A Multi-Scale Framework for Graph Based Machine Learning Problems

Donghyuk Shin
Dept of Computer Science
UT-Austin

Ph.D Proposal
May 26, 2015

Supervising Professor: Inderjit S. Dhillon

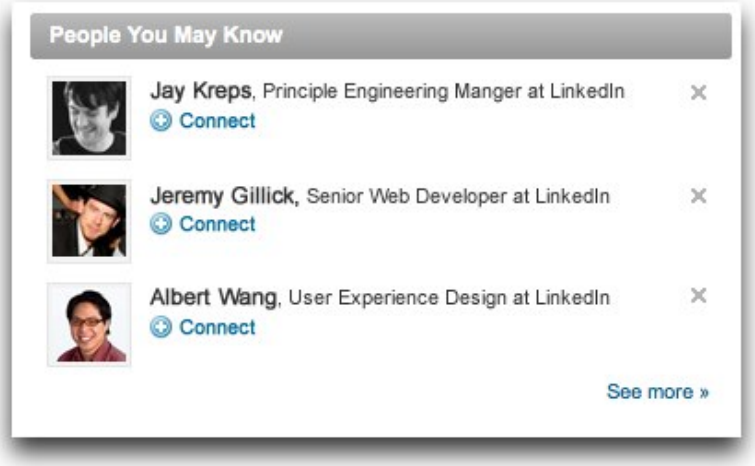
Outline

1. Introduction: Multi-Scale Framework
2. Part 1: Efficient and Scalable Computation
   - Multi-Scale Spectral Decomposition
3. Part 2: Models at Multiple Scales
   1. Link Prediction in Social Networks: Multi-Scale Link Prediction
   2. Recommender Systems beyond the User-Item Matrix: Collaborative Filtering with Interactional Context
4. Proposed Work

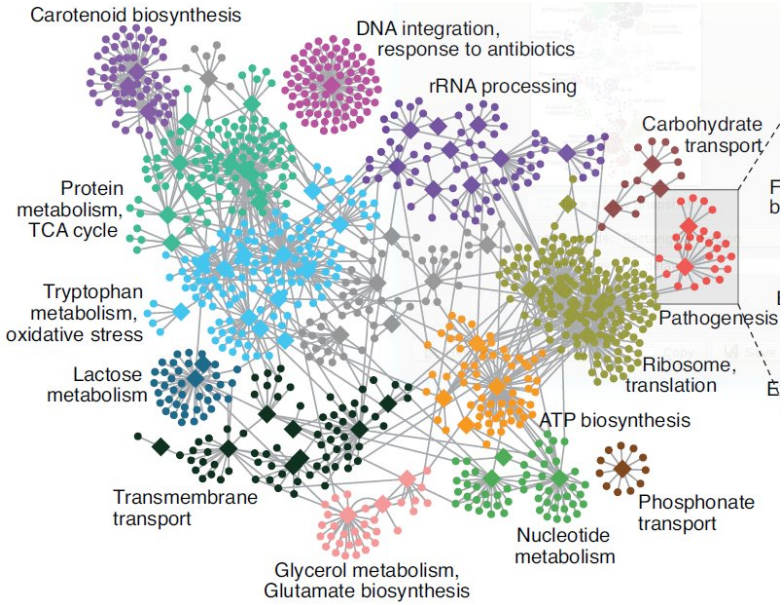
Machine Learning Problems involving Graphs

Predicting New Links

LinkedIn: people you may know

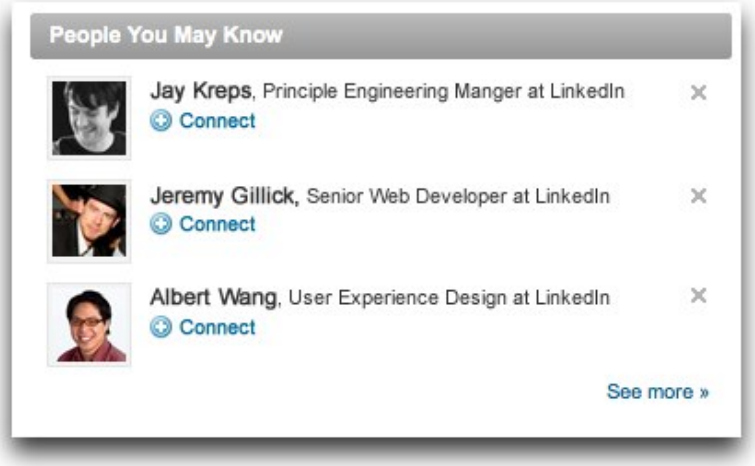


Gene-gene network [Marbach, et.al. 2012]

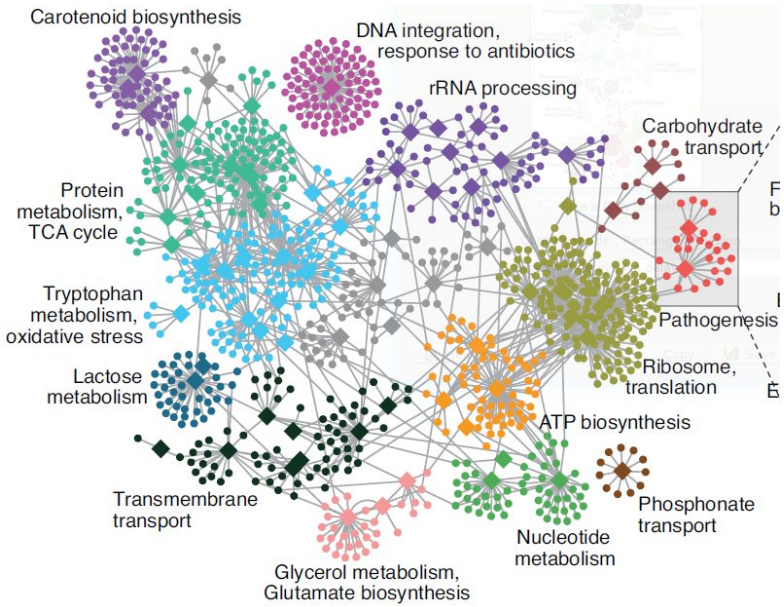


Predicting New Links

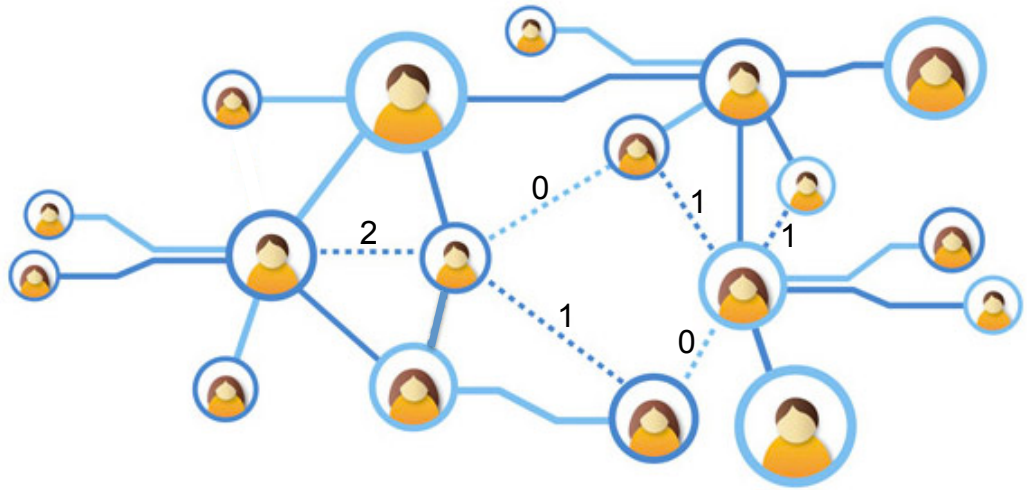
LinkedIn: people you may know



Gene-gene network [Marbach, et.al. 2012]

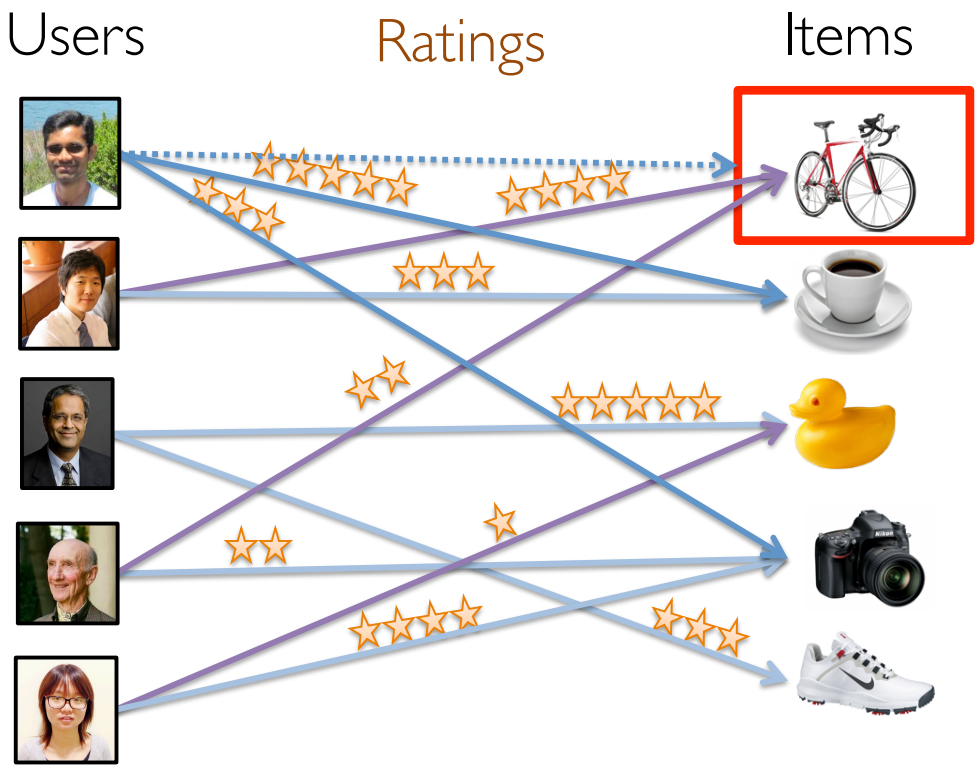


Proximity measures (common neighbors)



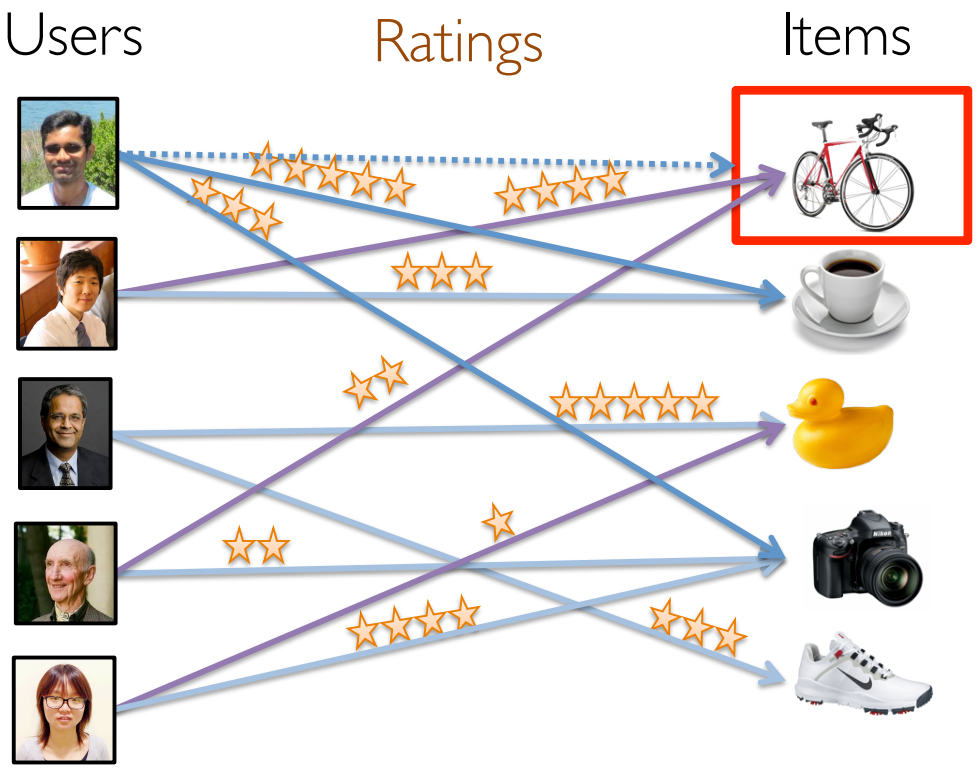
Recommending Products

Netflix, Amazon, etc ...

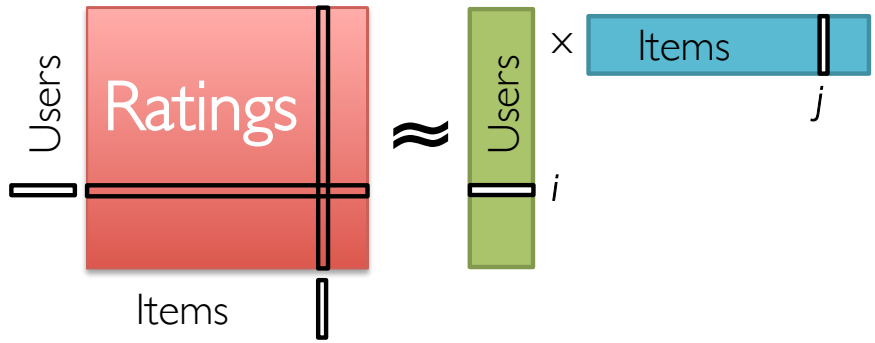


Recommending Products

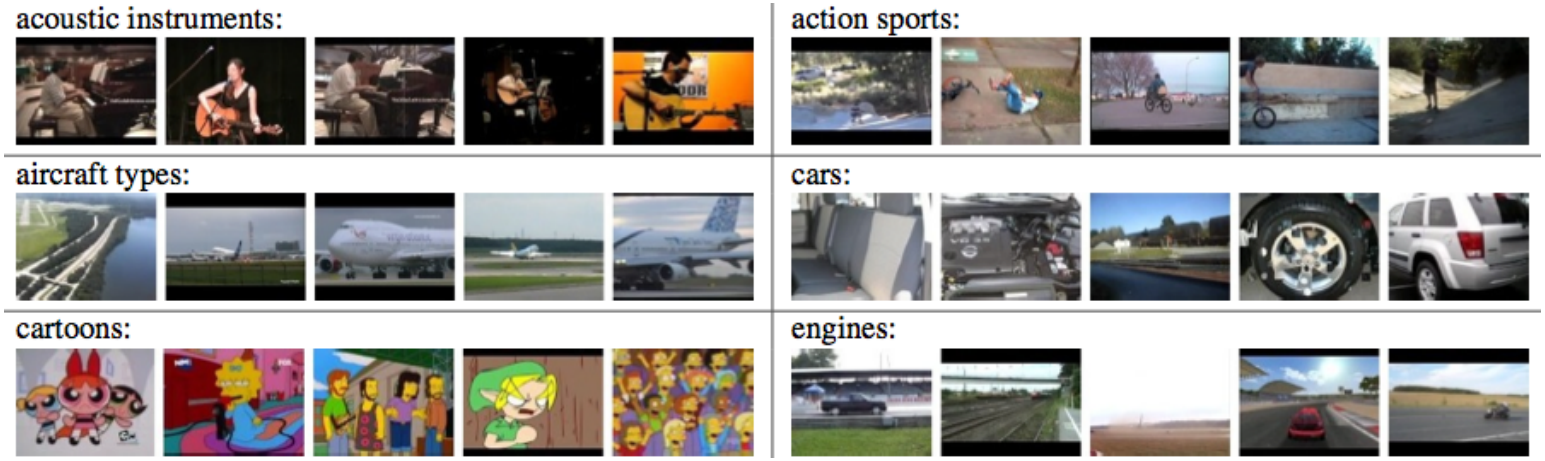
Netflix, Amazon, etc ...



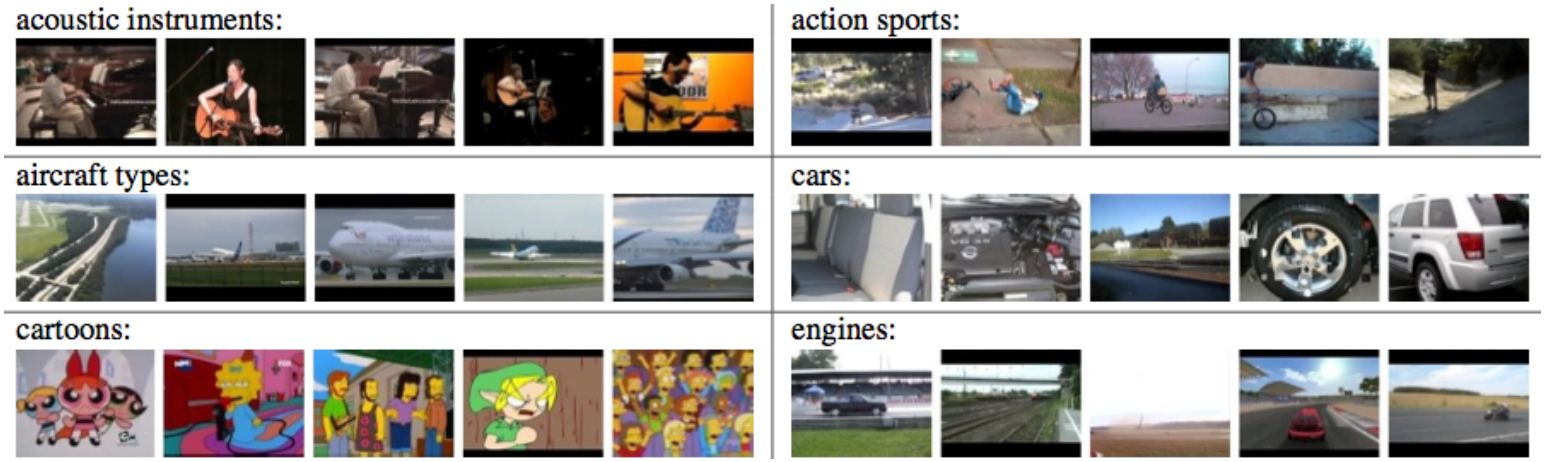
Low-rank matrix factorization:



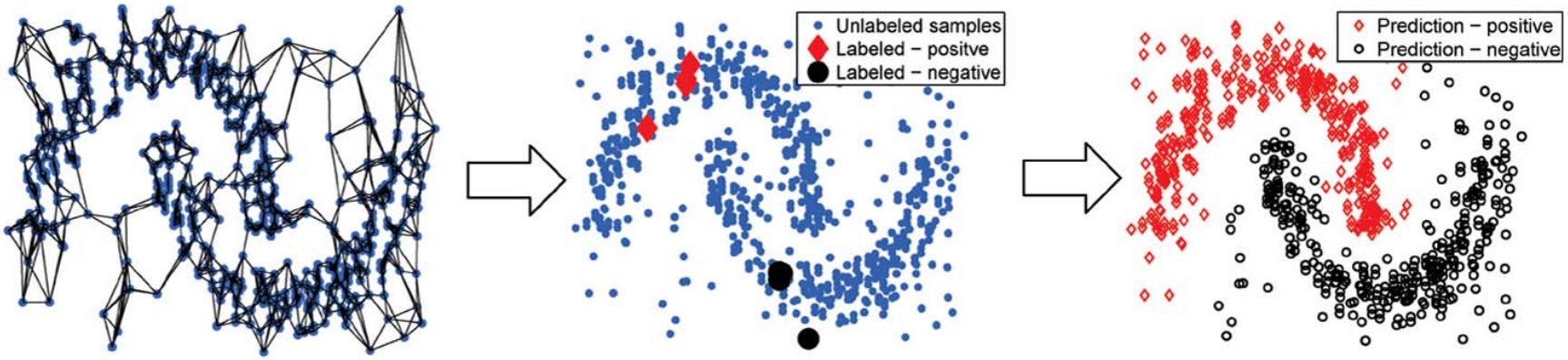
Classifying Images



Classifying Images



Graph-based semi-supervised classification:



[Liu, et.al. 2012]

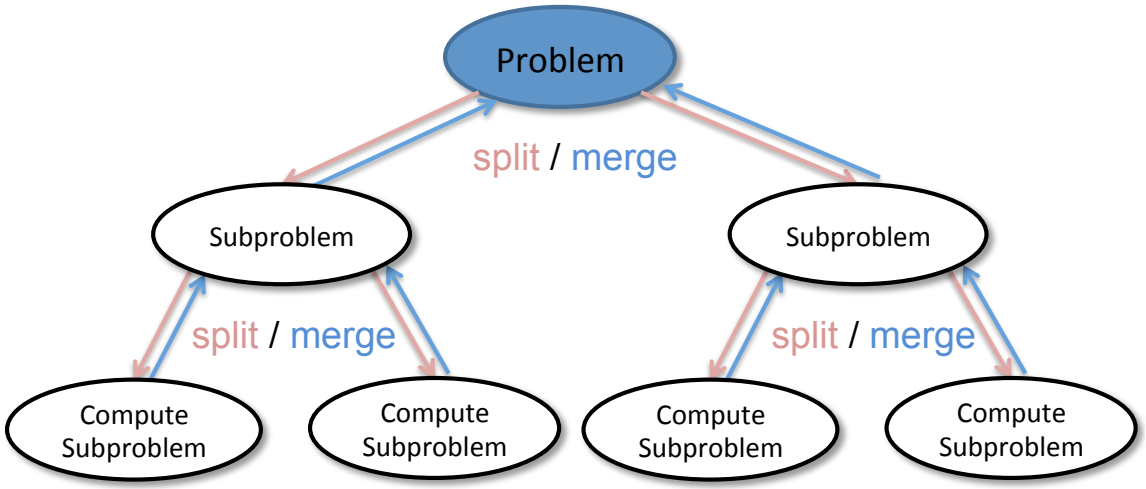
Many Graph-based ML Problems

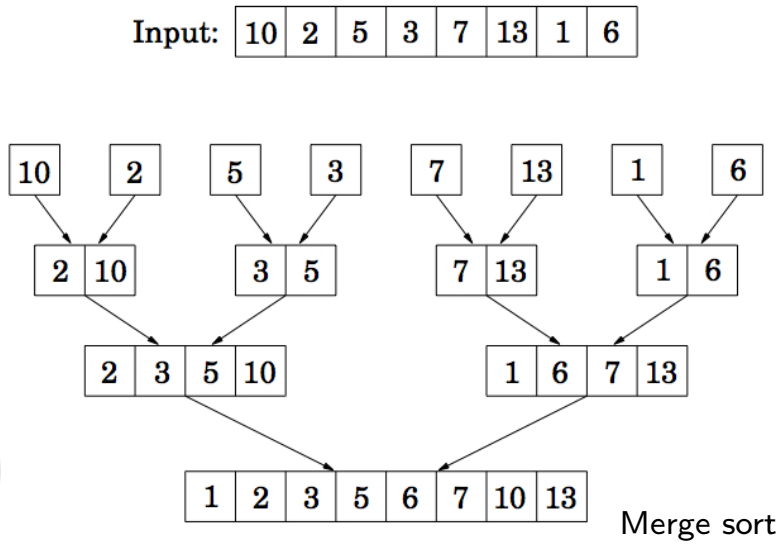
- Link prediction: proximity measures
- Recommender systems: collaborative filtering
- Semi-supervised learning: graph-based SSL
- Structured prediction: loopy belief propagation, gibbs sampling
- Community detection: triangle counting, k-core decomposition
- Graph analytics: PageRank, densest k-subgraph, graph matching
- ...

