Mining Graph Data

Cam Tu Nguyen

阮锦绣

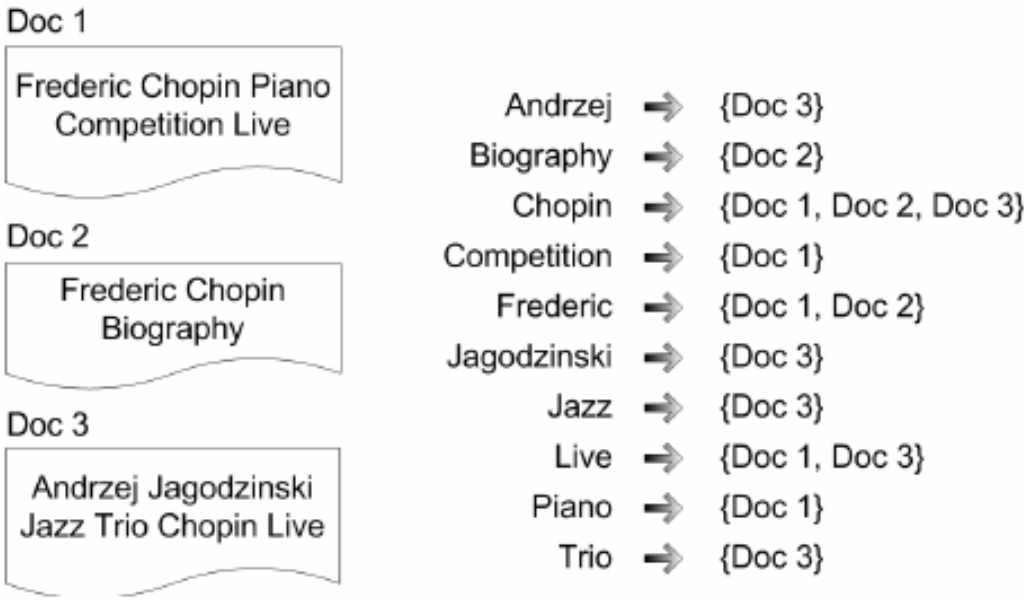
Software Institute, Nanjing University
nguyenct@lamda.nju.edu.cn
ncamtu@gmail.com

Outline

- PageRank for Web Graph
  - PageRank
- Mining Social-Network Graphs
  - Social Networks as Graphs
  - Finding Communities
    - Using Betweeness
    - Finding Complete Bipartie SubGraphs
  - Partitioning of Graphs
  - SimRank

Early Search Engine and Term Spam

- Search engines before Google mostly worked by crawling the Web and listing the terms in an **inverted index**.



- Term Spam: Fool search engines to into leading people to their page
  - A shirt-selling website include popular terms such as “movie” in the content (with background color).

Early Search Engine and Term Spam

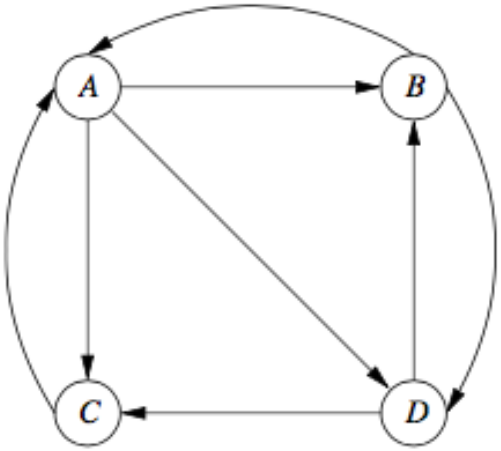
- Search engines before Google mostly worked by crawling the Web and listing the terms in an **inverted index**.

- Term Spam: Fool search engines to into leading people to their page
  - A shirt-selling website include popular terms such as “movie” in the content (with background color).

- Google introduced two innovations
  - **PageRank** was used to simulate where Web surfers, starting at a random page, would tend to gather at some important pages if they following hyperlinks.
  - The content of a page was judged not only by *the terms appearing on that page, but by the terms used in or near the links to that page*.

Definition of PageRank

- PageRank is a function that assigns a real number to each page in the Web.
  - The higher the PageRank of a page, the more "important" it is.
- We consider the ideal case where the Web is a connected component graph.
  - m_{ij} in row i and column j has value of 1/k if j has k arcs out, and one of them is to page i.



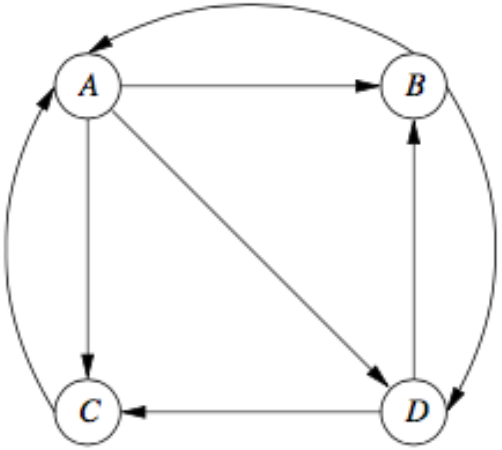
Transition Matrix

$$M = \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Vector showing the probabilities of moving from node B to A and D.

