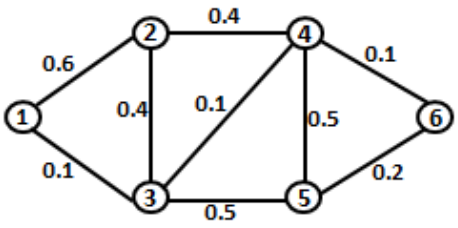

Fig. 1. Item-graph

According to criterion 1 and criterion 2 we must find minimum cardinality path with maximum reliability in G which is a kind of bicriteria path optimization problem and called MinSum-MaxMin bicriterion path optimization problem [10].

As it is highly unlikely to find a path from node a, to node b which achieves both the minimum cardinality and maximum reliability, we have to settle with something less, namely finding the set of efficient paths from a to b.

Definition 4.3: A path $P \in \prod_{ab}$ is efficient if and only if no other path $P' \in \prod_{ab}$ has a better value for one criterion and not worse value for the other one.

A path which is not efficient is thus dominated by at least one efficient path.

Definition 4.4: Two efficient paths are equivalent if and only if their value agree for both criteria.

Definition 4.5: A set $C_{ab} \subset \prod_{ab}$ of efficient paths is complete, if any path $P' \notin C_{ab}$ is either dominated or equivalent to at least one efficient path $P \in C_{ab}$.

Definition 4.6: A complete set C_{ab} is minimal if and only if no two of its efficient paths are equivalent.

According to these definitions, in the MinSum-MaxMin optimization problem we will be interested to determine a minimal complete set C^*_{ab} of efficient paths in G.

By solving a MinSum-MaxMin path optimization problem for each two non-adjacent nodes a and b in G, we find a complete set C^*_{ab} of efficient paths. Let $C^*_{ab} = \{P_1, P_2, ..., P_\ell\}$ be the complete set associated with nodes a and b. Also let $\lambda_1, ..., \lambda_\ell$ and $\mu_1, ..., \mu_\ell$ be the reliability and cardinality of the paths $P_1, P_2, ..., P_\ell$ respectively. If $\Theta_{ab} = \max\{\theta_i : \theta_i = \lambda_i/\mu_i \ for \ i = 1, 2, ..., \ell\}$, then we define the similarity between items a and b as follows:

$$\xi_{ab} = \lambda_i : \theta_i = \Theta_{ab}. \tag{10}$$

As an example consider the item graph in Figure 1 and suppose we are considering the introduction of a similarity link between items 1 and 6. By solving the bicriteria optimization problems (1) and (2) we find the minimal complete set $C^*_{16} = \{P_1, P_2\}$. Where $P_1 = (1, 2, 4, 6)$ with $\lambda_1 = 0.1$, $\mu_1 = 3$ and $P_2 = (1, 2, 4, 5, 6)$ with $\lambda_2 = 0.2$, $\mu_1 = 4$. $\Theta_{16} = \max\{\frac{0.1}{3}, \frac{0.2}{4}\}$. By using (9) the similarity weight between items 1 and 6 is found as:

$$\xi_{16} = 0.2$$

corresponding to path P_2.

V. Experimental Results

In this section we present the quality of our graph-based optimization algorithm and the recommender algorithms presented in Section III on two standard datasets: *MovieLens* [17] and a subset of the *Netflix* [3]. Both are publicly available movie rating datasets. Collected ratings are in a 1-to-5 star scale. Table I summarizes their statistical properties.

A. Testing methodology

The testing methodology adopted in this study is similar to the one described in [4]. For each dataset, known ratings are split into two subsets: training set M and test set T. The test set T contains only 5-stars ratings. So we can reasonably state that T contains items relevant to the respective users.

The original Netflix dataset was released partitioned into two parts: training-set and probe-set. Ratings in the training set M were a subset of the original Netflix training set. Some items (and the corresponding ratings) were removed because of the lack of complementary data, while other items were merged because of the different editions of the same movie existing. Ratings in the test set T were a subset of the Netflix probe-set. Again, some items and their corresponding ratings were removed or merged. Moreover, only 5-star ratings (or 1-star ratings) were retained from the Netflix probe-set, thus leading to the creation of two test sets T. The first test set only contained 5-stars and was used for the computation of recall, while the second test set only contained 1-stars and was used for the computation of fallout (see section V-B for a definition of *recall* and *fallout*).

We adopted a similar procedure for the Movielens dataset [17]. We randomly sub-sampled 1.4% of the ratings from the dataset in order to create a probe set. The training set M contains the remaining ratings. The test set T contains all the 5-star ratings from the probe set.

The same training set was used across all the algorithms, and a standard hold-out technique was adopted for the testing methodology. For each rating in the testing set, we predicted the rating together with the ratings of an additional 1000 unrated random items. The corresponding list was sorted and recommended to the user. Thus, the testing methodology used the whole training set to build the model, and recommended a list of 1000+1 items to the user.

