Improve Collaborative Filtering Through Bordered Block Diagonal Form Matrices

Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science & Technology, Tsinghua University, Beijing, 100084, China
zhangyf07@gmail.com {z-m,yiqunliu,msp}@tsinghua.edu.cn

ABSTRACT

Collaborative Filtering-based recommendation algorithms have achieved widespread success on the Web, but little work has been performed to investigate appropriate user-item relationship structures of rating matrices. This paper presents a novel and general collaborative filtering framework based on (Approximate) Bordered Block Diagonal Form structure of user-item rating matrices. We show formally that matrices in (A)BBDF structures correspond to community detection on the corresponding bipartite graphs, and they reveal relationships among users and items intuitionally in recommendation tasks. By this framework, general and special interests of a user are distinguished, which helps to improve prediction accuracy in collaborative filtering tasks. Experimental results on four real-world datasets, including the Yahoo! Music dataset, which is currently the largest, show that the proposed framework helps many traditional collaborative filtering algorithms, such as User-based, Item-based, SVD and NMF approaches, to make more accurate rating predictions. Moreover, by leveraging smaller and denser submatrices to make predictions, this framework contributes to the scalability of recommender systems.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Filtering; H.3.5 [**Online Information Services**]: Web-based services

Keywords

Collaborative Filtering; Community Detection; Block Diagonal Form; Graph Partitioning

1. INTRODUCTION

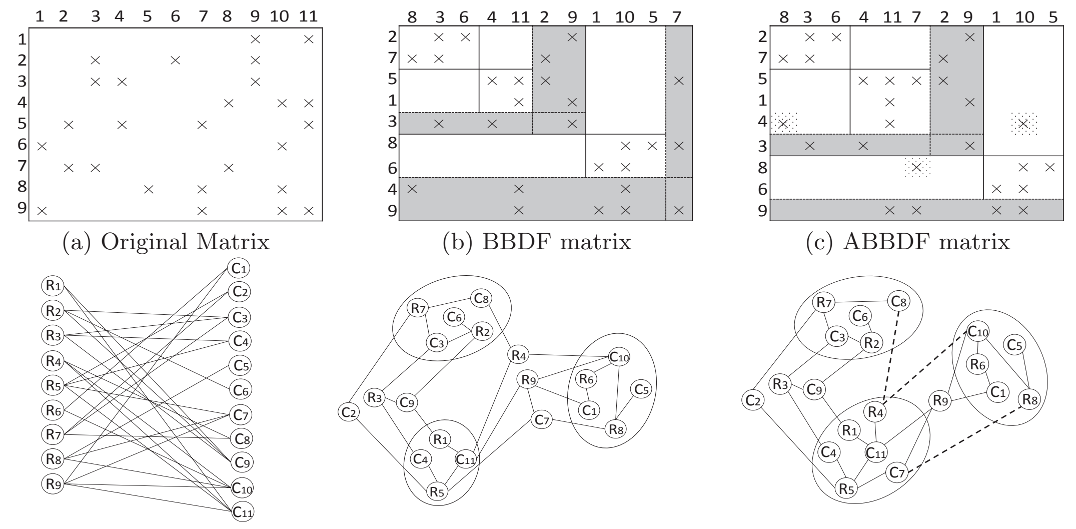
Recommender Systems play an important role on the Web which is becoming more and more personalized. With their ability to discover various items of potential interest to users, recommender systems benefit users by saving time when the users are looking for what they want, and they benefit online shops that seek to expand their marketing efforts.

Collaborative Filtering (CF) [22] recommendation algorithms based on user-item rating matrices have achieved great success in recommender systems. Typically, they take a rating matrix (Figure 1(a)) as input to make rating predictions, where each row/column/cross represents a user/item/rating. An important advantage of CF-based algorithms is their ability to make recommendations without clear content descriptions of the items, which is the reason why they are widely applied in recommender systems thus far [22].

However, CF-based recommendation algorithms also suffer from several drawbacks. First is the data sparsity problem, which usually decreases the accuracy of the rating predictions. Second is the problem of scalability, which is usually caused by the presence of computationally expensive training components. Finally, the lack of distinguishing users and items from different communities, which makes them incapable of detecting users' special interests and decreases their abilities to make long tail recommendations.

Some previous research has been performed attempting to address these problems, which mainly focus on various matrix-clustering [11, 26, 27, 39, 34] or community detection [6, 36, 23] techniques. Clustering-based approaches cluster users and/or items for CF. However, in real-world applications, clusters are usually difficult to interpret. Moreover, they usually assume that a user or an item should fall into one particular cluster, which might not be a reasonable assumption in reality. Community detection approaches based on user-item bipartite graphs attempt to improve accuracy and diversity by detecting user-item communities. Distinguishing the general and special interests of a given user helps to make better recommendations, but it is difficult for them to take advantage of various successful CF techniques on rating matrices in real-world recommender systems.

In fact, rating matrices and bipartite graphs can be equivalently transformed into each other. Before problem formalization, we would like to use an intuitional example to introduce the matrix structures that are leveraged in this framework. Figure 1(a) shows a rating matrix, and its corresponding bipartite graph is shown in Figure 1(d), where each row/column/non-zero of the matrix is represented as an R-node/C-node/Edge in the graph. Figure 1(b) is a *Bordered Block Diagonal Form (BBDF)* structure of the original matrix, where Row4, Row9 and Column7 are permuted to 'borders', and the remaining parts are thus permuted into two 'diagonal blocks'. The permuting procedure is conducted recursively on the first diagonal block. The *Approximate*



(a) Original Matrix (b) BBDF matrix (c) ABBDF matrix

(d) Original Graph (e) BBDF Graph (f) ABBDF Graph

Figure 1: An example of User-Item Rating Matrices and their corresponding Bipartite Graphs.

Bordered Block Diagonal Form (ABBDF) structure is a generalization of BBDF where scattered non-zeros are allowed in off-diagonal blank areas, as shown in Figure 1(c).

It is important to note that the ABBDF structure of rating matrices is capable of representing community detection results on the corresponding bipartite graphs. In Figure 1(f), for example, each community in the graph corresponds to a diagonal block in Figure 1(c); nodes out of communities constitute borders, and dotted inter-community edges correspond to the non-zeros in off-diagonal blank areas. In this sense, BBDF structure is a special case of ABBDF when the community detection result contains no inter-community edge, as shown by the example in Figure 1(e) and 1(b).

The CF framework based on (A)BBDF structures of rating matrices proposed in this paper is capable of combining the advantages of community detection and matrix clustering techniques as well as making use of various CF algorithms. An important advantage of this framework is that it does not rely on a specific CF algorithm. Any CF algorithm that is based on rating matrices can be integrated into this framework, such as User-based, Item-based, SVD and NMF. Experimental results show that this framework helps these CF algorithms to improve the prediction accuracy and, at the same time, benefits system scalability.

In summary, the contributions of the paper are four-fold:

- The relationship between (A)BBDF structures of rating matrices and community detection on the corresponding bipartite graphs is investigated formally.
- Two density-based algorithms are designed to transform sparse rating matrices into (A)BBDF structures.
- We propose a general collaborative filtering framework based on these structures to make rating predictions in recommendation tasks.
- Both the efficiency and effectiveness of the proposed framework are verified through experimental studies on four benchmark datasets.

In the remaining part of this paper, section 2 reviews some related work, and section 3 introduces some definitions and theorems. Section 4 presents the proposed algorithms and framework. Experimental settings and results are shown in section 5. In section 6, we present a discussion, and section 7 concludes the work and provides future directions.

2. RELATED WORK

Collaborative Filtering (CF) [22, 2] algorithms based on user-item rating matrices focus on the core task of making rating predictions. They attempt to discover and leverage the knowledge of users' preferences when making recommendations. Unlike Content-Based Filtering (CBF) [28], which makes recommendations by analyzing the item features of a user's historical items, CF algorithms take advantage of the wisdom of crowds, and they usually have no special requirements on items or domains.

