Localized Matrix Factorization for Recommendation based on Matrix Block Diagonal Forms

Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma, Shi Feng
Tsinghua University, Beijing, China
zhangyf07@gmail.com

Background

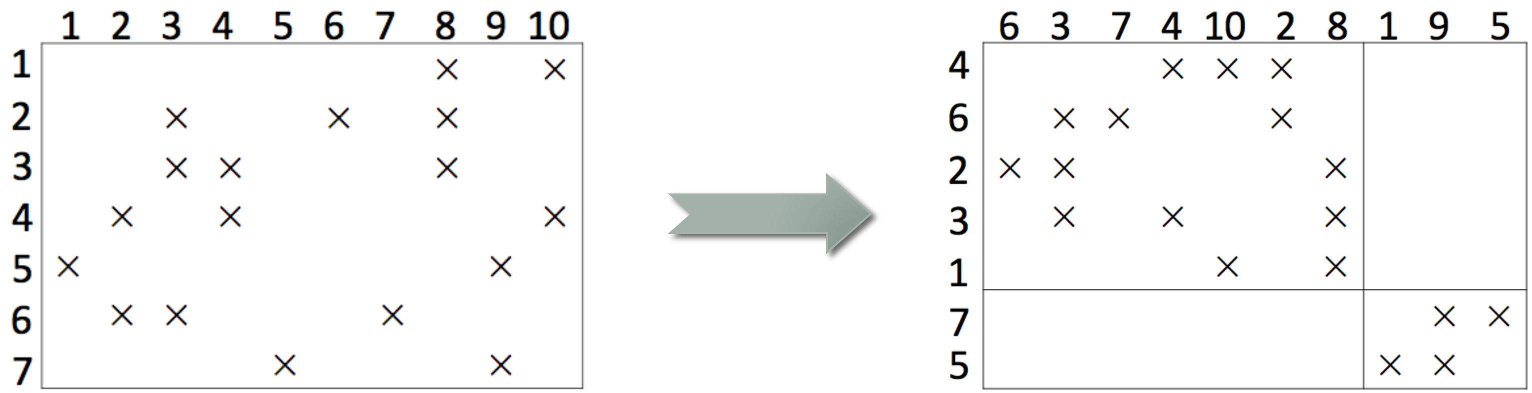
- Collaborative Filtering
  - Has achieved important success
  - Latent Factor Models based on Matrix Factorization techniques

- The Challenges
  - Data Sparsity
    - Very sparse user-item rating matrices
    - Usually, density < 1%

  - Scalability
    - Millions or even billions of users, items and ratings
    - Frequent model retraining

The Intuitional Idea

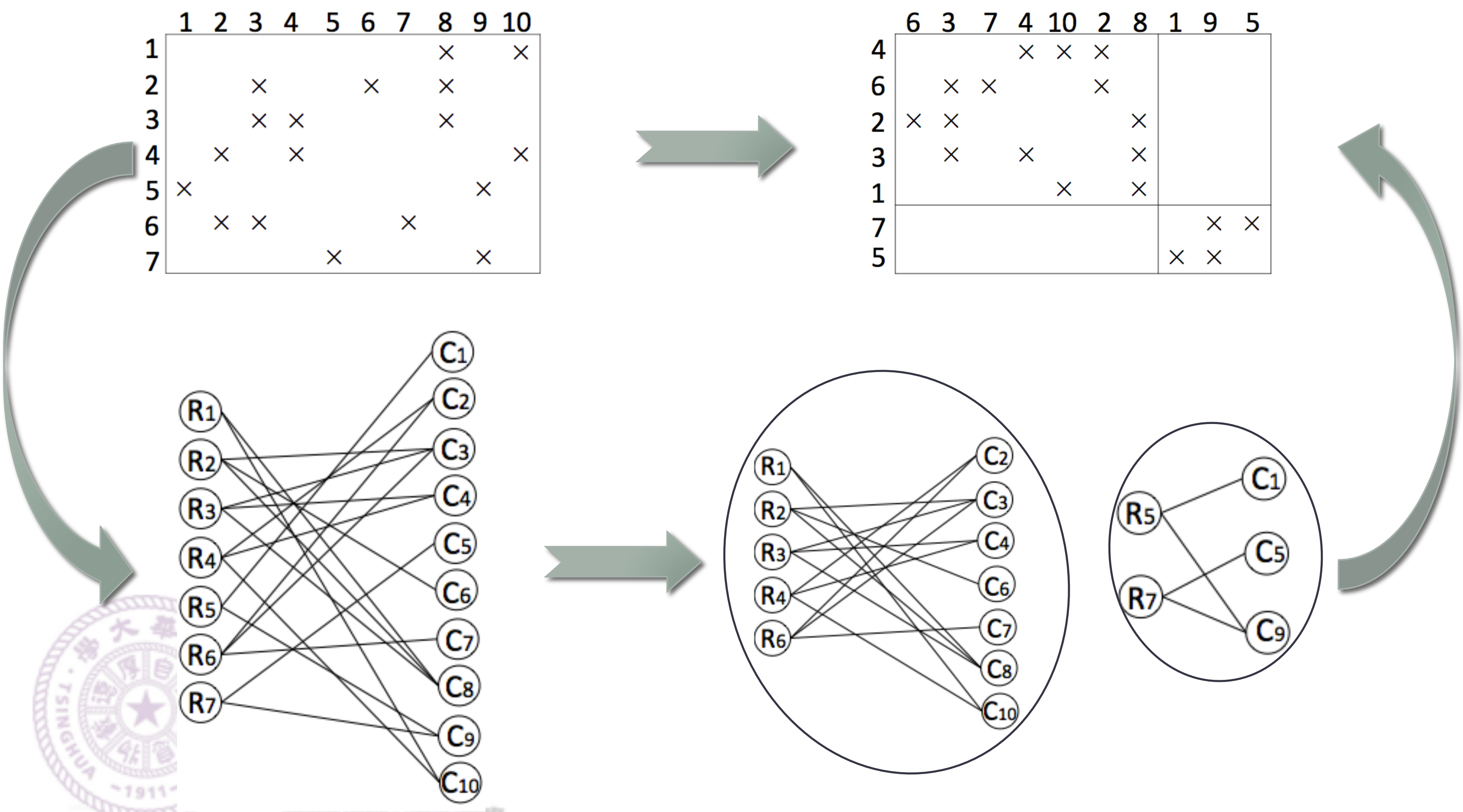
- Permute a matrix into Block Diagonal Form (BDF) structure.