The scale of problems are massive: millions of nodes and billions of edges

Divide & Conquer Method

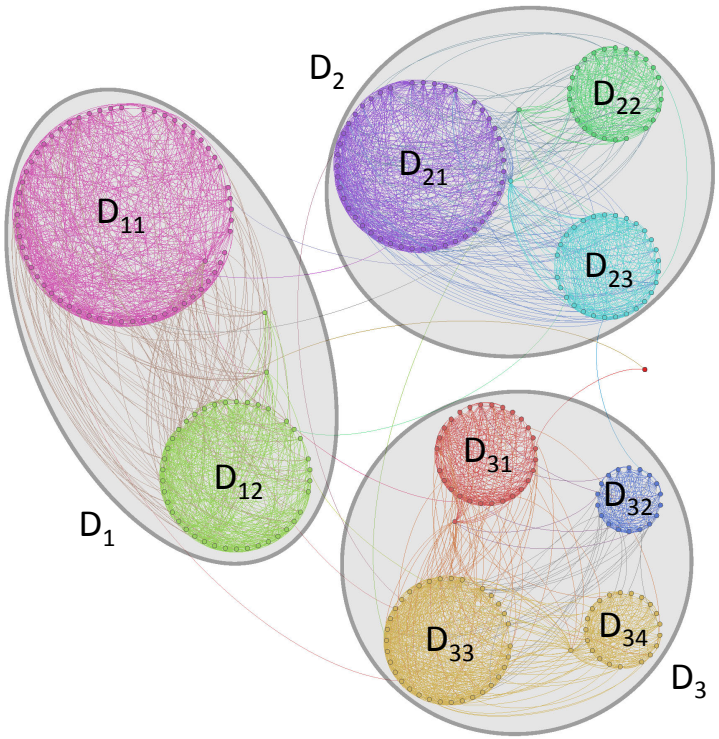
1. Divide original (large) problem into smaller instances
2. Solve the smaller instances
3. Obtain solution to original problem by combining these solutions

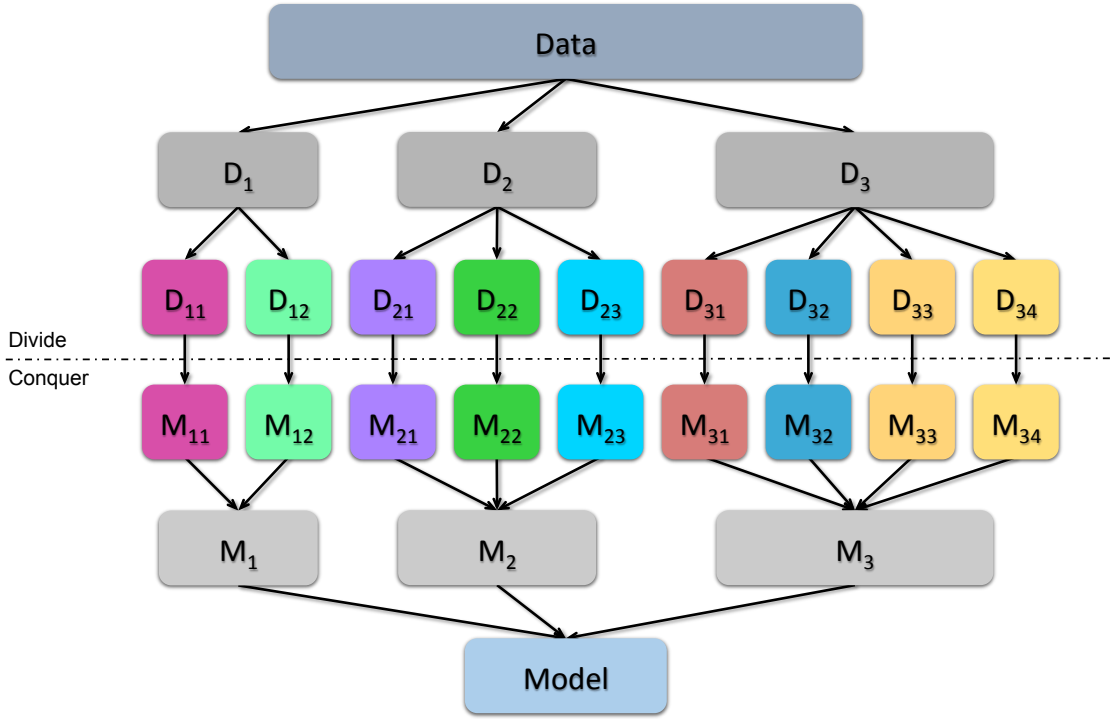




Merge sort

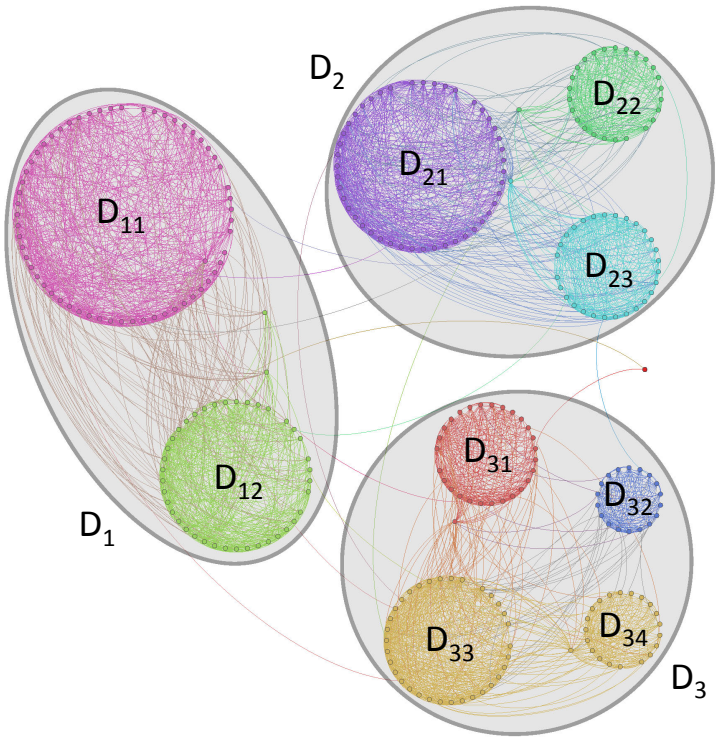
Multi-Scale Framework

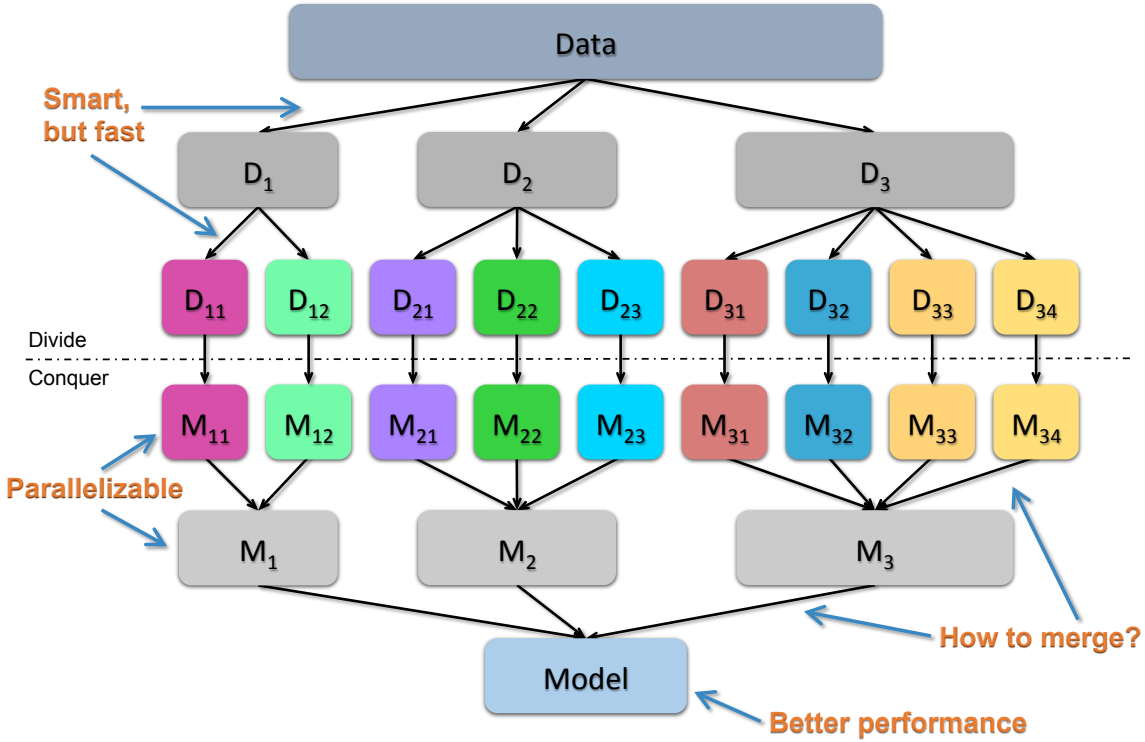




Efficiently compute solution of original problem

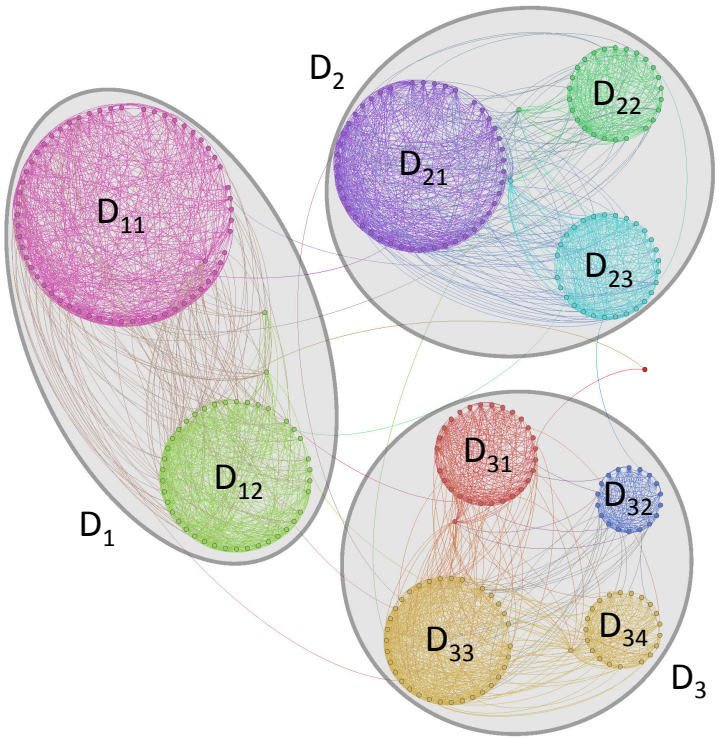
Multi-Scale Framework

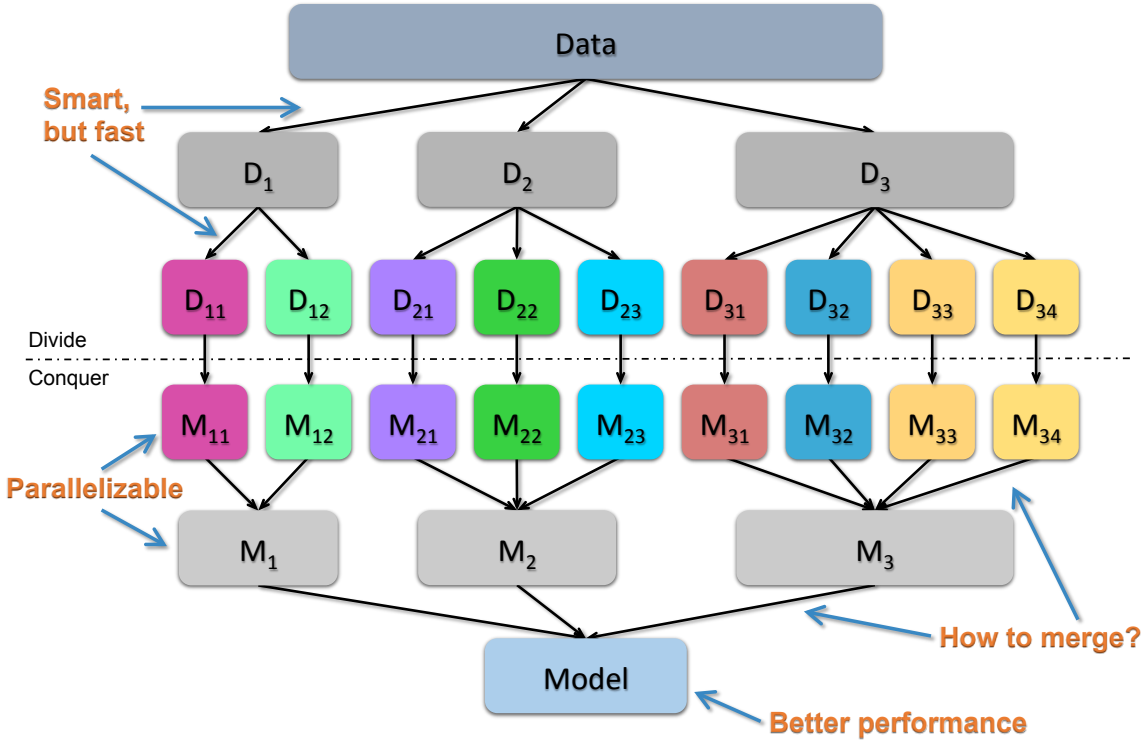




Efficiently compute solution of original problem

Multi-Scale Framework

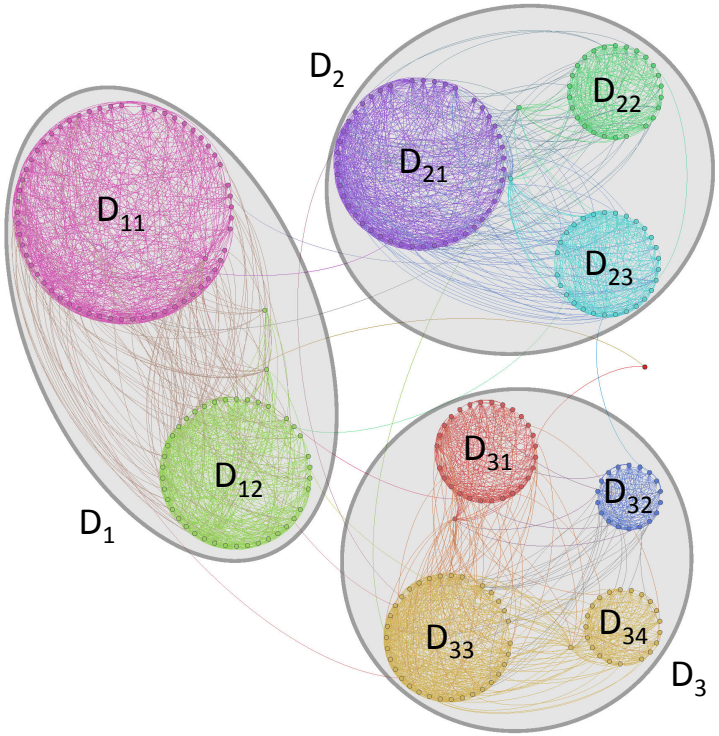


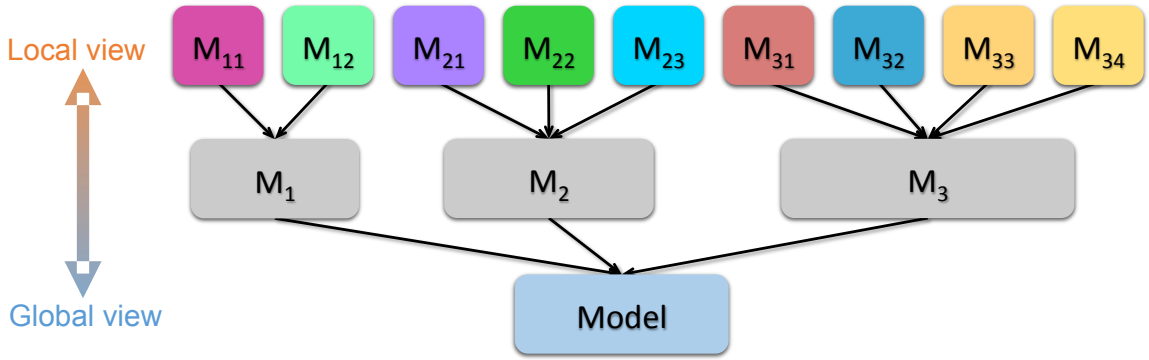


Efficiently compute solution of original problem

Part 1: Spectral Decomposition of Graphs

Multi-Scale Framework

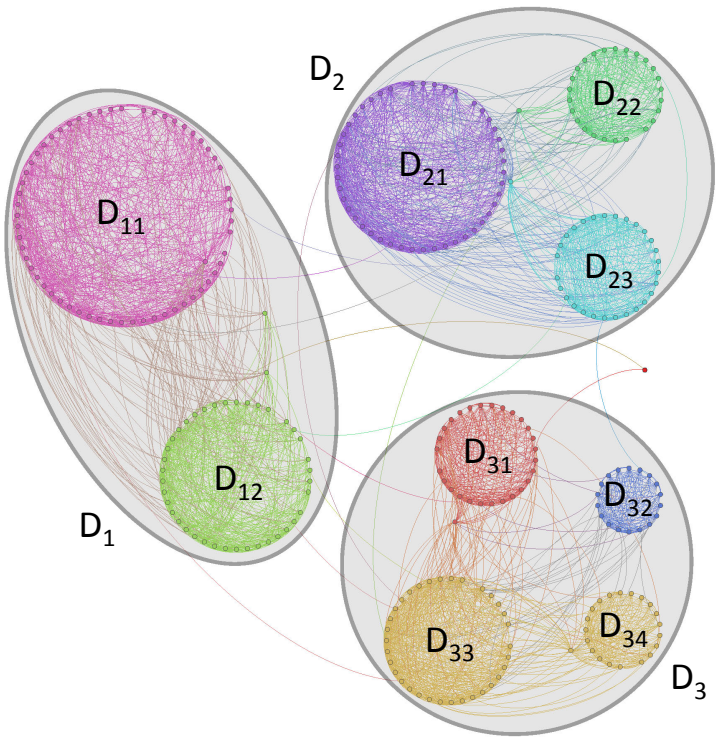


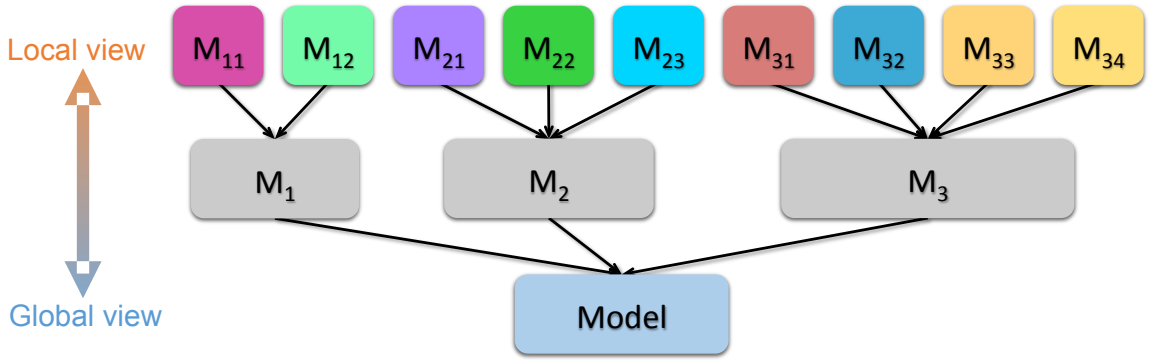


Subproblems as localized models

Models at multiple scales yield predictions at different levels of granularity

Multi-Scale Framework





Subproblems as localized models

Models at multiple scales yield predictions at different levels of granularity

Part 2: Link Prediction, Recommender Systems

Part 1: Efficient and Scalable Computation

Multi-Scale Spectral Decomposition

S. Si, D. Shin, I. S. Dhillon and B. N. Parlett. *Multi-Scale Spectral Decomposition of Massive Graphs*. NIPS'14.

Spectral Decomposition of Graphs

Goal: Given a graph $\mathcal{G}$ and its $n \times n$ adjacency matrix A, we seek to compute a rank-k approximation with reasonably large k (say $k \sim 10^2$ or 10^3):

$$A \approx U_k \Lambda_k U_k^{\mathsf{T}} = \sum_{j=1}^{k} \lambda_j \mathbf{u}_j \mathbf{u}_j^{\mathsf{T}}$$

- $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_k|$ are the leading eigenvalues of A
- $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ are the corresponding eigenvectors of A

Spectral Decomposition of Graphs

Goal: Given a graph $\mathcal{G}$ and its $n \times n$ adjacency matrix A, we seek to compute a rank-k approximation with reasonably large k (say $k \sim 10^2$ or 10^3):

$$A \approx U_k \Lambda_k U_k^{\mathsf{T}} = \sum_{j=1}^{k} \lambda_j \mathbf{u}_j \mathbf{u}_j^{\mathsf{T}}$$

- $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_k|$ are the leading eigenvalues of A
- $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ are the corresponding eigenvectors of A

A key operation needed in many ML tasks:

- Label propagation for semi-supervised and multi-label learning
- Recommender systems with side-information
- Link prediction in social network analysis
- Principal Component Analysis, Latent Semantic Indexing
- Graph matching
- ...

Spectral Decomposition

Standard iterative methods:

- Single-vector versions:
  - Power method — restricted to the leading eigenvector
  - Lanczos algorithm — difficulty with multiplicities of eigenvalues
- Block versions — slow convergence due to random initialization
  - Randomized SVD (Subspace Iteration)
  - Block Lanczos

Software Packages:

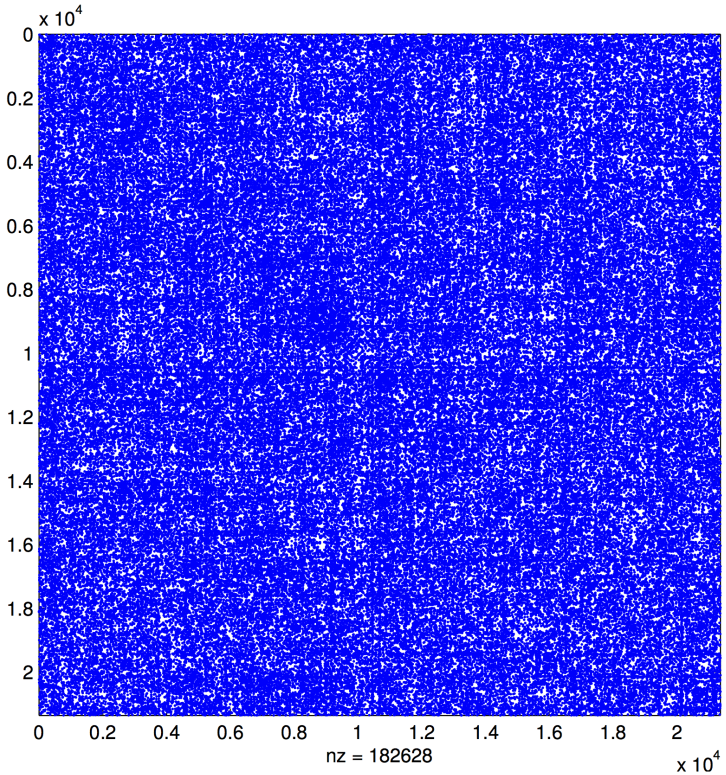
- ARPACK (Matlab's `eigs`), PROPACK, SVDPACK, SLEPc, etc

Motivation

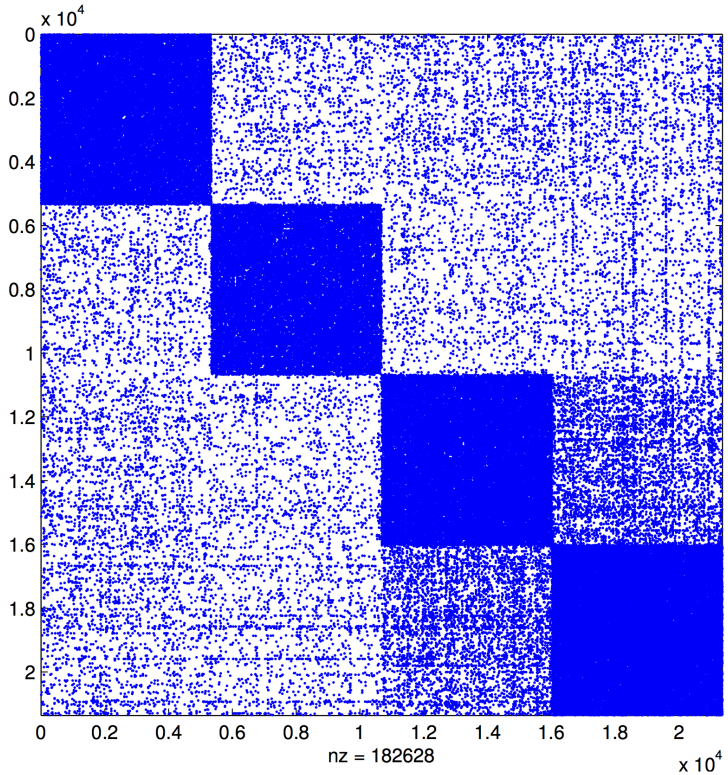
Cluster Structure

Graphs exhibit cluster structure.

CondMat dataset: 21,362 nodes and 182,628 edges (collaboration network); using Metis.



(a) Original graph (A)



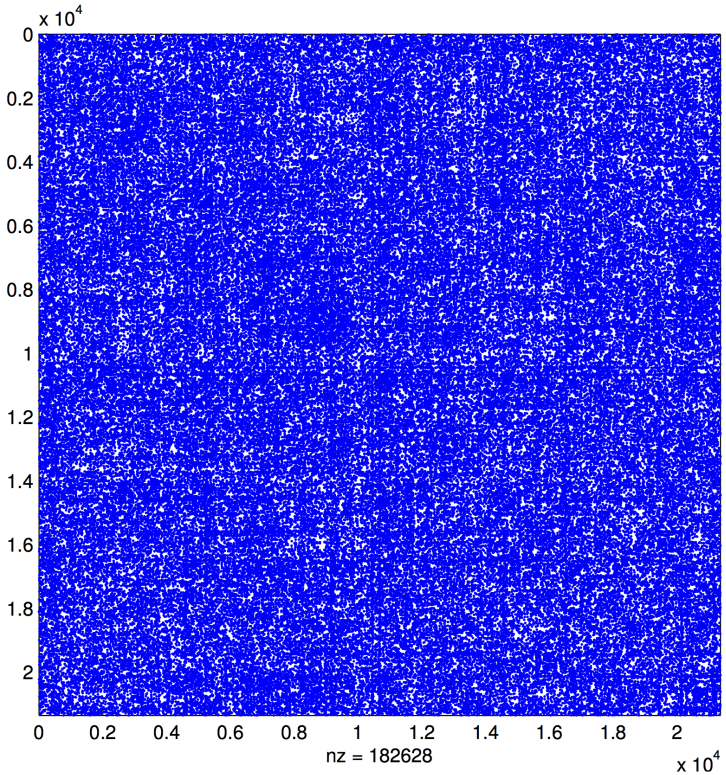
(b) Clustered graph ($A = D + \Delta$)

More than 86% of edges appear within clusters.