One of the most widely used forms of CF is the nearest neighbor approach [22]. User-based [29] and Item-based [31] CF algorithms are two best-known methods that fall into this category. Nearest neighbors can be determined by various similarity measures, such as Pearson correlation and cosine similarity. User-based CF attempts to find the neighborhood of like-minded users for each user and predicts a user's ratings according to the ratings given by the user's neighbors. Similarly, Item-based CF takes advantage of the similar items of each item, and predictions for a user are determined by the user's historical ratings. Nearest neighbor approaches are usually unable to detect item synonymies and are also computationally expensive in real-world recommender systems.

The Matrix Factorization (MF) [16] approaches attempt to factorize a rating matrix into products of real-valued component matrices. A reconstruction error objective function is usually defined, and gradient descent optimization procedures are usually conducted [25]. SVD [33, 37] and NMF [17, 40, 19] methods are typical algorithms that are investigated. However, computationally intensive training components of these techniques make them not scalable enough and impractical to conduct frequent model re-training. Incremental and distributed versions of SVD and NMF algorithms [4, 37, 32, 21, 10] partially alleviate this problem, but they are still not efficient enough because the effects of small updates to the rating matrix are not localized.

Various matrix clustering techniques have been investigated in an attempt to address the problems of efficiency, scalability and sparsity. User clustering and item clustering methods [26] cluster user or item vectors first, and nearest neighbors of a user or item are restricted to its cluster. Some other matrix clustering techniques, such as co-clustering [7, 18, 11], ping-pang algorithm [27] and clustered low-rank approximation [35], cluster users and items at the same time, and the procedure of rating prediction takes advantage of these user-item clusters. By utilizing clusters, the scalability of recommender systems is usually improved, but clusters are usually difficult to interpret. In addition, these approaches usually force a user or item to fall into a single cluster, which might not be a reasonable assumption in real-world applications.

Recently, community detection techniques based on graphs have been investigated extensively with the rapid growth of social networks [20, 24, 23, 6], which helps to improve both the accuracy and diversity of the recommender systems by extracting user or item communities.

In fact, user-item rating matrices can be equivalently transformed into bipartite graphs [1, 3], and community detection results on the bipartite graphs can be represented as (A)BBDF structures on the corresponding rating matrices. Any CF method can still be applied to the permuted matrix without any modification, but by leveraging user-item community information therein, more accurate and specific recommendations can be made.

3. DEFINITIONS AND THEOREMS

Several definitions and theorems are presented in this section, which will be the basis of the (A)BBDF permutation algorithms and the collaborative filtering framework to be proposed in Section 4.

Definition 1. **Bordered Block Diagonal Form (BBDF).** *Matrix A is in Bordered Block Diagonal Form if:*

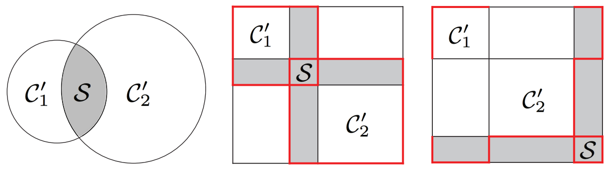
$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} & A_{1B} \\ A_{21} & A_{22} & \cdots & A_{2k} & A_{2B} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{k1} & A_{k2} & \cdots & A_{kk} & A_{kB} \\ A_{B1} & A_{B2} & \cdots & A_{Bk} & A_{BB} \end{bmatrix} = \begin{bmatrix} D_1 & & & & C_1 \\ & D_2 & & & C_2 \\ & & \ddots & & \vdots \\ & & & D_k & C_k \\ R_1 & R_2 & \cdots & R_k & B \end{bmatrix} \quad (1)$$

Namely, $A_{ij} = \mathbf{0}$ $(i \neq j, 1 \leq i, j \leq k)$. Each D_i $(1 \leq i \leq k)$ is a 'diagonal block'; $R = [R_1 \cdots R_k B]$ and $C = [C_1^T \cdots C_k^T B^T]^T$ are 'borders'. Recursively, each of the diagonal blocks D_i can also be in the BBDF structure. □

BBDF structure is a generalization of Block Diagonal Form (BDF) matrices, for example, $A = \text{diag}(D_1 D_2 \cdots D_k)$, where the latter has no border.

Definition 2. **Approximate Bordered Block Diagonal Form (ABBDF).** *Matrix A is in Approximate Bordered Block Diagonal Form if:*

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} & A_{1B} \\ A_{21} & A_{22} & \cdots & A_{2k} & A_{2B} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{k1} & A_{k2} & \cdots & A_{kk} & A_{kB} \\ A_{B1} & A_{B2} & \cdots & A_{Bk} & A_{BB} \end{bmatrix} = \begin{bmatrix} D_1 & A_{12} & \cdots & A_{1k} & C_1 \\ A_{21} & D_2 & \cdots & A_{2k} & C_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{k1} & A_{k2} & \cdots & D_k & C_k \\ R_1 & R_2 & \cdots & R_k & B \end{bmatrix} \quad (2)$$



(a) Communities (b) Middle result (c) ABBDF

Figure 2: Community detection and ABBDF, $k = 2$

Namely, A_{ij} $(i \neq j, 1 \leq i, j \leq k)$ can also contain scattered non-zeros compared with the BBDF structure. $D_1 D_2 \cdots D_k$ might also be in the ABBDF structure. □

ABBDF is a generalization of BBDF in that scattered non-zeros are allowed in off-diagonal blank areas. We also refer to BBDF as Accurate BBDF to distinguish it from Approximate BBDF.

Definition 3. **Community Detection.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a community in $\mathcal{G}$ is a set of vertices $\mathcal{C}_i \subseteq \mathcal{V}$, and a community detection with k $(k \geq 1)$ communities is $\mathcal{C} = \{\mathcal{C}_1 \mathcal{C}_2 \cdots \mathcal{C}_k\}$, witch satisfies the following:*

i. $\mathcal{C}_i \neq \emptyset$ $(1 \leq i \leq k)$

ii. $\mathcal{C}_i \cap \left(\bigcup_{j=1, j\neq i}^{k} \mathcal{C}_j \right) \neq \mathcal{C}_i$ $(1 \leq i \leq k)$

The second requirement means that each community contains at least one monopolized node. It's important to clarify that no definition of community detection is universally accepted until now because definitions often depend on a specific application and algorithm [9]. This definition is solely for introducing the relationships between the (A)BBDF structure and community detection in this paper. Even so, Definition 3 is general enough to describe the output of many frequently used community detection algorithms, including both independent and overlapping communities, depending on $\mathcal{C}_i \cap \left(\bigcup_{j=1, j\neq i}^{k} \mathcal{C}_j \right) = \emptyset$ $(1 \leq i \leq k)$ or not.

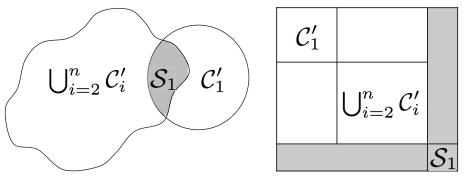
Theorem 1. *Any community detection result with k communities $\mathcal{C} = \{\mathcal{C}_1 \mathcal{C}_2 \cdots \mathcal{C}_k\}$ on a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be represented as an ABBDF structure with k diagonal blocks on its corresponding rating matrix A.*

Proof. Without loss of generality, we assume that $\bigcup_{i=1}^{k} \mathcal{C}_i = \mathcal{V}$, in other words, that each node belongs to at least one community. Otherwise, we permute the nodes in $\mathcal{V} - \bigcup_{i=1}^{k} \mathcal{C}_i$ to the borders first. We will make a proof by induction.

If $k = 1$, then the property holds naturally, and the matrix is viewed as a single diagonal block.

If $k = 2$, then permute the nodes in $\mathcal{S} = \mathcal{C}_1 \cap \mathcal{C}_2$ to borders, followed by permuting the nodes in $\mathcal{C}_1' = \mathcal{C}_1 - \mathcal{S}$ and $\mathcal{C}_2' = \mathcal{C}_2 - \mathcal{S}$ to construct two diagonal blocks, shown in Figure 2.