- Diagonal blocks are independent
  - Can be trained independently
  - Benefits computational time
- Diagonal blocks become denser
  - May Benefit prediction accuracy

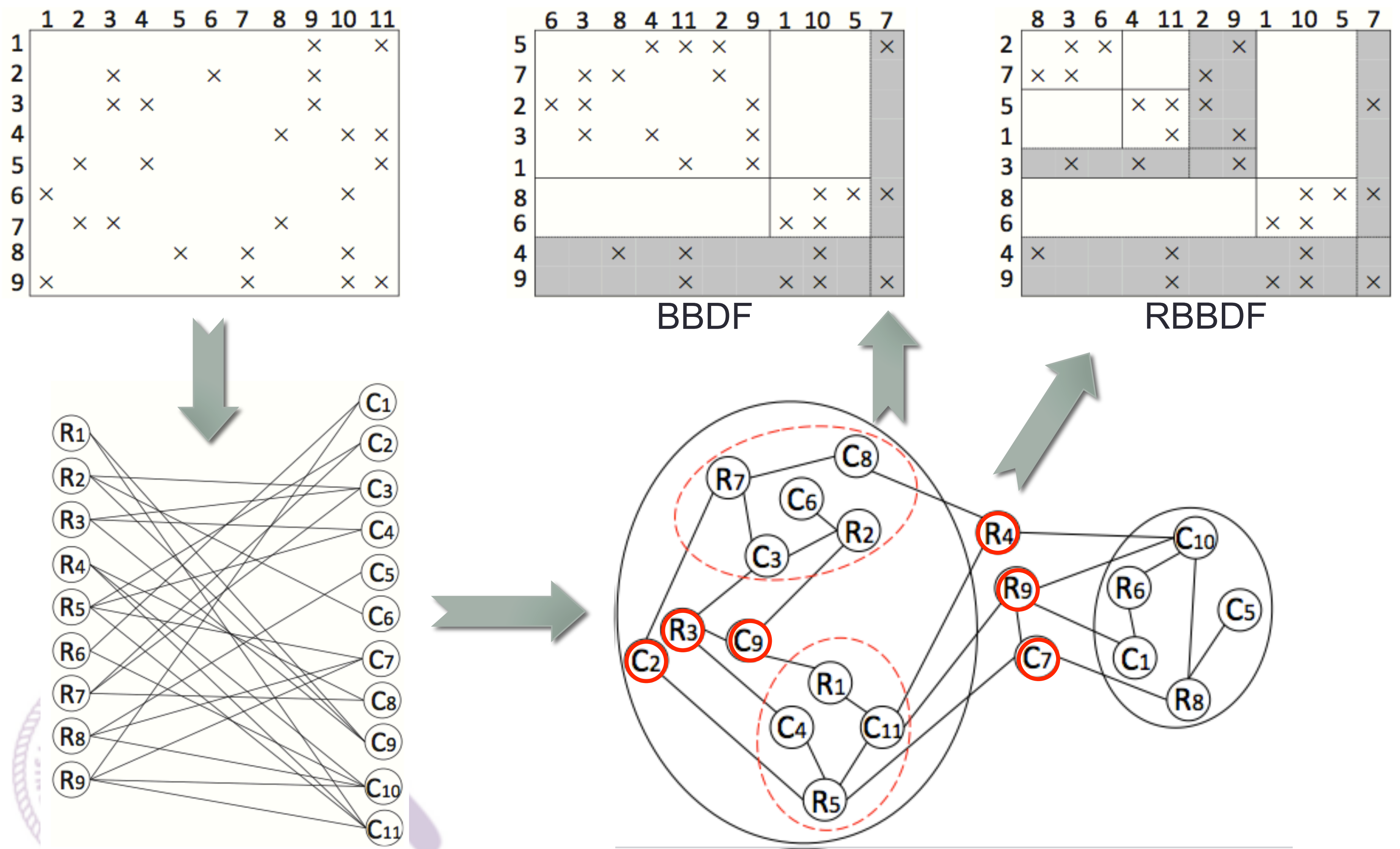
The Intuitional Idea (cont.)

- Problem of the BDF structure
  - Not all matrices can be permuted into BDF structures



A generalization of BDF structure

- Bordered Block Diagonal Form (BBDF) structure

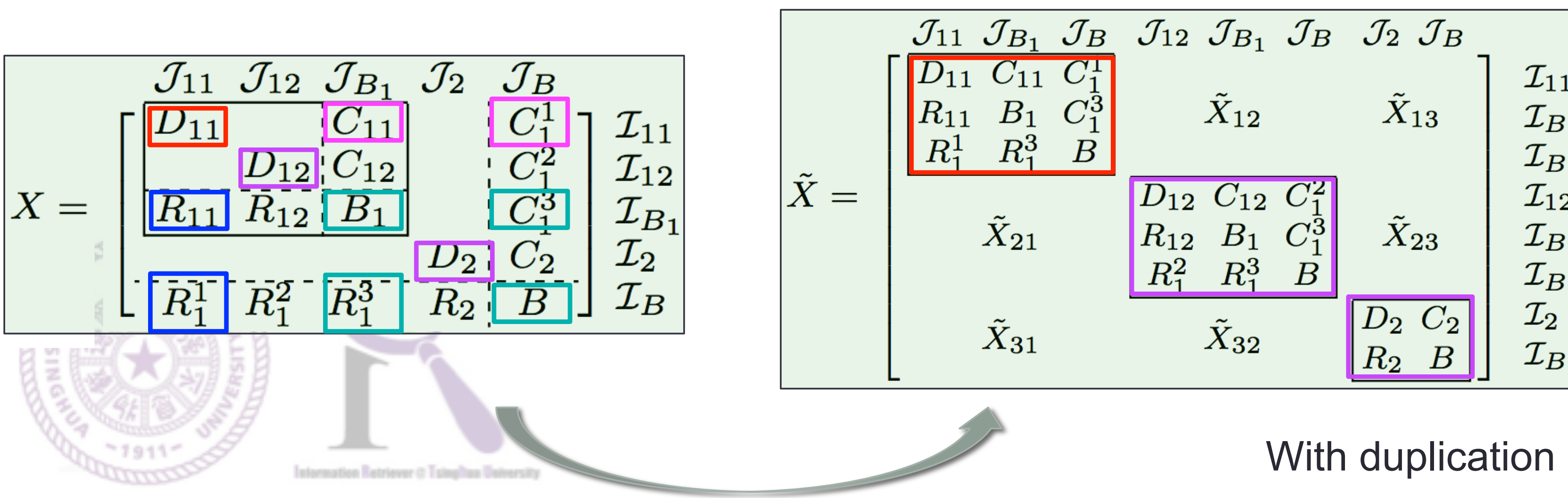


Graph Partitioning by Vertex Separator (GPVS)

Properties of (R)BBDF structure

- BBDF and RBBDF structures have many important properties
  - They make many MF algorithms decomposable
  - Naturally suitable for parallelization
  - The theoretical basis of the framework to be introduced
  - See detailed propositions and theorems in the paper

- Construct a BDF matrix from an RBBDF matrix



The LMF framework

- A sparse matrix is permuted into RBBDF structure.
- A BDF matrix is constructed from this structure.

$$\tilde{X} = \mathrm{diag}(\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_k)$$

- Conduct rating prediction within 3 steps:
  - Factorize each diagonal block independently

$$\tilde{X}_i \approx f(U_i V_i^T)$$

  - Approximate the off-diagonal zero blocks:

$$\tilde{X}_{ij} \approx f(U_i V_j^T)$$

  - Average duplicated sub-blocks:

$$X^*_{\mathcal{I}_* \sim \mathcal{J}_*} = \frac{1}{k} \sum_{t=1}^{k} \tilde{X}^{*(i_t j_t)}_{\mathcal{I}_* \sim \mathcal{J}_*}$$

BBDF permutation algorithm

- The relationship of BBDF structure and GPVS
  - Construct bipartite graph and use GPVS result to permute a matrix
  - Use the GPVS routine in Metis* for graph partitioning

- Balance the size of subgraphs? Perhaps not!
  - Communities may not be evenly divided.
  - Dense subgraphs usually represent actual communities.
  - Widely used in community detection tasks.

