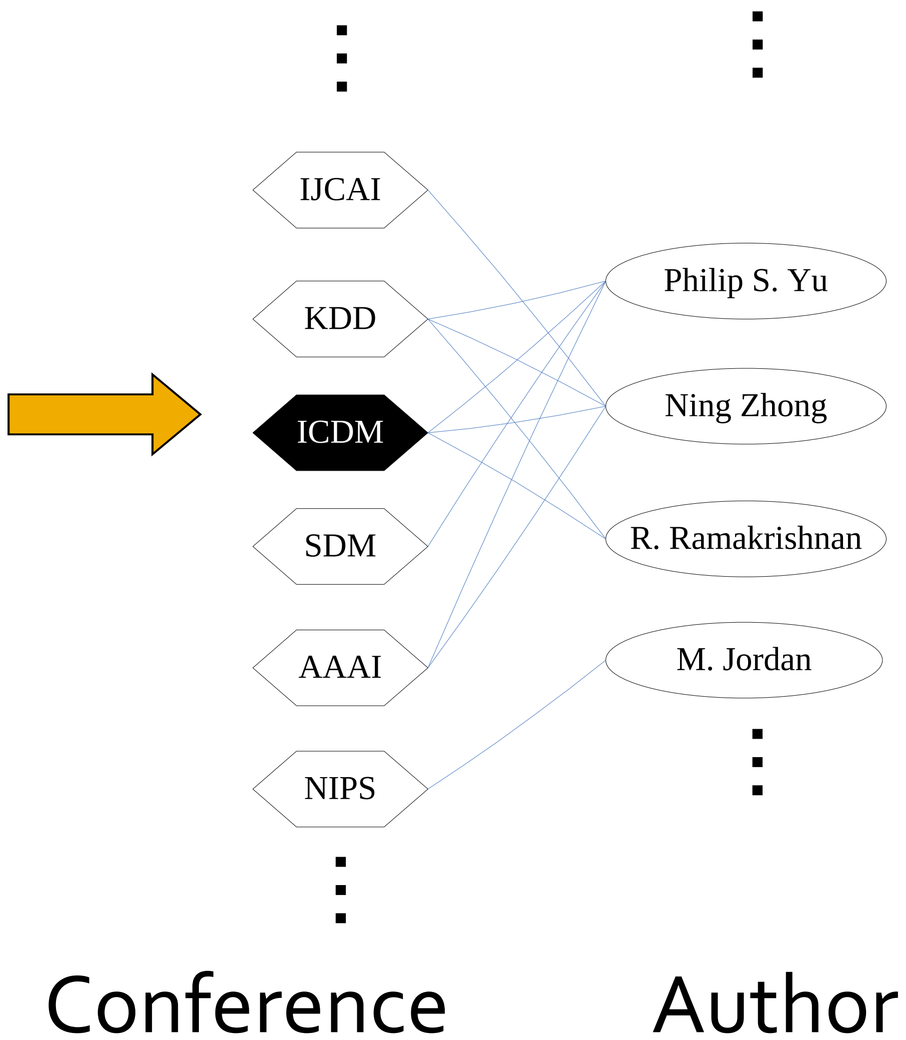
Proximity in Graphs by using Random Walks

CS345a: Data Mining
Jure Leskovec and Anand Rajaraman
Stanford University

Neighborhood Formulation

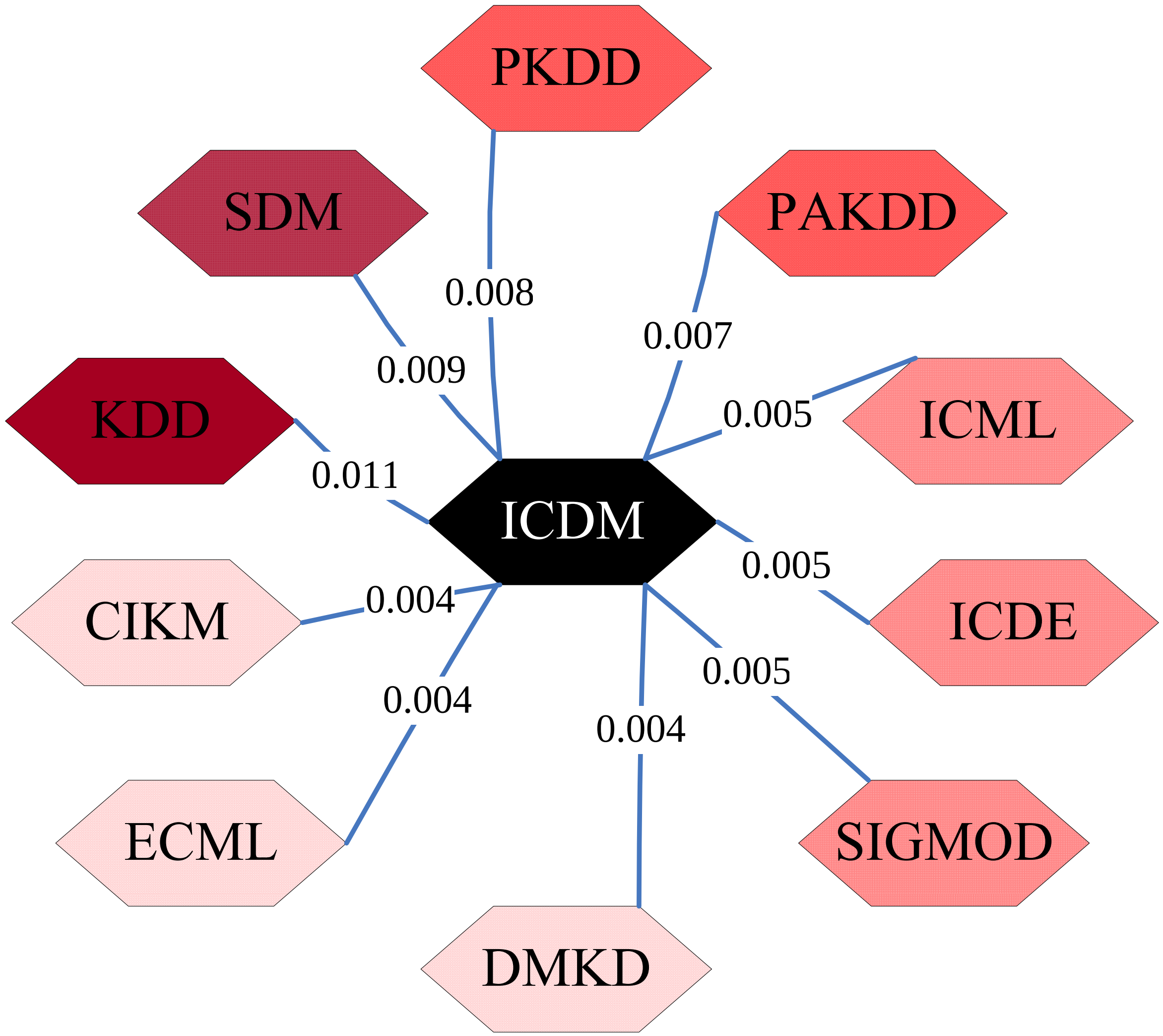


Q: what is most related conference to ICDM

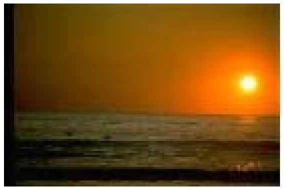
A: Random walks!

[Sun, ICDM2005]

Neighborhood Search on Graphs



Automatic Image Captioning



{Sea Sun Sky Wave}

...

{Cat Forest Grass Tiger}

?

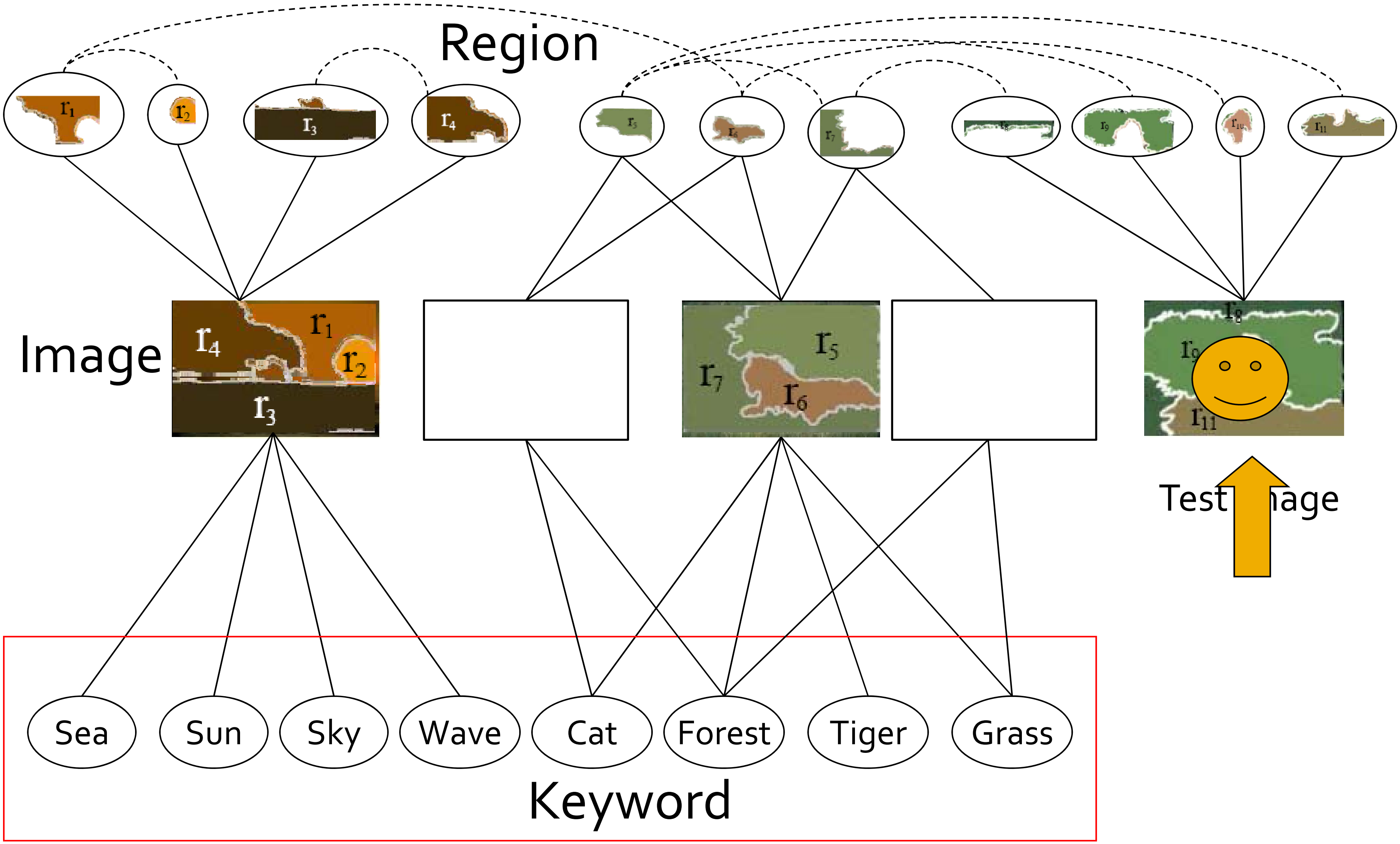
{?, ?, ?,}

A: Proximity on grpahs!