Suppose that the property holds for $k = n - 1$. When $k = n$, as shown in Figure 3, let $\mathcal{S}_1 = \mathcal{C}_1 \cap \left(\bigcup_{i=2}^{n} \mathcal{C}_i \right)$ and $\mathcal{C}_i' = \mathcal{C}_i - \mathcal{S}_1$ $(1 \leq i \leq n)$. Note that $\mathcal{C}_i' \neq \emptyset$ $(1 \leq i \leq n)$,



$\bigcup_{i=2}^{n} \mathcal{C}_i'$
$\mathcal{S}_1$
$\mathcal{C}_1'$
$\mathcal{C}_1'$
$\bigcup_{i=2}^{n} \mathcal{C}_i'$
$\mathcal{S}_1$

Figure 3: Community detection and ABBDF, $k = n$

according to Definition 3. Permute the vertices in $\mathcal{S}_1$ to the borders, permute $\mathcal{C}'_1$ to the upper-left corner and permute $\bigcup_{i=2}^{n} \mathcal{C}'_i$ to the bottom-right corner. According to the inductive assumption, $\bigcup_{i=2}^{n} \mathcal{C}'_i$ can be permuted into an ABBDF structure with $n-1$ diagonal blocks in the same manner. As a result, the final matrix is an ABBDF structure with n diagonal blocks, and the final border is $\mathcal{S} = \bigcup_{i=1}^{n-1} \mathcal{S}_i$. $\square$

It is helpful to look at the case of $k = 3$, as shown in Figure 4. First, let $\mathcal{S}_1 = \mathcal{C}_1 \cap (\mathcal{C}_2 \cup \mathcal{C}_3)$ and $\mathcal{C}'_i = \mathcal{C}_i - \mathcal{S}_1$ $(i = 1, 2, 3)$. Permute $\mathcal{S}_1$ to the borders, and there will be two diagonal blocks in the matrix, corresponding to vertex sets $\mathcal{C}'_1$ and $\mathcal{C}'_2 \cup \mathcal{C}'_3$. Moreover, let $\mathcal{S}_2 = \mathcal{C}'_2 \cap \mathcal{C}'_3$ and $\mathcal{C}''_i = \mathcal{C}'_i - \mathcal{S}_2$ $(i = 2, 3)$. By recursively permuting $\mathcal{S}_2$ to the borders, one of the diagonal blocks is again permuted into an ABBDF structure, which gives us two new diagonal blocks that correspond to $\mathcal{C}''_2$ and $\mathcal{C}''_3$. The final number of diagonal blocks is 3, and the border is $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$.

Corollary 1. *Given a community detection result $\mathcal{C} = \{\mathcal{C}_1 \mathcal{C}_2 \cdots \mathcal{C}_k\}$ on a bipartite graph, let $\mathcal{S}_i = \mathcal{C}_i \cap \big(\bigcup_{j=1, j \neq i}^{k} \mathcal{C}_j\big)$ and $\mathcal{C}'_i = \mathcal{C}_i - \mathcal{S}_i$ for each $1 \leq i \leq k$. Then, $\mathcal{C}$ corresponds to an Accurate BBDF structure if and only if there is no edge from $\mathcal{C}'_i$ to $\mathcal{C}'_j$ when $i \neq j$.*

Proof. The result follows from the fact that a non-zero in the off-diagonal blank areas in an ABBDF structure corresponds to an edge that connects two nodes, u and v, from two communities, $\mathcal{C}(u)$ and $\mathcal{C}(v)$, where u and v are monopolized by $\mathcal{C}(u)$ and $\mathcal{C}(v)$, correspondingly. $\square$

Intuitionally, a diagonal block in an (A)BBDF matrix is a 'user-item community', with its users and items being its 'dominant' users and items. The borders can be viewed as 'super' users and items among communities. Super users are users whose tastes are relatively broad and fall into different communities. Super items are items favored by users from different communities. Users might indeed focus their main attentions on certain fields, but they do step into other fields occasionally, which is important when detecting the potential interests of a user. In this sense, an Approximate BBDF structure is more natural than an Accurate BBDF structure.

The following definitions and theorems concern how to permute a sparse rating matrix into accurate and approximate BBDF structures.

Definition 4. **Graph Partitioning by Vertex Separator (GPVS)-based Community Detection.**
Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $Adj(v)$ denotes the set of vertices adjacent to v. For a vertex subset $\mathcal{V}' \subset \mathcal{V}$, $Adj(\mathcal{V}') = \{v_j \in \mathcal{V} - \mathcal{V}' : \exists v_i \in \mathcal{V}' s.t.\ v_j \in Adj(v_i)\}$.
$\mathcal{V}_S \subset \mathcal{V}$ is a vertex separator if the subgraph induced by $\mathcal{V} - \mathcal{V}_S$ has $k \geq 2$ connected components. Formally, GPVS is defined as $\Gamma_v = \{\mathcal{V}_1 \mathcal{V}_2 \cdots \mathcal{V}_k; \mathcal{V}_S\}$, where $\mathcal{V}_i \neq \emptyset$, $\mathcal{V}_i \cap \mathcal{V}_S = \emptyset$, $Adj(\mathcal{V}_i) \subset \mathcal{V}_S$ for $1 \leq i \leq k$, $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for $1 \leq i < j \leq k$, and $(\bigcup_{i=1}^{k} \mathcal{V}_i) \cup \mathcal{V}_S = \mathcal{V}$. Note that $\mathcal{V}_S = \emptyset$ is allowed.
$\mathcal{C} = \{\mathcal{C}_1 \mathcal{C}_2 \cdots \mathcal{C}_k\}$ is the community detection result, where $\mathcal{C}_i = \mathcal{V}_i \cup \mathcal{V}_S$ $(1 \leq i \leq k)$. $\square$

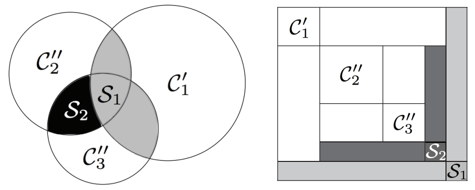
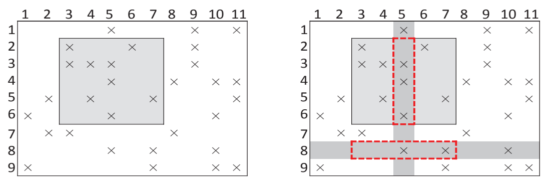


Figure 4: Community detection and ABBDF, $k = 3$


(a) A block in matrix (b) Restricted lines
Figure 5: Block, Line and Line Density

Intuitively in GPVS, the removal of a vertex separator splits the graph into k connected components. GPVS is a type of community detection algorithm that corresponds to accurate BBDF structures. This conclusion can be derived directly from Corollary 1 and Definition 4 because $\mathcal{S}_i = \mathcal{C}_i \cap \big(\bigcup_{j=1, j \neq i}^{k} \mathcal{C}_j\big) = \mathcal{V}_S$ and $\mathcal{C}'_i = \mathcal{C}_i - \mathcal{S}_i = \mathcal{V}_i$ are disconnected when $i \neq j$.

The reader can refer to Figure 1(b) and 1(e) for an example. By first removing nodes R_4 R_9 and C_7 which are represented by shaded borders, the remaining nodes are partitioned into two parts, corresponding to the two main diagonal blocks in the rating matrix. Furthermore, by removing nodes R_3 C_2 and C_9, one of the diagonal blocks is again permuted into the BBDF structure with two diagonal blocks.