- Design a density based algorithm.
  - Some definitions

$$\rho(A) = \frac{\mathrm{n}(A)}{\mathrm{area}(A)} \quad \bar{\rho}(A_1 \cdots A_k) = \frac{\sum_{i=1}^{k} \mathrm{n}(A_i)}{\sum_{i=1}^{k} \mathrm{area}(A_i)}$$

*G. Karypis. Metis-A Software Package for Partitioning Unstructured Graphs, Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices (v5.0), 2011.

RBBDF permutation algorithm (cont.)

$$
\left[\begin{array}{ccc|c|c}
D_{11} & & C_{11} & & C_1^1 \\
& D_{12} & C_{12} & & C_1^2 \\
R_{11} & R_{12} & B_1 & & C_1^3 \\
& & & D_2 & C_2 \\
\hline
R_1^1 & R_1^2 & R_1^3 & R_2 & B
\end{array}\right]
$$

Algorithm 1 RBBDF$(X, \hat{\rho}, k)$

Require:
User-item rating matrix: X
Average density requirement: $\hat{\rho}$
Current number of diagonal blocks in X: k

Ensure:
Matrix X be permuted into RBBDF structure
BDF matrix $\tilde{X}$ which is constructed from X

```
ρ ← ρ̄(X̃_1 X̃_2 ··· X̃_k)
if ρ ≥ ρ̂ then
    return X̃ ▷ Density requirement has been reached
else
    [D_{s_1} D_{s_2} ··· D_{s_k}] ← Sort([D_1 D_2 ··· D_k]) ▷ Sort diagonal blocks by size in decreasing order
    for i ← 1 to k do
        [D^1_{s_i} D^2_{s_i}] ← MetisNodeBisection(D_{s_i}) ▷ Partition D_{s_i} into 2 diagonals using core routine of Metis
        if ρ̄(X̃_{s_1} ··· X̃_{s_{i-1}} X̃^1_{s_i} X̃^2_{s_i} X̃_{s_{i+1}} ··· X̃_{s_k}) > ρ then
            X' ← Permute D_{s_i} into [D^1_{s_i} D^2_{s_i}] in X
            RBBDF(X', ρ̂, k + 1) ▷ Recurse
            break ▷ No need to check the next diagonal
        end if
    end for
    return X̃ ▷ No diagonal improves average density
end if
```

The expected minimum average density of diagonal blocks.

Continue to split diagonal blocks if the average density has not reached density requirement.

Try to split a diagonal block in decreasing order of block size.

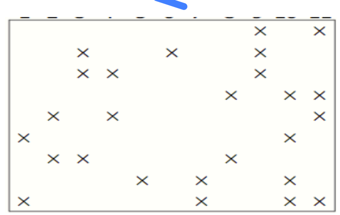
Stop this round and continue if the split increases average density.

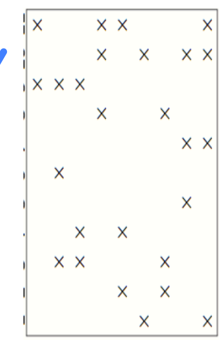
Stop and exit if no split increases average density.

Experiments

- Four real-world datasets:
  - MovieLens-100k, MovieLens-1m, DianPing* and Yahoo! Music.

	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	256,804,235
#ratings/user	106.045	165.598	43.059	256.550
#ratings/item	59.453	253.089	22.828	410.912
average density	0.0630	0.0419	0.00193	0.000411





- Experimented the LMF framework on 4 MF algorithms
  - SVD, NMF, PMF, fast MMMF
- Root Mean Square Error $\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(r_i - \hat{r}_i)^2}{N}}$

*A famous restaurant rating website in China (The Chinese version of Yelp)

Analysis of RBBDF algorithm

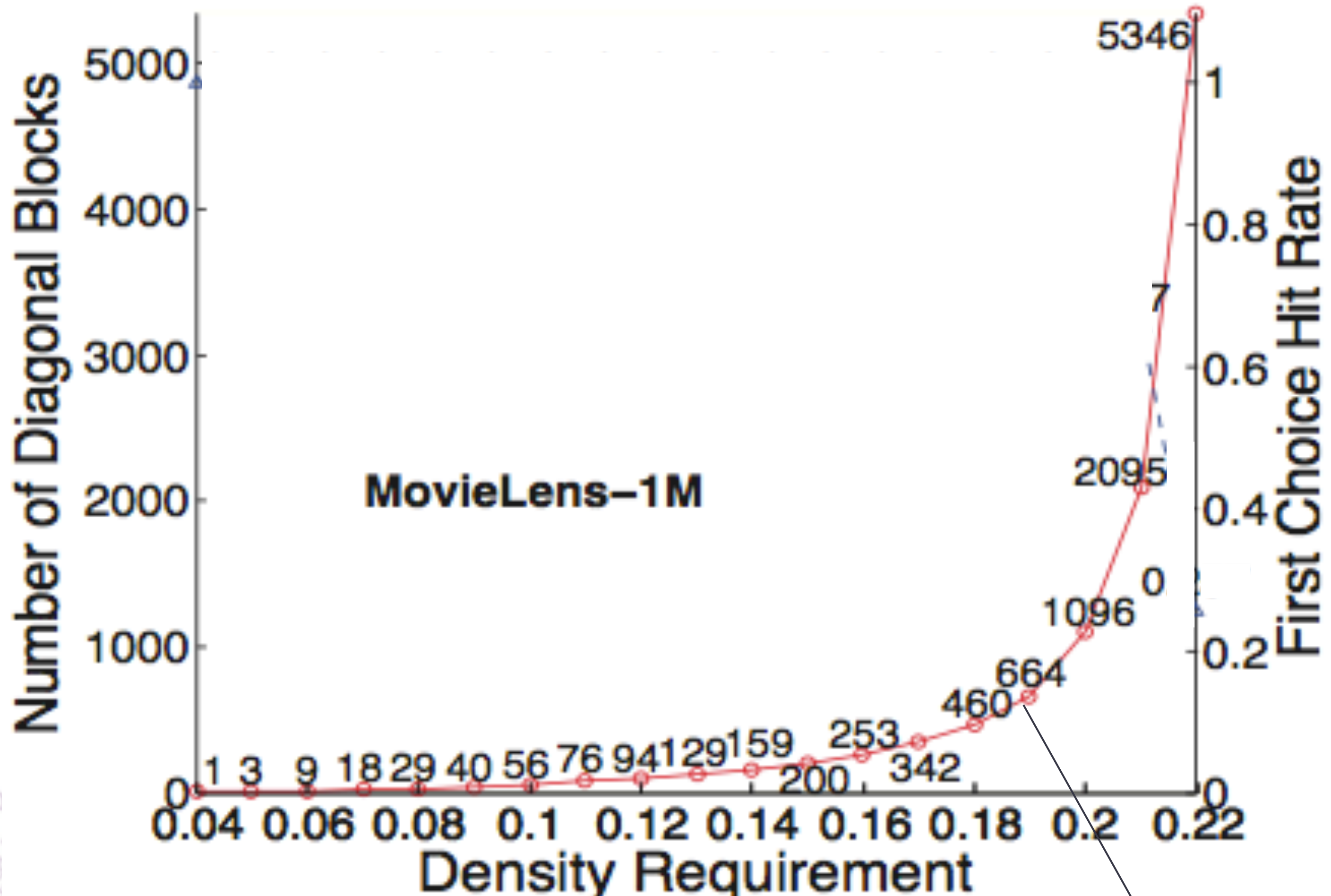
- Relationship of density requirement and # diagonal blocks
  - Low density -> A small number of big communities
  - High density -> A large number of small communities

- Example of RBBDF permutation results on DianPing

(a) BBDF $\rho = 0.005$ (b) BBDF $\rho = 0.01$

Analysis of RBBDF algorithm (cont.)