[Pan KDD2004]

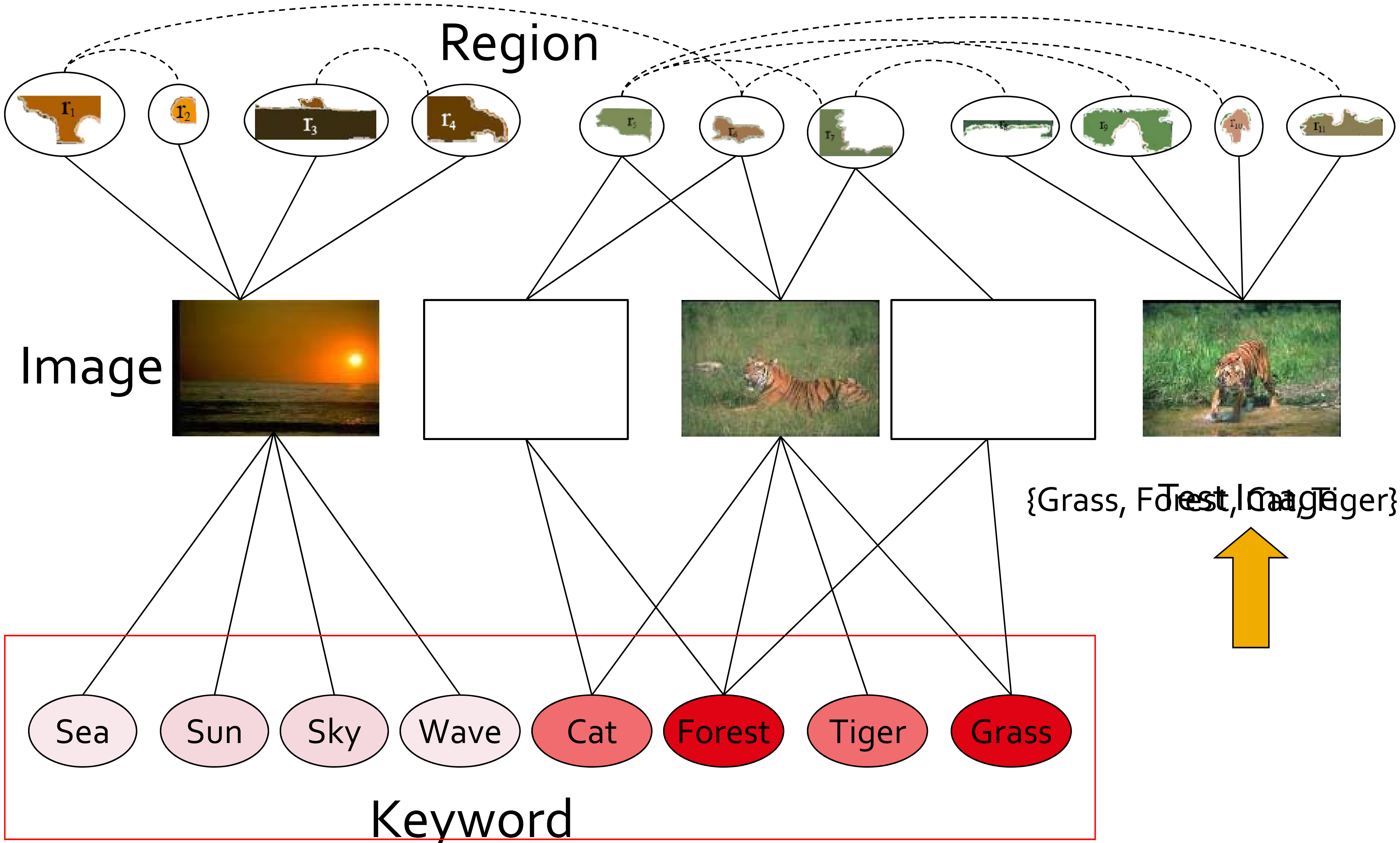
Automatic Image Captioning

[Pan, KDD '04]



[Pan, KDD '04]

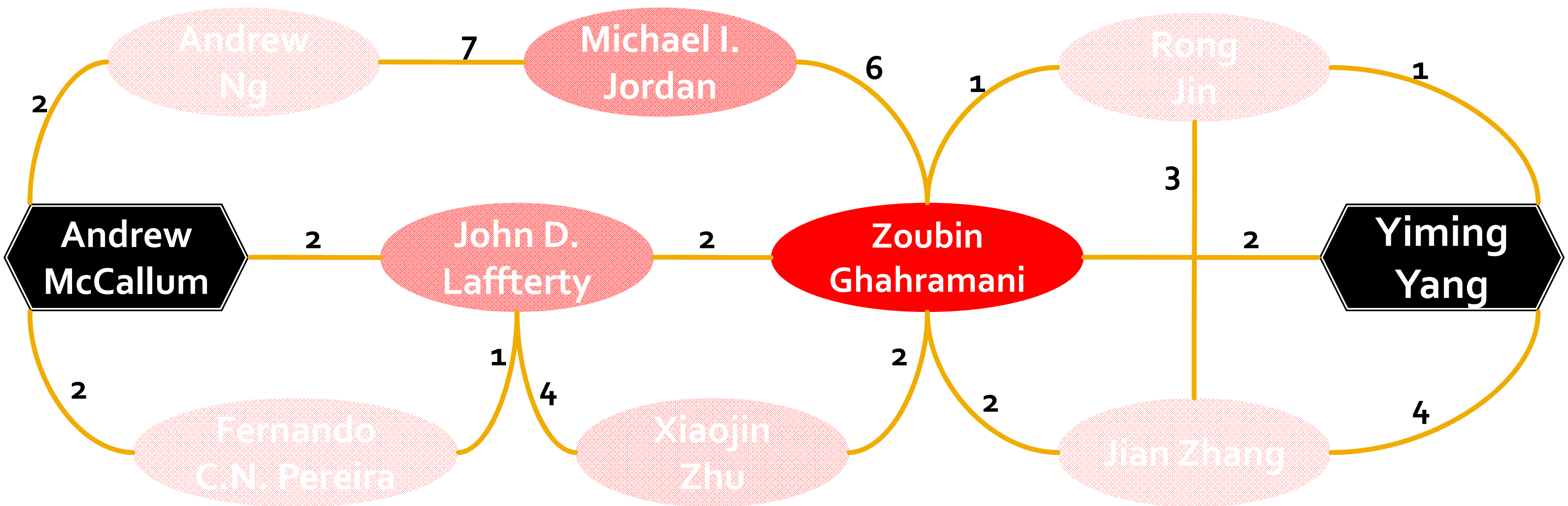
Automatic Image Captioning



[Tong-Faloutsos, '06]

Center-Piece Subgraphs

- Connection subgraphs:
  - What is the most likely connection between Andrew McCallum and Yiming Yang:

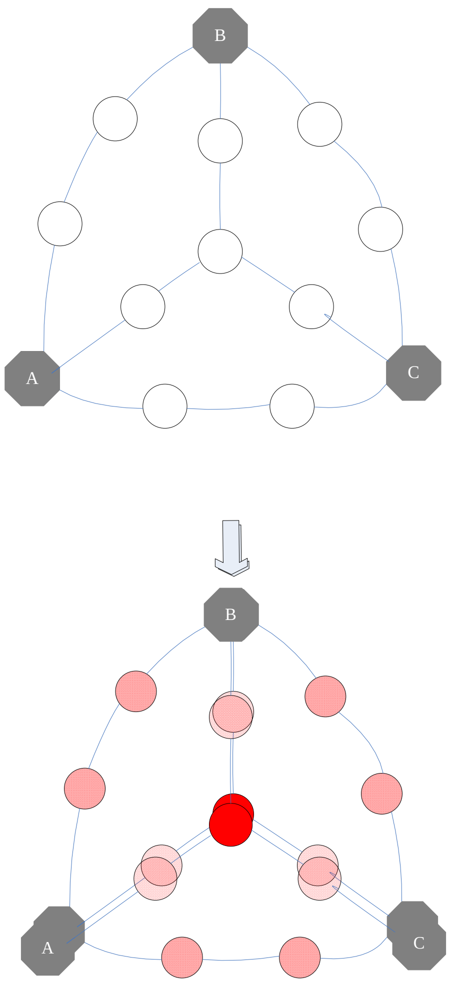


[Tong-Faloutsos, '06]

Center-Piece Subgraph (CEPS)

- Given Q query nodes
- Find Center-piece subgraph on *b* nodes

- Input of CEPS:
  - Q Query nodes
  - Budget b
  - k-softAND coefficient



[Tong-Faloutsos, ‘06]