Definition 5. **Graph Partitioning by Edge Separator (GPES)-based Community Detection.**
Consider the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
$\mathcal{E}_S \subset \mathcal{E}$ is an edge separator if the removal of $\mathcal{E}_S$ gives $\Gamma_e = \{\mathcal{V}_1 \mathcal{V}_2 \cdots \mathcal{V}_k\}$ $(k \geq 2)$, where $\mathcal{V}_i \neq \emptyset$ for $1 \leq i \leq k$, $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for $1 \leq i < j \leq k$, $\bigcup_{i=1}^{k} \mathcal{V}_i = \mathcal{V}$, and the subgraphs induced by $\mathcal{V}_i$ and $\mathcal{V}_j$ are disconnected for $i \neq j$.
$\mathcal{C} = \{\mathcal{C}_1 \mathcal{C}_2 \cdots \mathcal{C}_k\}$ is the community detection result, where $\mathcal{C}_i = \mathcal{V}_i$ $(1 \leq i \leq k)$. $\square$

GPES-based community detection corresponds to an ABBDF structure with no border because $\mathcal{S}_i = \mathcal{C}_i \cap \big(\bigcup_{j=1, j \neq i}^{k} \mathcal{C}_j\big) = \emptyset$. This structure can be used to further construct an ABBDF structure with borders when necessary, which will be presented by the algorithms in the following sections.

Definition 6. **Density.** *Let A be an $m \times n$ $(m, n \geq 1)$ matrix, let $\mathrm{n}(A)$ be the number of non-zeros in A, and let $\mathrm{area}(A) = m \times n$ be the area of A.*
The density of A is $\rho(A) = \frac{\mathrm{n}(A)}{\mathrm{area}(A)}$, and the average density of k matrices $A_1 \cdots A_k$ is $\bar{\rho}(A_1 \cdots A_k) = \frac{\sum_{i=1}^{k} \mathrm{n}(A_i)}{\sum_{i=1}^{k} \mathrm{area}(A_i)}$. Let $\mathcal{G}$ denote the bipartite graph of A; then, $\rho(\mathcal{G}) = \rho(A)$, $\bar{\rho}(\mathcal{G}_1 \cdots \mathcal{G}_k) = \bar{\rho}(A_1 \cdots A_k)$.
A row or a column of matrix A is referred to as a 'line'. The density of line l restricted to a block B is $\rho(l(B))$, where $l(B)$ denotes the sub-vector on line l that is restricted to block B, shown as the red dashed rectangles in Figure 5(b). $\square$

The density of graphs or matrices has been widely used in various community detection algorithms and tasks [9]. Taking Figure 5 as an example, the density of the shaded block is $\frac{9}{25}$ in Figure 5(a). In Figure 5(b), the density of column 5 is $\frac{5}{9}$, and the density of row 8 restricted to the shaded block is $\frac{2}{5}$.

4. ALGORITHMS

4.1 Accurate BBDF Permutation

4.1.1 Basic Procedure of BBDF Permutation

In accurate BBDF permutation, a basic procedure is performed recursively, which is to permute some lines to borders and to permute the remaining part to construct several diagonal blocks. This recursive framework is known as *George's nested dissection approach* [3, 14], which has been widely used in fill-reducing ordering of sparse matrices. This basic procedure is investigated in this subsection, and the accurate BBDF permutation algorithm in the following subsection leverages this procedure.

The basic step is equivalent to GPVS [1] on the corresponding user-item bipartite graph, which has been shown in preliminaries. It is typical that a graph has more than one vertex separator, and a proper GPVS algorithm attempts to find the one with the least number of vertices, namely, the *minimum vertex separator*. Unfortunately, the minimum vertex cut problem is known to be NP-hard [5], but this problem has been investigated extensively, and many efficient and high-quality heuristic-based methods have been proposed [15], such as the multilevel approach, spectral partitioning and kernel-based methods. It has been verified both theoretically and experimentally that multilevel approaches usually provides both fast execution time and very high quality partitions [15, 30, 1, 3]. Perhaps the most widely known and used package for graph partitioning is Metis by Karypis [13], which is based on a multilevel approach, and we chose the core multilevel graph partitioning routine implemented in Metis as the basic GPVS algorithm.

In this study, we utilize the *density* of user-item communities to control the procedure of BBDF permutation because dense subgraphs are usually interpreted as actual communities. This approach has been widely used in community detection tasks [9]. Algorithm 1 shows the basic procedure.

Algorithm 1 Basic-BBDF-Permutation$(A, \mathcal{G})$

```
Require:
    User-Item rating matrix $A$.
    Bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) = (\mathcal{R} \cup \mathcal{C}, \mathcal{E})$ of $A$. ▷ $\mathcal{R}/\mathcal{C}$ are row/column vertex sets of $\mathcal{V}$ correspondingly.
Ensure:
    Average density of resulting diagonal blocks $\bar{\rho}$.
1: $\Gamma_v \leftarrow \{\mathcal{V}_1\mathcal{V}_2 \cdots \mathcal{V}_k; \mathcal{V}_S\} \leftarrow$ GPVS$(\mathcal{G})$
2: Permute rows of $A$ in order of $\mathcal{R}_1\mathcal{R}_2 \cdots \mathcal{R}_k\mathcal{R}_S$
3: Permute columns of $A$ in order of $\mathcal{C}_1\mathcal{C}_2 \cdots \mathcal{C}_k\mathcal{C}_S$
4: return $\bar{\rho}(D_1D_2 \cdots D_k)$ ▷ $D_i$ denotes the $i$-th diagonal block which corresponds to vertex set $\mathcal{V}_i = \mathcal{R}_i \cup \mathcal{C}_i$
```

We expect the average density of diagonal blocks to be improved compared with the density of the original matrix A after permutation; we discuss the relevant issues here.

Taking the notations in Definition 6, we have:

$$\rho(A) = \frac{\mathrm{n}(A)}{\mathrm{area}(A)}, \quad \bar{\rho}(D_1 \cdots D_k) = \frac{\sum_{i=1}^{k} \mathrm{n}(D_i)}{\sum_{i=1}^{k} \mathrm{area}(D_i)}$$

Let $n = \mathrm{n}(A), n_1 = \sum_{i=1}^{k} \mathrm{n}(D_i), n_2 = n - n_1$; $s = \mathrm{area}(A)$, $s_1 = \sum_{i=1}^{k} \mathrm{area}(D_i), s_2 = s - s_1$. We have:

$$\rho_A = \rho(A) = \frac{n}{s}, \quad \rho_1 = \bar{\rho}(D_1 \cdots D_k) = \frac{n_1}{s_1}, \quad \rho_2 = \frac{n_2}{s_2}$$

where ρ_2 represents the average density of off-diagonal blank areas plus borders. Let $\bar{\rho}(D_1 \cdots D_k) > \rho(A)$, namely, $\rho_1 > \rho_A$; then, we have:

$$\frac{n_1}{s_1} > \frac{n}{s} = \frac{n_1 + n_2}{s_1 + s_2} \quad \Leftrightarrow \quad \frac{n_1}{s_1} > \frac{n_2}{s_2} \quad \Leftrightarrow \quad \rho_1 > \rho_2$$

This statement means that the average density of the resulting diagonal blocks will increase if and only if it is greater than the average density of the remaining parts. This relationship is usually satisfied because the GPVS algorithm attempts to find a separator with the minimum number of vertices. Furthermore, the total area of off-diagonal blocks plus borders tends to be much larger than the area of the diagonal blocks [1]. For this reason, we choose the average density to be an important controller in the algorithm.

4.1.2 Accurate BBDF Permutation Algorithm

The density-based BBDF permutation algorithm requires a parameter ρ as an input, which is a pre-defined requirement on the minimum average density of diagonal blocks. It conducts Basic-BBDF-Permutation on a matrix and recurses on each of the resulting diagonal blocks until the density of a diagonal block has reached the density requirement ρ or the Basic-BBDF-Permutation cannot improve the average density any more. Algorithm 2 shows the procedure.