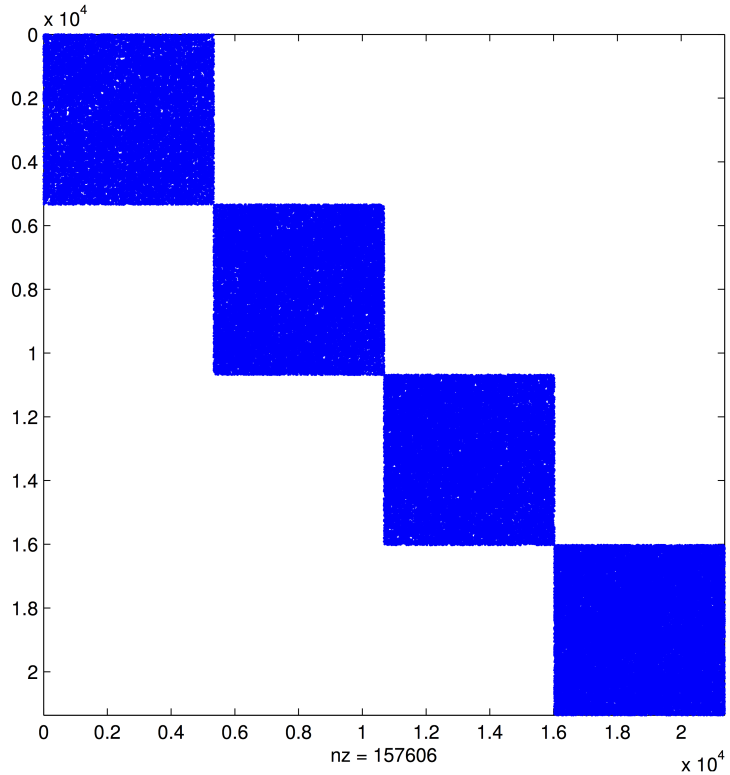
Cluster Structure

Graphs exhibit cluster structure.

CondMat dataset: 21,362 nodes and 182,628 edges (collaboration network); using Metis.



(a) Original graph (A)

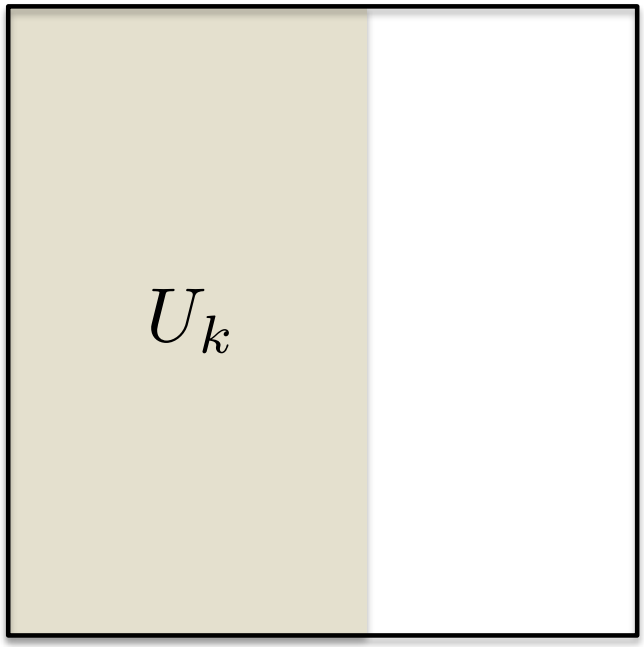


(b) Clustered graph (D)

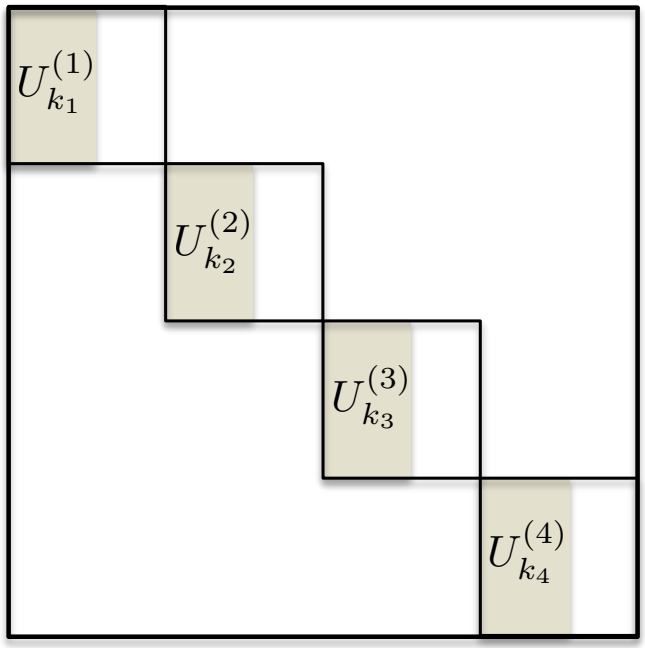
More than 86% of edges appear within clusters.

Subspace of Clusters

Subspace of entire graph U_k and union of cluster's subspace $\Omega = U_{k_1}^{(1)} \oplus \cdots \oplus U_{k_4}^{(4)}$.



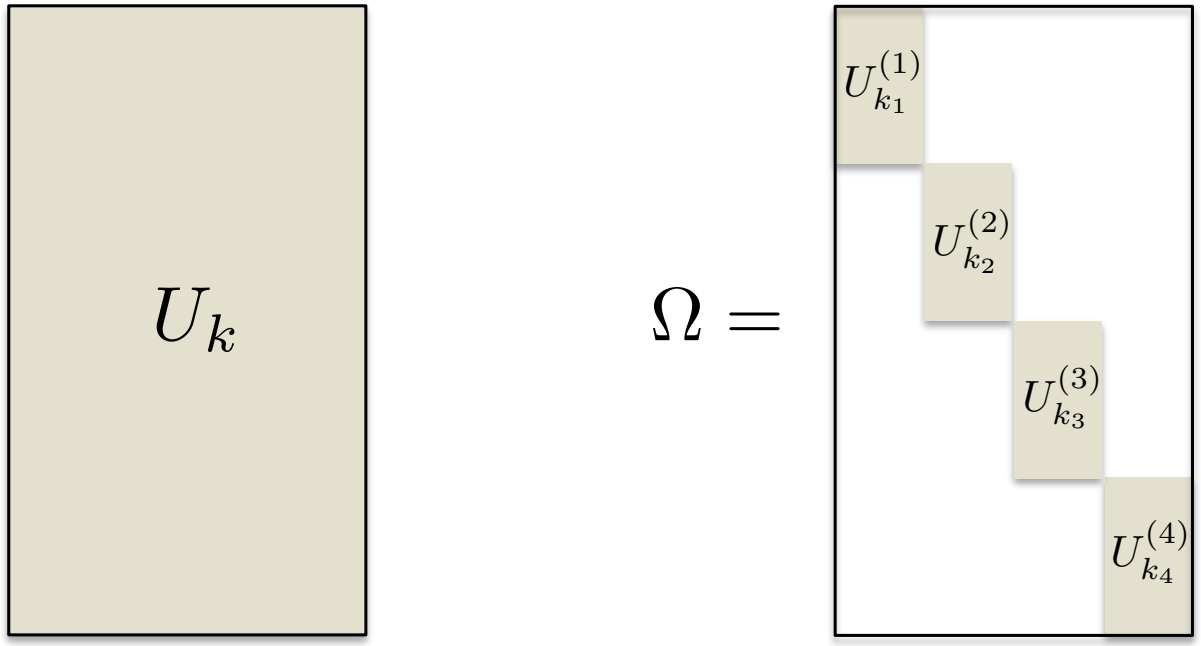
(a) Original graph (A)



(b) Clustered graph (D)

Subspace of Clusters

Subspace of entire graph U_k and union of cluster's subspace $\Omega = U_{k_1}^{(1)} \oplus \cdots \oplus U_{k_4}^{(4)}$.



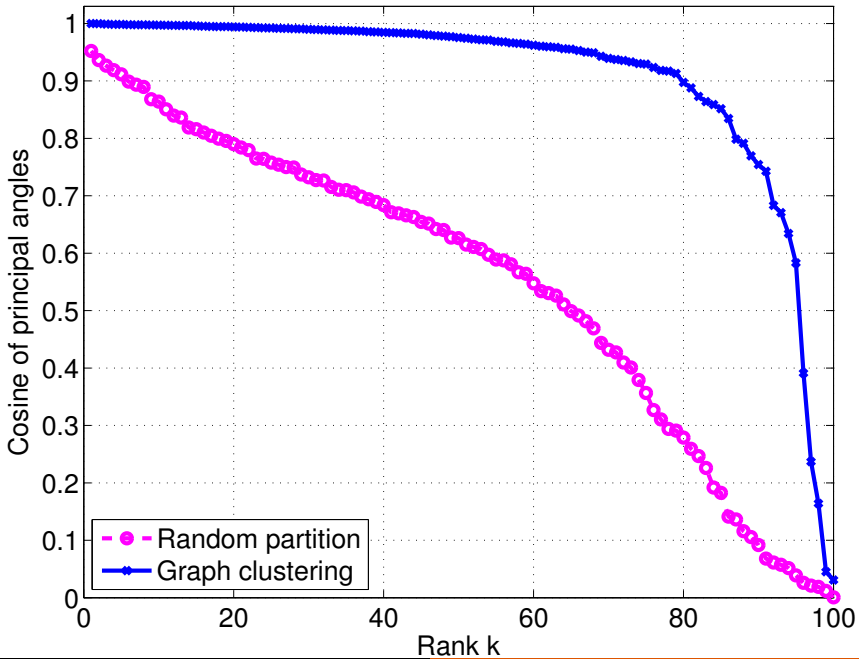
(a) Subspace of entire graph (b) Union of cluster's subspace

Subspace of Clusters

Union of all cluster's subspaces Ω has significant overlap with the dominant subspace U_k of the original graph.

$$\begin{bmatrix} \vdots \\ U_k \\ \vdots \end{bmatrix} \xleftrightarrow{\text{large overlap}} \Omega = \begin{bmatrix} U_{k_1}^{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & U_{k_c}^{(c)} \end{bmatrix}$$

Constructing Ω — graph clustering vs. random partition



Principal Angles between U_k and Ω

Theorem

Let U_k be the 'true' eigenvectors of A and let $\Omega = U_{k_1}^{(1)} \oplus \cdots \oplus U_{k_c}^{(c)}$, where $U_{k_i}^{(i)}$ are the k_i dominant eigenvectors of A_{ii}. For some $\eta \geq 0$, the principal angles $\Theta(\Omega, U_k)$ between Ω and U_k are related to Δ as

$$\|\sin(\Theta(\Omega, U_k))\|_2 \leq \frac{\|\Delta\|_2}{\eta}, \quad \|\sin(\Theta(\Omega, U_k))\|_F \leq \sqrt{k}\frac{\|\Delta\|_F}{\eta}.$$

$$A = D + \Delta, \quad D = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{cc} \end{bmatrix}, \quad \Delta = \begin{bmatrix} 0 & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & 0 \end{bmatrix}$$

How to Divide?

$$A = D + \Delta, \quad D = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{cc} \end{bmatrix}, \quad \Delta = \begin{bmatrix} 0 & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & 0 \end{bmatrix}$$

We want to find a partition of A such that $\|\Delta\|$ is small in order to make $\|\sin(\Theta(\Omega, U_k))\|$ small $\Rightarrow$ use graph clustering

Fast graph clustering algorithms:

- Metis [Karypis and Kumar 1999]
- Graclus [Dhillon et al. 2007]
- GEM [Whang et al. 2012]
- Nerstrand [LaSalle and Karypis 2014].

How to Divide?

$$A = D + \Delta, \quad D = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{cc} \end{bmatrix}, \quad \Delta = \begin{bmatrix} 0 & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & 0 \end{bmatrix}$$

We want to find a partition of A such that $\|\Delta\|$ is small in order to make $\|\sin(\Theta(\Omega, U_k))\|$ small $\Rightarrow$ use graph clustering

Better cluster quality (small $\|\Delta\|$) implies higher accuracy:

Vertices shuffled (%)	0	20	40	60	80	100
Within-cluster edges (%)	86.31	64.57	47.08	35.43	27.42	24.92
Avg. $\cos(\Theta(\bar{U}_k, U_k))$	0.9980	0.9757	0.9668	0.9475	0.9375	0.9268

Standard Eigensolvers

Subspace Iteration algorithm:

- works on subspace span$\{A^j Q\}$ at j-th iteration

1: Initialize Q
2: **for** $j = 1, 2, \ldots, t$ **do**
3: $\quad Y = AQ$
4: $\quad QR = \mathrm{qr}(Y)$ (QR-factorization of Y, overwrites Q)
5: **end for**
6: $B = Q^{\mathsf{T}} AQ$
7: $V_k \Sigma_k V_k^{\mathsf{T}} \approx B$ (Eigendecomposition of B)
8: Output $U_k = QV_k$ and $\Lambda_k = \Sigma_k$ as the approximate eigenpairs

Randomized SVD: $Q \leftarrow$ Gaussian random matrix

Standard Eigensolvers

Block Lanczos algorithm:

- works on Krylov subspace span$\{Q_1, AQ_1, A^2Q_1, \cdots, A^{j-1}Q_1\}$ at j-th iteration

1: Initialize Q_1 (set $B_0 = 0,\ Q_0 = 0$)
2: **for** $j = 1, 2, \ldots, t$ **do**
3: $\quad R = AQ_j - Q_{j-1}B_{j-1}^{\mathsf{T}}$ (R orthogonal to Q_{j-1})
4: $\quad D_j = Q_j^{\mathsf{T}}R$ (Obtain D_j by projecting R onto Q_j)
5: $\quad R = R - Q_jD_j$ (R orthogonal to Q_j)
6: $\quad Q_{j+1}B_j = \mathrm{qr}(R)$ (QR-factorization of R)
7: **end for**
8: Form block tri-diagonal matrix T_j
9: $V_k\Sigma_kV_k^{\mathsf{T}} \approx T_j$ (Eigendecomposition of T_j)
10: Output $U_k = [Q_1 \cdots Q_j]V_k$ and $\Lambda_k = \Sigma_k$ as the approximate eigenpairs

Standard Eigensolvers

Block Lanczos algorithm:

- works on Krylov subspace span$\{Q_1, AQ_1, A^2Q_1, \cdots, A^{j-1}Q_1\}$ at j-th iteration

1: Initialize Q_1 (set $B_0 = 0,\ Q_0 = 0$)
2: **for** $j = 1, 2, \ldots, t$ **do**
3: $\quad R = AQ_j - Q_{j-1}B_{j-1}^{\mathsf{T}}$ (R orthogonal to Q_{j-1})
4: $\quad D_j = Q_j^{\mathsf{T}}R$ (Obtain D_j by projecting R onto Q_j)
5: $\quad R = R - Q_jD_j$ (R orthogonal to Q_j)
6: $\quad Q_{j+1}B_j = \mathrm{qr}(R)$ (QR-factorization of R)
7: **end for**
8: Form block tri-diagonal matrix T_j
9: $V_k\Sigma_kV_k^{\mathsf{T}} \approx T_j$ (Eigendecomposition of T_j)
10: Output $U_k = [Q_1 \cdots Q_j]V_k$ and $\Lambda_k = \Sigma_k$ as the approximate eigenpairs

Standard iterative eigensolvers start with a random initial matrix

$\Rightarrow$ slow convergence

How to Merge?

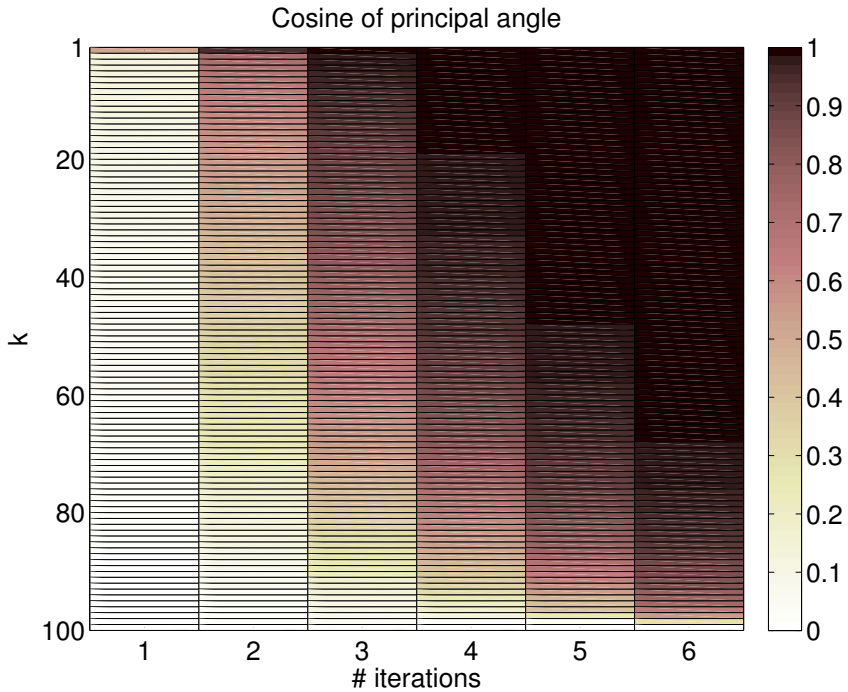
Main Idea

Use Ω obtained from the clusters of A as a good initialization to standard eigensolvers.

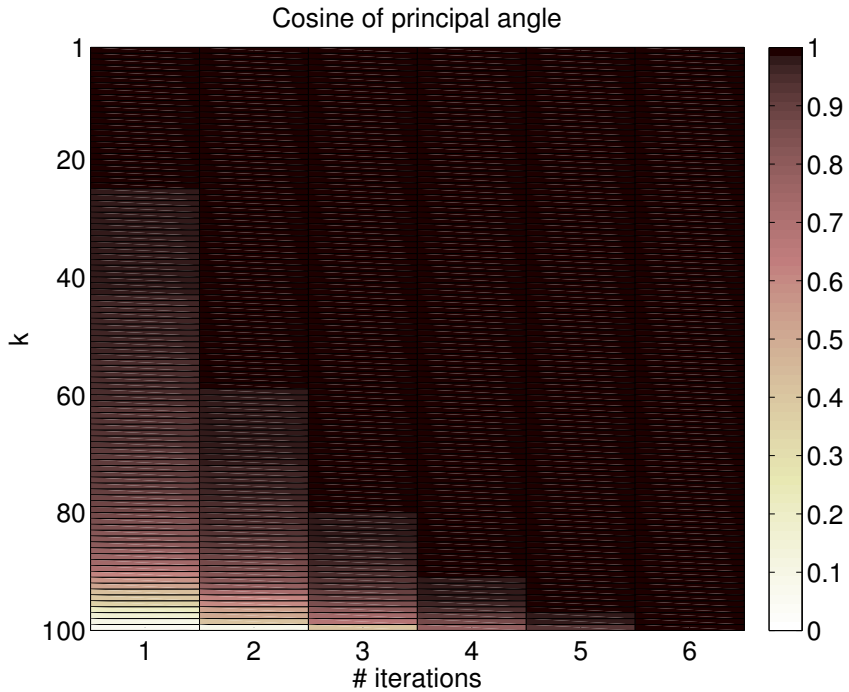
How to Merge?

Main Idea

Use Ω obtained from the clusters of A as a good initialization to standard eigensolvers.



(c) Random initialization



(d) Start with Ω

Single-level MSEIGS

Main Idea

Use Ω obtained from the clusters of A as a good initialization to standard eigensolvers.

Multi-Scale Spectral Decomposition (MSEIGS) with single level:

1: Generate c clusters $A_{11}, \cdots, A_{cc}$ via graph clustering.
2: Compute top-r eigenpairs $(\lambda_j^{(i)}, \mathbf{u}_j^{(i)})$ of A_{ii} using standard eigensolvers $(r \leq k)$.
3: Select top-k eigenpairs from the c clusters.
4: Form block diagonal matrix $\Omega = U_{k_1}^{(1)} \oplus \cdots \oplus \Omega_{k_c}^{(c)}$ $(\sum_i k_i = k)$.
5: Apply block Lanczos with initialization $Q_1 = \Omega$.

Number of Clusters

Trade-off in choosing the number of clusters c:

- Small c will give:
  - Larger clusters — increases time to compute eigenpairs of A_{ii}
  - Smaller $\|\Delta\|$ — faster convergence of single-level MSEIGS

Number of Clusters

Trade-off in choosing the number of clusters c:

- Small c will give:
  - Larger clusters — increases time to compute eigenpairs of A_{ii}
  - Smaller $\|\Delta\|$ — faster convergence of single-level MSEIGS
- Large c will give:
  - Smaller clusters — faster to compute eigenpairs of A_{ii}
  - Larger $\|\Delta\|$ — slower convergence of single-level MSEIGS

Number of Clusters

Trade-off in choosing the number of clusters c:

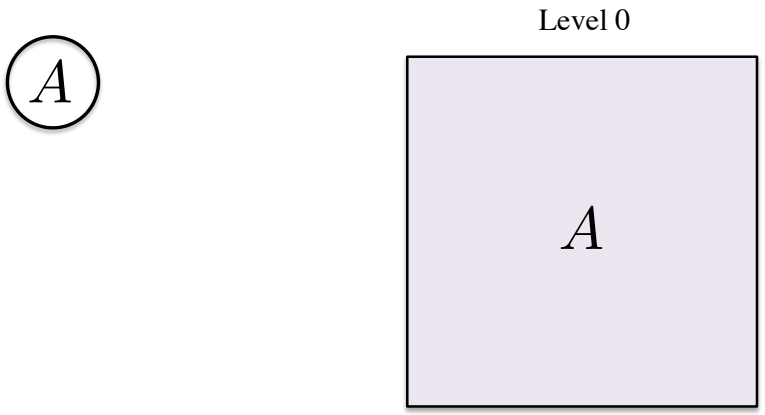
- Small c will give:
  - Larger clusters — increases time to compute eigenpairs of A_{ii}
  - Smaller $\|\Delta\|$ — faster convergence of single-level MSEIGS
- Large c will give:
  - Smaller clusters — faster to compute eigenpairs of A_{ii}
  - Larger $\|\Delta\|$ — slower convergence of single-level MSEIGS

$\Rightarrow$ Use hierarchical clustering:

- Further partition A_{ii} into smaller clusters until each cluster is small enough.
- Recursively apply single-level MSEIGS from lower to upper levels.

Multi-Scale Spectral Decomposition (MSEIGS)

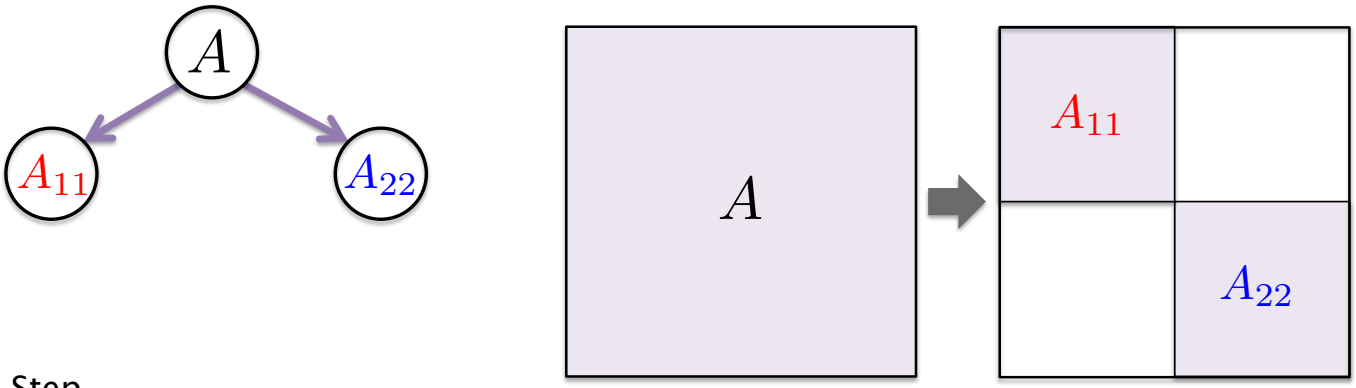
1. Construct hierarchical clustering of A.
2.
3.
4.



Divide Step

Multi-Scale Spectral Decomposition (MSEIGS)

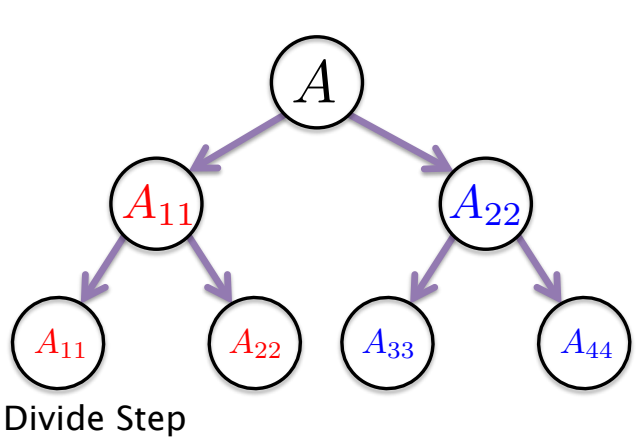
1. Construct hierarchical clustering of A.
2.
3.
4.