CEPS: 3 steps

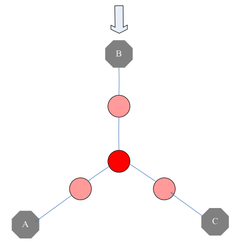
- Individual Score Calculation:
  - Measure proximity of each node with respect to individual query node

$$r(i, j) = \begin{pmatrix} \\ n \times Q \\ \\ \end{pmatrix}$$

- Combine Individual Scores:
  - Measure importance of a node to the whole query set

$$r(Q, j) = \begin{pmatrix} \\ n \times 1 \\ \\ \end{pmatrix}$$

- “Extract” the connection subgraph: $\arg\max_H \, g(H)$



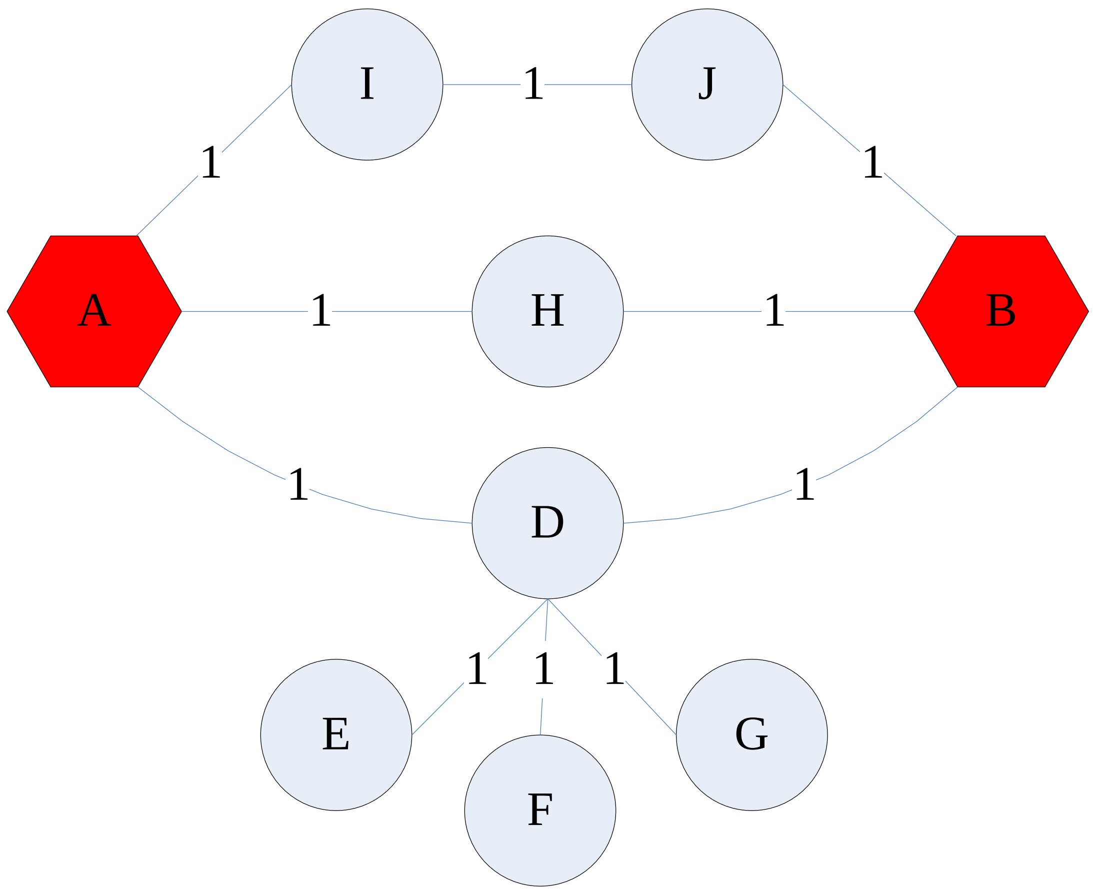
Node Score Calculation

[Tong-Faloutsos, '06]

- Goal:
  - Calculate importance score $r(i,j) = r_{i,j}$
  - for each node j and each query node i
- **How to do that?**

[Tong-Faloutsos, '06]

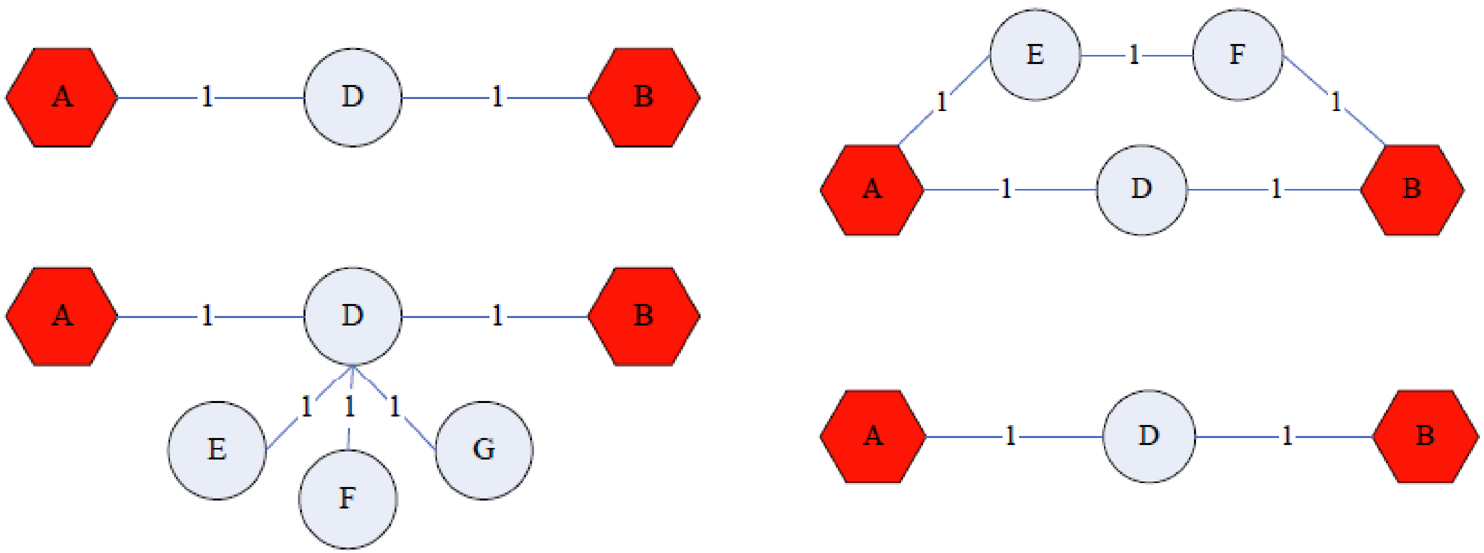
Proximity on Graphs



a.k.a.: Relevance, Closeness, 'Similarity'…

Good proximity measure?

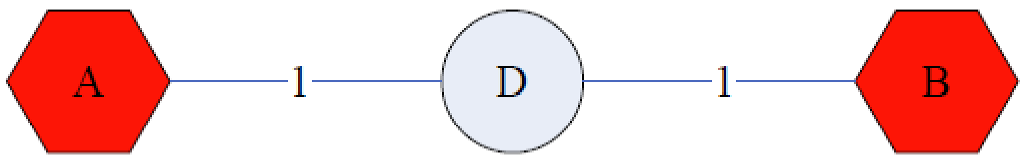
- Shortest path is not good:

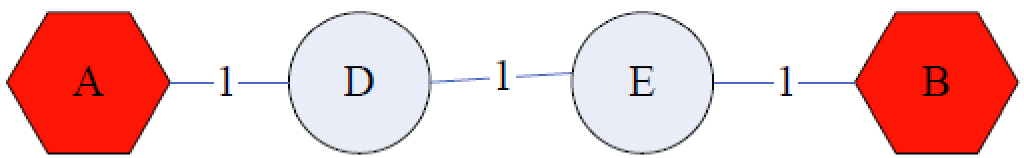


- No influence for degree-1 nodes (E, F)!
  - known as 'pizza delivery guy' problem in undirected graph
- Multi-faceted relationships

Good proximity measure?

- Max-flow is not good:

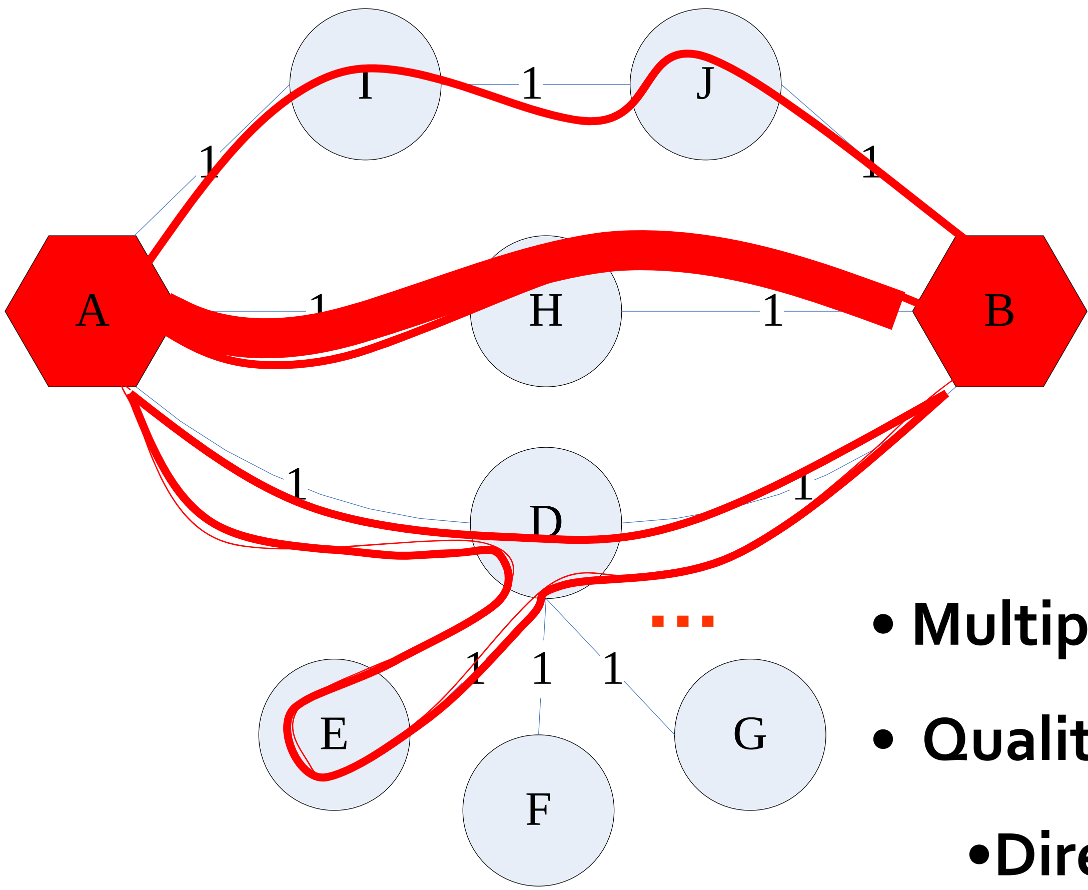




- Does not punish long paths

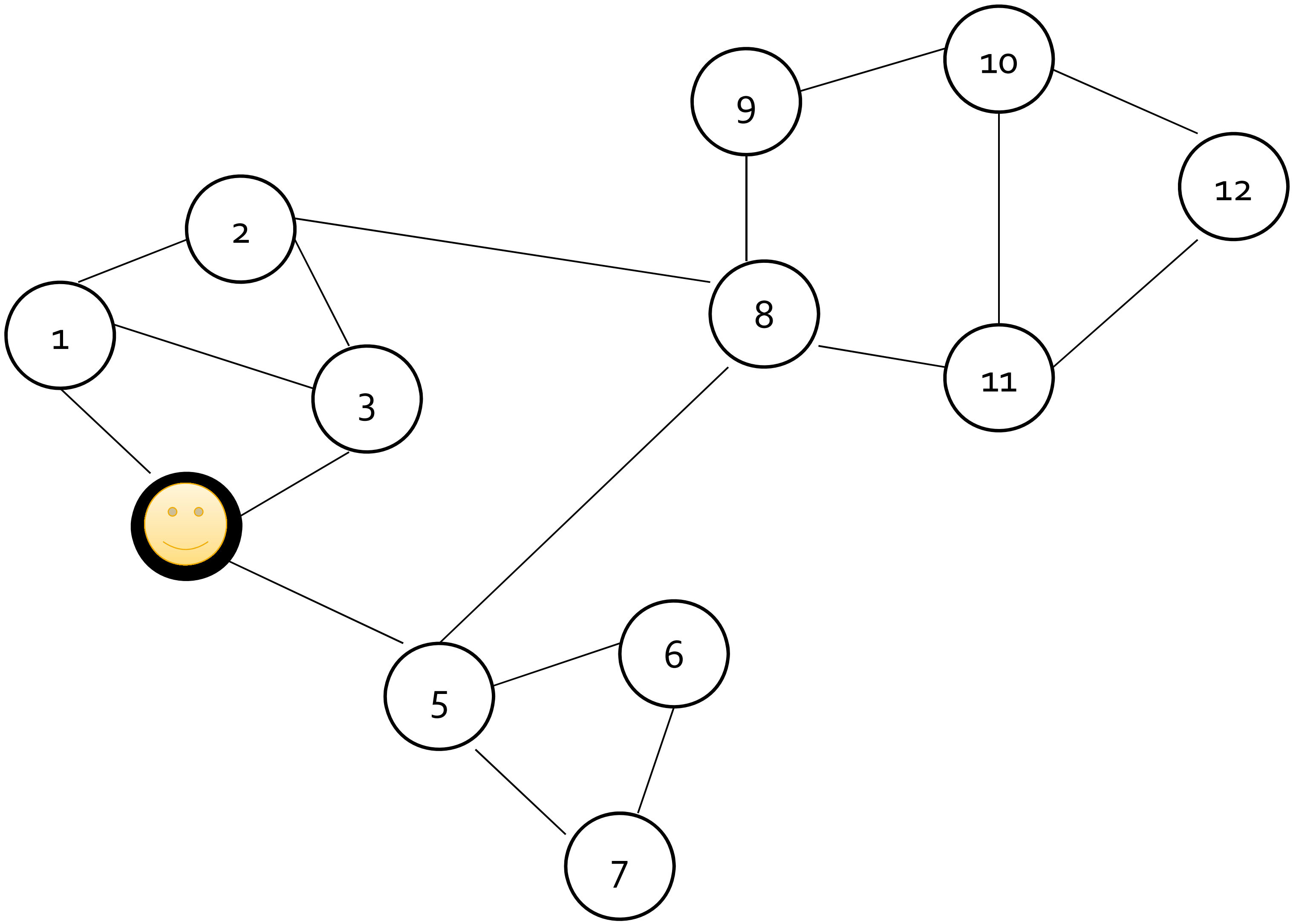
[Tong-Faloutsos, ‘06]

What is good notion of proximity?

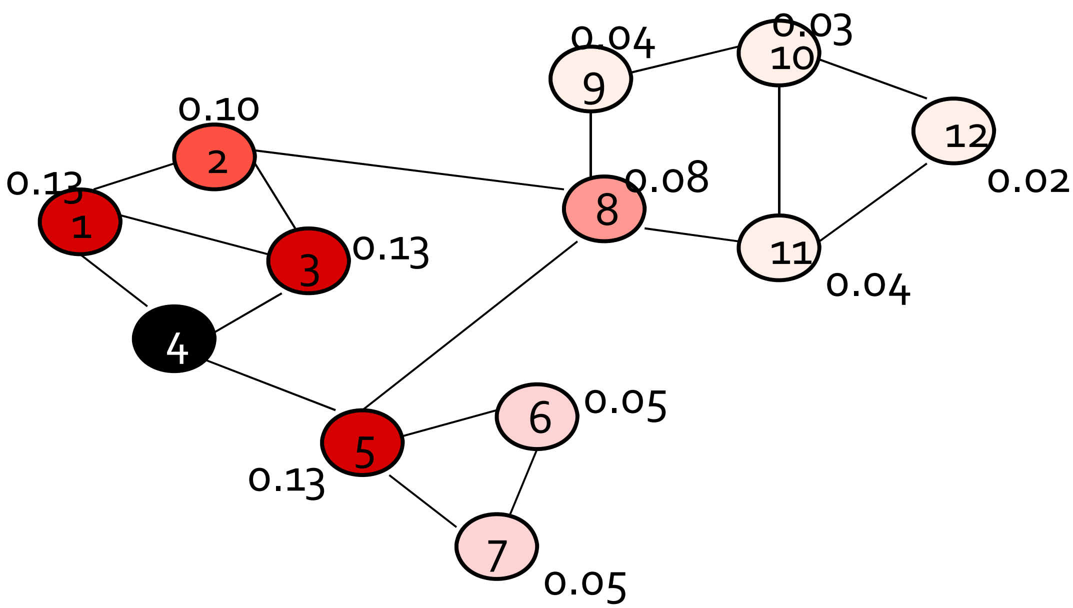


- Multiple Connections
- Quality of connection
  - Direct & In-direct connections
  - Length, Degree, Weight…

Random Walk with Restarts



Random Walks with Restarts



	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	0.22
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

Nearby nodes, higher scores

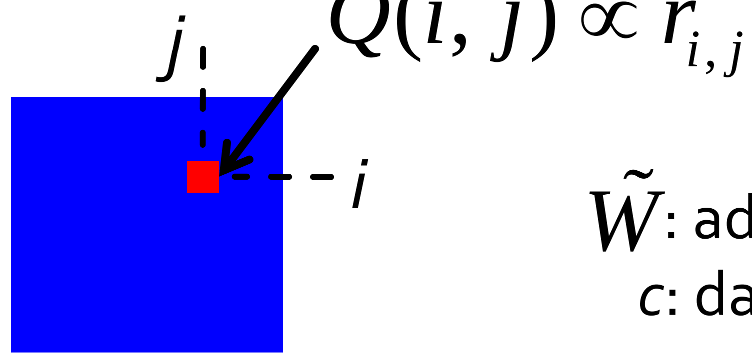
More red, more relevant

Ranking vector

$\vec{r}_4$

Why is RWR a good score?

$$Q = (I - c\tilde{W})^{-1} =$$



$\tilde{W}$: adjacency matrix.

c: damping factor

$$Q = I + c\,\tilde{W} + c^2\,\tilde{W}^2 + c^3\,\tilde{W}^3 + \cdots$$

all paths from *i* to *j* with length 1

all paths from *i* to *j* with length 2

all paths from *i* to *j* with length 3

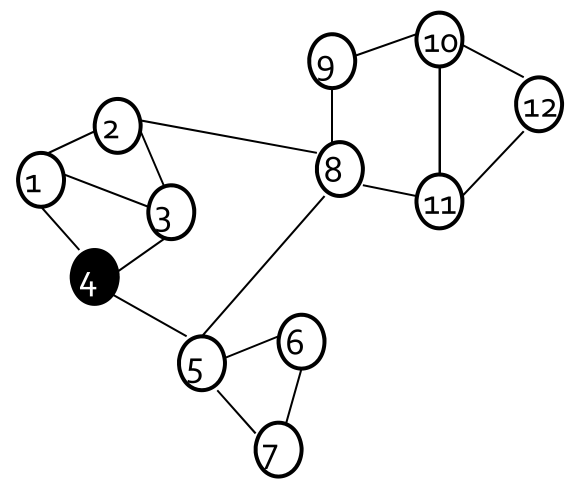
Computing the score

$$\vec{r}_i = c\tilde{W}\vec{r}_i + (1-c)\vec{e}_i$$

Ranking vector Transition matrix Restart prob. Starting vector

$$\begin{pmatrix}0.13\\0.10\\0.13\\0.22\\0.13\\0.05\\0.05\\0.08\\0.04\\0.03\\0.04\\0.02\end{pmatrix} = 0.9\times\begin{pmatrix}
 & 1/3 & 1/3 & 1/3 & & & & & & & & \\
1/3 & & 1/3 & & & & & 1/4 & & & & \\
1/3 & 1/3 & & 1/3 & & & & & & & & \\
1/3 & & 1/3 & & 1/4 & & & & & & & \\
 & & & 1/3 & & 1/2 & 1/2 & 1/4 & & & & \\
 & & & & 1/4 & & 1/2 & & & & & \\
 & & & & 1/4 & 1/2 & & & & & & \\
 & 1/3 & & & 1/4 & & & & 1/2 & & 1/3 & \\
 & & & & & & & 1/4 & & 1/3 & & \\
 & & & & & & & & 1/2 & & 1/3 & 1/2 \\
 & & & & & & & 1/4 & & 1/3 & & 1/2 \\
 & & & & & & & & & 1/3 & 1/3 &
\end{pmatrix}\begin{pmatrix}0.13\\0.10\\0.13\\0.22\\0.13\\0.05\\0.05\\0.08\\0.04\\0.03\\0.04\\0.02\end{pmatrix} + 0.1\times\begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$$