Algorithm 2 BBDF-Permutation$(A, \mathcal{G}, \rho)$

```
Require:
    User-Item rating matrix $A$.
    Bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of $A$.
    Density requirement $\rho$.
Ensure:
    Matrix $A$ permuted into BBDF structure.
1: $\rho_A \leftarrow \rho(A)$
2: if $\rho_A < \rho$ then            ▷ else do nothing
3:     $\bar{\rho} \leftarrow$ Basic-BBDF-Permutation$(A, \mathcal{G})$
4:     if $\bar{\rho} > \rho_A$ then        ▷ else do nothing
5:         for each diagonal block $D_i$ in $A$ do
6:             BBDF-Permutation$(D_i, \mathcal{G}_{\mathcal{V}_i}, \rho)$ ▷ $\mathcal{V}_i$ denotes the vertex set of $D_i$, $\mathcal{G}_{\mathcal{V}_i}$ is the subgraph induced by $\mathcal{V}_i$
7:         end for
8:     end if
9: end if
```

Note that, in the 4-th line, we do nothing if the average density of the resulting diagonal blocks is not improved compared with the original matrix, although some diagonals might not have reached the density requirement ρ. Such a diagonal D is viewed as a sparse user-item community, and such a case occurs when the density requirement ρ is set too high. An important reason for using the average density to prevent such diagonals from recursion is that they would result in many small scattered communities with only a few users and items. Proper density requirements give better BBDF structures, and this issue will be discussed with the experimentation below.

4.2 Approximate BBDF Permutation

Approximate BBDF can be achieved by GPES, and we also make use of the multilevel graph partitioning approach that is implemented in Metis. The ABBDF permutation algorithm also has a density requirement of ρ, as input. It computes a GPES on the bipartite graph of a rating matrix

A, and by ignoring the non-zeros corresponding to the edge separator, matrix A is permuted to several diagonal blocks.

Unlike an accurate BBDF permutation, which accomplishes nothing if the average density is not improved after the permutation, the approximate BBDF permutation algorithm moves lines from diagonal blocks to borders to further improve their average density, unless the average density is greater than the original matrix. Note that this goal can always be reached because the density of a single non-zero is one. This procedure is performed recursively in each diagonal block until the density requirement is reached, as shown in Algorithm 3.

```
Algorithm 3 ABBDF-Permutation($A, \mathcal{G}, \rho$)
Require:
    User-Item rating matrix $A$.
    Bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) = (\mathcal{R} \cup \mathcal{C}, \mathcal{E})$ of $A$.
    Density requirement $\rho$.
Ensure:
    Matrix $A$ permuted into ABBDF structure.
 1: if $\rho(A) \geq \rho$ then
 2:     return
 3: else
 4:     $\Gamma_e \leftarrow \{\mathcal{V}_1\mathcal{V}_2\cdots\mathcal{V}_k\} \leftarrow$ GPES($\mathcal{G}$)
 5:     Permute rows of $A$ in order of $\mathcal{R}_1\mathcal{R}_2\cdots\mathcal{R}_k$
 6:     Permute columns of $A$ in order of $\mathcal{C}_1\mathcal{C}_2\cdots\mathcal{C}_k$
 7:     $\{\mathcal{V}'_1\mathcal{V}'_2\cdots\mathcal{V}'_k; \mathcal{V}'_S\} \leftarrow$Improve-Density($A, \mathcal{G}, \Gamma_e$)
 8:     for each diagonal block $D_i$ in $A$ do
 9:         ABBDF-Permutation($D_i, \mathcal{G}_{\mathcal{V}'_i}, \rho$)
10:     end for
11: end if
```

The sub-procedure Improve-Density selects the specific line whose removal improves the average density the most from the diagonal blocks at each time and inserts it into the borders. This process is shown in Algorithm 4.

```
Algorithm 4 Improve-Density($A, \mathcal{G}, \Gamma_e$)
Require:
    User-Item rating matrix $A$.
    Bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) = (\mathcal{R} \cup \mathcal{C}, \mathcal{E})$ of $A$.
    GPES result $\Gamma_e = \{\mathcal{V}_1\mathcal{V}_2\cdots\mathcal{V}_k\}$ of $\mathcal{G}$.
Ensure:
    Average density of diagonal blocks greater than $\rho(A)$.
 1: $\{\mathcal{V}'_1\mathcal{V}'_2\cdots\mathcal{V}'_k; \mathcal{V}'_S\} \leftarrow \{\mathcal{V}_1\mathcal{V}_2\cdots\mathcal{V}_k; \emptyset\}$
 2: while $\bar{\rho}(D_1D_2\cdots D_k) < \rho(A)$ do
 3:     $l', i' \leftarrow 0, \bar{\rho}' \leftarrow 0$
 4:     for each diagonal block $D_i$ do
 5:         for each line $l$ in $D_i$ do
 6:             $\bar{\rho} \leftarrow \frac{\sum_{j=1}^{k} \mathrm{n}(D_j) - \mathrm{n}(l(D_i))}{\sum_{j=1}^{k} \mathrm{area}(D_j) - \mathrm{area}(l(D_i))}$
 7:             if $\bar{\rho} > \bar{\rho}'$ then
 8:                 $l' \leftarrow l, i' \leftarrow i, \bar{\rho}' \leftarrow \bar{\rho}$
 9:             end if
10:         end for
11:     end for
12:     Permute line $l'$ to borders
13:     $\mathcal{V}'_{i'} \leftarrow \mathcal{V}'_{i'} - \{\mathrm{node}(l')\}$
14:     $\mathcal{V}'_S \leftarrow \mathcal{V}'_S \cup \{\mathrm{node}(l')\}$ ▷ node($l'$) denotes the node in
        $\mathcal{V}'_{i'}$ corresponding to line $l'$
15: end while
16: return $\{\mathcal{V}'_1\mathcal{V}'_2\cdots\mathcal{V}'_k; \mathcal{V}'_S\}$
```

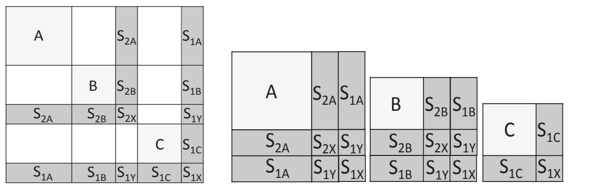
Note that, when looking for the line that improves the average density the most, there is no need to check all of the lines in each diagonal block in a real implementation. In each diagonal block, we only need to check the line with the minimum density restricted to it, namely the line that contains the fewest non-zeros in it.

4.3 (A)BBDF-based Rating Prediction

A great advantage of the CF framework based on the accurate or approximate BBDF structure is that any CF algorithm can be used on a submatrix that is made up of diagonal blocks and borders. In this paper, we do not propose new CF algorithms on (A)BBDF structures but instead propose a general CF framework, which makes use of user-item communities. By utilizing the community information, the prediction accuracy tends to be improved. Furthermore, conducting collaborative filtering on smaller and denser submatrices contributes to the scalability of CF algorithms.

The intuitive example shown in Figure 6 will be used to introduce the framework. For each diagonal block, we reconstruct its corresponding community by combining it with borders from different levels. Specifically, we use dominant users and items together with the corresponding super users and items to make predictions. In Figure 6, three submatrices are constructed, which correspond to the diagonal blocks A, B and C. Note that, in the ABBDF structure, scattered non-zeros in off-diagonal areas are ignored because they are viewed as special interests or unprofessional ratings of users.

On each submatrix, any CF algorithm can be conducted to make rating predictions. There could be more than one prediction for user-item pairs from border crosses because they are shared by different submatrices. These predictions are averaged as the final prediction. It is a natural idea that the predictions can be averaged with different weights, but in this paper, we take only the average value. The averaging strategy could be investigated in future work.