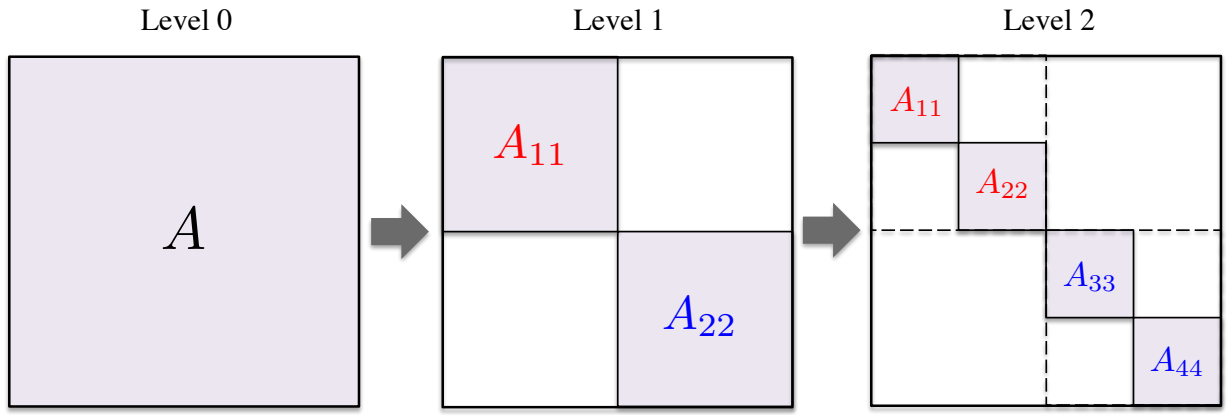


Divide Step

Multi-Scale Spectral Decomposition (MSEIGS)

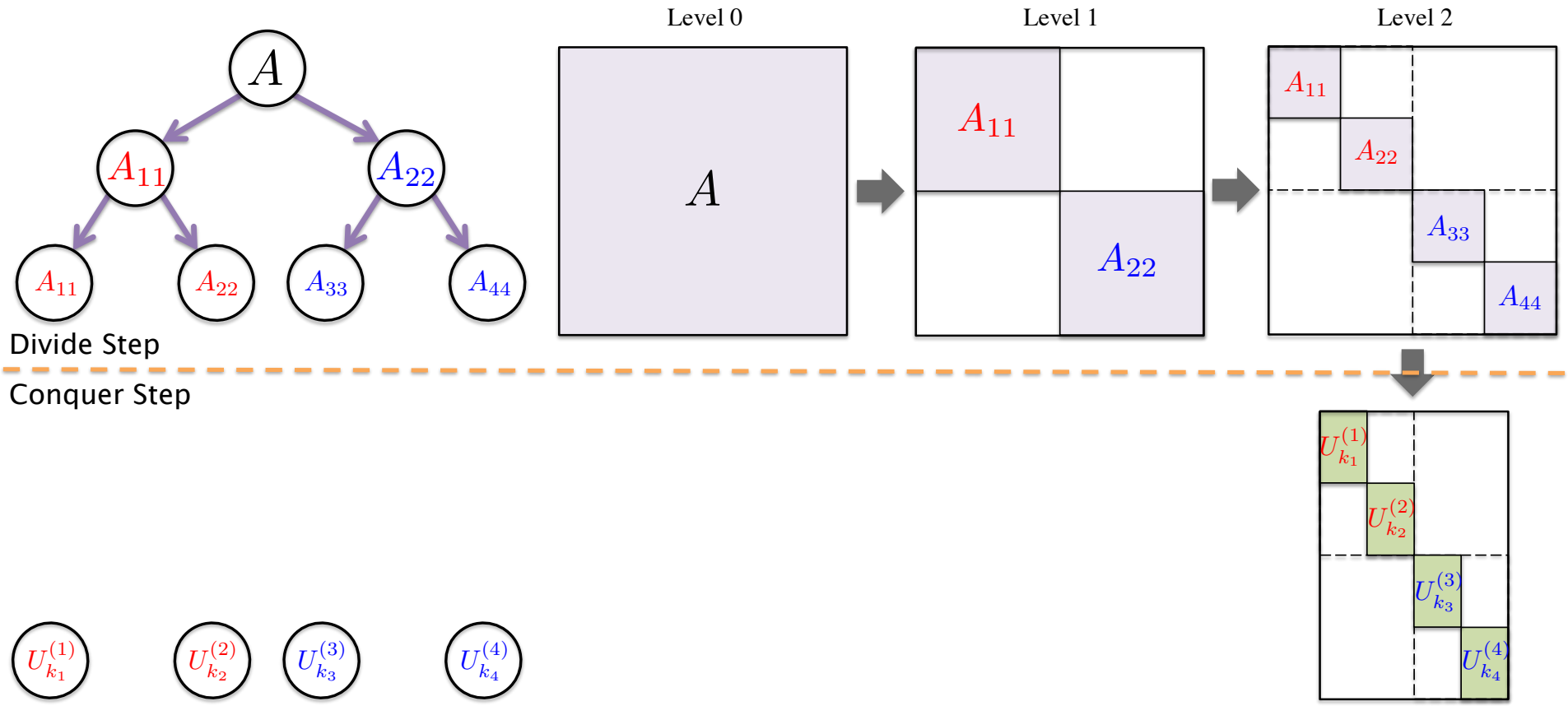
1. Construct hierarchical clustering of A.
2.
3.
4.





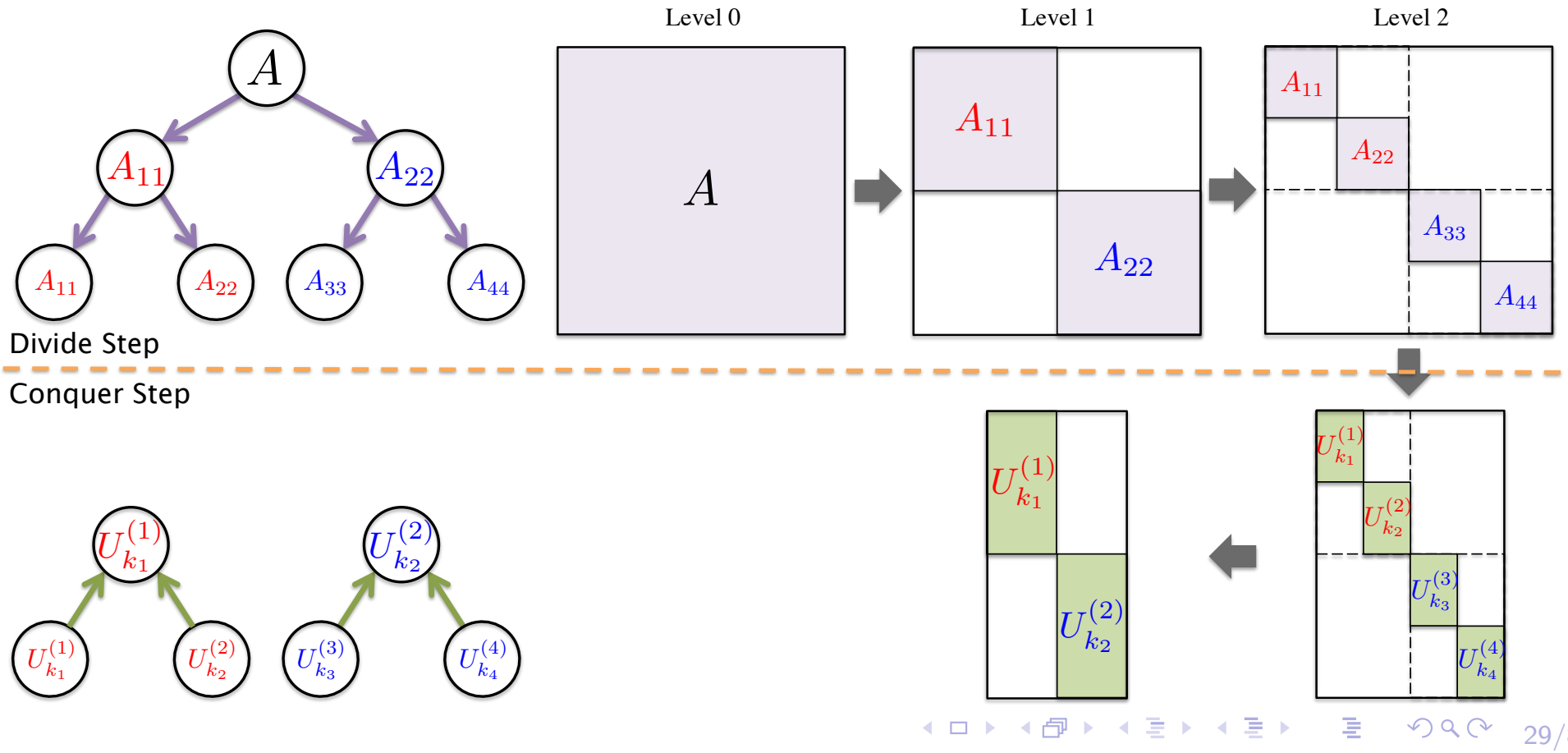
Multi-Scale Spectral Decomposition (MSEIGS)

1. Construct hierarchical clustering of A.
2. Compute approximation at the bottom level (leaf clusters).
3.
4.



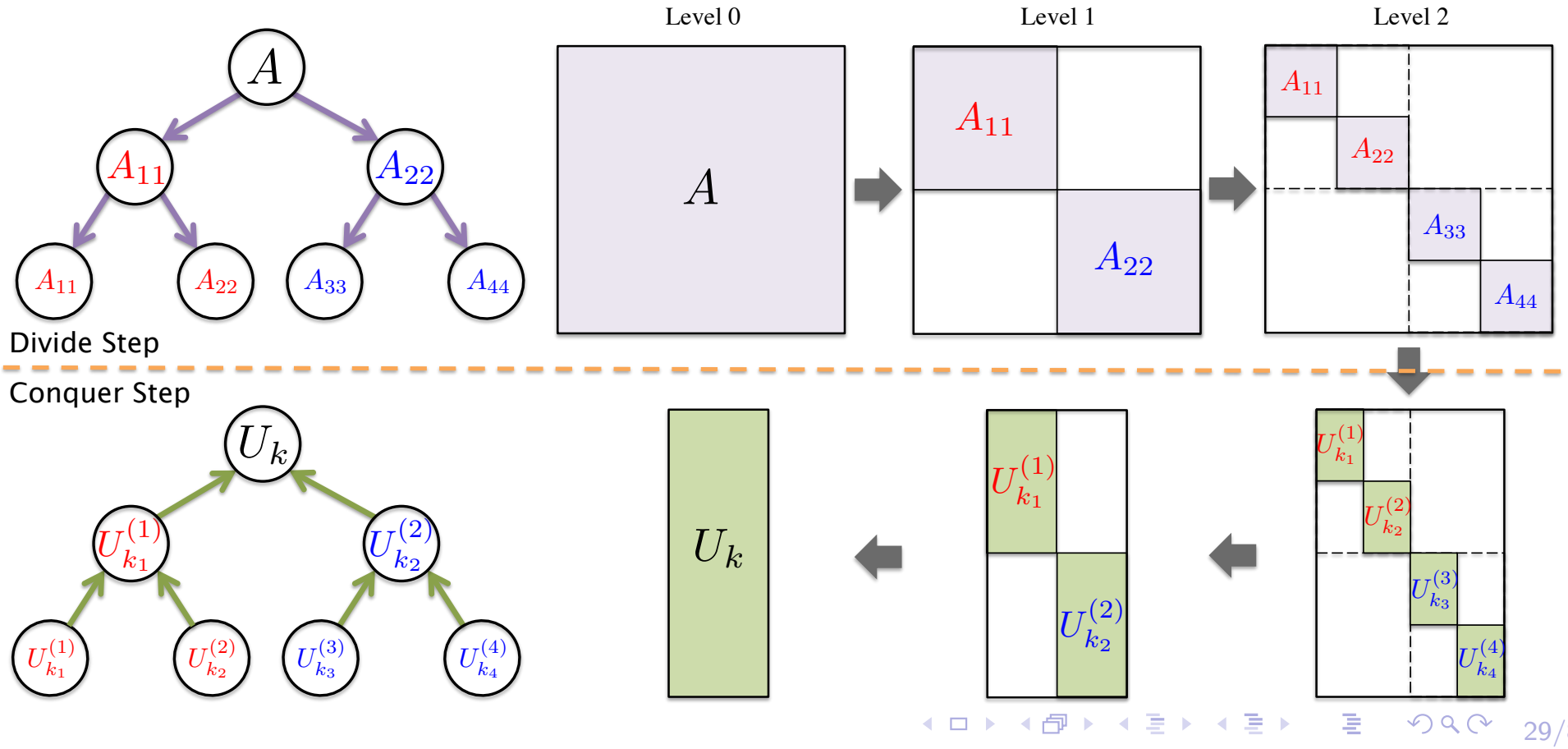
Multi-Scale Spectral Decomposition (MSEIGS)

1. Construct hierarchical clustering of A.
2. Compute approximation at the bottom level (leaf clusters).
3. Compute approximation of parent cluster using child cluster's subspace.
4.



Multi-Scale Spectral Decomposition (MSEIGS)

1. Construct hierarchical clustering of A.
2. Compute approximation at the bottom level (leaf clusters).
3. Compute approximation of parent cluster using child cluster's subspace.
4. Top level outputs (approximate) eigenpairs U_k and Λ_k of A.



Experimental Results

Multi-core Shared-memory

- All methods implemented using C/C++, OpenMP, Intel MKL
- Datasets:

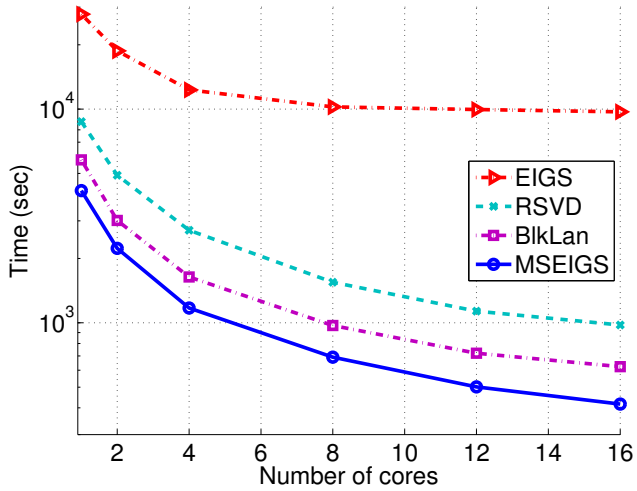
Dataset	CondMat	Amazon	RoadCA	LiveJournal	Friendster	SDWeb
nodes	21,263	334,843	1,965,206	3,997,962	10.00M	82.29M
nonzeros	182,628	1,851,744	5,533,214	69,362,378	83.67M	3.68B
rank k	100	100	200	500	100	50

Multi-core Shared-memory

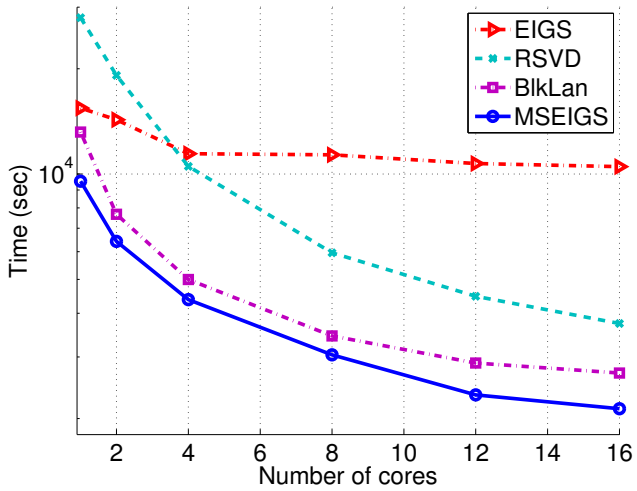
- All methods implemented using C/C++, OpenMP, Intel MKL
- Datasets:

Dataset	CondMat	Amazon	RoadCA	LiveJournal	Friendster	SDWeb
nodes	21,263	334,843	1,965,206	3,997,962	10.00M	82.29M
nonzeros	182,628	1,851,744	5,533,214	69,362,378	83.67M	3.68B
rank k	100	100	200	500	100	50

Number of cores vs. Time to compute similar approximations:



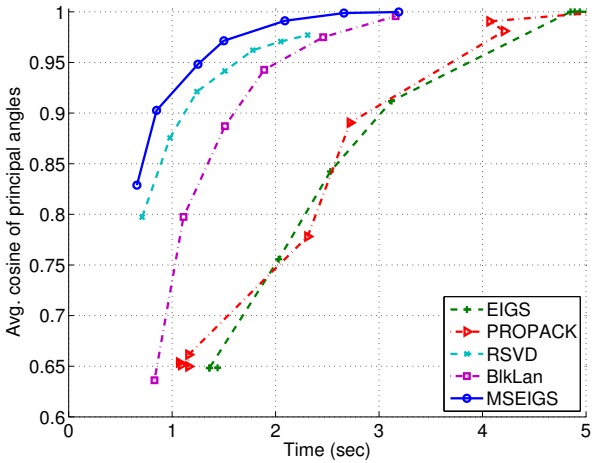
(c) LiveJournal



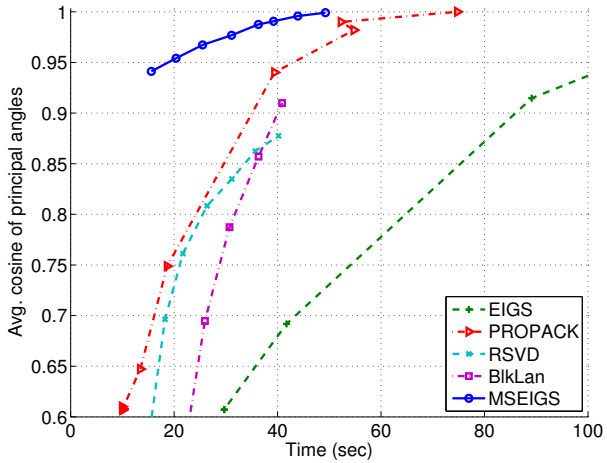
(d) SDWeb

Approximation Results

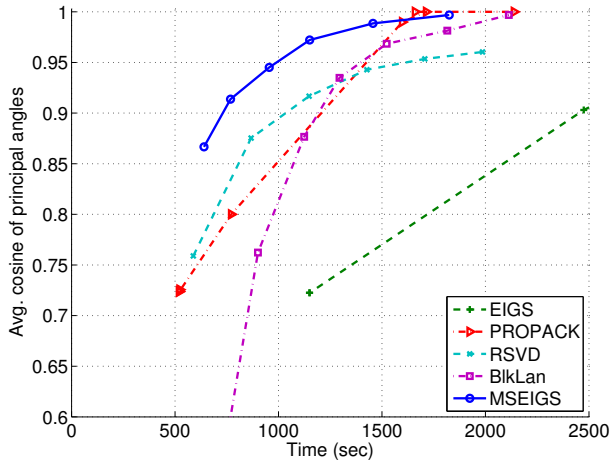
Time vs. Average cosine of principal angles:



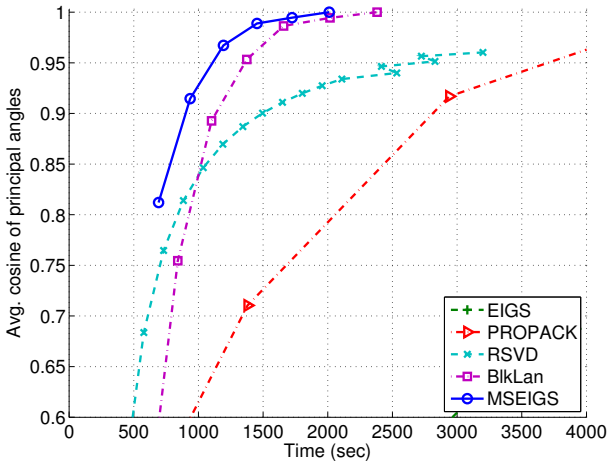
(a) CondMat



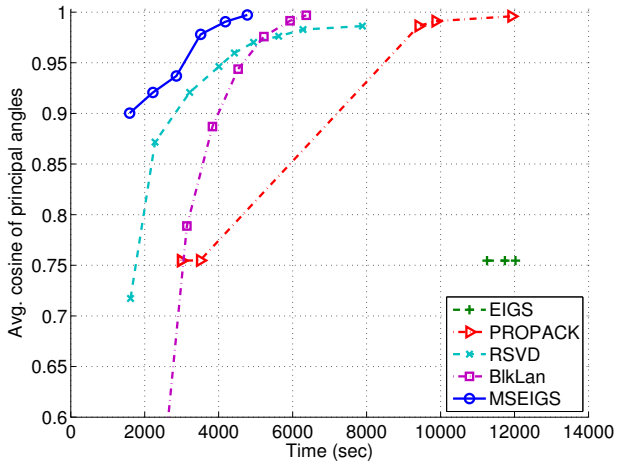
(b) Amazon



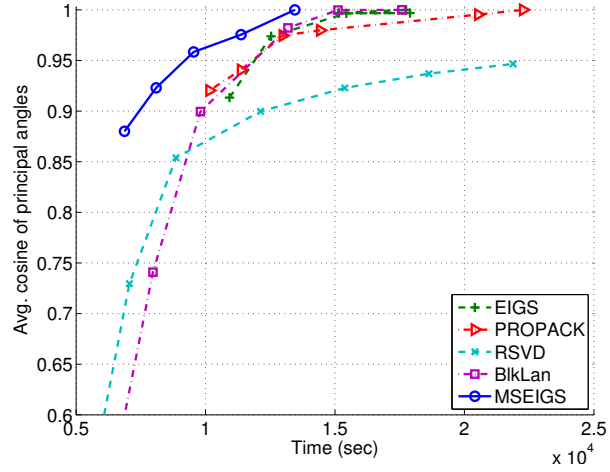
(c) FriendsterSub



(d) RoadCA



(e) LiveJournal



(f) SDWeb

Part 2: Models at Multiple Scales

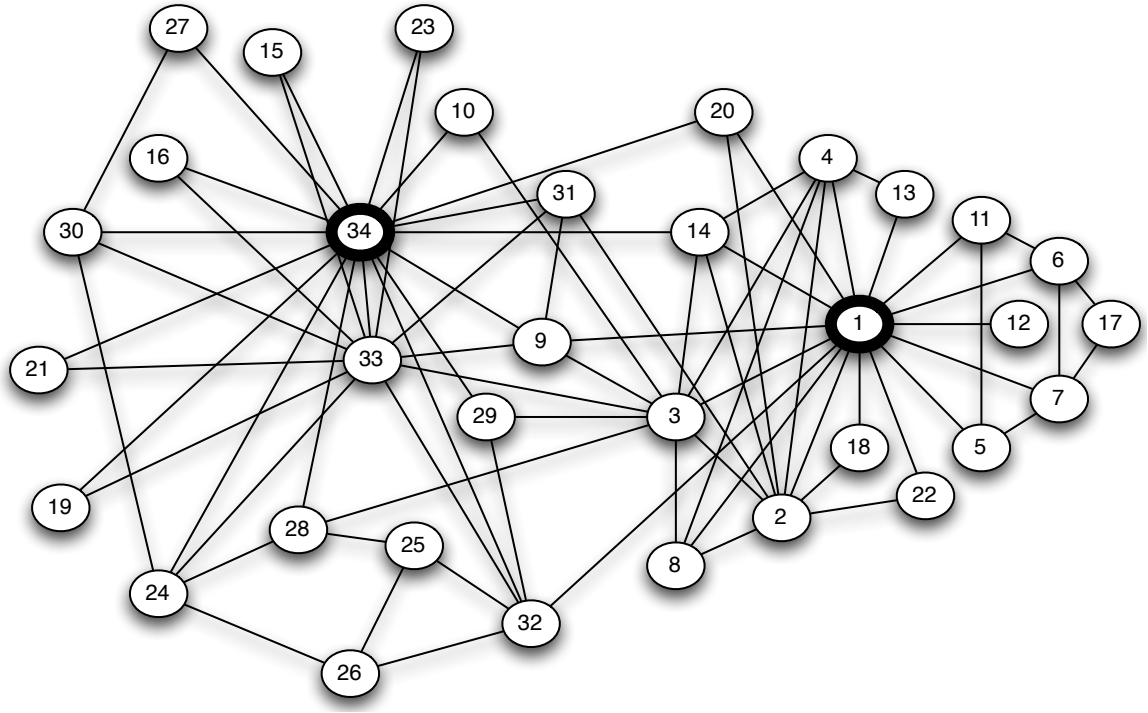
Multi-Scale Link Prediction

D. Shin, S. Si and I. S. Dhillon. *Multi-Scale Link Prediction*. CIKM'12.

Social Networks

- Nodes: individual actors (people or groups of people)
- Edges: relationships (social interactions) between nodes

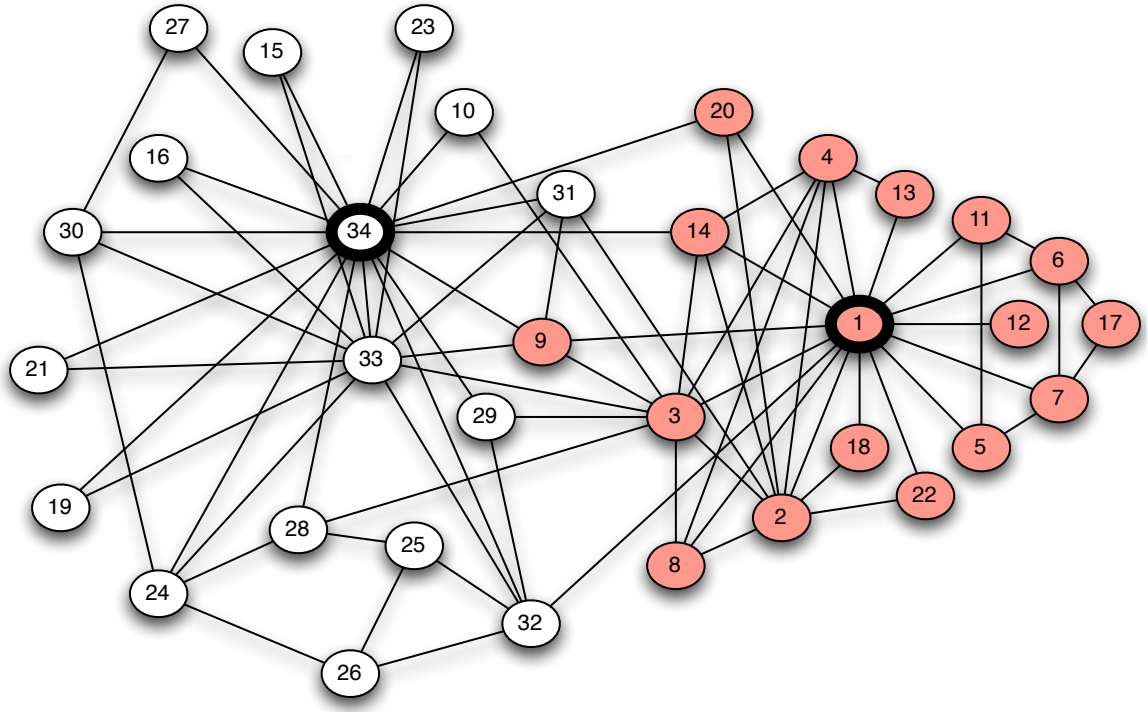
Example: Karate Club network



Social Networks

- Central cores contain most of the links
- Naturally splits into multiple tightly-connected regions

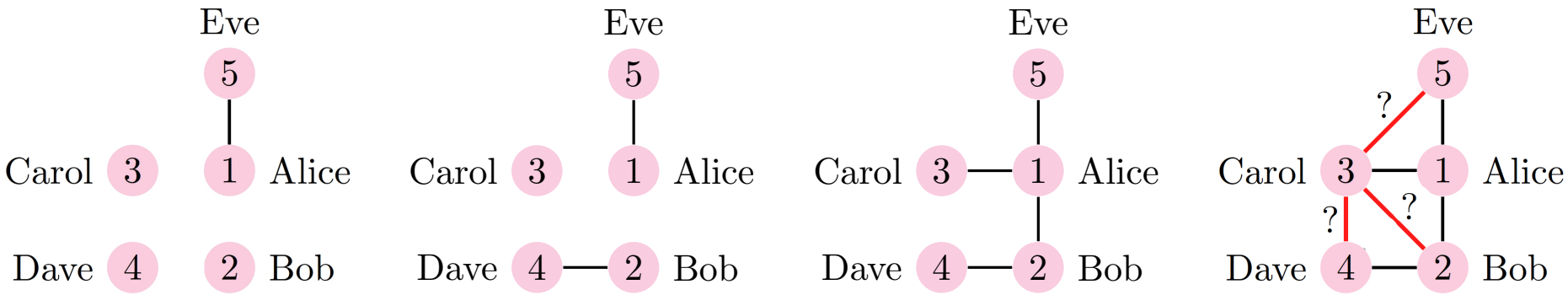
Example: Karate Club network



Link Prediction Problem

Consider a social network that evolves with time,

$$\ldots \longrightarrow A^{(t-2)} \longrightarrow A^{(t-1)} \longrightarrow A^{(t)} \xrightarrow{?} A^{t+1}$$

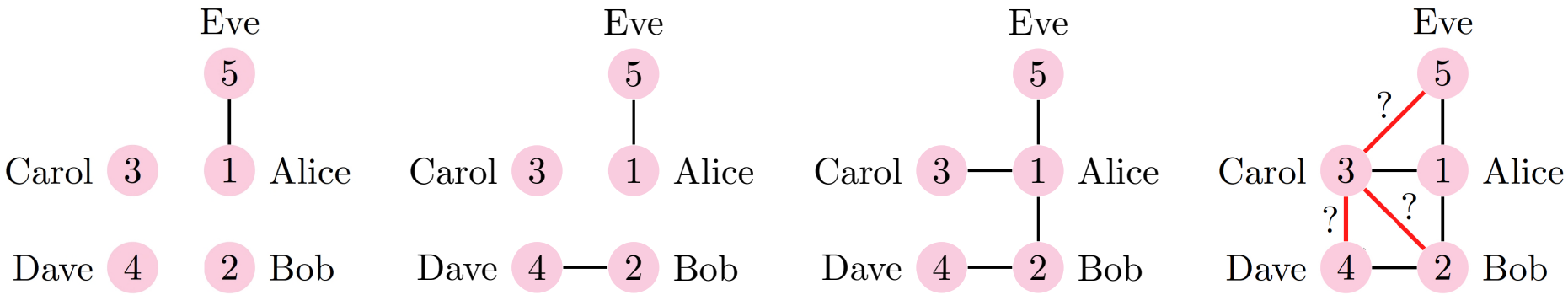


Q: Can we predict links that will form at time step $t + 1$?