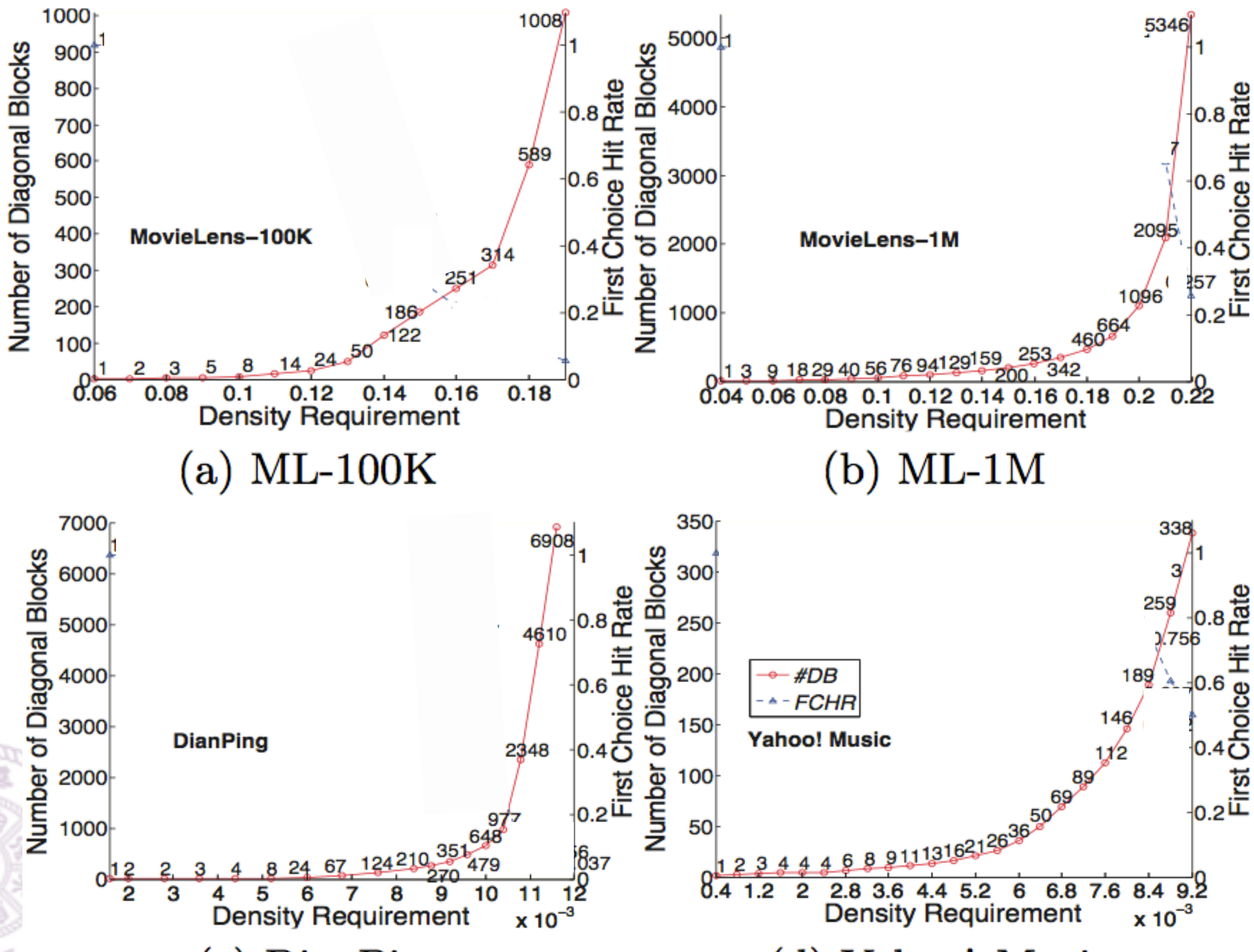
- Relationship of density requirement and # diagonal blocks



diagonal blocks grows faster and faster with the increasing of the pre-set density requirement

Analysis of RBBDF algorithm (cont.)

- Relationship of density requirement and # diagonal blocks



(a) ML-100K (b) ML-1M

(c) DianPing (d) Yahoo! Music

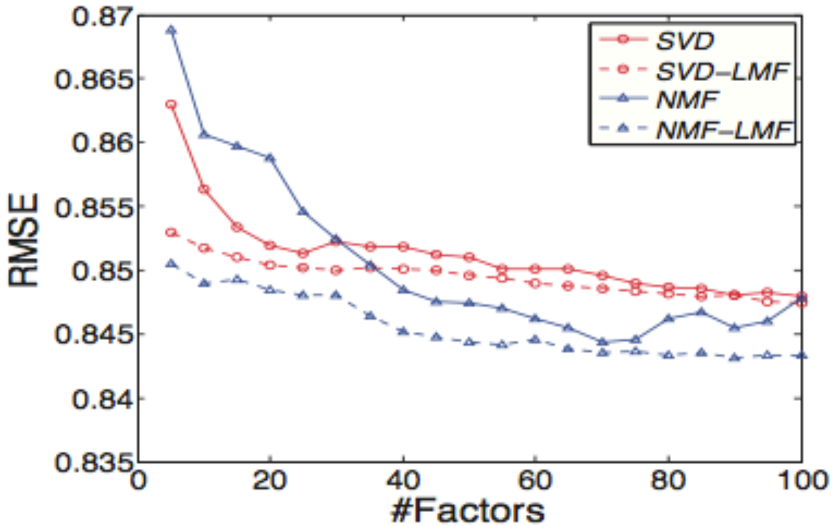
Prediction Accuracy

- RMSE v.s. Number of latent factors (on MovieLens-1m)
  - Density requirement = 0.055, # diagonal blocks = 4

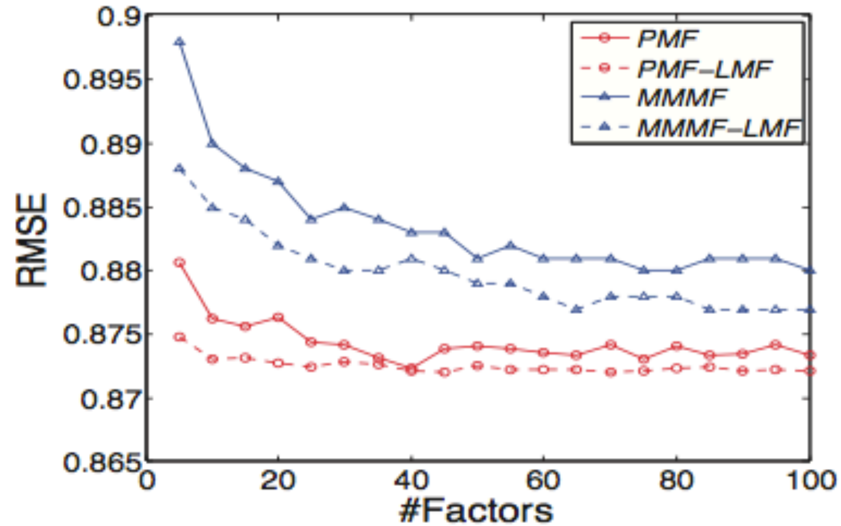
	$\tilde{X}_1$	$\tilde{X}_2$	$\tilde{X}_3$	$\tilde{X}_4$
#users	1,507	1,683	1,743	1,150
#items	2,491	3,108	3,616	3,304
#ratings	118,479	259,665	462,586	192,267
density	0.0316	0.0496	0.0734	0.0506

- Experimentation
  - 1. Approximate the whole matrix with r factors, record RMSE
  - 2. Approximate each diagonal block with r factors using the LMF framework and record RMSE

Prediction Accuracy (cont.)