n x 1 n x n n x 1



Computing the score

$$
\begin{pmatrix} \\ \\ \\ \\ \\ \\ \textbf{?} \\ \\ \\ \\ \\ \\ \end{pmatrix} = 0.9
\begin{pmatrix}
 & 1/3 & 1/3 & 1/3 & & & & & & & & \\
1/3 & & 1/3 & & & & & 1/4 & & & & \\
1/3 & 1/3 & & 1/3 & & & & & & & & \\
1/3 & & 1/3 & & 1/4 & & & & & & & \\
 & & & 1/3 & & 1/2 & 1/2 & 1/4 & & & & \\
 & & & & 1/4 & & 1/2 & & & & & \\
 & & & & 1/4 & 1/2 & & & & & & \\
 & 1/3 & & & 1/4 & & & & 1/2 & & 1/3 & \\
 & & & & & & & 1/4 & & 1/3 & & \\
 & & & & & & & & 1/2 & & 1/3 & 1/2 \\
 & & & & & & & 1/4 & & 1/3 & & 1/2 \\
 & & & & & & & & & 1/3 & 1/3 &
\end{pmatrix}
\begin{pmatrix} \\ \\ \\ \\ \\ \\ \textbf{?} \\ \\ \\ \\ \\ \\ \end{pmatrix}
+ 0.1 \times
\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

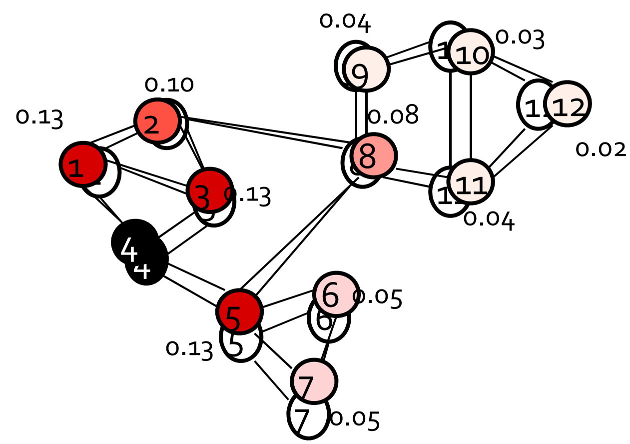
Query

Ranking vector　　Adjacency matrix　　Ranking vector　　Starting vector

Computing r_i

$$\vec{r}_i[t+1] = c\tilde{W}\vec{r}_i[t] + (1-c)\vec{e}_i$$

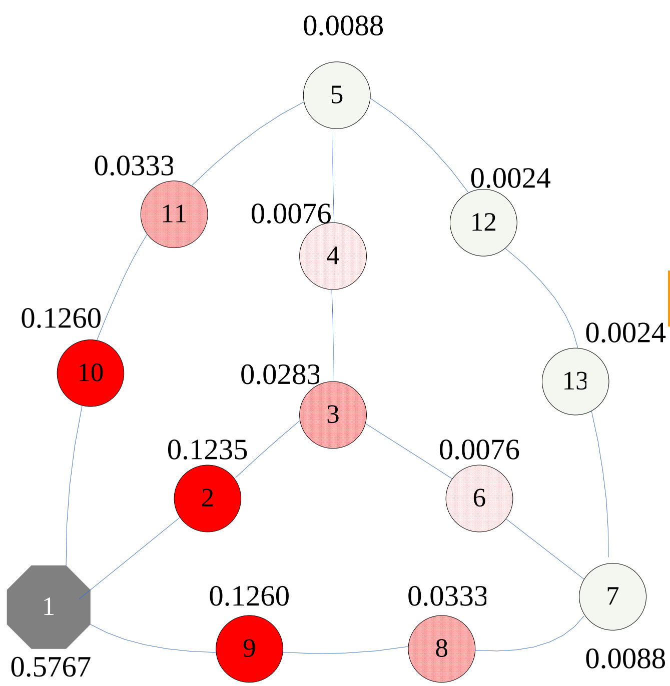
$$\vec{r}_i = 0.9 \times \underbrace{\begin{pmatrix}
 & 1/3 & 1/3 & 1/3 & & & & & & & & \\
1/3 & & 1/3 & & & & & 1/4 & & & & \\
1/3 & 1/3 & & 1/3 & & & & & & & & \\
1/3 & & 1/3 & & 1/4 & & & & & & & \\
 & & & 1/3 & & 1/2 & 1/2 & 1/4 & & & & \\
 & & & & 1/4 & & 1/2 & & & & & \\
 & & & & 1/4 & 1/2 & & & & & & \\
 & 1/3 & & & 1/4 & & & & 1/2 & & 1/3 & \\
 & & & & & & & 1/4 & & 1/3 & & \\
 & & & & & & & & 1/2 & & 1/3 & 1/2 \\
 & & & & & & & 1/4 & & 1/3 & & 1/2 \\
 & & & & & & & & & 1/3 & 1/3 &
\end{pmatrix}}_{\tilde{W}} \underbrace{\begin{pmatrix} 0.13 \\ 0.10 \\ 0.13 \\ 0.22 \\ 0.13 \\ 0.05 \\ 0.05 \\ 0.08 \\ 0.04 \\ 0.03 \\ 0.04 \\ 0.02 \end{pmatrix}}_{\vec{r}_i} + 0.1 \times \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\vec{e}_i}$$



No pre-computation / light storage

Slow on-line response O(mE)

Calculating Score r(i,j)



	Q1
Node 1	0.5767
Node 2	**0.1235**
Node 3	0.0283
Node 4	0.0076
Node 5	0.0088
Node 6	0.0076
Node 7	0.0088
Node 8	0.0333
Node 9	**0.1260**
Node 10	**0.1260**
Node 11	0.0333
Node 12	0.0024
Node 13	0.0024

Combining the scores

- The RWR score r(i,j)
  - i … query node, $i \in Q$
  - j … node in the network
- Combine r(i,j) to get the score r(Q,j)
  - **1)** AND query: Meeting probability
    - prob. that |Q| random particles coincide on node *j*

$$r(\mathcal{Q}, j) \triangleq r(\mathcal{Q}, j, Q) = \prod_{i=1}^{Q} r(i, j)$$

Combining the scores: k-SoftAND

$$r(\mathcal{Q}, j) \triangleq r(\mathcal{Q}, j, Q) = \prod_{i=1}^{Q} r(i, j)$$

- OR query:
  - At least one particle arrives to node *j*:
    (i.e., node *j* is important to at least 1 query node)

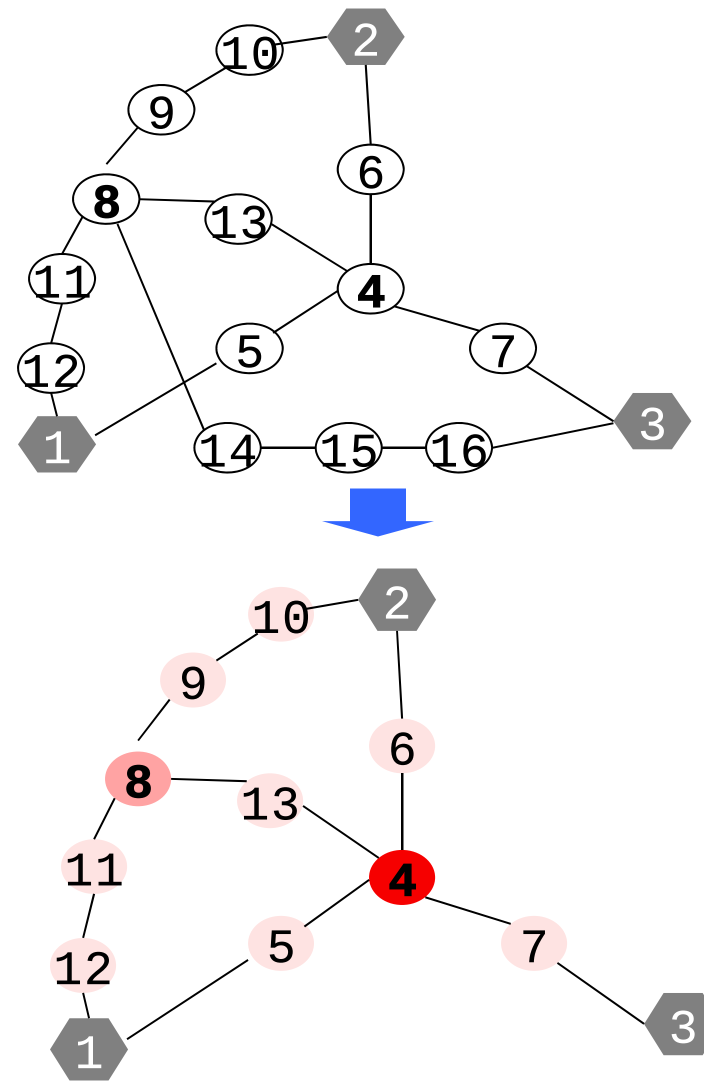
$$r(\mathcal{Q}, j) \triangleq r(\mathcal{Q}, j, 1) = 1 - \prod_{i=1}^{Q} (1 - r(i, j))$$

- k-SoftAND query:
  - Node *j* is important to at least *k* query nodes
  - Can be computed recursively:

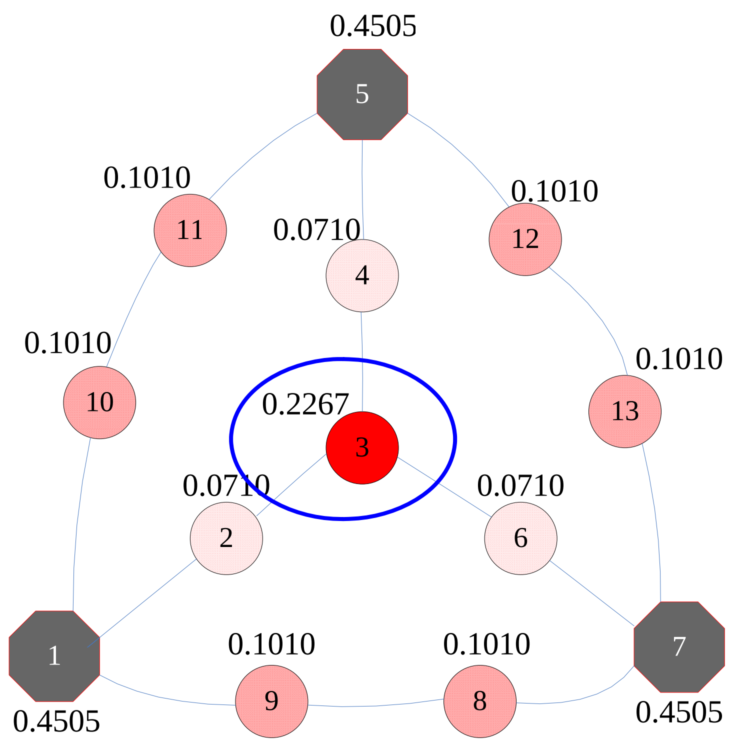
$$r(\mathcal{Q}, j, k) = r(\acute{\mathcal{Q}}, j, k-1) \cdot r(Q, j) + r(\acute{\mathcal{Q}}, j, k) \cdot (1 - r(Q, j))$$

Extracting Connection Subgraph

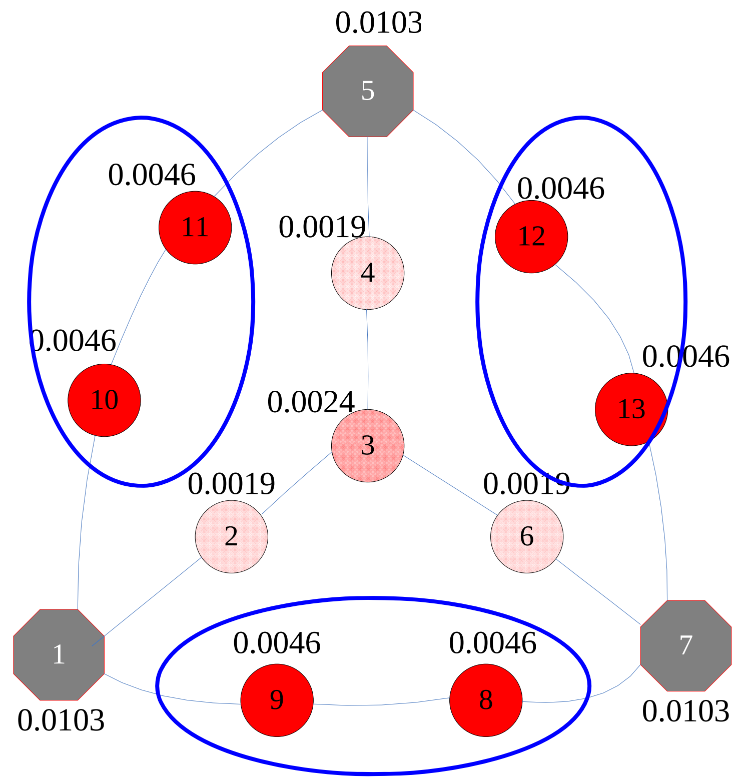
- Goal:
  - Extract a sub-graph *S* on *b* nodes that maximizes the $\sum_{u \in S} r(Q, u)$
- Idea:
  - Iterate until budget is reached
    - Pick not-yet-selected node $j = \arg\max_j r(Q,j)$
    - Find good paths to all query nodes
    - Add the paths to S



AND vs. k-SoftAND

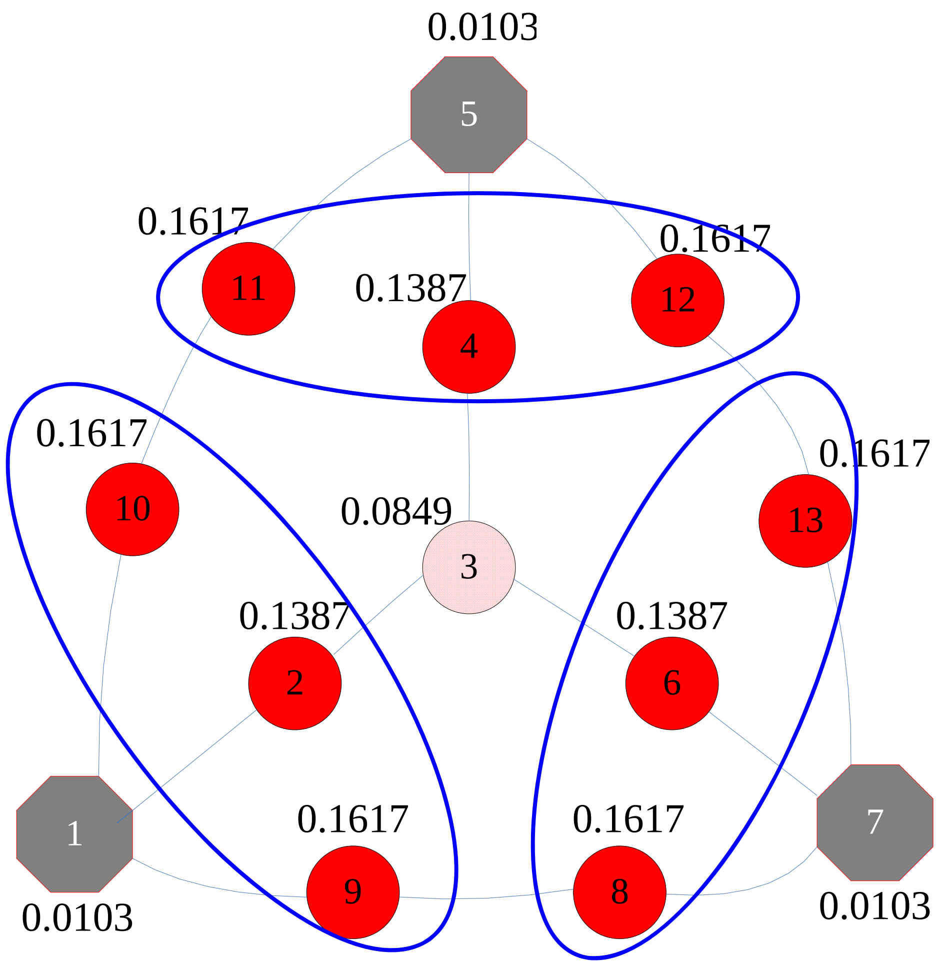


AND



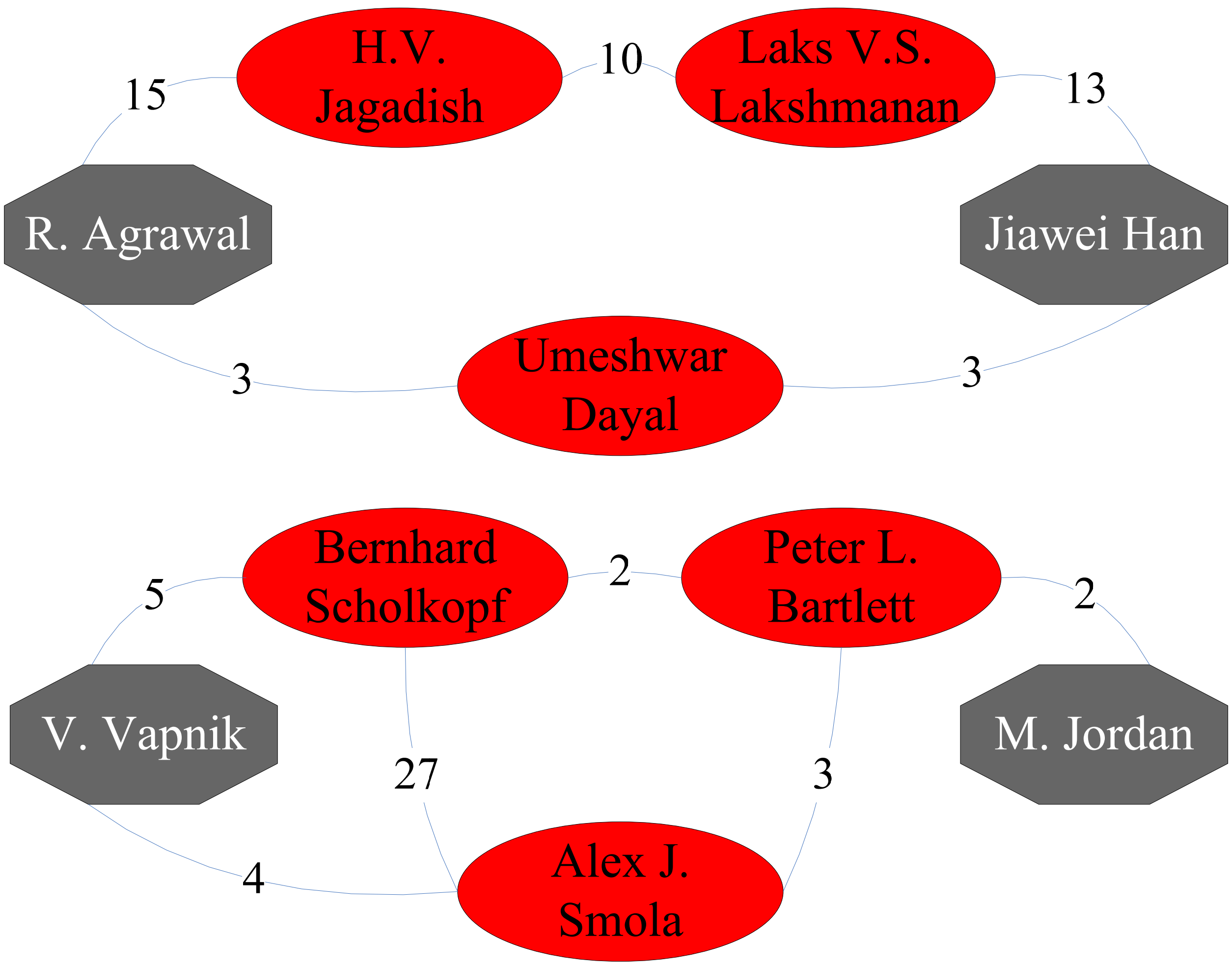
2-SoftAND

1-SoftAND = OR



0.0103
5
0.1617
11
0.1387
4
0.1617
12
0.1617
10
0.0849
3
0.1617
13
0.1387
2
0.1387
6
0.1617
9
0.1617
8
1
0.0103
7
0.0103