Link Prediction Problem

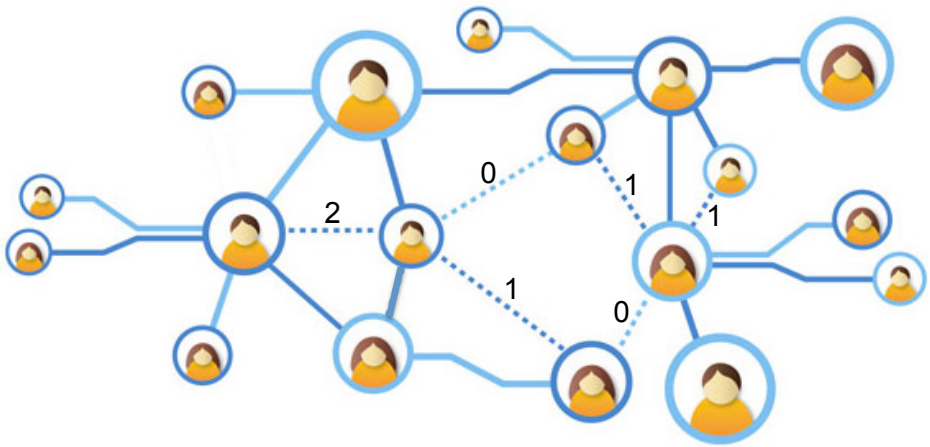
Consider a social network that evolves with time,

$$\ldots \longrightarrow A^{(t-2)} \longrightarrow A^{(t-1)} \longrightarrow A^{(t)} \xrightarrow{?} A^{t+1}$$



Q: Can we predict links that will form at time step $t + 1$?

A: Many models exist, e.g. common neighbors, the Katz measure, etc



Low-Rank Approximations

- The adjacency matrix $A = [a_{ij}]$ of graph $G = (\mathcal{V}, \mathcal{E})$:

$$a_{ij} = \begin{cases} 1, & \text{if there is an edge between nodes } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$$

- Powers of adjacency matrix A:
  $(A^n)_{ij}$ counts the number of length-n walks between nodes i and j

Low-Rank Approximations

- The adjacency matrix $A = [a_{ij}]$ of graph $G = (\mathcal{V}, \mathcal{E})$:

$$a_{ij} = \begin{cases} 1, & \text{if there is an edge between nodes } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$$

- Powers of adjacency matrix A:
  $(A^n)_{ij}$ counts the number of length-n walks between nodes i and j
- Matrix Functions, e.g., Katz measure:

$$f_{katz}(A) = (I - \beta A)^{-1} = I + \beta A + \beta^2 A^2 + \beta^3 A^3 + \cdots$$

Low-Rank Approximations

- The adjacency matrix $A = [a_{ij}]$ of graph $G = (\mathcal{V}, \mathcal{E})$:

$$a_{ij} = \begin{cases} 1, & \text{if there is an edge between nodes } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$$

- Powers of adjacency matrix A:
  $(A^n)_{ij}$ counts the number of length-n walks between nodes i and j
- Matrix Functions, e.g., Katz measure:

$$f_{katz}(A) = (I - \beta A)^{-1} = I + \beta A + \beta^2 A^2 + \beta^3 A^3 + \cdots$$

- Truncated SVD:

$$A^i \approx U \Lambda^i U^{\mathsf{T}}$$

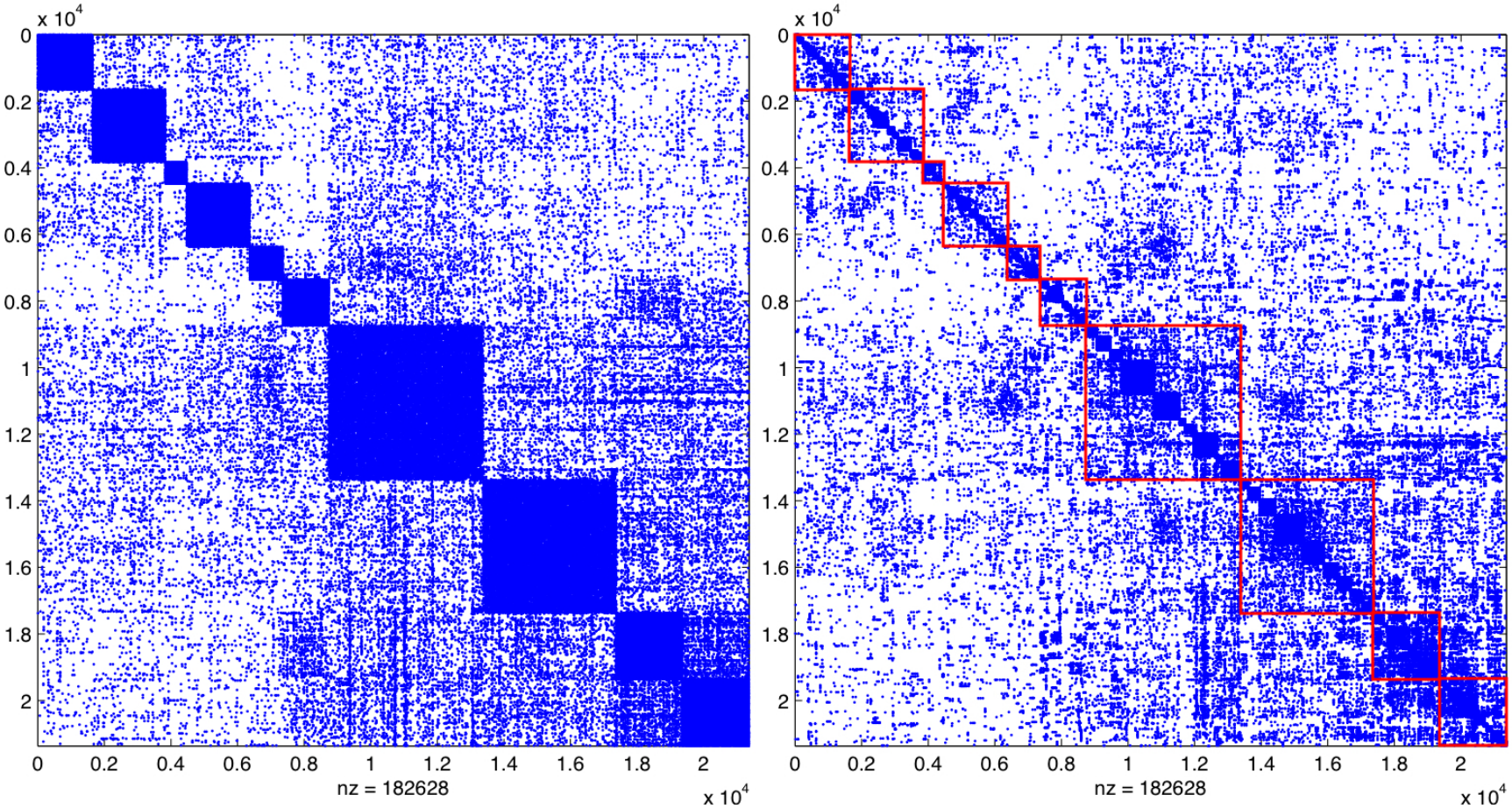
  allows approximations of matrix functions:

$$(I - \beta A)^{-1} \approx U(I - \beta \Lambda)^{-1} U^{\mathsf{T}}$$

Hierarchical Clustering

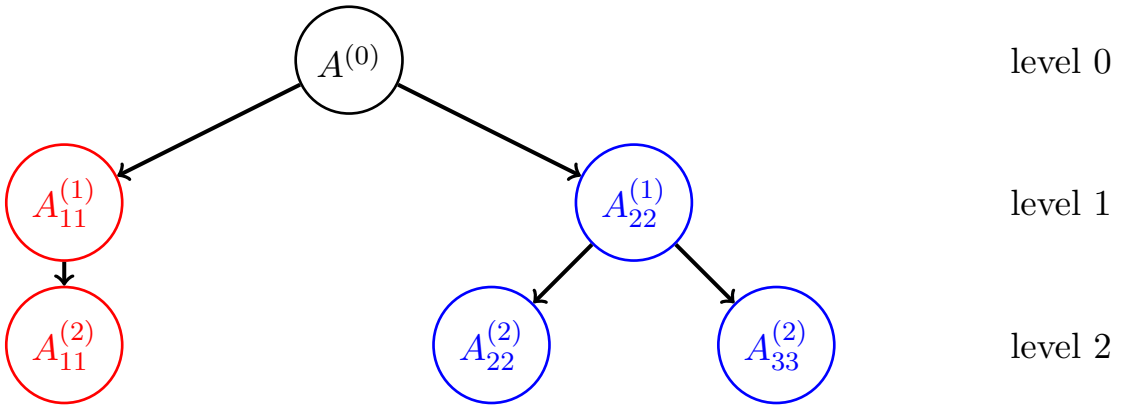
Networks exhibit a hierarchical structure.

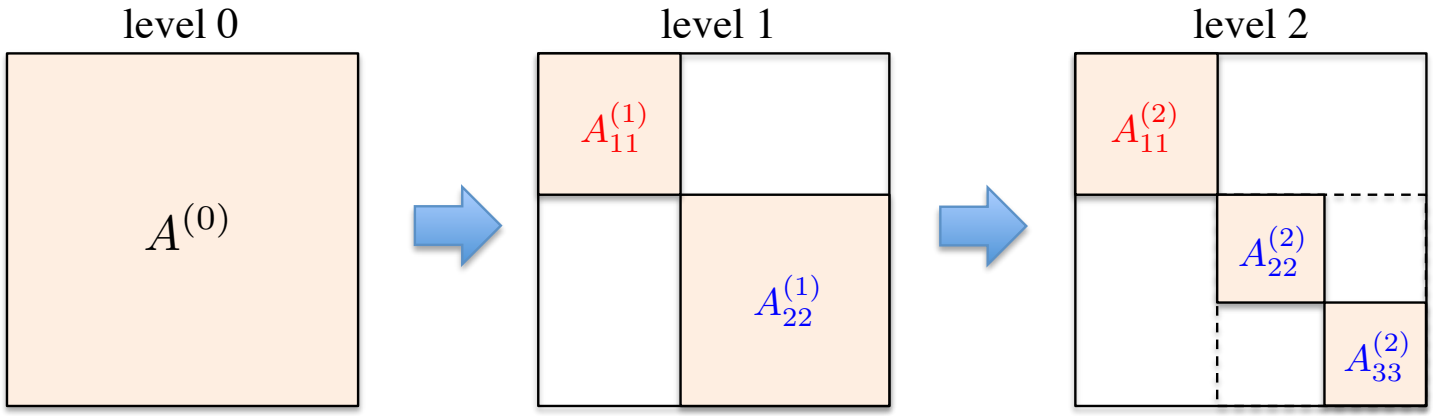
CondMat dataset: 21,362 nodes and 182,628 edges (collaboration network); using Graclus.



Hierarchical Clustering

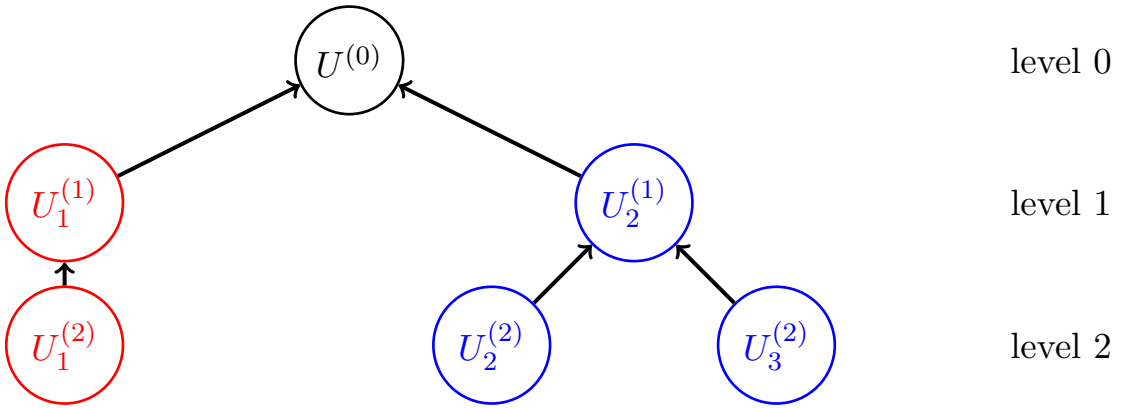
- Divide the graph into smaller subgraphs via hierarchical clustering

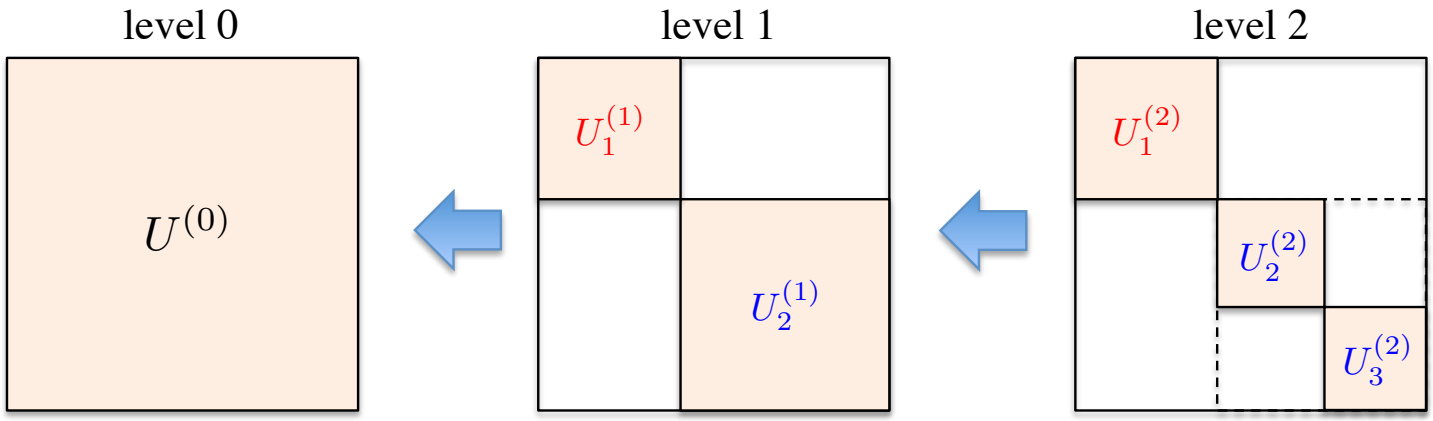




Subspace Approximation

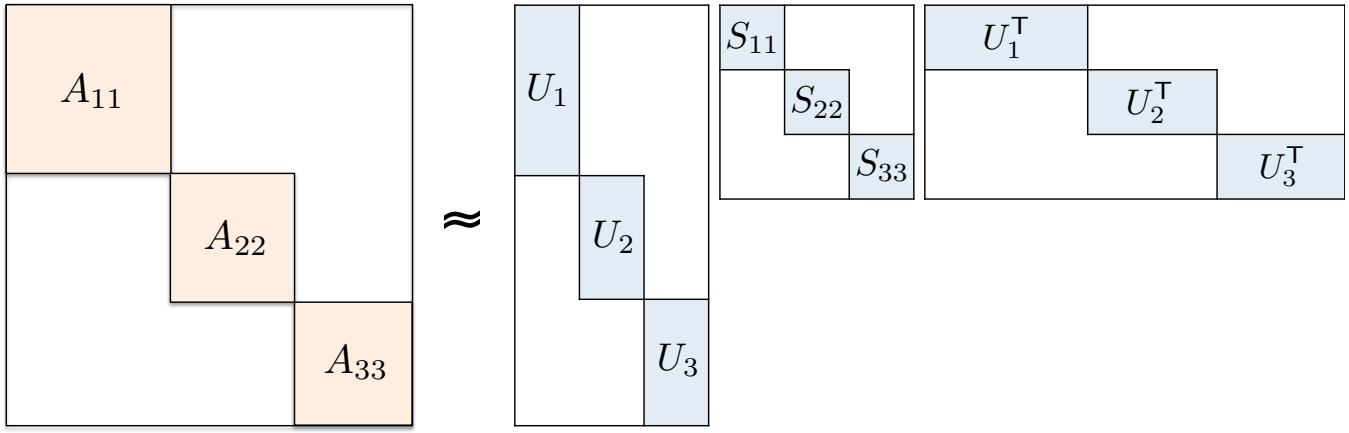
- Apply MSEIGS to efficiently obtain low-rank approximation for higher levels using lower level's approximation





Subspace Approximation

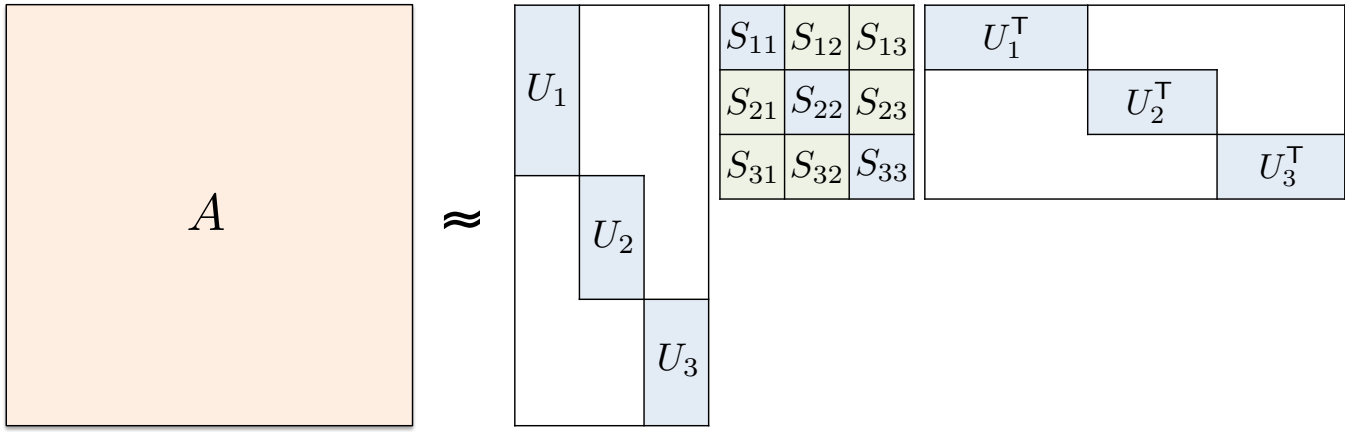
- At level 2:



Cluster-wise approximations: only approximates diagonal blocks

Subspace Approximation

- At level 2:



$$S_{ij} = U_i^T A_{ij} U_j \quad \text{for} \quad i \neq j$$

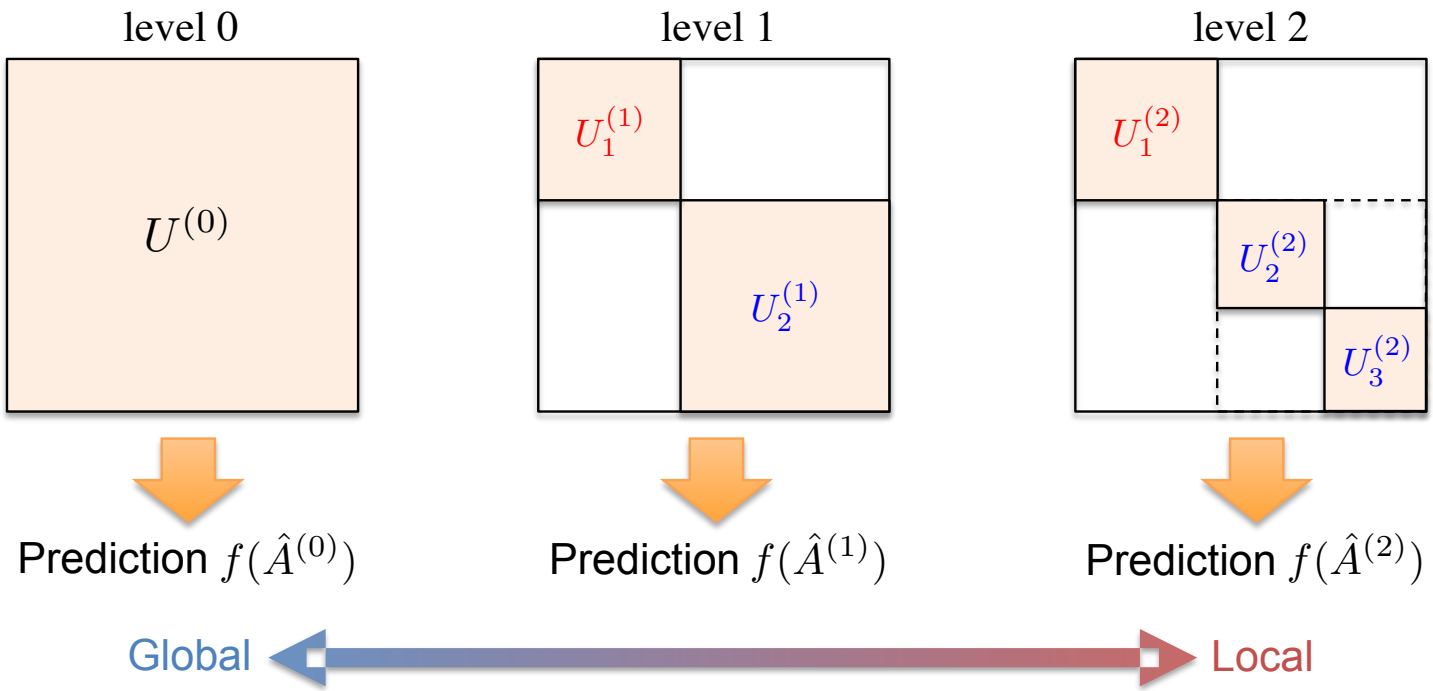
S captures the associations between clusters

Extend to an approximation for the entire matrix A

(Clustered Low Rank Approximation [Savas and Dhillon 2011])