(a) SVD & NMF



(b) PMF & MMMF

- Solid line: RMSE of making predictions directly
- Dotted line: RMSE of making predictions in LMF framework

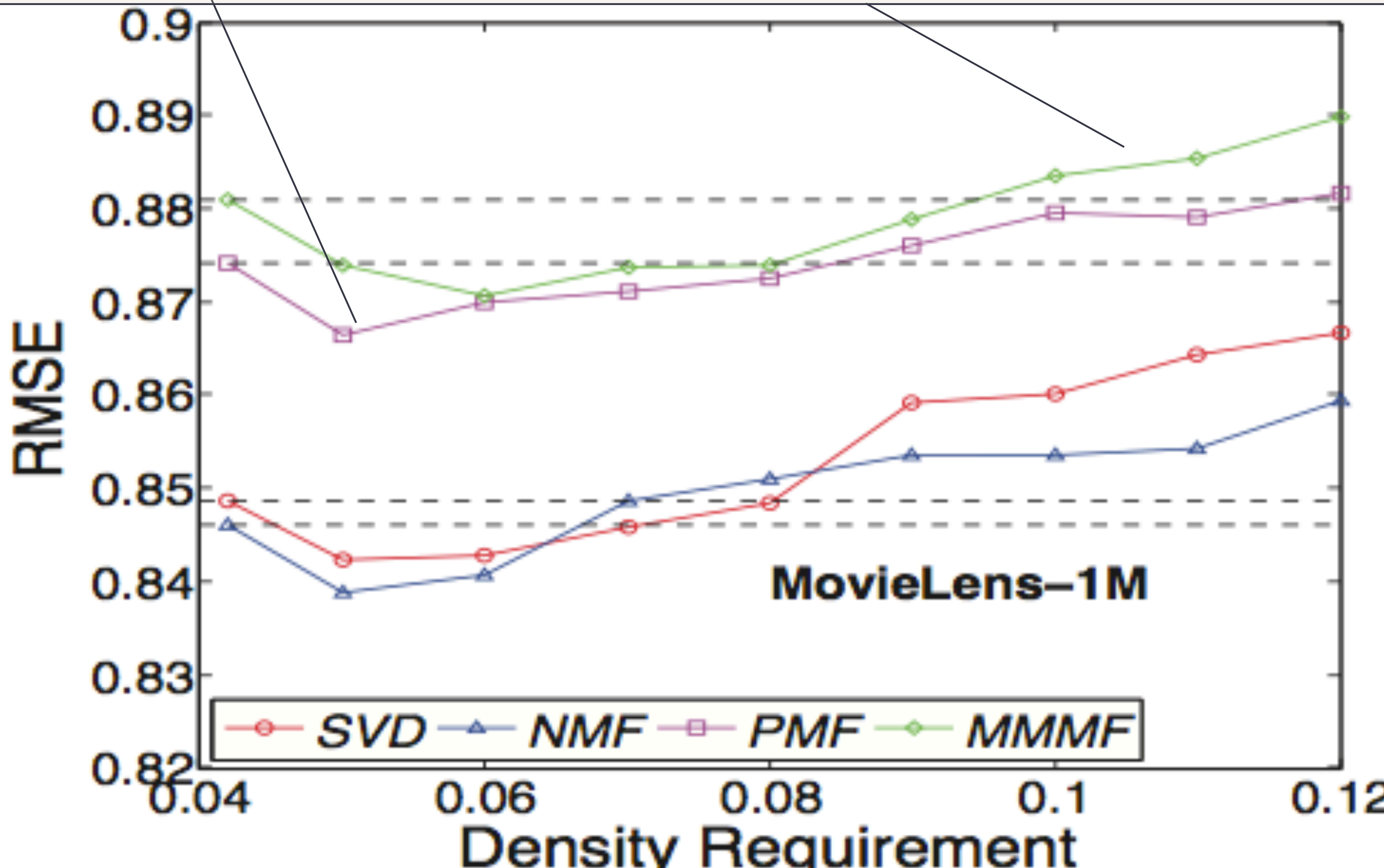
- Some observations
  - The LMF framework gains better prediction accuracy
  - Advantage is more obvious given small number of latent factors
    - Small number of latent factors is not sufficient to approximate the whole matrix directly, but sufficient to approximate a relatively small matrix

Prediction Accuracy (cont.)

- RMSE v.s. Density requirements (given r = 60)

Gains better prediction accuracy if density requirement is not too high

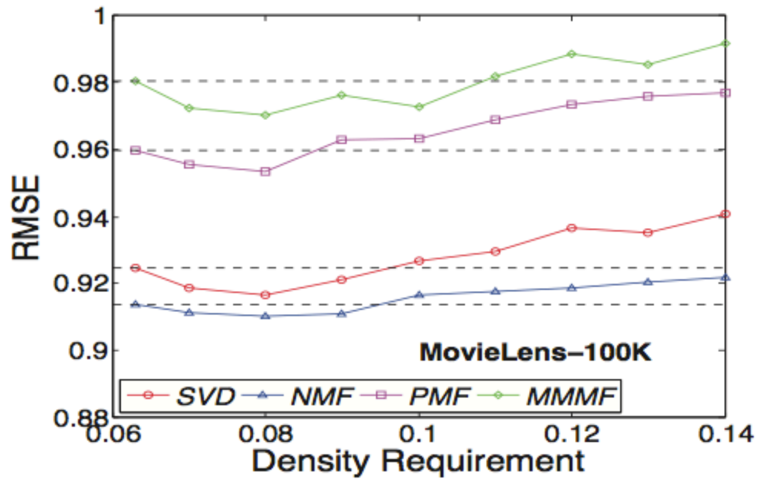
Performance goes worse than the base performance given high density requirements



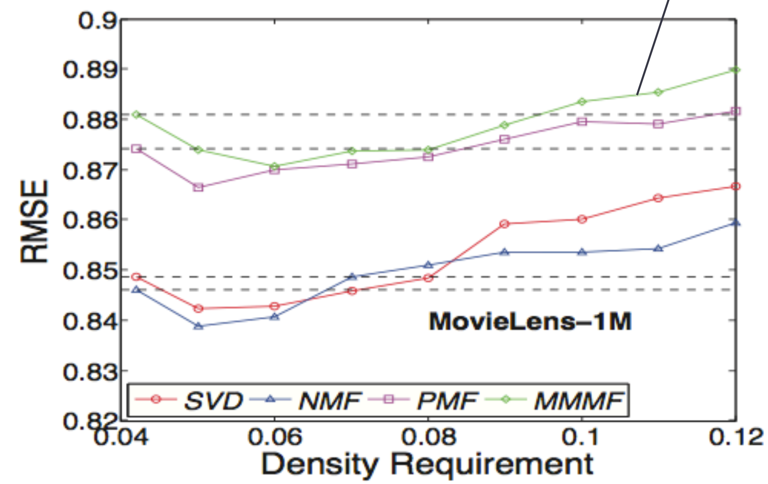
Prediction Accuracy (cont.)