In order to measure recall, we first trained the algorithm using the ratings in M. Then, for each item i in T that was rated 5 stars by user u, we followed these steps:

1) We randomly selected 1,000 additional items that were not rated by user u. We assumed that the user u was not interested in most of them.
2) We predicted the ratings for the test item i and for the additional 1,000 items.
3) We formed a top-N recommendation list by picking the N items with the largest predicted ratings.

Overall, we generated a number of recommendation lists equal to the number of elements in T. For each list we had a hit (e.g., a successful recommendation) if the test item was in the list.

TABLE I
STATISTICAL PROPERTIES OF MOVIELENS AND NETFLIX.

Dataset	Users	Items	Density
Movielens	6,040	3,883	4.26%
subset of Netflix	247,939	6,489	0.55%

Therefore, the overall recall r was computed by counting the number of successful recommendations over the total number of recommendations

$$r = \frac{\text{\# times the element is in the list}}{\text{\# elements in } T} \tag{11}$$

A similar approach was used to measure fallout, with the only difference being the composition of the test set T. In this case it only contained part of the 1-star ratings. Therefore we can reasonably state that this test set contained items that were not relevant to the users. The fallout f was defined as

$$f = \frac{\text{\# times the element is in the list}}{\text{\# elements in } T} \tag{12}$$

B. Evaluation Metric

To evaluate the algorithms we employed the receiver operator characteristic (ROC) curve [22] advocated for recommender system evaluation by Herlocker [12]. ROC curves are suited for tracking performance in binary classification tasks while varying a parameter of the classifier (usually, the number of classified items). RS applications are cast as binary classification when we classify a user/item pair as like/does-not-like (rating prediction) or purchased/did-not-purchase (implicit rating prediction). To create the ROC curve we vary the number of recommended items in a ranked list that we use as recommendations. ROC curves plot the miss rate (fallout) on the x-axis against the hit rate (recall) on the y-axis. Recall measures the percentage of items in the catalog interesting for the user and that the recommender system is able to suggest to the user. Fallout measures the percentage of items in the catalog not interesting for the the user and that the recommender system erroneously suggests to the user. Ideally, a good algorithm should have high recall (i.e. it should be able to recommend items of interest to the user) and low fallout (i.e. it should avoid recommending items of no interest to the user).

Figures 2 and 3 represent the results by using ROC curves. For both datasets, the graph-based algorithm outperforms other algorithms, as the best results are obtained when a curve is close to the upper-left corner of the diagram (i.e., low fall-out and large recall).

The improvement of quality is more evident on the Netflix dataset. This is an expected result, as in our experiments the Netflix dataset is sparser than the Movielens dataset and traditional CF techniques suffer from the sparsity problem. For instance, the number of similarity links between items in traditional item-based algorithms – such as NNCosKnn and DR – is low if the input user rating matrix is too sparse. On the contrary, our graph-based approach is able to overcame this problem by finding additional similarity links.

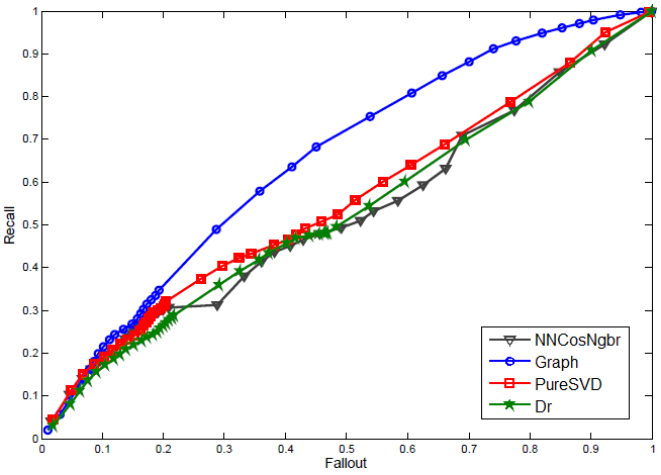

Fig. 2. ROC curves (Netflix data set)

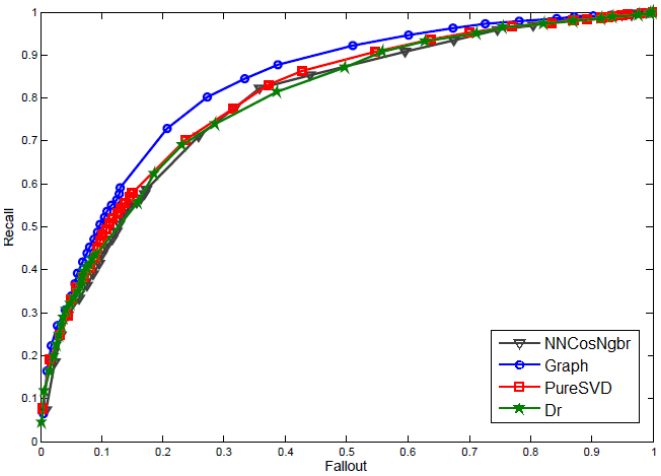

Fig. 3. ROC curves (MovieLens data set)