Example:



H.V. Jagadish
10
Laks V.S. Lakshmanan
15
13
R. Agrawal
Jiawei Han
3
Umeshwar Dayal
3
Bernhard Scholkopf
2
Peter L. Bartlett
5
2
V. Vapnik
M. Jordan
27
3
4
Alex J. Smola

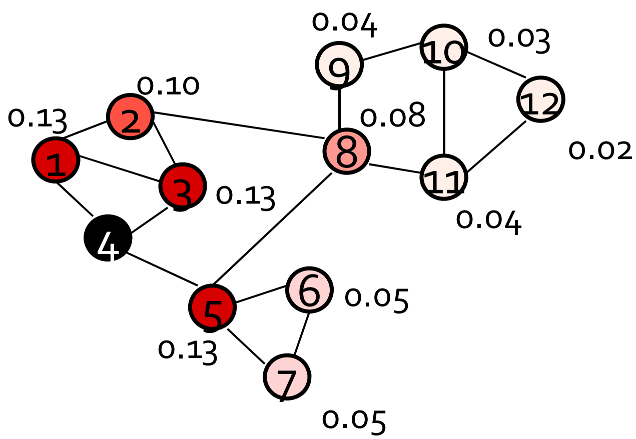
Speeding up the response time

- Solving the random walk for each query separately is time consuming
  - Not appropriate for real-time
- Can we do better?
- Idea 1) Pre-compute scores for each possible query node i

Pre-compute all the scores

$\mathbf{Q^{-1}}$:

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}
0.20	0.13	0.14	0.13	0.68	0.56	0.56	0.63	0.44	0.35	0.39	0.34
0.28	0.20	0.13	0.96	0.64	0.53	0.53	0.85	0.60	0.48	0.53	0.45
0.14	0.13	0.20	1.29	0.68	0.56	0.56	0.63	0.44	0.35	0.39	0.33
0.13	0.10	0.13	2.06	0.95	0.78	0.78	0.61	0.43	0.34	0.38	0.32
0.09	0.09	0.09	1.27	2.41	1.97	1.97	1.05	0.73	0.58	0.66	0.56
0.03	0.04	0.04	0.52	0.98	2.06	1.37	0.43	0.30	0.24	0.27	0.22
0.03	0.04	0.04	0.52	0.98	1.37	2.06	0.43	0.30	0.24	0.27	0.22
0.08	0.11	0.04	0.82	1.05	0.86	0.86	2.13	1.49	1.19	1.33	1.13
0.03	0.04	0.03	0.28	0.36	0.30	0.30	0.74	1.78	1.00	0.76	0.79
0.04	0.04	0.04	0.34	0.44	0.36	0.36	0.89	1.50	2.45	1.54	1.80
0.04	0.05	0.04	0.38	0.49	0.40	0.40	1.00	1.14	1.54	2.28	1.72
0.02	0.03	0.02	0.21	0.28	0.22	0.22	0.56	0.79	1.20	1.14	2.05



$$Q^{-1} = (I - c\tilde{W})^{-1}$$

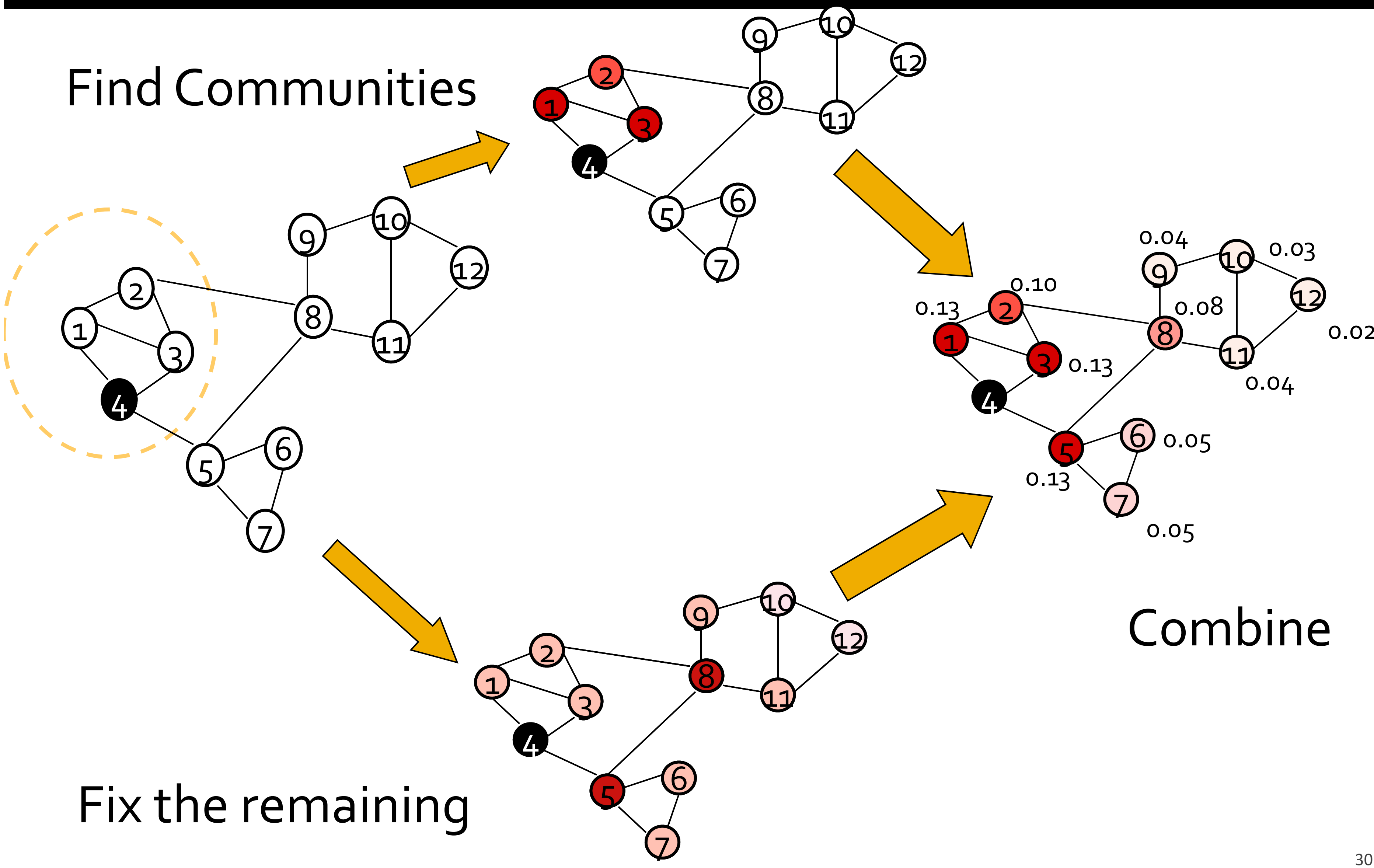
Fast on-line response

Heavy pre-computation/storage cost

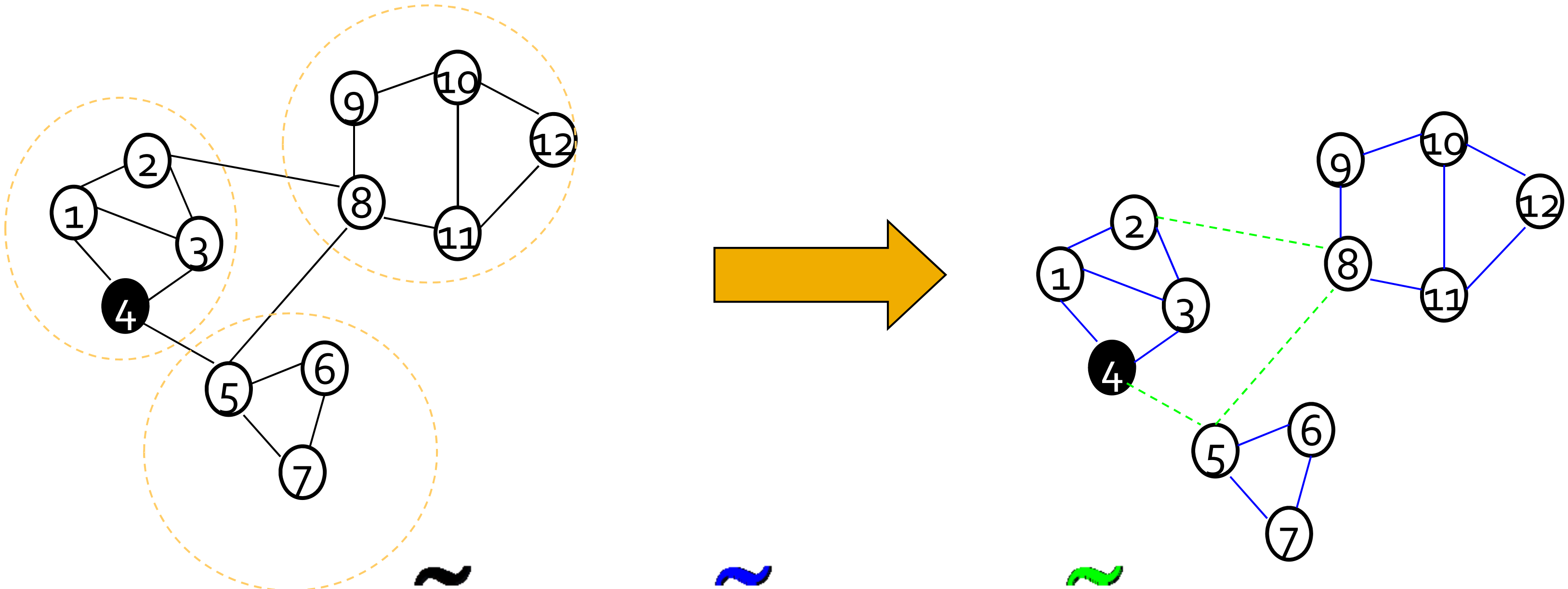
$O(n^3)$ $O(n^2)$

Idea 2: Approximation

[Tong-Faloutsos-Pan, ‘06]