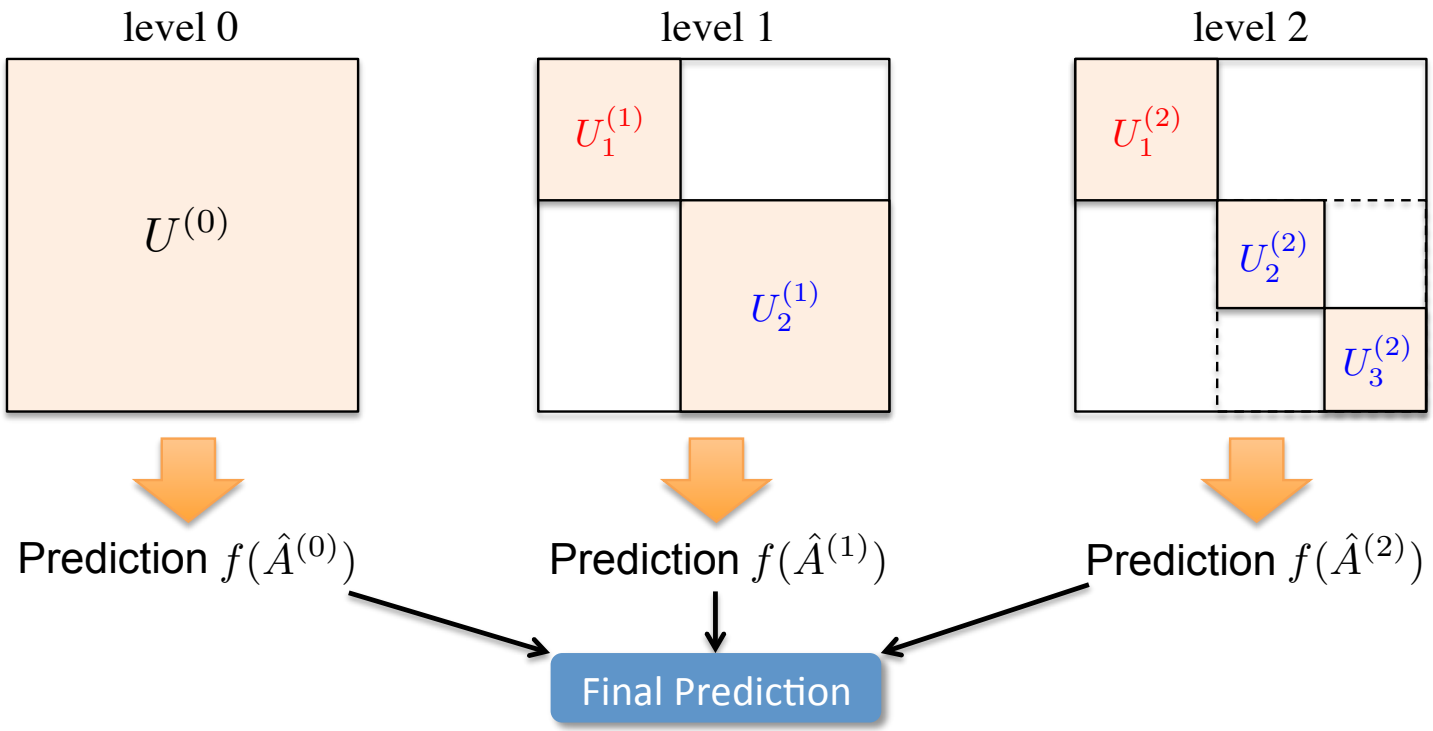
Multi-Scale Predictions

- Predictions at multiple scales from local to global

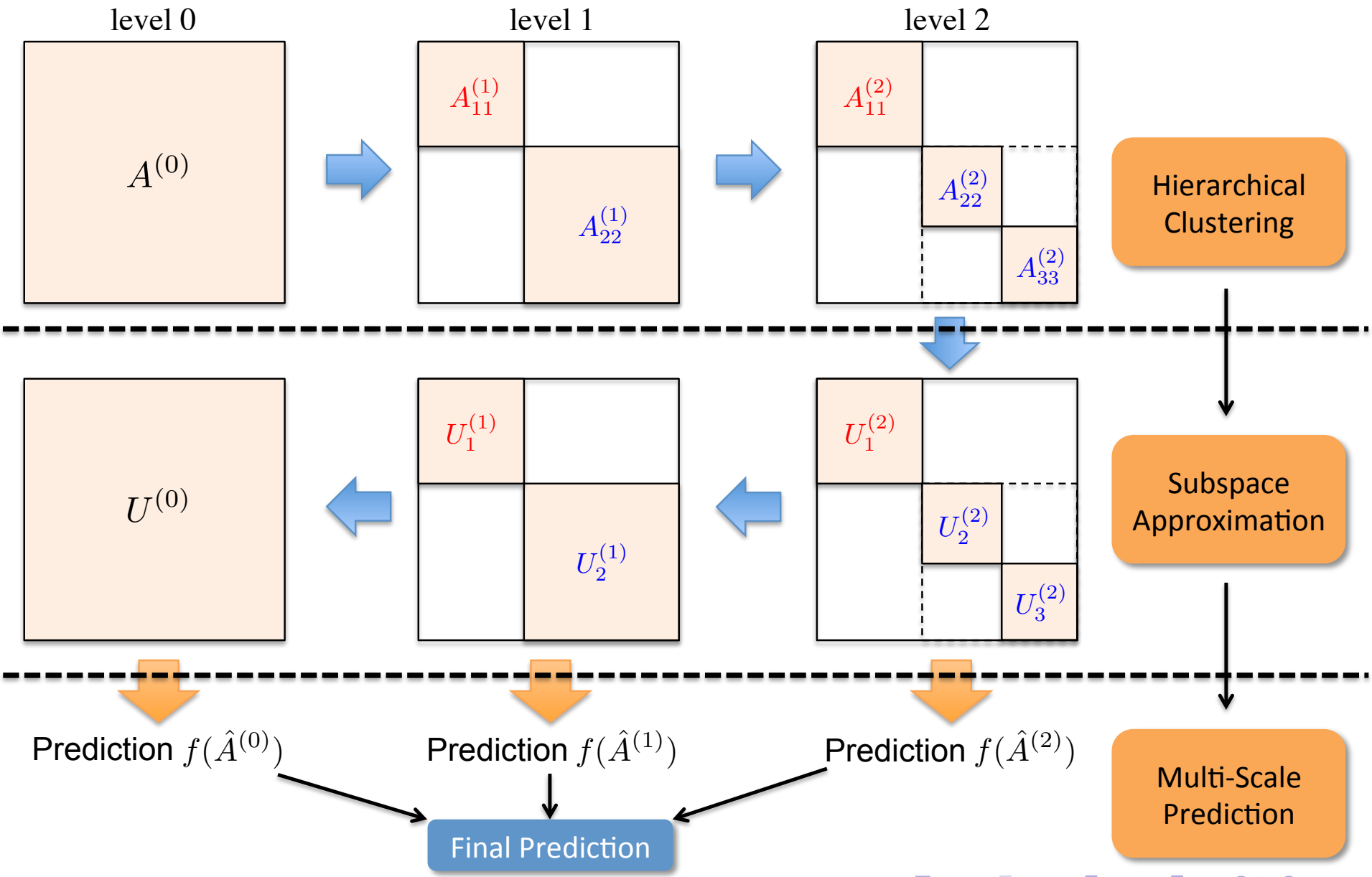


Multi-Scale Predictions

- Combine predictions at each level to make final predictions

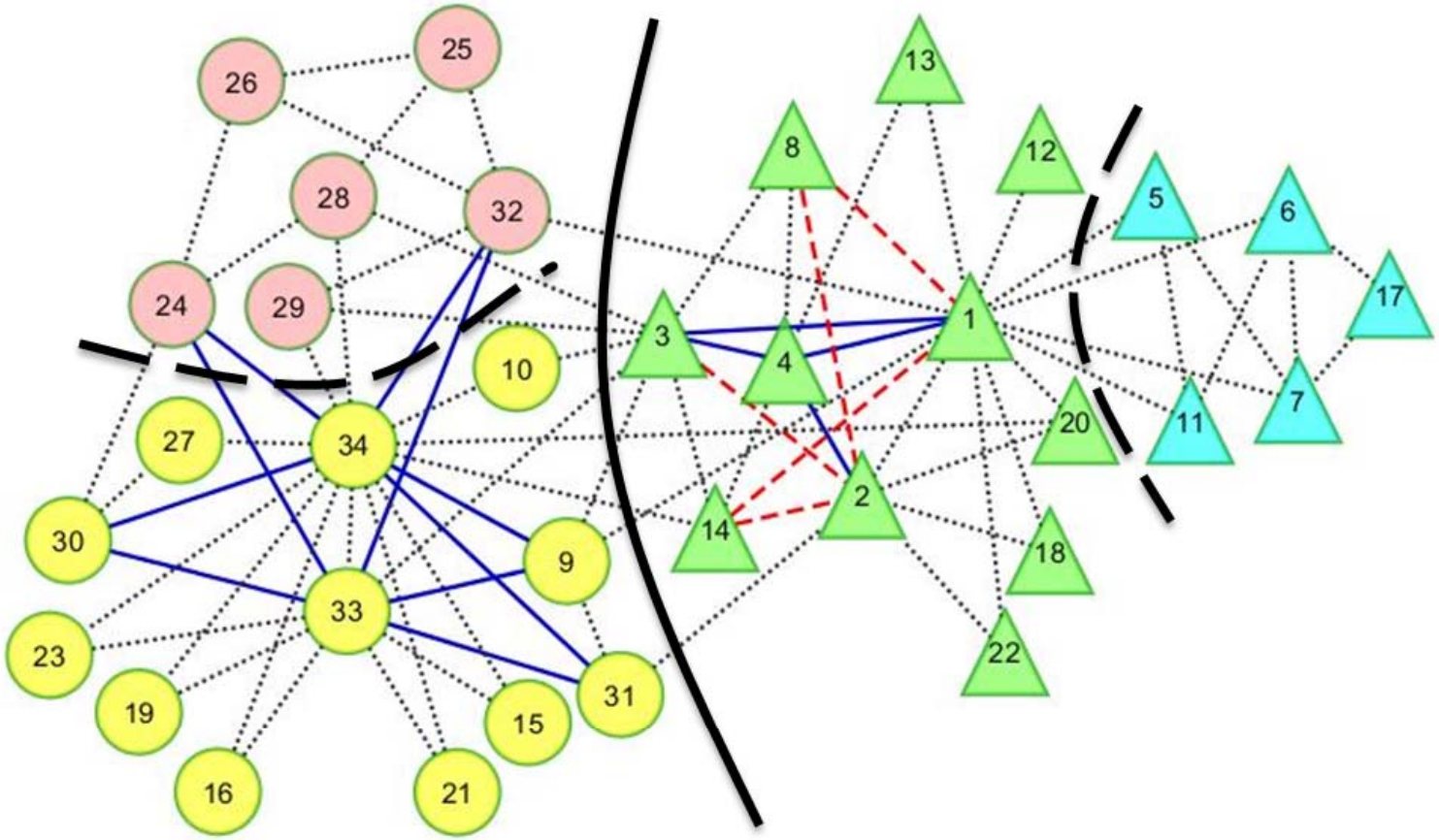


Multi-Scale Link Prediction (MSLP)



Case Study: Karate Club

Top-3 hit results — CLRA: red / MSLP: red & blue



Experimental Results

- Three large datasets: Flickr(FL), LiveJournal(LJ), MySpace(MS)
- About 2M nodes, 40M to 90M edges
- $c = 2$, $\ell = 5$, $r = 100$

Table: Precision at top-100

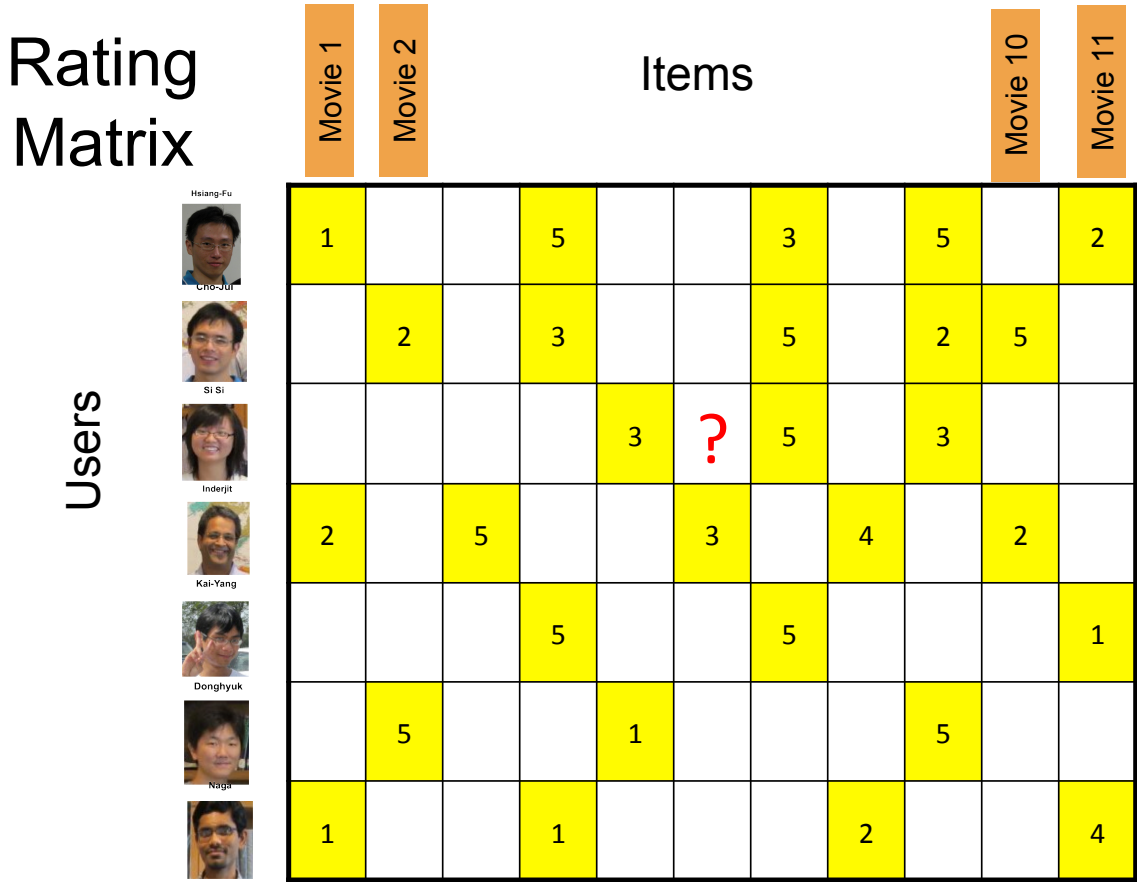
Method	FL	LJ	MS
PA (Preferential Attachment)	1.02	1.32	4.57
AA (Adamic-Adar)	7.29	5.93	7.44
RWR (Random Walk w/ Restart)	5.49	3.46	1.30
LR (Logistic Regression)*	2.54	2.23	4.95
CN (Common Neighbors)	7.08	5.94	7.18
CLRA-CN	6.91	5.21	6.88
MSLP-CN	7.03	5.59	7.05
Katz	7.17	5.86	6.18
CLRA-Katz	12.13	6.11	7.64
MSLP-Katz	**13.34**	**6.72**	**8.38**

*using network-based features.

Recommender System with Additional Information

User-Item Matrix

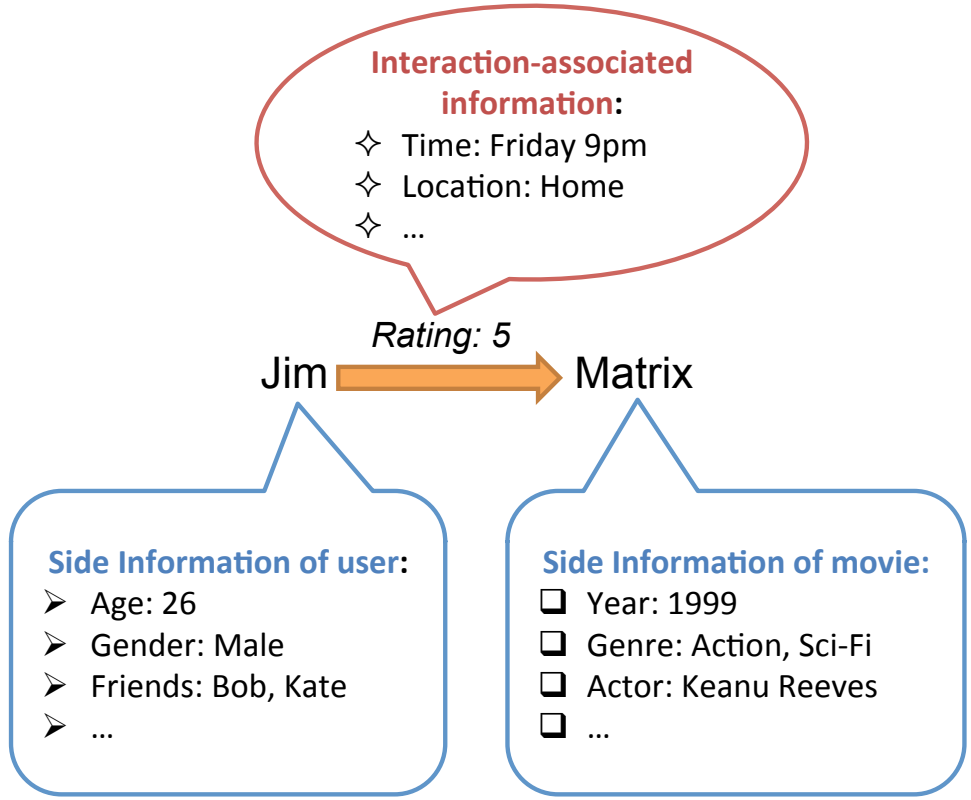
Main data source:



Additional Information

Two main types of additional information:

- Interaction information associated with the interplay of users and items:
  - Contextual information
  - Location
  - Timestamps
  - ...
- Rich side-information of users and items:
  - Attribute information
  - Social networks
  - User-generated content
  - ...



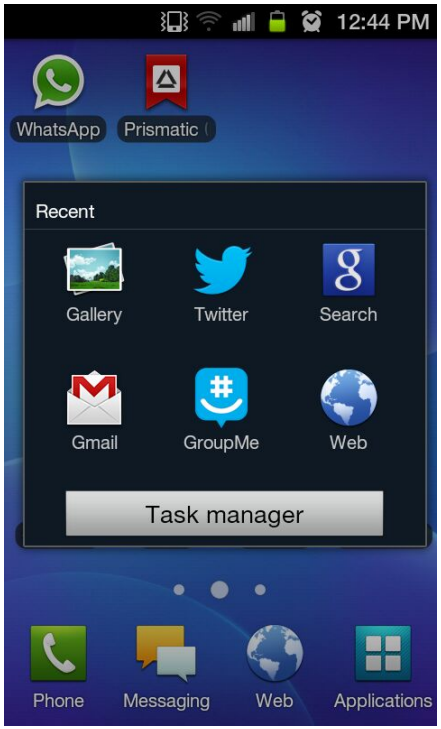
Collaborative Filtering with Interactional Context

(Mobile App Recommendation)

N. Natarajan, D. Shin and I. S. Dhillon. *Which App Will You Use Next? Collaborative Filtering with Interactional Context*. RecSys'13

Which App Will You Use Next?

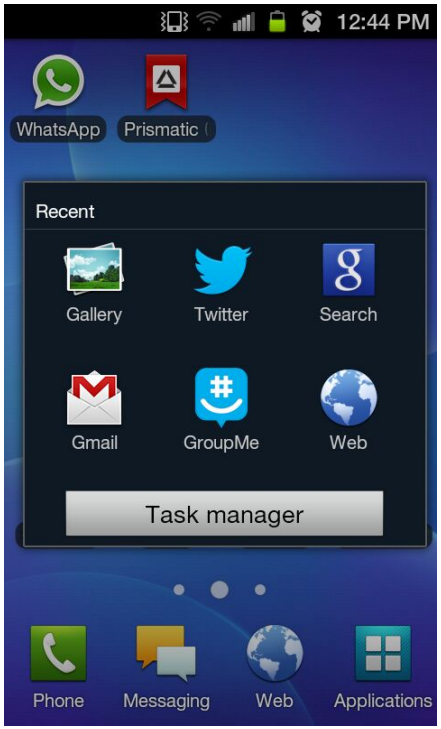
Given a sequence of apps used *recently*, recommend an app to use *next*



Applicable where items are used repeatedly

Which App Will You Use Next?

Given a sequence of apps used *recently*, recommend an app to use *next*



Applicable where items are used repeatedly

Given a user $u \in \mathcal{U}$

with session *in progress* $s = \langle a_{i_1} \rightarrow a_{i_2} \rightarrow \cdots \rightarrow a_{i_t} \rangle$, for some $t \geq 1$

recommend the best candidate item $a_{i_{t+1}} \in \mathcal{I}$

Implicit Feedback via Click Sequences

- Session data examples:

$\langle \text{browser} \rightarrow \text{maps} \rightarrow \text{messages} \rangle$
$\langle \text{mail} \rightarrow \text{phone} \rightarrow \text{settings} \rightarrow \text{phone} \rightarrow \text{browser} \rightarrow \text{phone} \rightarrow \text{browser} \rightarrow \text{search} \rangle$
$\langle \text{phone} \rightarrow \text{calendar} \rightarrow \text{messages} \rightarrow \text{camera} \rangle$
$\langle \text{settings} \rightarrow \text{mail} \rightarrow \text{browser} \rightarrow \text{mail} \rightarrow \text{browser} \rangle$
...

Implicit Feedback via Click Sequences

- Session data examples:

$\langle \text{browser} \rightarrow \text{maps} \rightarrow \text{messages} \rangle$
$\langle \text{mail} \rightarrow \text{phone} \rightarrow \text{settings} \rightarrow \text{phone} \rightarrow \text{browser} \rightarrow \text{phone} \rightarrow \text{browser} \rightarrow \text{search} \rangle$
$\langle \text{phone} \rightarrow \text{calendar} \rightarrow \text{messages} \rightarrow \text{camera} \rangle$
$\langle \text{settings} \rightarrow \text{mail} \rightarrow \text{browser} \rightarrow \text{mail} \rightarrow \text{browser} \rangle$
...