- Density requirements (given r = 60)

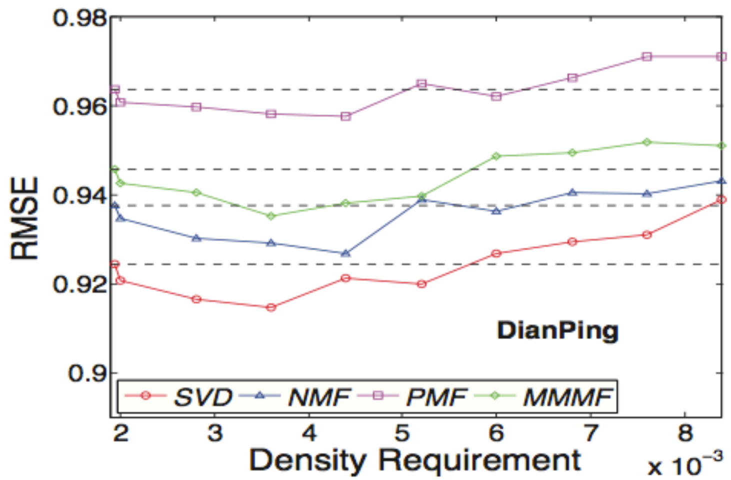
The matrix is split into too many small scattered sub-matrices



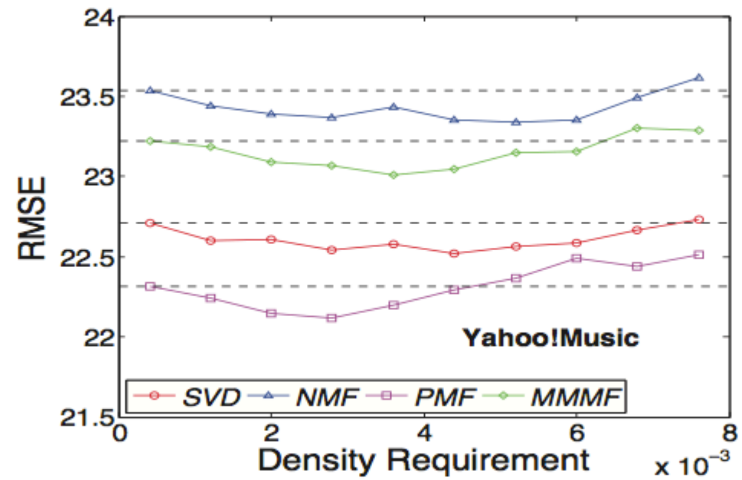
(a) MovieLens-100K



(b) MovieLens-1M



(c) DianPing



(d) Yahoo! Music

Speedup by parallelization

- As for the decomposable properties in LMF framework
  - Easy to train each diagonal block with simple parallelization techniques.

- Three steps
  - Permute the original matrix into 8 diagonal blocks, t_1
  - Factorize each diagonal block in parallel, t_2
  - Approximate the original matrix using LMF, t_3

- Metric
  - Use t as the time used for approximating the whole matrix directly
  - Use $t' = t_1 + t_2 + t_3$ as the time using the LMF framework

$$Speedup = \frac{t}{t'}$$

Speed up by parallelization (cont.)

- Results
  - Speedup is achieved on all four datasets and algorithms using simple penalization techniques
  - The sparser a matrix is, the higher speedup we tend to gain.

Method	MovieLens-100K			MovieLens-1M		
	Base	LMF	Speedup	Base	LMF	Speedup
SVD	23.9s	7.7s	3.10	184.9s	43.4s	4.26
NMF	8.7s	3.9s	2.23	86.6s	22.1s	3.92
PMF	43.8s	11.6s	3.78	265.1s	60.1s	4.41
MMMF	19.6min	4.71min	4.16	1.73h	21.5min	4.83

Method	DianPing			Yahoo!Music		
	Base	LMF	Speedup	Base	LMF	Speedup
SVD	143.7s	35.7	4.03	6.22h	1.21h	5.14
NMF	64.4s	16.6s	3.88	4.87h	1.05h	4.64
PMF	190.5s	44.1s	4.32	7.91h	1.48h	5.34
MMMF	48.5min	10.2min	4.75	38.8h	6.22h	6.24

Conclusions

- In this work
  - Investigated RBBDF structure of rating matrices in terms of matrix factorization problems
  - Designed density-based algorithm to transform a matrix into RBBDF structure
  - Proposed the LMF framework for recommendation tasks
  - Experimented on four real-world datasets

- Future directions
  - May be hard to find an appropriate density requirement
  - Investigate other kinds of RBBDF permutation algorithms

Thanks!

Experiments (cont.)

- Computational time of RBBDF algorithm
  - Experiment on an 8-core 3.1GHz 64G RAM Linux server.

k	5	10	15	20	50	100	150	200
ML-100K / ms	160	180	196	208	224	340	422	493
ML-1M / s	4.45	5.61	6.25	6.76	8.31	9.51	10.25	10.74
DianPing / s	6.01	9.69	11.61	12.84	14.64	15.06	16.18	16.95
Yahoo! / min	8.03	9.54	10.95	12.08	17.67	21.83	23.35	24.73

  - It takes less time to partition a submatrix as they become smaller.
  - The time used by the RBBDF algorithm is much less than that used for training an MF model on the whole rating matrix.

Why use block size as a heuristic

$$\Delta\rho = \rho' - \rho = \frac{n + \Delta n}{s - \Delta s_1 + \Delta s_2} - \frac{n}{s} = \frac{s\Delta n + n\Delta s}{s(s - \Delta s)} \quad (17)$$

where ρ and ρ' are the average densities of diagonal blocks in $\tilde{X}$ before and after partitioning D_i, and $\Delta s \triangleq \Delta s_1 - \Delta s_2$.

Because $s - \Delta s > 0$, we have the following:

$$\Delta\rho > 0 \leftrightarrow s\Delta n + n\Delta s = s\Delta n + n(\Delta s_1 - \Delta s_2) > 0 \quad (18)$$

If $\Delta s > 0$, then (18) holds naturally. Otherwise, the following is required:

$$\frac{n}{s} < \frac{\Delta n}{\Delta s_2 - \Delta s_1} \quad (19)$$

Although not guaranteed, (19) can usually be satisfied as the following property usually holds:

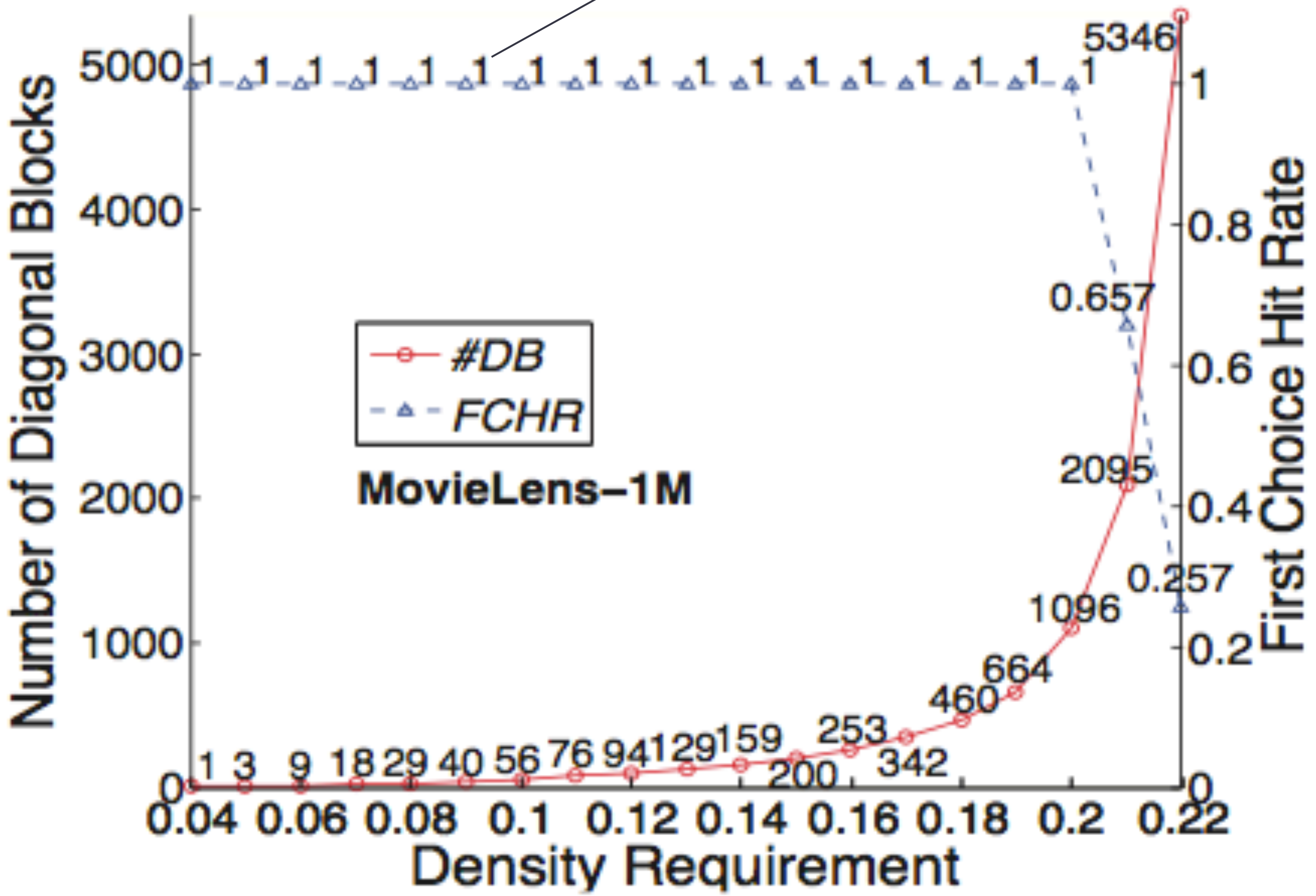
$$\frac{n}{s} < \frac{\Delta n}{\Delta s_2} < \frac{\Delta n}{\Delta s_2 - \Delta s_1} \quad (20)$$

Analysis of RBBDF algorithm (cont.)

- Verification of the heuristic

$$\text{FCHR} = \frac{\#\ recursions\ where\ D_{s_1}\ is\ chosen}{\#\ recursions\ in\ total}$$

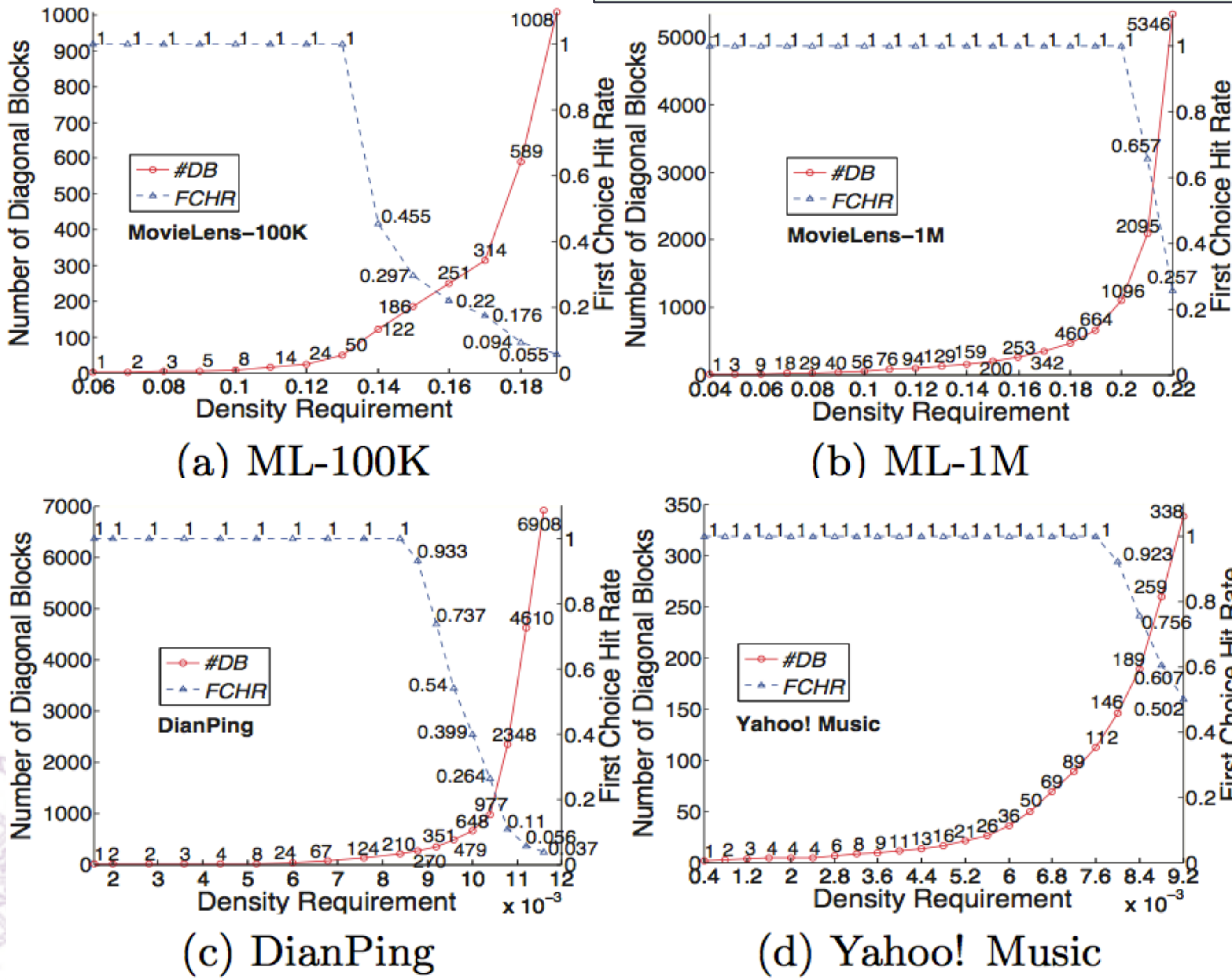
1. FCHR remains 1 when density requirement is not too high -> No computational wastes
2. A relatively low density requirement is usually enough in practical applications



Analysis of RBBDF algorithm (cont.)