VI. CONCLUSION

In order to overcome the problem of sparsity in item-based CF, we introduced a new optimization approach which is based on the graph representation of the item similarity matrix. To find a new similarity link between two items with unknown similarity, in the item graph, we proposed two optimization criteria: (i) paths with maximum reliability and (ii) paths with minimum cardinality in terms of number of "hops". By solving a bicriterion path optimization problem we found all possible efficient paths in the graph then we chose the best similarity weight by using a simple optimization approach. Eventually we use the best path in the candidate set to assign the similarity weight to the new link. The experiments showed that our new

bicriteria optimization framework is effective in improving the prediction accuracy of collaborative filtering and dealing with the data sparsity problem.

REFERENCES

[1] C. C. Aggarwal, J. L. Wolf, K. Wu, P. S. Yu. Horting hatches an egg. A new graph-theoretic approach to collaborative filtering. In: KDD '99: Proc. of the 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 201-212, ACM, New York, NY, USA, 1999.

[2] R.Bambinip, P. Cremonesi, R. Turrin. Recommender Systems Handbook. to appear, Springer, Chapter A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment,2010.

[3] J. Bennet, S. Lanning. The netflix prize.Proc. of the KDD Cup and Workshop, 2007.

[4] P. Cremonesi, Y. Koren, R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In RecSys, pp. 39-46, 2010.

[5] M. Deshpande and G. Karypis. Item-based top-N recommendation algorithms. ACM Transactions on Information Systems (TOIS), 22(1), pp. 143Ű177, 2004.

[6] F. Fouss, J.M. Renders, A. Pirotte, M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Transactions on Knowledge and Data Engineering 19(3), pp. 355-369, 2007.

[7] K. Goldberg, T. Roeder, D. Gupta, C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval 4(2), pp. 133-151, 2001.

[8] G.H. Golub, C.F. Van Loan. Matrix computations (3rd ed.). Johns Hopkins University Press,1996.

[9] M. Gori, A. Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In Proc. of the 2007 IJCAI Conf., pp. 2766-2771,2007.

[10] P. Hansen. Bicriterion path problems. In: Multiple criteria decision making: theory and applications. Heidelberg: Springer, pp. 109-27, 1980.

[11] J.Herlocker, J. Konstan, L. Teveen, J. Riedl. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS) 22, 1, pp. 5Ű53,2004.

[12] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In Proceedings of the Conference on Research and Development in Information Retrieval, 1999.

[13] Z. Huang, H. Chen, D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Transactions on Information Systems 22(1), pp. 116-142, 2004.

[14] L. Katz. A new status index derived from sociometric analysis. Psychometrika 18(1), pp. 39-43, 1953.

[15] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In KDD Š08: In Proc. of the 4th ACM SIGKDD int. Conf on Knowledge discovery and data mining. ACM, New York, NY, USA, pp. 426Ű434,2008.

[16] H. Luo, C. Niu, R. Shen, C. Ullrich. A collaborative filtering framework based on both local user similarity and global user similarity. Machine Learning 72(3), pp. 231-245, 2008.

[17] B. Miller, I. Albert, S. Lam, J. Konstan, J. Riedl. MovieLens unplugged: experiences with an occasionally connected recommender system. Proc. of the 8th Int. Conf on Intelligent user interfaces, pp. 263-266, 2003.

[18] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In Proc. of KDD Cup and Workshop, 2007.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In Proc. of the WWW Conf., 2001.

[20] B. Sarwar, G. Karypis, J. Konstan, J. Riedl. Application of Dimensionality Reduction in Recommender System-A Case Study. Defense Technical Information Center,2000.

[21] RE. Steuer. Multiple criteria optimization: theory, computation, and application New York: Wiley, 1986.

[22] J. A. Swets. Measuring the accuracy of diagnostic systems. Science 240, pp. 1285-1293, 1988.

[23] G. Takńacs, I. Pilńaszy, B. Nńemeth, D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In Proc. of the 2nd KDD Workshop on Large Scale Recommender Systems and the Netflix Prize Competition, 2008.

[24] F. Wang, S. Ma, L. Yang, and T. Li. Recommendation on item graphs, In Proc. of the Sixth Int. Conf. on Data Mining, ser. ICDM '06. Washington,DC, USA: IEEE Computer Society, pp. 1119-1123, 2006.

[25] R. C. Wilson, E. R. Hancock, B. Luo. Pattern Vectors from Algebraic Graph Theory. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2005.

[26] G.R. Xue, C. Lin, Q. Yang, W. Xi, H.J. Zeng, Y. Yu, Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In Proc. of SIGIR, 2005.