- Simple and effective way to model sequences $\rightarrow$ Markov models
  - State space: set of items $\mathcal{I}$
  - State transition probability matrix $M \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$
    - $M(i, j)$: probability that item j is clicked immediately after item i (first-order Markov model)
    - Estimated from training session data

Markov Modeling

Markov models at different scales:

- Individual-level $M^{(u)}$
  - Estimated from session data of each user u
  - Training data for a given user is too sparse to estimate $|\mathcal{I}| \times |\mathcal{I}|$ parameters

Markov Modeling

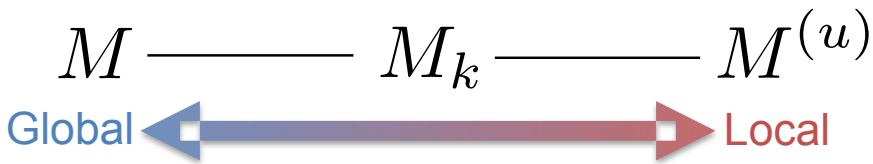
Markov models at different scales:

- Individual-level $M^{(u)}$
  - Estimated from session data of each user u
  - Training data for a given user is too sparse to estimate $|\mathcal{I}| \times |\mathcal{I}|$ parameters
- Cluster-level M_k
  - Each user u is represented by Markov transition probability matrix $M^{(u)}$
  - Group users with common navigational patterns via k-means

Markov Modeling

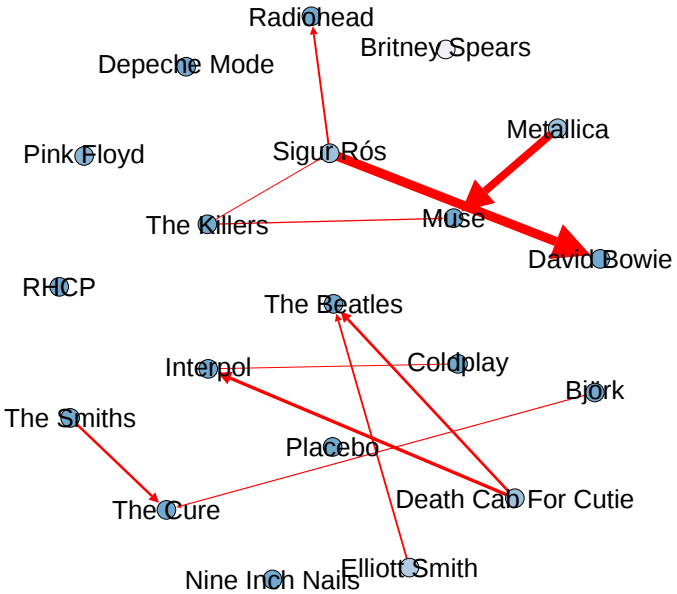
Markov models at different scales:

- Individual-level $M^{(u)}$
  - Estimated from session data of each user u
  - Training data for a given user is too sparse to estimate $|\mathcal{I}| \times |\mathcal{I}|$ parameters
- Cluster-level M_k
  - Each user u is represented by Markov transition probability matrix $M^{(u)}$
  - Group users with common navigational patterns via k-means
- Global-level M
  - Estimated from all session data
  - Hard to capture diverse transition behavior of different users

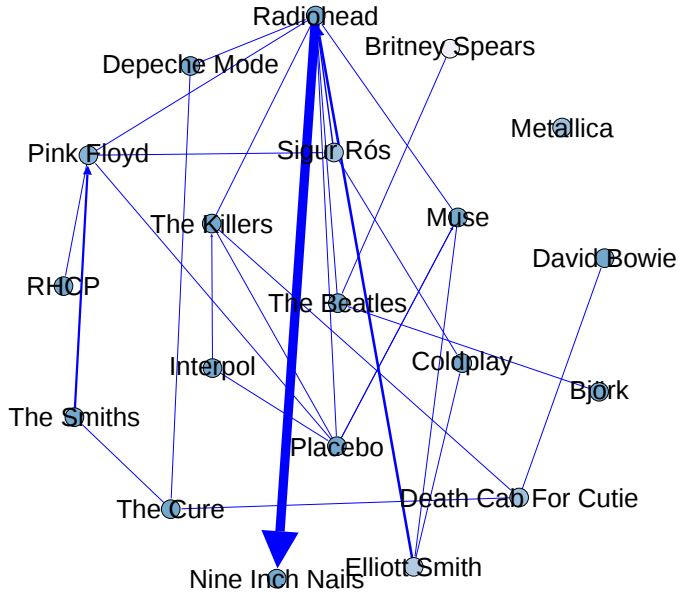


Behavioral Clusters

Transitions between top-20 artists

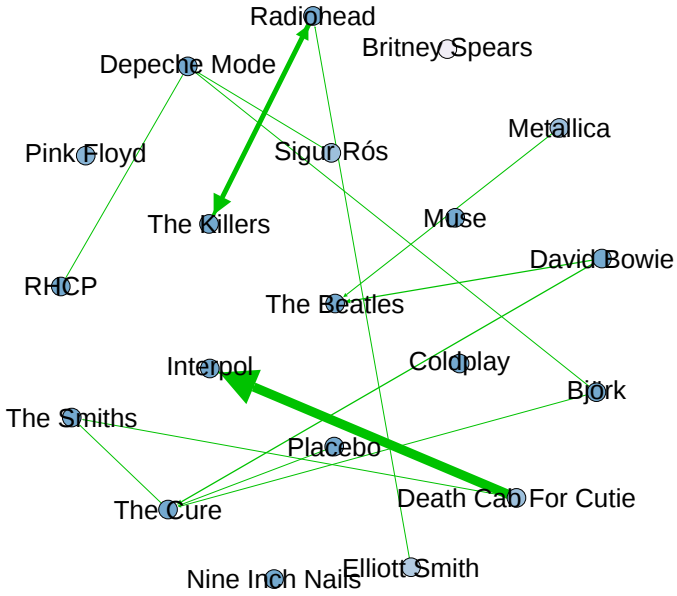


Cluster 1



Cluster 3

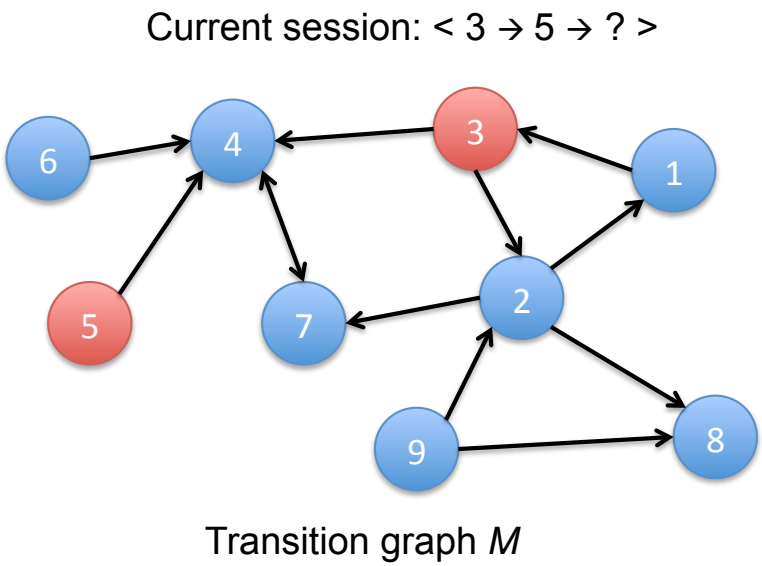
Cluster 2



Transition Behavior at Multiple Scales

Proposed method (iConRank):

- Adapts a neighborhood-based collaborative filtering method — leads to a personalized PageRank formulation
- Utilizes transition patterns at multiple scales

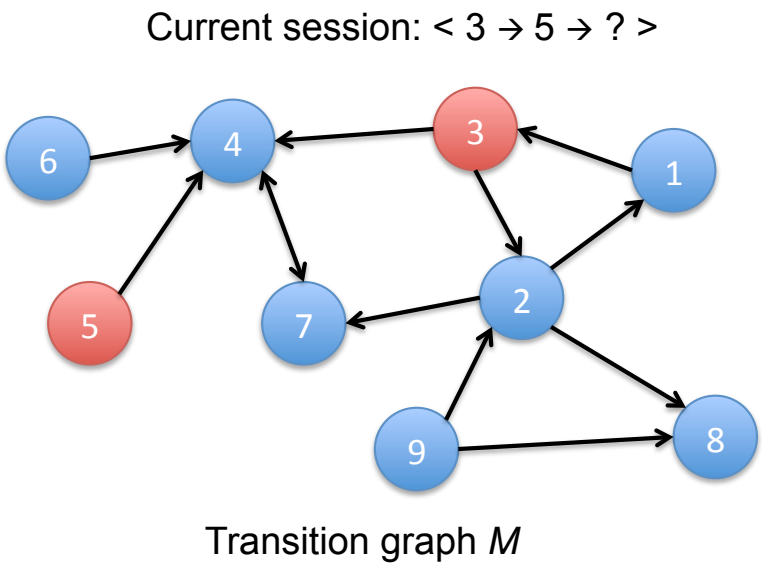


Transition graph *M*

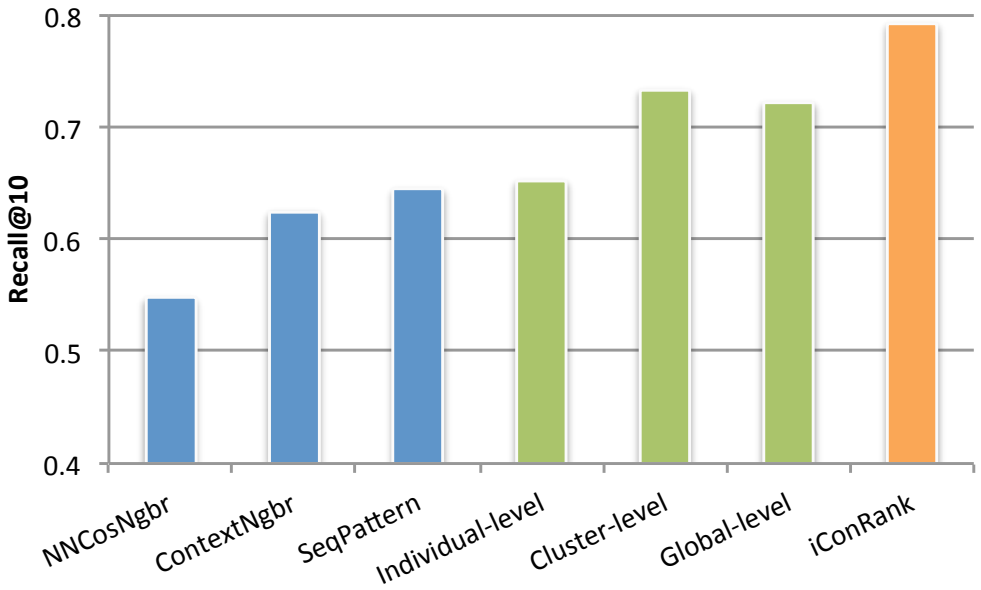
Transition Behavior at Multiple Scales

Proposed method (iConRank):

- Adapts a neighborhood-based collaborative filtering method — leads to a personalized PageRank formulation
- Utilizes transition patterns at multiple scales



Transition graph *M*

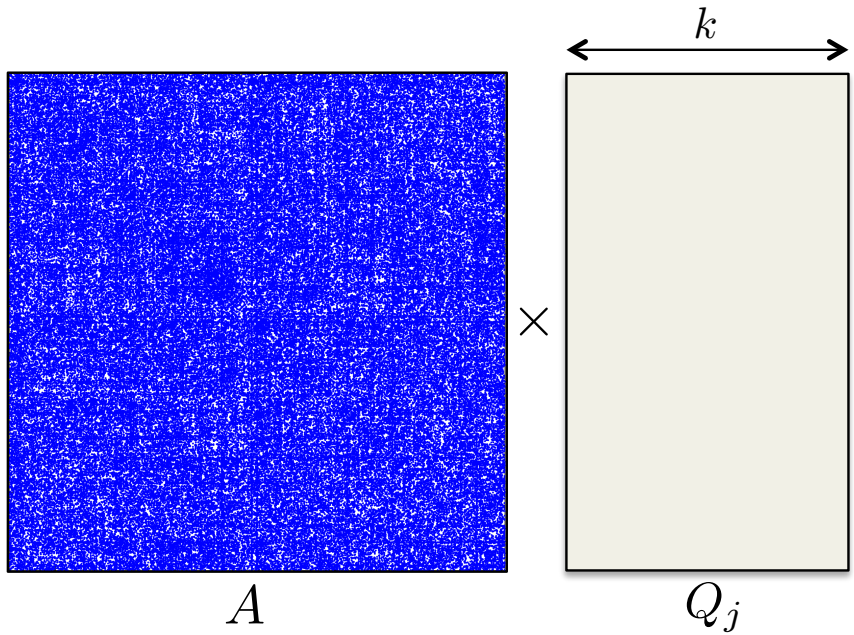


Proposed Work

MSEIGS with Extreme Accuracy

Computing extremely accurate eigenpairs (e.g., up to machine precision):

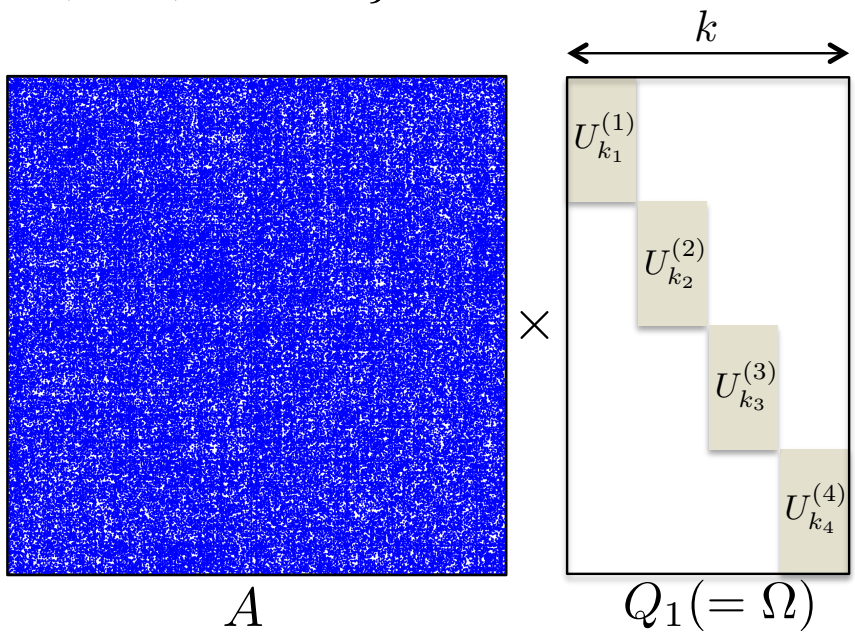
- Block size k of the block Lanczos method $\Rightarrow$ size of Krylov subspace span$\{Q_1, AQ_1, A^2Q_1, \cdots, A^{j-1}Q_1\}$



MSEIGS with Extreme Accuracy

Computing extremely accurate eigenpairs (e.g., up to machine precision):

- Block size k of the block Lanczos method $\Rightarrow$ size of Krylov subspace span$\{Q_1, AQ_1, A^2Q_1, \cdots, A^{j-1}Q_1\}$

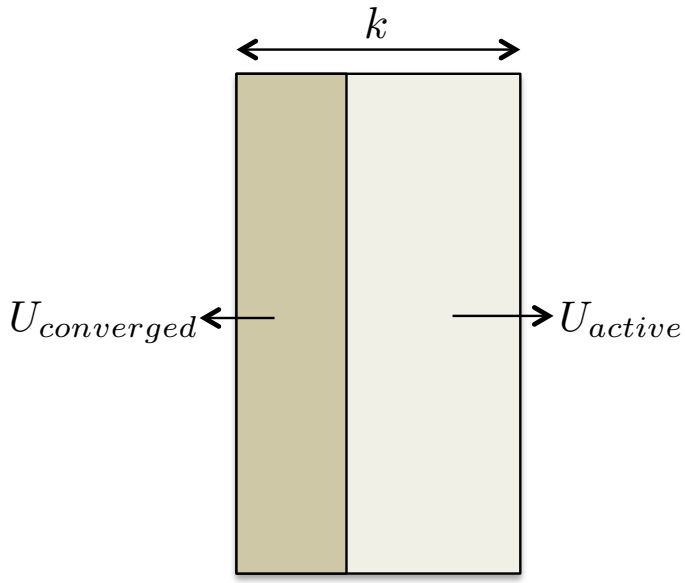


- Smaller block size seems favorable over larger block size in terms of computational cost.
- For a smaller block size, we would have to discard useful information from child clusters.

MSEIGS with Extreme Accuracy

Computing extremely accurate eigenpairs (e.g., up to machine precision):

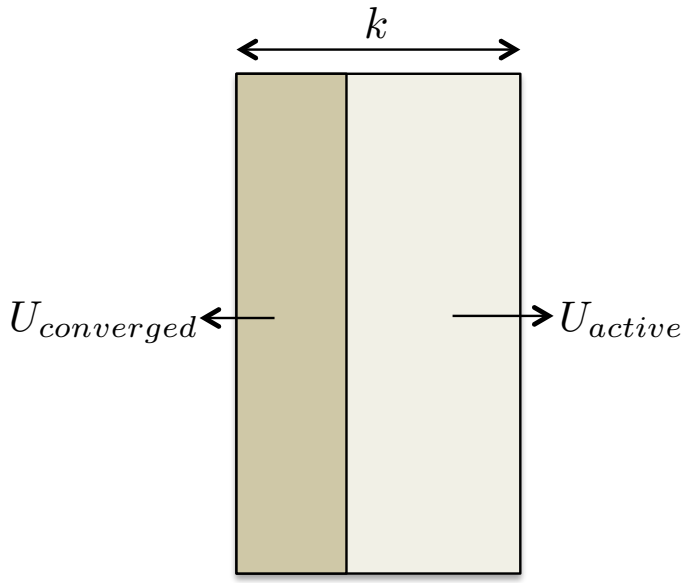
- Restart and deflation: $U = [U_{converged} | U_{active}]$
  - Fix $U_{converged}$ and work on only U_{active}
  - Must orthogonalize U_{active} against $U_{converged}$



MSEIGS with Extreme Accuracy

Computing extremely accurate eigenpairs (e.g., up to machine precision):

- Restart and deflation: $U = [U_{converged} | U_{active}]$
  - Fix $U_{converged}$ and work on only U_{active}
  - Must orthogonalize U_{active} against $U_{converged}$

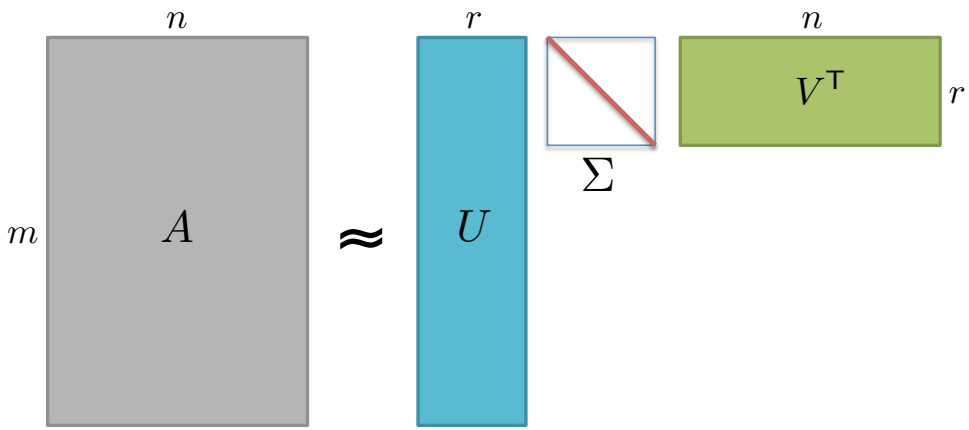


Improve results for *normalized* matrices (e.g., normalized Laplacian matrix):

- Widely used: nonlinear dimensionality reduction, graph-based SSL, etc
- Existing methods suffer slow convergence due to the distribution of eigenvalues

Multi-Scale SVD

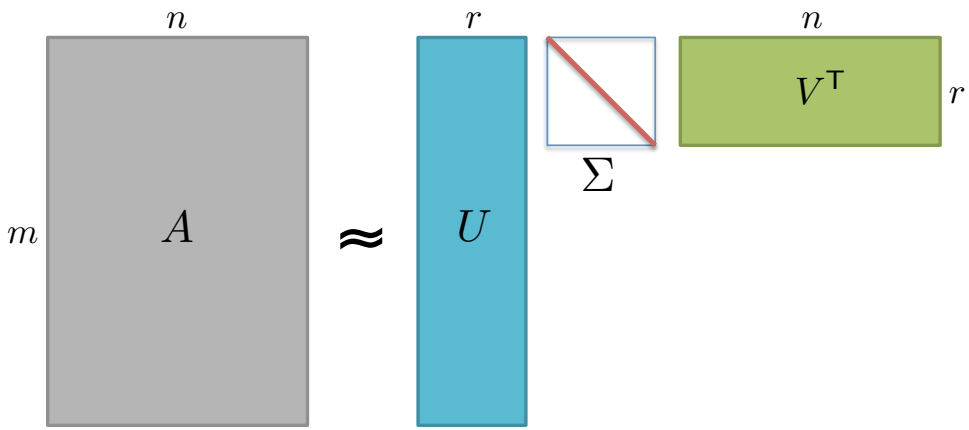
Computing the SVD of $A \in \mathbb{R}^{m \times n}$:



Generally, algorithms for SVD are analogs of symmetric eigensolvers

Multi-Scale SVD

Computing the SVD of $A \in \mathbb{R}^{m \times n}$:

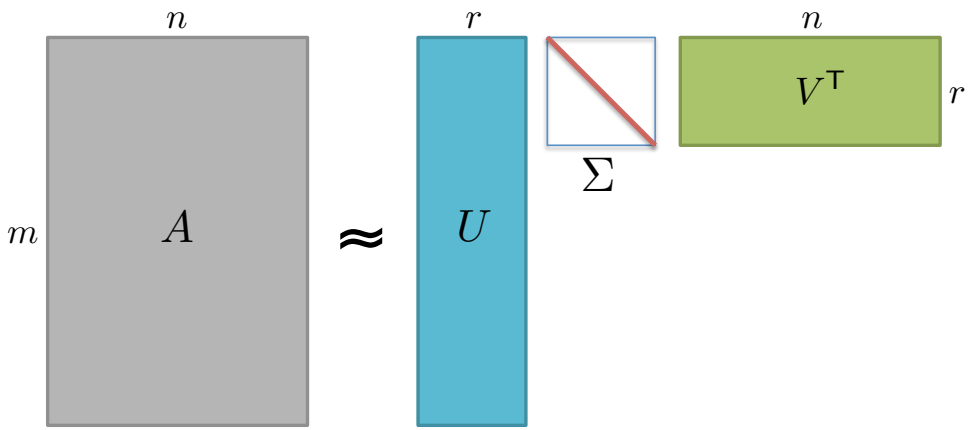


Generally, algorithms for SVD are analogs of symmetric eigensolvers

- Method 1: $C = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} = \begin{bmatrix} U & U \\ V & -V \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix} \begin{bmatrix} U & U \\ V & -V \end{bmatrix}^T$
  - Doubles the size of the matrix that needs to be clustered
  - Vectors have length $m + n \Rightarrow$ waste of memory and unnecessary work
  - Numerically stable (Matlab's `svds`)

Multi-Scale SVD

Computing the SVD of $A \in \mathbb{R}^{m \times n}$:



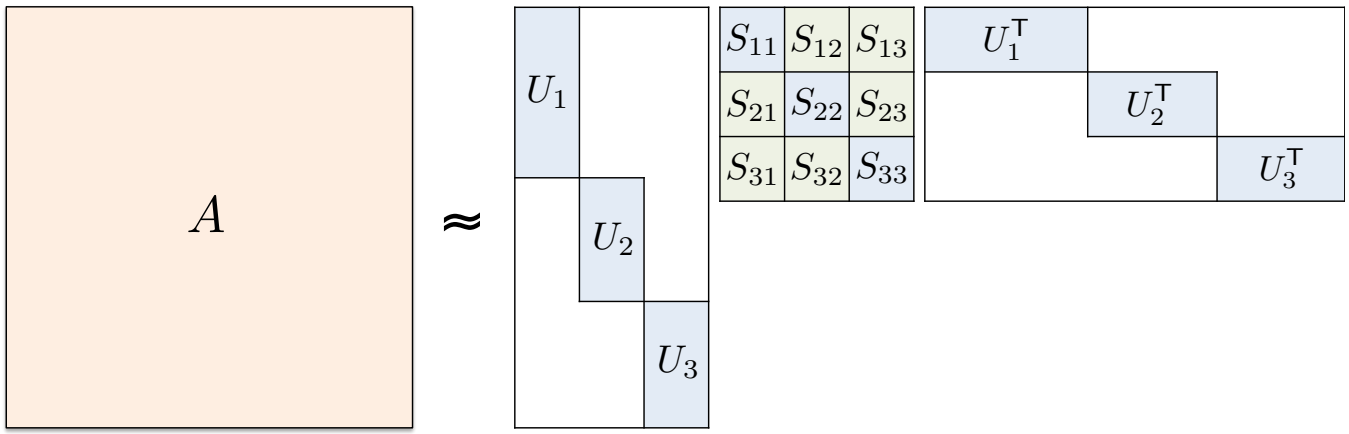
Generally, algorithms for SVD are analogs of symmetric eigensolvers

- Method 2: $A^T A = V \Sigma^2 V^T$
  - Severe loss of accuracy of smaller singular values if A is ill-conditioned
    $\Rightarrow$ not a problem with the dominant singular values
  - Fast when $n \ll m$, since only vectors of length-n need to be stored

Combining Predictions at Multiple Scales

Better way to combine predictions at multiple scales:

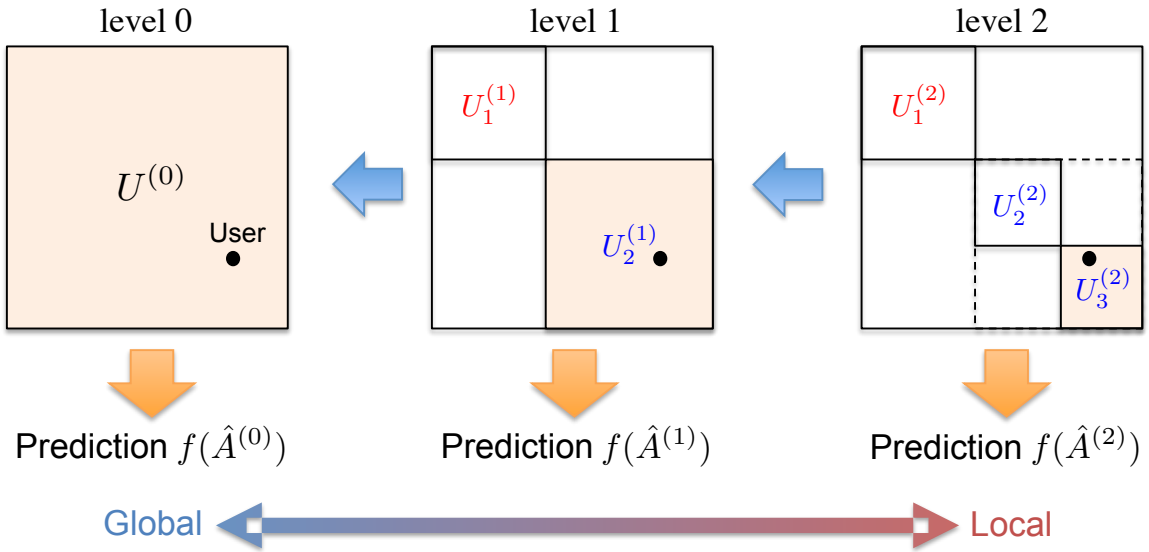
- In MSLP, we use CLRA that requires computation of off-diagonal blocks — time consuming as the rank k or number of cluster c increases ($S \in \mathbb{R}^{ck \times ck}$).



Combining Predictions at Multiple Scales

Better way to combine predictions at multiple scales:

- In MSLP, we use CLRA that requires computation of off-diagonal blocks — time consuming as the rank k or number of cluster c increases ($S \in \mathbb{R}^{ck \times ck}$).
- A more intuitive approach would be to make predictions for a given user with only clusters that he or she belongs to.