(a) BBDF matrix (b) Submatrices extracted

Figure 6: A toy example for extracting communities

5. EXPERIMENTS

5.1 Dataset Description

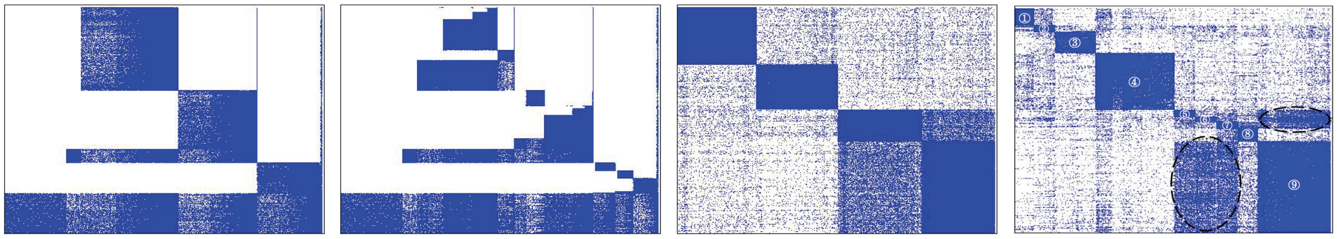
We conducted a series of experiments on four real-world datasets: MovieLens-100K, MovieLens-1M, Dianping and Yahoo! Music to validate the effectiveness of the proposed framework. Among these datasets, MovieLens-100K and MovieLens-1M are from the well-known MovieLens dataset.

Table 1: Statistics of the four datasets

	ML-100K	ML-1M	Dianping	Yahoo! Music
#users	943	6,040	11,857	1,000,990
#items	1,682	3,952	22,365	624,961
#ratings	100,000	1,000,209	510,551	262,810,175
#ratings/user	106.045	165.598	43.059	262.550
#ratings/item	59.453	253.089	22.828	420.523
average density	0.0630	0.0419	0.00193	0.000421

Table 2: Community analysis for Approximate BBDF on Dianping when density requirement is 0.005

Community	1	2	3	4	5	6	7	8	9
City	Chengdu	Shenzhen	Tianjin	Beijing	Nanjing	Suzhou	Hangzhou	Guangzhou	Shanghai
#users	323	288	922	2903	684	262	295	845	4531
#restaurants	1189	1359	2548	5011	1327	1443	1309	1274	4586
%accuracy	89.6	90.4	92.6	94.4	91.6	88.7	89.2	90.7	91.1



(a) BBDF $\rho = 0.005$ (b) BBDF $\rho = 0.01$ (c) ABBDF $\rho = 0.005$ (d) ABBDF $\rho = 0.01$

Figure 7: Accurate BBDF (BBDF) and Approximate BBDF (ABBDF) structures on Dianping dataset

Besides, we also collected a year's data from a famous restaurant rating web site *Dianping.com* (http://www.dianping.com) in China, and selected those users who made 20 or more ratings. The ratings also range from 1 to 5 like the MovieLens dataset. The Yahoo Music dataset [8] is from KDD Cup 2011, and its ratings range from 1 to 100. Statistics on these four datasets are presented in Table 1.

These datasets are chosen because they have different sizes and densities. Additionally, two of them have more users than items, and the other two are the opposite. We expect to verify whether the framework works regardless of the size or density of the matrix. A significant difference between Dianping and other frequently used datasets is that Dianping is a location-based service in which the longitude and latitude of each restaurant is recorded. By analyzing the locations of user-item communities discovered by (A)BBDF algorithms, we expect to obtain an intuitional observation of their effects.

5.2 Algorithms and Evaluation Metrics

Four popular CF algorithms were experimented on using the framework. The User-based and Item-based CF algorithms are famous memory based approaches, while SVD and NMF are known to be famous matrix factorization approaches.

User-based: The Pearson correlation was used for user similarities, as suggested in [29]. The neighborhood size is $k = 100$.

Item-based: Adjusted cosine similarity was used because it is reported to give the best performance in [31], and $k = 100$ is also used for neighborhood size.

SVD: We use the popular SVD prediction strategy presented in [16]. Here, the number of factors k is 100, and the regularization coefficient λ is 0.015.

NMF: The most commonly used non-negative matrix factorization algorithm in [17] was used to make predictions, and we also used $k = 100$ and $\lambda = 0.015$.

To make a comparison with the literature, we used the Root Mean Square Error (RMSE) to measure the prediction accuracy in this work. Five-fold cross validation was conducted on the MovieLens and Dianping datasets, and the average RMSE was calculated. For the Yahoo! Music dataset, we used its training set and validation set for training and evaluation, respectively.

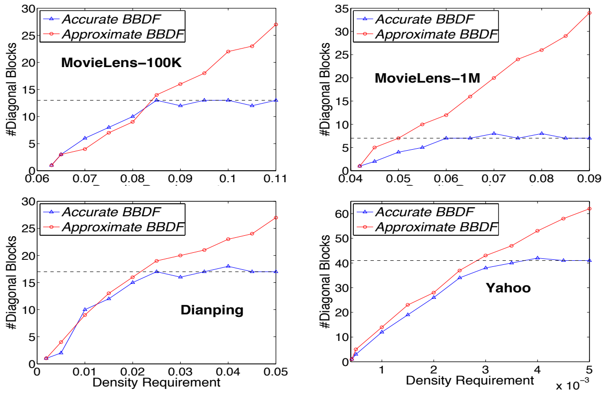


Figure 8: #communities v.s. density requirement

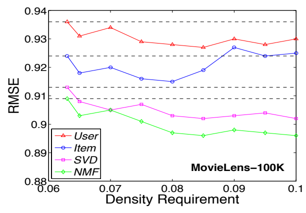
5.3 Community Analysis

The most important parameter in the density-based BBDF and ABBDF permutation algorithms is the density requirement ρ. Low density requirements lead to fewer and larger user-item communities, and high density requirements result in more and smaller communities, as shown in Figure 7, where $\rho = 0.005$ and $\rho = 0.01$ are applied to the accurate and approximate BBDF algorithms on the Dianping dataset. Note that the matrix is not as dense as it appears to be in these figures; The appearance of density arises because more than half a million points are restricted to a small canvas.

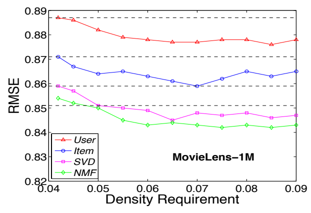
An appropriate density requirement is important. If the density requirement is too low, then the user-item communities hidden in the original rating matrix cannot be extracted properly and completely. However, if the density requirement is too high, then it will result in many small scattered communities, which could lead to over-fitting problems.

An appropriate density requirement gives reasonable and meaningful communities. See the ABBDF structure in Figure 7(d), for example. Nine communities are extracted, numbered 1 through 9, from the upper-left to bottom-right corner. It is interesting to note that they represent nine main cities in China. Table 2 presents these cities and shows how high a percentage of restaurants in each community truly belong to that corresponding city.

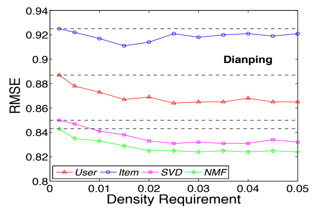
One can find that communities 5, 6, 7 and 9 are highly related to each other by the dense pseudo-blank areas marked by circles. By viewing the dense areas horizontally, we know



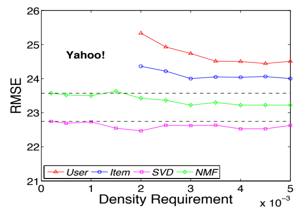
(a) acc.BBDF ML-100K



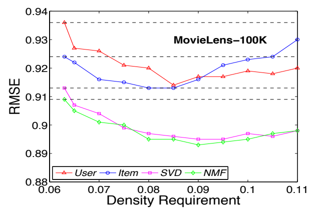
(b) acc.BBDF ML-1M



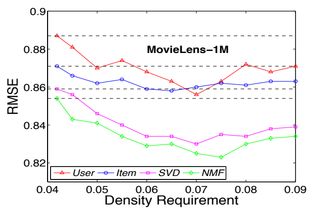
(c) acc.BBDF Dianping



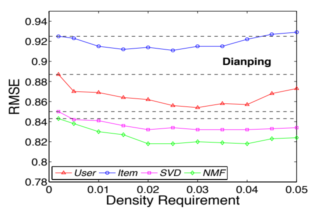
(d) acc.BBDF Yahoo!