- Verification of the heuristic

$$\text{FCHR} = \frac{\#\ \textit{recursions where } D_{s_1} \textit{ is chosen}}{\#\ \textit{recursions in total}}$$



(a) ML-100K (b) ML-1M (c) DianPing (d) Yahoo! Music

Decomposable regularizer & why fast version of MMMF

- L-p norm regularizer is decomposable:

$$\mathcal{R}(U,V) = \lambda_U \|U\|_p^p + \lambda_V \|V\|_p^p$$
$$= \sum_{i=1}^{k} \left(\lambda_U \|U_i\|_p^p + \lambda_V \|V_i\|_p^p\right) = \sum_{i=1}^{k} \mathcal{R}(U_i, V_i)$$

The Frobenius norm is ℓ_p-*norm* where $p = 2$. The basic MMMF algorithm takes the trace-norm $\|X\|_\Sigma$ (the sum of singular values of X) [34], which is unfortunately not a decomposable regularizer. However, a fast MMMF algorithm based on the equivalence $\|X\|_\Sigma = \min_{X=UV^T} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2)$ is proposed in [23], which also takes ℓ_p-*norm* regularizers.

Proof of the theorem

PROOF. i. Consider the optimization problem defined in (1) with decomposable properties of prediction link f, loss function $\mathcal{D}_W$, hard constraint $\mathcal{C}$, and regularizer $\mathcal{R}$; we have:

$$\begin{aligned}
&(U,V) = \mathcal{P}(X,r) \\
=\ & \underset{(U,V)\in\mathcal{C}}{\operatorname{argmin}} \left[\mathcal{D}_W(X, f(UV^T)) + \mathcal{R}(U,V)\right] \\
=\ & \underset{(U,V)\in\mathcal{C}}{\operatorname{argmin}} \sum_{i=1}^{k} \left[\mathcal{D}_{W_i}(X_i, f(UV^T)_i) + \mathcal{R}(U_i, V_i)\right] \\
=\ & \underset{(U,V)\in\mathcal{C}}{\operatorname{argmin}} \sum_{i=1}^{k} \left[\mathcal{D}_{W_i}(X_i, f(U_i V_i^T)) + \mathcal{R}(U_i, V_i)\right] \\
=\ & \bigwedge_{i=1}^{k} \left\{ \underset{(U_i,V_i)\in\mathcal{C}}{\operatorname{argmin}} \left[\mathcal{D}_{W_i}(X_i, f(U_i V_i^T)) + \mathcal{R}(U_i, V_i)\right] \right\} \\
=\ & \bigwedge_{i=1}^{k} \left\{\mathcal{P}(X_i, r)\right\} = \bigwedge_{i=1}^{k} \left\{(U_i, V_i)\right\}
\end{aligned}$$

thus, $U = [U_1^T U_2^T \cdots U_k^T]^T$ and $V = [V_1^T V_2^T \cdots V_k^T]^T$.

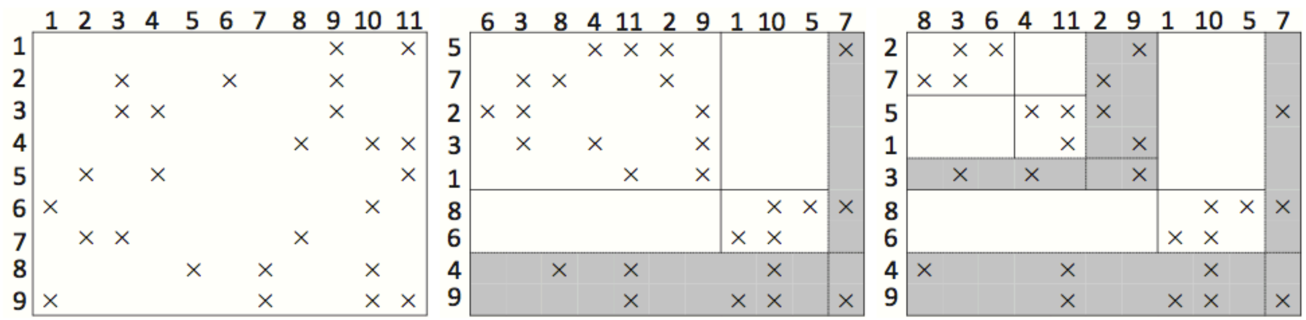
ii. This can be derived directly from the decomposable property of prediction link f in (10):

$$X_{ij} \approx f(UV^T)_{ij} = f(U_i V_j^T)$$

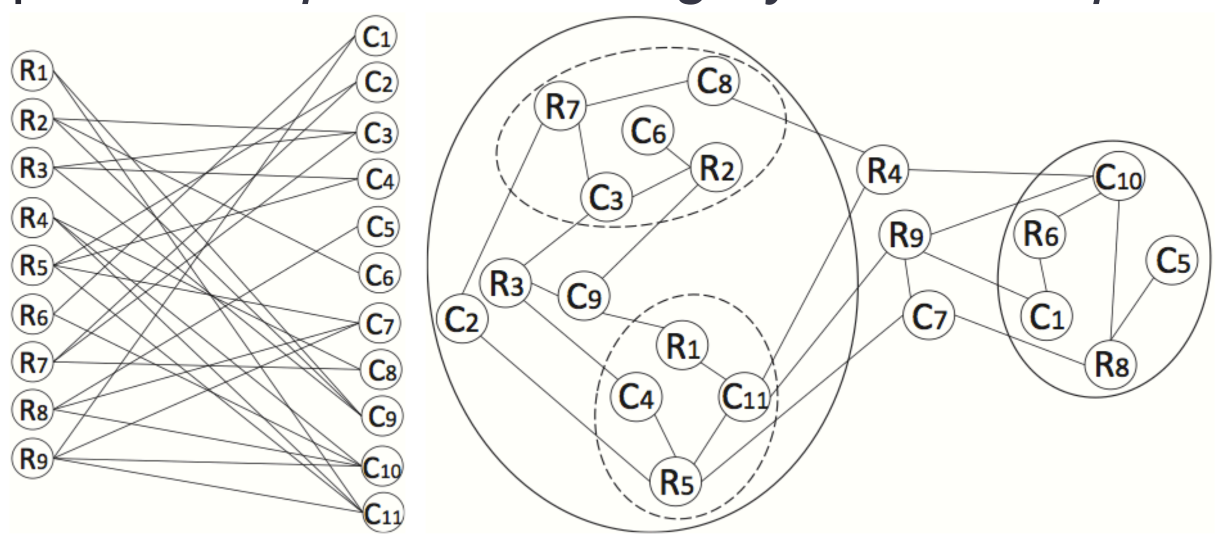
and it holds for any $1 \leq i, j \leq k$, including zero submatrices where $i \neq j$. □

Our Approach – The LMF framework

- Localized Matrix Factorization
  - Based on (Recursive) Bordered Block Diagonal Form



  - General and compatible with many widely-adopted MF algorithms
  - Naturally suitable for parallelization
- Relationship with *Graph Partitioning by Vertex Separator* (GPVS)



Future work

- Rating matrix changes dynamically in practical systems
  - The prediction accuracy decreases with time
  - To train the MF model periodically is time consuming
  - Only to retrain some of the diagonal blocks in LMF

Related Work

- Matrix Clustering techniques
  - Clustered low rank approximation (Savas, 2011)
  - Collaborative filtering via user-item subgroups (Xu, 2012)
  - Scalable CF with cluster-based smoothing (Xue, 2005)

- Incremental or distributed MF algorithms
  - Incremental singular value decomposition (Sarwar, 2002)
  - Distributed non-negative matrix factorization (Liu, 2010)
  - Distributed stochastic gradient descent (Gemulla, 2011)