Step 1: Partition the graph

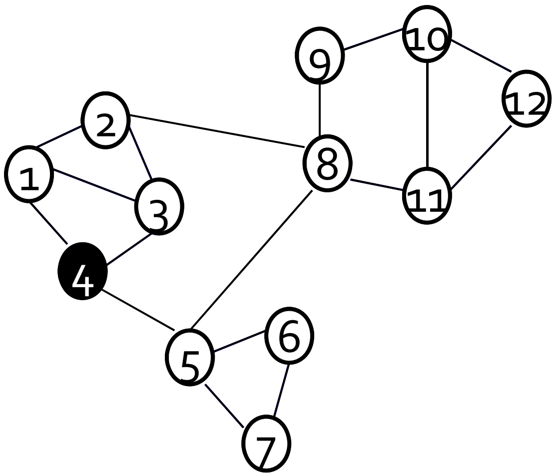


$$\tilde{W} = \tilde{W}_1 + \tilde{W}_2$$

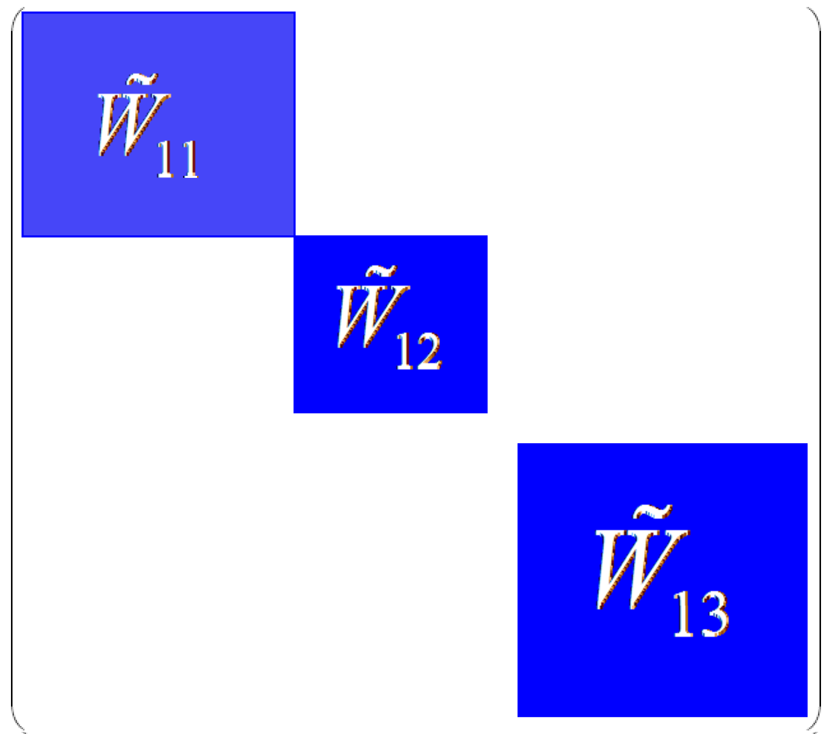
Within-partition links

cross-partition links

Step 2: Make W_1 block-diagonal

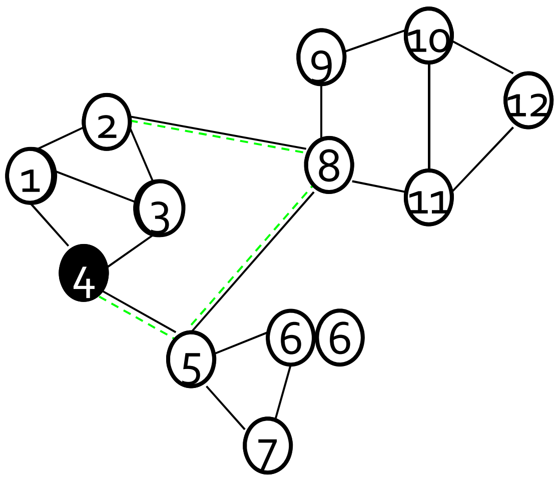


$$\tilde{W}_1 = \begin{pmatrix} \tilde{W}_{11} & & \\ & \tilde{W}_{22} & \\ & & \tilde{W}_{33} \end{pmatrix} =$$

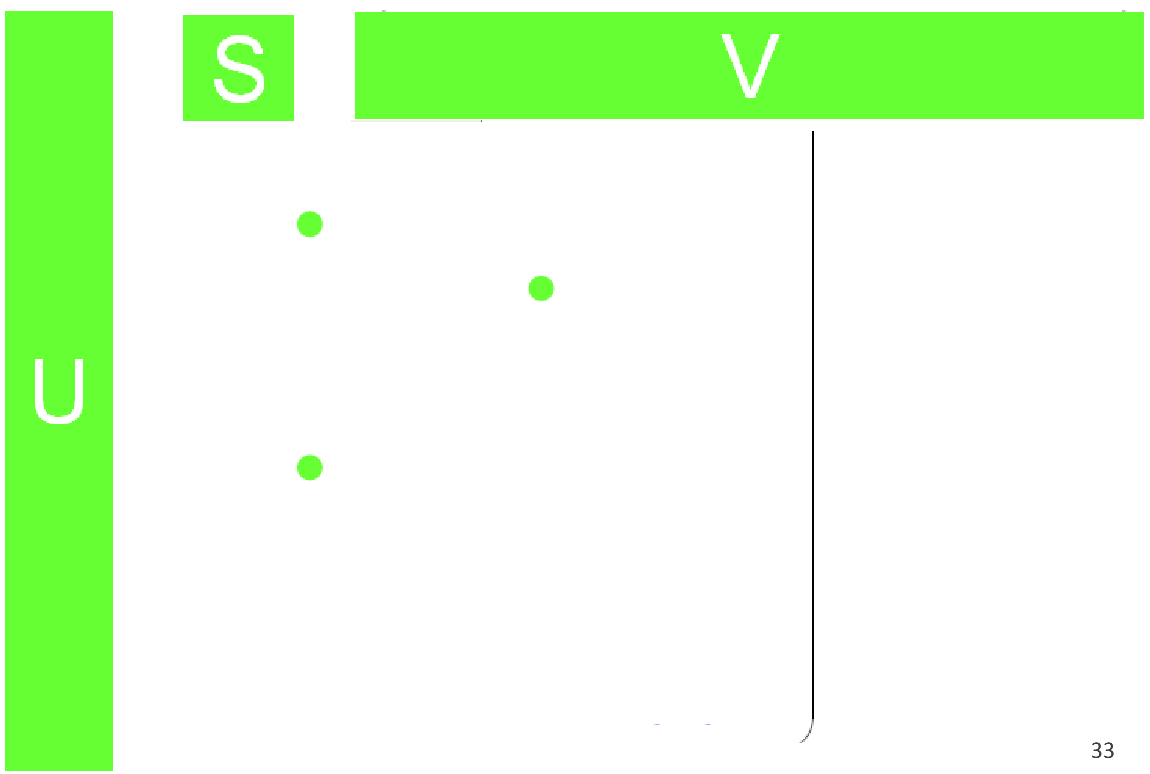


$$Q_{1,i}^{-1} = (\mathrm{I} - \tilde{\mathrm{W}}_{1,i})^{-1}$$

Step 3: Cross community links



$$\tilde{W}_2 \approx USV =$$



$|S| << |\tilde{W}_2|$

Final algorithm: Offline strage

- Offline stage:

$$Q_1^{-1} = \begin{pmatrix} Q_{1,1} & & & \\ & Q_{1,2} & & \\ & & \ddots & \\ & & & Q_{1,k} \end{pmatrix}$$

Communities

$$\begin{pmatrix} 0.39 & -0.31 & 0.03 & -0.03 \\ -0.28 & 1.18 & -0.10 & 0.06 \\ -0.00 & -0.05 & 0.17 & 0.03 \\ -0.03 & 0.01 & 0.03 & 0.19 \end{pmatrix}$$

$$\tilde{\Lambda} = (S^{-1} - cVQ_1^{-1}U)^{-1}$$

Bridges

$$\tilde{W} = \tilde{W}_1 + \tilde{W}_2$$

One more thing:

- Sherman-Morrison-Woodbury matrix identity:
  - Inverse of a rank-*k* correction of matrix X can be computed by doing a rank-*k* correction to X^{-1}:

$$(\mathbf{X} - \mathbf{USV})^{-1} = \mathbf{X}^{-1} + \mathbf{X}^{-1}\mathbf{U}\tilde{\mathbf{\Lambda}}\mathbf{V}\mathbf{X}^{-1}$$

$$\text{where } \tilde{\mathbf{\Lambda}} = (\mathbf{S}^{-1} - \mathbf{V}\mathbf{X}^{-1}\mathbf{U})^{-1}$$

- In our case:
  $Q^{-1}=c(I-W)^{-1}=c(I-W_1-W_2)^{-1}=$

$$Q^{-1} \approx Q_1^{-1} + cQ_1^{-1}U\tilde{\Lambda}VQ_1^{-1}$$

Final algorithm: Online stage

- Online stage:
  - Compute column i of:

$$Q^{-1} \approx Q_1^{-1} + cQ_1^{-1} U \tilde{\Lambda} V Q_1^{-1}$$

  - Using:

$$\vec{r_i} = (1-c)(\mathbf{Q}_1^{-1}\vec{e_i} + c\mathbf{Q}_1^{-1}\mathbf{U}\tilde{\mathbf{\Lambda}}\mathbf{V}\mathbf{Q}_1^{-1}\vec{e_i})$$

Acknowledgment

- Most slides are borrowed from the tutorial by Hanghang Tong and Christos Faloutsos from CMU
- http://www.cs.cmu.edu/~htong/tut/cikm2008/cikm_tutorial.html