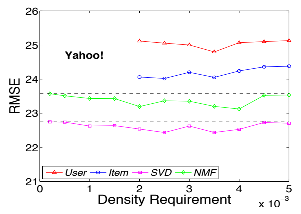
(e) app.BBDF ML-100K



(f) app.BBDF ML-1M



(g) app.BBDF Dianping



(h) app.BBDF Yahoo!

Figure 9: RMSE on the four datasets based on accurate (acc.) and approximate (app.) BBDF structures

that users in one city can usually go to restaurants in another city, and vertically we know that restaurants in the other city can be frequently visited by users from the first city. These observations are reasonable in reality because these four cities are geographically very close to each other, as shown in Figure 10. Traffic among these cities is developed, and their business relationships are close. In fact, this area is exactly the famous Yangtze River Delta of China.

Figure 8 shows the relationship between the number of extracted communities and the density requirement on the four datasets, respectively. For accurate BBDF, the number of communities rises first, and then tends to be stable after a certain density requirement is met because a diagonal block will not be permuted recursively if its average density does not increase after graph partitioning. However, for approximate BBDF, the number of communities increases consistently because it moves lines to borders to increase the average density in such cases. This strategy makes the algorithm more flexible but could also lead to too many small scattered communities if the density requirement is too high. The impact of the density requirement on the prediction accuracy will be analyzed next.

5.4 Prediction Accuracy

Experimental results on four datasets show that the proposed CF framework helps existing CF algorithms to improve their accuracy in a large range of density requirements.

The experimental results on the RMSE versus the density requirement is shown in Figure 9. Each sub-figure presents

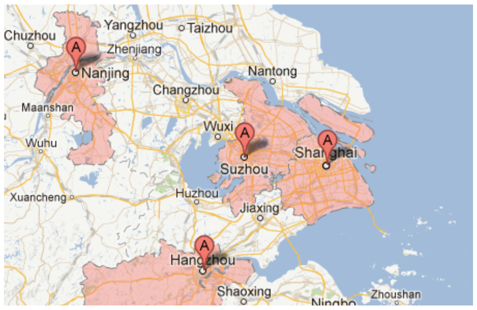


Figure 10: Location of the four highly related cities.

the performance of all of the four CF algorithms on the corresponding dataset. Note that the beginning point of each curve represents the base performance of the CF method on the dataset, and the density requirement of this point is set to the density of the whole rating matrix. As a result, points on a curve below its beginning point mean an improvement on the prediction accuracy, and vice versa.

We must note the fact that the very large number of users and items in the Yahoo! Music dataset makes it unrealistic for current hardware to conduct memory-based CF algorithms. To the best of our knowledge, there is no report of direct user-based or item-based collaborative filtering on this dataset. Strategies such as KNN with SVD features [12] or parallelization [38] were attempted in KDD cup, but these strategies would make the results incomparable with other datasets. As a result, we did not perform KNN on the whole dataset. However, we found that our CF framework makes it possible to run KNN predictions on a standalone machine when the number of communities is more than 27, and experiments were performed in such cases.

Experimental results show that the proposed CF framework on accurate BBDF benefits user-based, SVD, and NMF algorithms consistently on all of the four datasets. However, mixed results were obtained on the item-based CF method. On MovieLens-1M, Dianping and Yahoo! Music datasets, its performance is slightly improved, but on the MovieLens-100K dataset, its accuracy rises at first and begins to drop while the density requirement continues to rise. Furthermore, a negative effect can even be introduced if the density requirement is too high.

This effect could be the reason that the item-based algorithm utilizes historical ratings of a user, which is different from other CF algorithms that take advantage of neighborhood relationships. A user's historical ratings are reduced in the BBDF framework, especially on small datasets, which introduces bad predictions. However, it still benefits from BBDF structure given reasonable density requirements.

For approximate BBDF structure, prediction accuracy improves at first but tends to drop when the density requirement is too high, in almost all of the cases. The reason for the performance improvement could be that approximate BBDF plays a role in data denoising, and user-item communities help to extract professional ratings. However, small

Table 3: Best RMSE of the CF framework based on Accurate (Acc.) and Approximate (App.) BBDF structures on the four datasets and the corresponding density requirement ρ. The standard deviations are $\leq$ 0.006 on MovieLens and DianPing dataset and $\leq$ 0.05 on Yahoo! Music dataset.

Method	MovieLens-100K					MovieLens-1M					Dianping					Yahoo! Music				
	base	Acc.BBDF		App.BBDF		base	Acc.BBDF		App.BBDF		base	Acc.BBDF		App.BBDF		base	Acc.BBDF		App.BBDF	
	RMSE	ρ	RMSE	ρ	RMSE	RMSE	ρ	RMSE	ρ	RMSE	RMSE	ρ	RMSE	ρ	RMSE	RMSE	ρ	RMSE	ρ	RMSE
User-based	0.936	0.085	0.927	0.085	**0.914**	0.887	0.070	0.876	0.070	**0.856**	0.887	0.025	**0.864**	0.030	**0.854**	-	0.0045	24.443	0.0035	24.798
Item-based	0.924	0.080	0.915	0.080	0.913	0.871	0.070	0.859	0.065	0.858	0.925	0.015	0.911	0.025	0.911	-	0.0030	24.002	0.0025	24.020
SVD	0.913	0.085	0.902	0.090	0.895	0.859	0.065	0.845	0.070	**0.830**	0.850	0.025	0.831	0.020	0.832	22.746	0.0020	22.469	0.0025	22.434
NMF	0.909	0.085	0.896	0.090	0.893	0.854	0.075	0.842	0.075	**0.823**	0.843	0.030	0.824	0.020	**0.818**	23.572	0.0030	23.223	0.0040	**23.124**

scattered communities could give bad predictions because of over-fitting problems, which leads to a decrease in the performance when high density requirements are used.

Table 3 presents the best performance of each CF method on each dataset for accurate and approximate BBDF frameworks. A bold number indicates an obvious improvement, where the RMSE is reduced more than 0.02 on MovieLens and Dianping or more than 0.4 on Yahoo! Music.

When calculating the average RMSE on each dataset, five-fold cross-validation was conducted on MovieLens and DianPing, and experiments were conducted five times on Yahoo!Music. Standard deviations were $\leq$ 0.006 on MovieLens and DianPing and were $\leq$ 0.05 on Yahoo! Music.

We see that both accurate and approximate BBDF benefits CF algorithms in the best cases, and approximate BBDF tends to achieve better performance.

5.5 Scalability and Efficiency

Experiments were conducted on a linux server with 8 core 3.1GHz CPU and 64GB RAM. For both accurate and approximate BBDF algorithms on each dataset, we averaged the computational time consumed under different density requirements, as shown in Table 4.

Table 4: Computational time of the BBDF and ABBDF algorithms on the four datasets.

Dataset	ML-100K	ML-1M	Dianping	Yahoo!
accBDF	85.6ms	1.61s	21.2s	42.6min
appBDF	521.8ms	6.43s	80.3s	103.4min

Experiments show that the computational time of the BBDF algorithms increases along with the scale of rating matrices, but the time used for BBDF permutation is small compared with the CF prediction algorithms. Moreover, once BBDF structures have been constructed, they help to decrease the total prediction time by conducting collaborative filtering on smaller submatrices, especially for user-based and item-based methods. We calculated the average speedups for each CF algorithm on each dataset, as shown in Table 5, where speedup is defined as:

$$Sp = \frac{T_{CF}}{T_{BBDF} + T_{BBDF\_CF}}$$

T_{CF} is the time used by a CF prediction algorithm on the whole matrix, T_{BBDF} is the time of accurate or approximate BBDF permutation algorithms, and T_{BBDF_CF} is the time of CF prediction algorithms on (A)BBDF structures.

The experimental results show that in almost all of the cases, (A)BBDF structures speed up rating prediction; the speedup is obvious especially for nearest neighbor CF methods and large datasets.