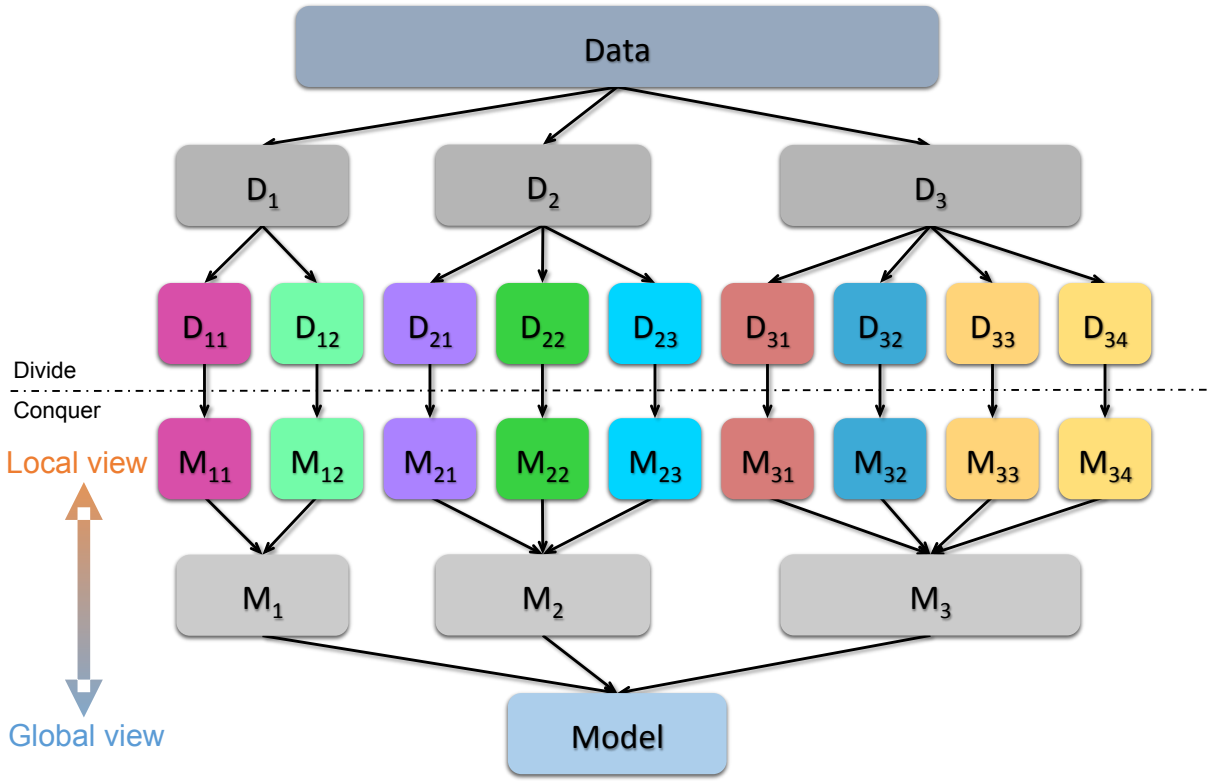


- Each level is likely to have different contributions to the final prediction.

Applications for Multi-Scale Framework

Exploring other problems that can benefit from our multi-scale framework:

- Graph-based semi-supervised classification
- Influence maximization
- Matrix completion, inductive matrix completion

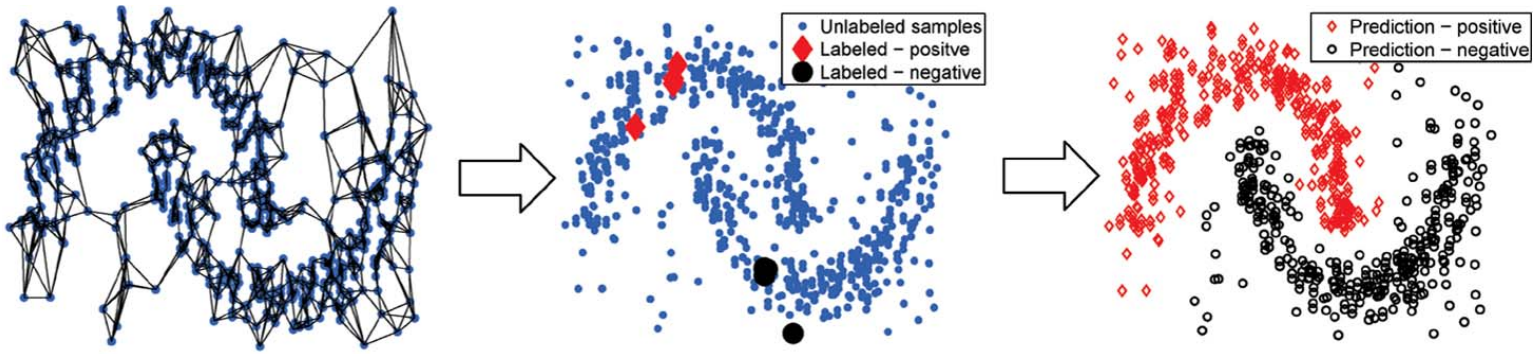


Label Propagation for Semi-supervised Learning

Label Propagation: $F^{(t+1)} = \alpha A F^{(t)} + (1-\alpha) Y$

$$F^*(A) = (1-\alpha)(I - \alpha A)^{-1} Y$$
$$\approx (1-\alpha) U_k (I - \alpha \Lambda_k)^{-1} U_k^{\mathsf{T}} Y$$

$A \approx U_k \Lambda_k U_k^{\mathsf{T}}$: $n \times n$ affinity matrix, Y: $n \times \ell$ initial label matrix, $0 \leq \alpha \leq 1$

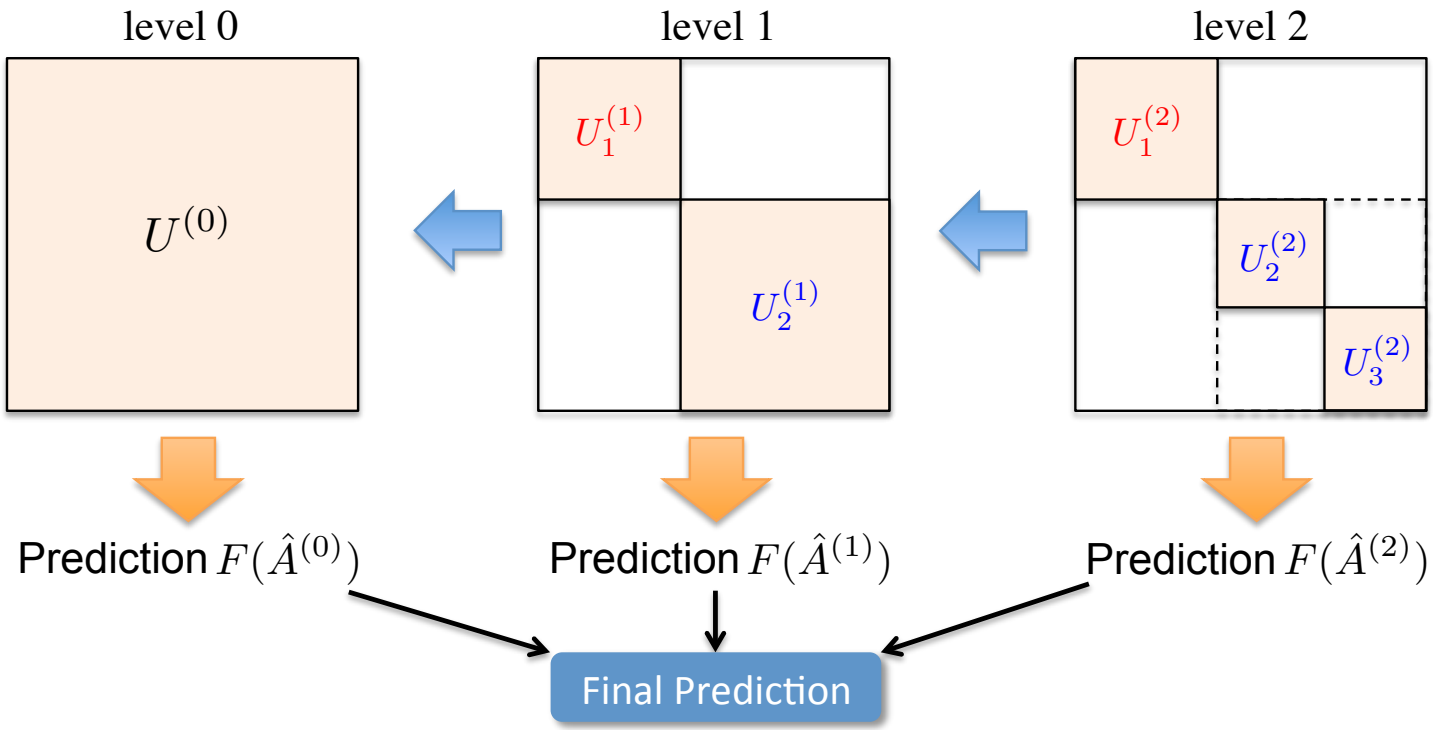


Label Propagation for Semi-supervised Learning

Label Propagation: $F^{(t+1)} = \alpha A F^{(t)} + (1-\alpha) Y$

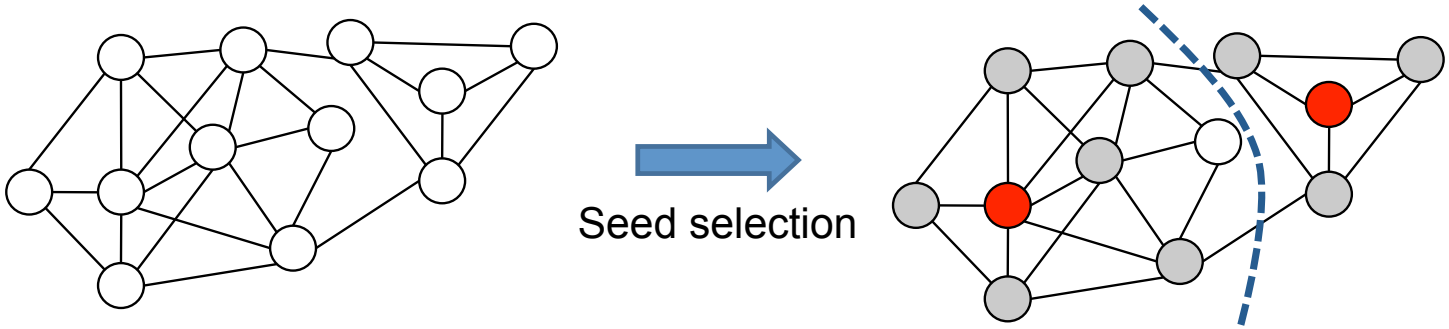
$$F^*(A) = (1-\alpha)(I - \alpha A)^{-1} Y$$
$$\approx (1-\alpha) U_k (I - \alpha \Lambda_k)^{-1} U_k^{\mathsf{T}} Y$$

$A \approx U_k \Lambda_k U_k^{\mathsf{T}}$: $n \times n$ affinity matrix, Y: $n \times \ell$ initial label matrix, $0 \leq \alpha \leq 1$



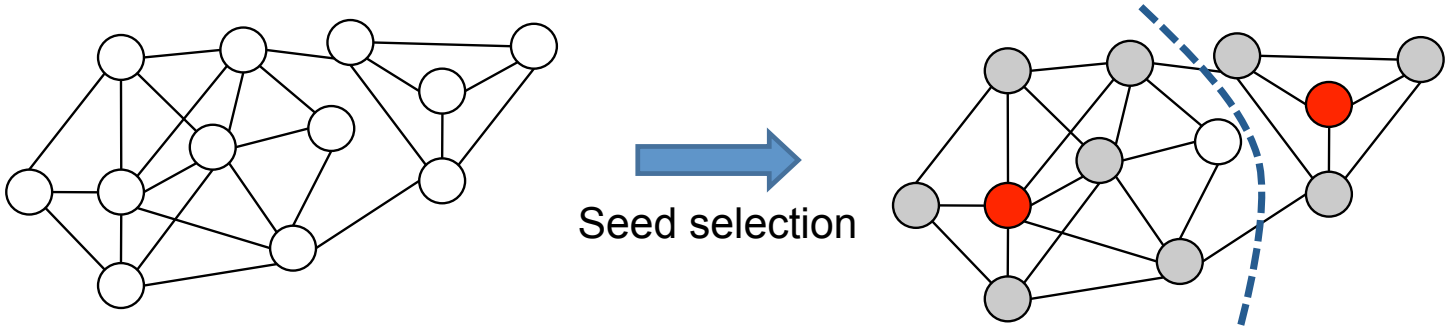
Influence Maximization

Goal: find b nodes (called seeds) such that the expected number of nodes activated by the seeds is the largest possible [Kempe, et.al. 2003]



Influence Maximization

Goal: find b nodes (called seeds) such that the expected number of nodes activated by the seeds is the largest possible [Kempe, et.al. 2003]

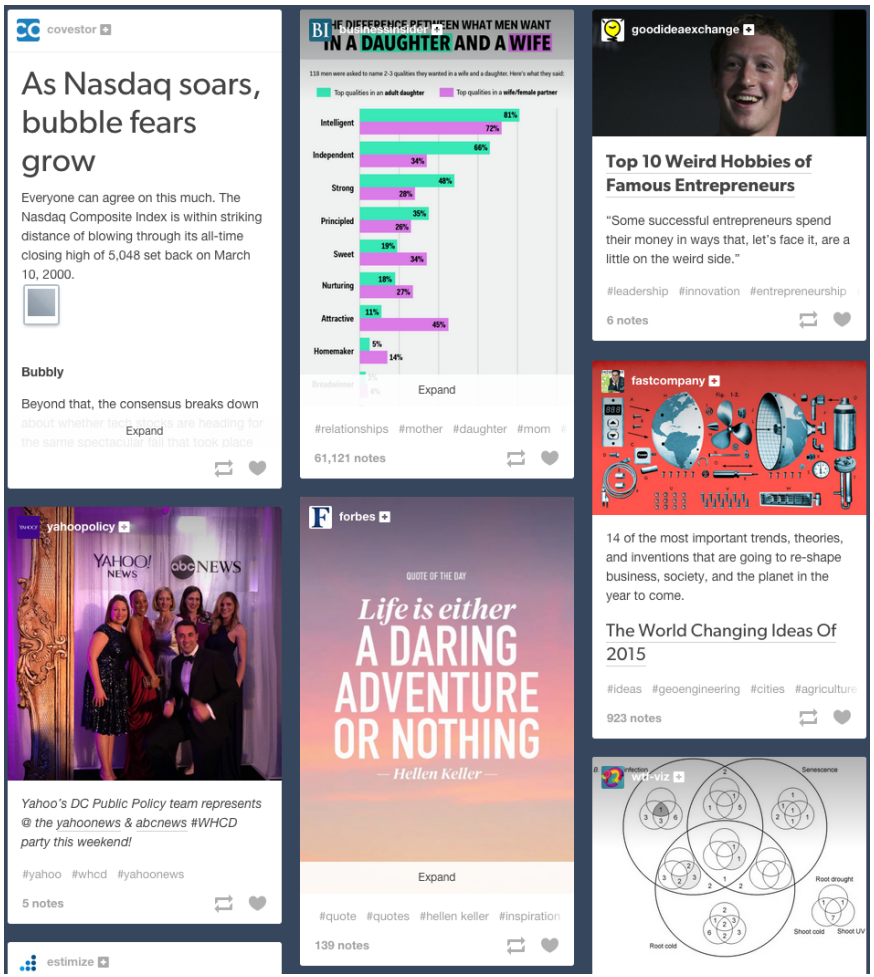


- Community-based methods have been successful [Wang, et.al. 2010, Chen, et.al. 2014]
- Multi-scale approach: use local seeds from lower levels to identify global seeds for the entire graph

Recommender Systems with Side Information

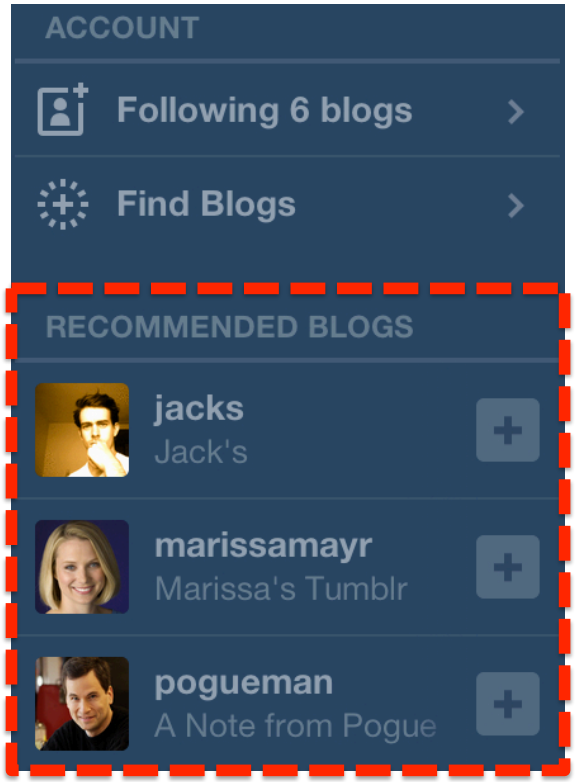
Tumblr — microblogging service:

- Blog consists of posts



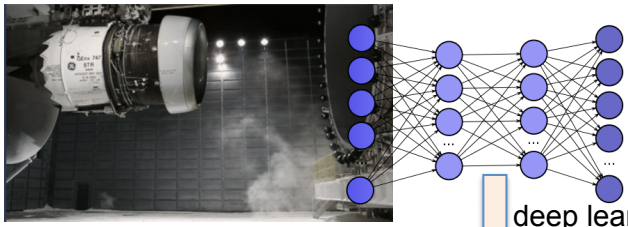
Blog recommendation:

- Who to follow? (230M blogs)



Inductive Matrix Completion

[Jain and Dhillon 2013]



Photo

deep learning

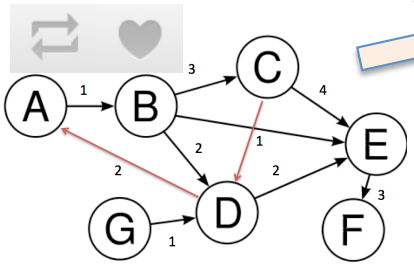
Text and Tags

Steam rises from the heated pad below a GEnx engine, following a nighttime ice test at GE Aviation in Winnipeg, Canada. Temperatures were around -20 degrees F.

#GIF #BLACK AND WHITE #GE

word vec

Like and Reblog Graphs



latent

Blogs

Users

$$A_{ij} = \mathbf{x}_i^\top Z \mathbf{y}_j$$

$$\approx$$

A X W H^T Y^T

$$\min_{W,H} \sum_{(i,j)\in\Omega} (A_{ij} - \mathbf{x}_i^T W H^T \mathbf{y}_j)^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2)$$

Experimental Results

- 1M randomly sampled users and blogs

Method	PRC@10	RCL@10	AUC
Global	1.03%	4.80%	0.8687
SVD	1.28%	5.10%	0.8530
MC	1.28%	5.07%	0.8515
Katz	1.90%	8.15%	0.9209
CMC	0.49%	2.41%	0.8996
IMC	2.93%	11.33%	0.9075

D. Shin, S. Cetintas and K. Lee. *Recommending Tumblr Blogs to Follow with Inductive Matrix Completion*. RecSys'14.

Inductive Matrix Completion

Adapting multi-scale framework:

$$A = D + \Delta, \quad D = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{cc} \end{bmatrix}, \quad \Delta = \begin{bmatrix} 0 & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & 0 \end{bmatrix}$$

$$D \approx \begin{bmatrix} X_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & X_c \end{bmatrix} \begin{bmatrix} W_1 H_1^{\mathsf{T}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W_c H_c^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} Y_1^{\mathsf{T}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Y_c^{\mathsf{T}} \end{bmatrix}$$

- How to cluster A?
- How are W_i and H_i related between different levels in the hierarchy?
- How to deal with off-diagonal blocks in Δ?

Thank You!

References

[1] D. Shin, S. Si and I. S. Dhillon *Multi-Scale Link Prediction*. CIKM. 2012.

[2] N. Natarajan, D. Shin and I. S. Dhillon. *Which App Will You Use Next? Collaborative Filtering with Interactional Context*. RecSys. 2013.

[3] S. Si, D. Shin, I. S. Dhillon and B. N. Parlett. *Multi-Scale Spectral Decomposition of Massive Graphs*. NIPS. 2014.

[4] D. Shin, S. Cetintas and K. Lee *Recommending Tumblr Blogs to Follow with Inductive Matrix Completion*. RecSys. 2014.

References

[5] D. Kempe, J. Kleinberg and E. Tardos *Maximizing the Spread of Influence through a Social Network*. KDD. 2003.

[6] B. Savas and I. S. Dhillon. *Clustered Low Rank Approximation of Graphs in Information Science Applications*. SDM. 2011.

[7] J. Whang, X. Sui and I. S. Dhillon *Scalable and Memory-Efficient Clustering of Large-Scale Social Networks*. ICDM. 2012.

[8] W. Liu, J. Wang and S. Chang *Robust and Scalable Graph-Based Semi-supervised Learning*. Proc. of the IEEE. 2012.

[9] P. Jain and I. S. Dhillon. *Provable Inductive Matrix Completion*. CoRR. 2013.

[10] N. Natarajan and I. S. Dhillon. *Inductive Matrix Completion for Predicting Gene-disease Associations*. Bioinformatics. 2014.

[11] D. LaSalle and G. Karypis. *Multi-threaded Modularity based Graph Clustering using the Multilevel Paradigm*. J. Parallel Dist. Comp. 2014.

[12] Y. Chen, W. Zhu, W. Peng, W. Lee and S. Lee. *Community-Based Influence Maximization in Social Networks*. ACM TIST. 2014.

Backup Slides

Early-Termination Strategy

- Computing the exact spectral decomposition of A can be quite time consuming.
- Highly accurate eigenpairs are not essential for many applications.

Fast early termination strategy (MSEIGS-Early):

- Approximate the eigenpairs of A by terminating MSEIGS at a certain level ℓ.
- From the top-r eigenpairs of each cluster, select the top-k eigenpairs from all c_ℓ clusters as an approximation of A.

Label Propagation in Semi-supervised Learning

Label Propagation: $F(t+1) = \alpha SF(t) + (1-\alpha)Y$

$$F^* = (1-\alpha)(I - \alpha S)^{-1} Y$$
$$\approx (1-\alpha) U_k (I - \alpha \Lambda_k)^{-1} U_k^{\mathsf{T}} Y$$

where S is the normalized affinity matrix; Y is the $n \times \ell$ initial label matrix.

Method	Aloi ($k = 1500$)		Delicious ($k = 1000$)		
	time(sec)	acc(%)	time(sec)	top3-acc(%)	top1-acc(%)
Truncated	1824.8	59.87	3385.1	45.12	48.89
CG	2921.6	60.01	1094.9	44.93	48.73
EIGS	3890.9	60.08	458.2	45.11	48.51
RSVD	964.1	59.62	359.8	44.11	46.91
BlkLan	1272.2	59.96	395.6	43.52	45.53
MSEIGS	767.1	60.03	235.6	44.84	49.23
MSEIGS-Early	176.2	58.98	61.36	44.71	48.22

Recommender Systems with Side-Information

Inductive Matrix Completion (IMC): given a user-item matrix R

$$\min_{W \in \mathbb{R}^{f_c \times r}, H \in \mathbb{R}^{f_d \times r}} \sum_{(i,j) \in \Omega} (R_{ij} - \mathbf{x}_i^{\mathsf{T}} W H^{\mathsf{T}} \mathbf{y}_j)^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2),$$

where $\mathbf{x}_i$ and $\mathbf{y}_j$ are user and item features, respectively.
Principal components of a social network are used as user features.

Method	Flixster ($k = 100$)		Amazon ($k = 500$)		LiveJournal ($k = 500$)	
	eig-time	RCL@20	eig-time	RCL@20	eig-time	RCL@20
Katz	-	0.1119	-	0.3224	-	0.2838
MC	-	0.0820	-	0.4497	-	0.2699
EIGS	120.51	0.1472	871.30	0.4999	12099.57	0.4259
RSVD	85.31	0.1491	369.82	0.4875	7617.98	0.4294
BlkLan	104.95	0.1465	882.58	0.4687	5099.79	0.4248
MSEIGS	36.27	0.1489	264.47	0.4911	2863.55	0.4253
MSEIGS-Early	21.88	0.1481	179.04	0.4644	1545.52	0.4246

Experiment Results

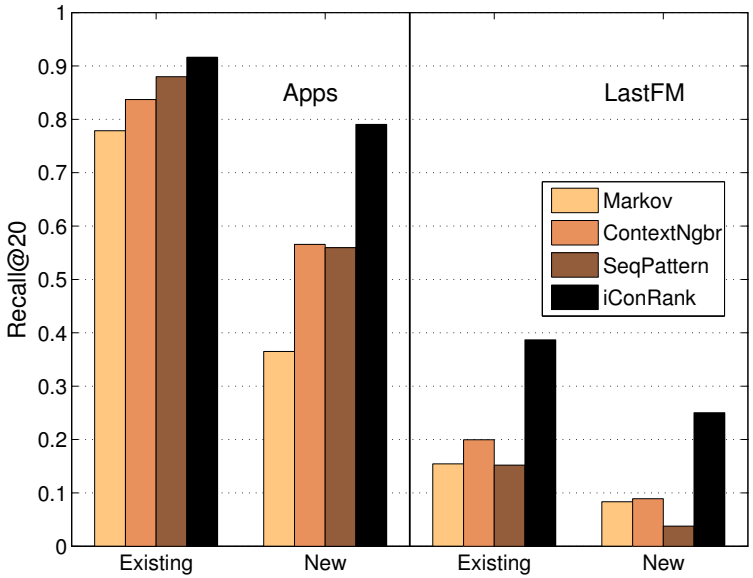
- Apps

Method	Recall@N			
	$N = 5$	$N = 10$	$N = 15$	$N = 20$
NNCosNgbr	0.4301	0.5478	0.6167	0.6636
SVD	0.4574	0.5853	0.6480	0.6851
Markov	0.4592	0.5744	0.6370	0.6754
ContextNgbr	0.5266	0.6248	0.6739	0.7045
SeqPattern	0.5517	0.6451	0.6899	0.7223
iConRank	**0.6701**	**0.7927**	**0.8386**	**0.8632**

- LastFM

Method	Recall@N			
	$N = 5$	$N = 10$	$N = 15$	$N = 20$
NNCosNgbr	0.0691	0.1044	0.1328	0.1560
SVD	0.0810	0.1286	0.1633	0.1922
Markov	0.0631	0.0905	0.1113	0.1285
ContextNgbr	0.0597	0.0775	0.0884	0.0971
SeqPattern	0.0371	0.0536	0.0656	0.0748
iConRank	**0.1277**	**0.1882**	**0.2304**	**0.2633**

- Existing and new items



- Session length