Table 5: Speedups of conducting collaborative filtering based on (A)BBDF structures on four datasets.

Dataset	ML-100K		ML-1M		Dianping		Yahoo!	
	acc.	app.	acc.	app.	acc.	app.	acc.	app.
User	1.28	1.25	1.24	1.21	1.33	1.21	-	-
Item	1.15	1.19	1.21	1.18	1.39	1.29	-	-
SVD	1.10	1.11	1.16	1.28	1.30	1.28	1.46	1.34
NMF	1.07	1.02	1.20	1.17	1.26	1.32	1.54	1.47

6. DISCUSSION

In practical real-world recommender systems, rating matrices are usually changing continuously as new ratings are made by users. However, (A)BBDF structures make it possible to re-predict the submatrices that are truly in need of re-prediction, for example, those whose RMSE have reached a criterion, rather than to re-predict the whole matrix, which might further contribute to the scalability in real-world recommender systems.

The cold-start problem has long been an important issue in the research of collaborative filtering. By inserting a new user or item into a proper community and making recommendations there, the framework could benefit the cold-start recommendation. Additionally, ABBDF structure distinguishes a user's special interests from the general interests of his or her community, which could contribute to the serendipity of the recommendations, and these issues will be investigated in future work.

Although the (A)BBDF permutation algorithms have only one tunable parameter density requirement ρ, it could be difficult to determine a proper ρ given a new dataset in real applications because too low or too high density requirements might both not give the best performance, which is currently a shortcoming of the algorithms. Further work will be performed to investigate the possibility of learning proper density requirements automatically.

7. CONCLUSIONS

In this paper, we investigated the relationship between (A)BBDF structures and community detection on user-item bipartite graphs, and we proposed algorithms that, in fact, need only one intuitional parameter density requirement to permute a matrix into (A)BBDF structures. We further proposed a general collaborative filtering framework that is based on (A)BBDF structures to make rating predictions.

Experimental results show that, by utilizing user-item communities contained in these structures, the proposed framework benefits many CF algorithms improving their prediction accuracies, and at the same time contributes to system scalability, which means that (A)BBDF structures tend to be a general and promising framework to improve the performance of existing CF algorithms.

8. ACKNOWLEDGEMENT

The authors thank the anonymous reviewers for their constructive reviews and suggestions. This work was supported by Natural Science Foundation (60903107,61073071) and National High Technology Research and Development (863) Program (2011AA01A205) of China.

9. REFERENCES

[1] C. Aykanat, A. Pinar, and U. V. Catalyurek. Permuting Sparse Rectangular Matrices into Block-Diagonal From. *SISC*, 2004.

[2] R. M. Bell and Y. Koren. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. *Proc. ICDM*, 2007.

[3] E. Boman and M. Wolf. A Nested Dissection approach to Sparse Matrix Partitioning for Parallel Computations. *Proc. AMM*, 2007.

[4] M. Brand. Fast online SVD revisions for lightweight recommender systems. *Proc. SIAM SDM*, 2003.

[5] T. N. Bui and C. Jones. Finding Good Approximate Vertex and Edge Partitions is NP-hard. *Inform. Process. Lett.*, 1992.

[6] J. Chen and Y. Saad. Dense Subgraph Extraction with Application to Community Detection. *TKDE*, 2012.

[7] I. S. Dhillon, S. Mallela, and D. S. Modha. Information Theoretic Co-clustering. *Proc. SIGKDD*, pages 89–98, 2003.

[8] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The Yahoo! Music Dataset and KDD-Cup'11. *KDDCUP*, 2011.

[9] S. Fortunato. Community Detection in Graphs. *Physics Reports*, 486:75–174, 2010.

[10] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale Matrix Factorization with Distributed Stochastic Gradient Descent. *KDD*, 2011.

[11] T. George and S. Merugu. A Scalable Collaborative Filtering Framework based on Co-clustering. *Proc. ICDM*, 2005.

[12] M. Jahrer and A. Toscher. Collaborative Filtering Ensemble. *KDDCUP*, 2011.

[13] G. Karypis. Metis-A Software Package for Partitioning Unstructured Graphs, Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices-Version 5.0. *University of Minnesota*, 2011.

[14] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SISC*, 1998.

[15] J. Kim, I. Hwang, Y. H. Kim, and B. R. Moon. Genetic Approaches for Graph Partitioning: A Survey. *Proc. CECCO*, pages 473–480, 2011.

[16] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42:30–37, 2009.

[17] D. D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. *NIPS*, 2001.

[18] K. W. Leung, D. L. Lee, and W. Lee. CLR: A Collaborative Location Recommendation Framework based on Co-Clustering. *Proc. SIGIR*, 2011.

[19] T. Li, C. Gao, and J. Du. A NMF-based Privacy-Preserving Recommendation Algorithm. *Proc. ICISE*, 2009.

[20] W. Lin, X. Kong, P. S. Yu, Q. Wu, Y. Jia, and C. Li. Community Detection in Incomplete Information Networks. *Proc. WWW*, pages 341–349, 2012.

[21] C. Liu, H. Yang, J. Fan, L. He, and Y. Wang. Distributed Nonnegative Matrix Factorization for Web-scale Dyadic Data Analysis on MapReduce. *Proc. WWW*, pages 681–690, 2010.

[22] J. Liu, M. Z. Q. Chen, J. Chen, F. Deng, H. Zhang, Z. Zhang, and T. Zhou. Recent Advances in Personal Recommender Systems. *Jour. Info. and Sys. Sci.*, 5:230–247, 2009.

[23] F. Luo, J. Z. Wang, and E. Promislow. Exploring Local Community Structures in Large Networks. *Proc. WI*, 2006.

[24] T. Nakahara and H. Morita. Recommender System for Music CDs Using a Graph Partitioning Method. *Proc. KES*, 2009.

[25] J. Noel, S. Sanner, K. Tran, P. Christen, and L. Xie. New Objective Functions for Social Collaborative Filtering. *Proc. WWW*, pages 859–868, 2012.

[26] M. O'Connor and J. Herlocker. Clustering Items for Collaborative Filtering. *Proc. SIGIR Workshop*, 1999.

[27] S. Oyanagi, K. Kubota, and A. Nakase. Application of Matrix Clustering to Web Log Analysis and Access Prediction. *Proc. WEBKDD*, pages 13–21, 2001.

[28] M. J. Pazzani and D. Billsus. Content-Based Recommendation Systems. *The Adaptive Web LNCS*, pages 325–341, 2007.

[29] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *CSCW*, 1994.

[30] P. Sanders and C. Schulz. Engineering Multilevel Graph Partitioning Algorithms. *ESA*, 2011.

[31] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. *Proc. WWW*, pages 285–295, 2001.

[32] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. *Proc. ICCIT*, 2002.

[33] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of Dimensionality Reduction in Recommender Systems - a case study. *WebKDD*, 2000.

[34] B. Savas and I. S. Dhillon. Clustered Low Rank Approximation of Graphs in Information Science Applications. *Proc. SIAM SDM*, pages 164–175, 2011.

[35] B. Savas and I. S. Dhillon. Clustered Low Rank Approximation of Graphs in Information Science Applications. *Proc. SIAM SDM*, pages 164–175, 2011.

[36] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding. Community Discovery using Nonnegative Matrix Factorization. *Journal of DMKD*, 22, 2011.

[37] Q. Wang, J. Xu, H. Li, and N. Craswell. Regularized Latent Sematic Indexing. *Proc. SIGIR*, 2011.

[38] Y. Wu, Q. Yan, D. Bickson, et al. Efficient Multicore Collaborative Filtering. *KDDCUP*, 2011.

[39] B. Xu, J. Bu, and C. Chen. An Exploration of Improving Collaborative Recommender Systems via User-Item Subgroups. *WWW*, pages 21–30, 2012.

[40] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from Incomplite Ratings Using Non-negative Matrix Factorization. *Proc. SIAM SDM*, 2006.