Definition of PageRank

- We consider the ideal case where the Web is a connected component graph.



$$M = \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Transition Matrix

- Consider a random surfer associated with a column vector whose jth component is the probability that the surfer is at page j (*idealized PageRank function).*

Definition of PageRank

- The distribution of a random surfer on the Web:
  - Initial probability vector v_0
  - After n steps, the distribution of the surfer will be $M^n v_0$

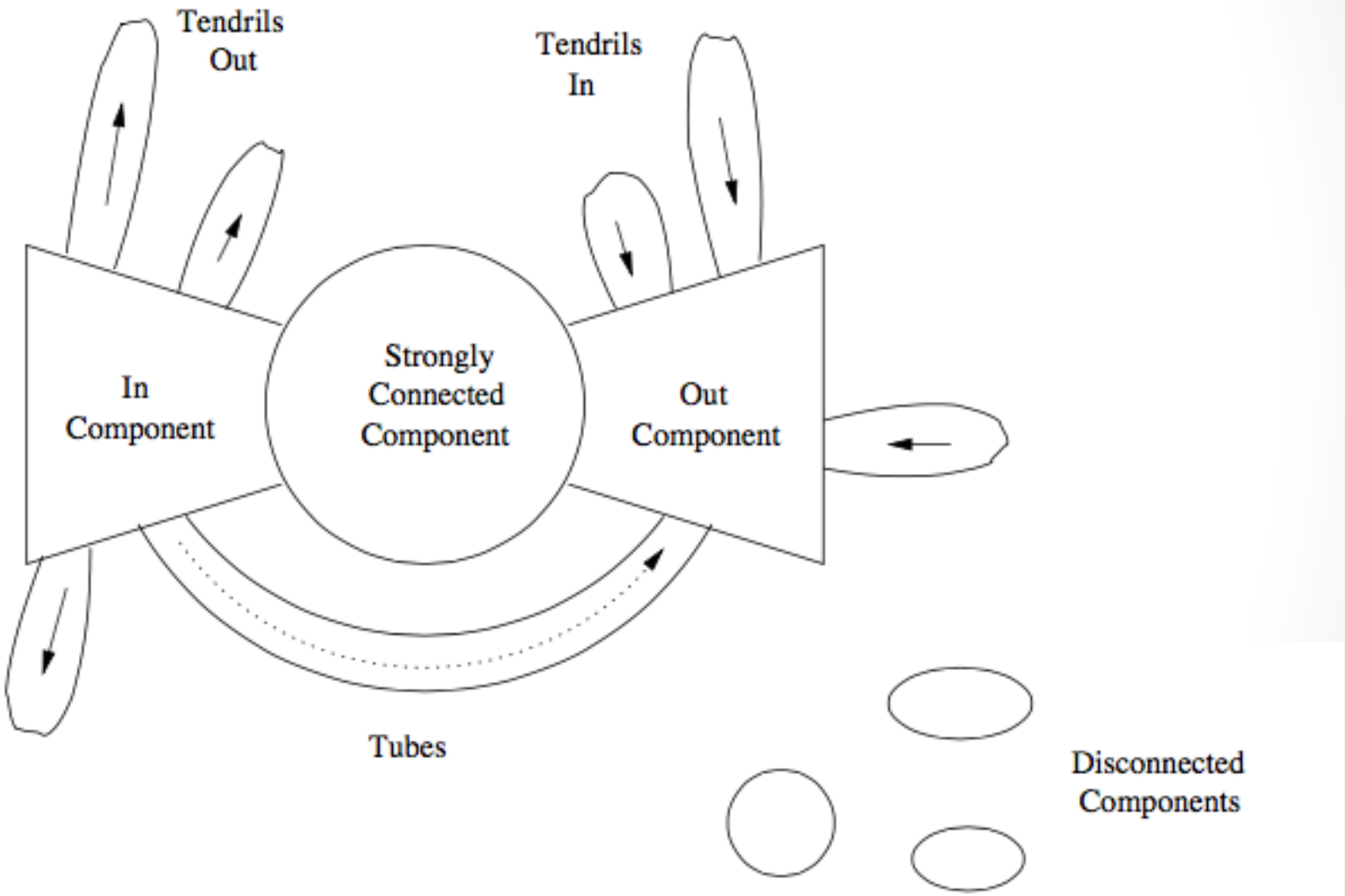
- The theory of Markov Process:
  - If the graph is strongly connected and there are no dead ends, the distribution of the surfer approaches a limiting distribution **v** that **v = Mv**
  - In other words, the limiting distribution **v** is the eigenvector of M.

Power Iteration Method

- Simple iterative scheme
  - Suppose there are N web pages
  - Initialize $v_0 = [1/N, \ldots, 1/N]^T$
  - Iterate $v^{k+1} = Mr^k$
  - Stop when $|v^{k+1} - v^k|_1 < \text{epsilon}$
    - $|x|$ is the L1 norm or Manhattan distance of vector x.
    - Can use other vector norm, e.g. Euclidean
- Example: apply to the previous example of Web graph

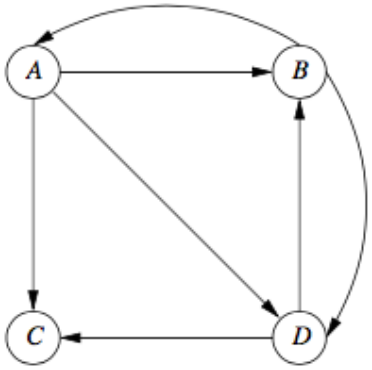
$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{bmatrix}, \begin{bmatrix} 11/32 \\ 7/32 \\ 7/32 \\ 7/32 \end{bmatrix}, \cdots, \begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}$$

Structure of the Web



Avoiding dead ends

- A page with no-link out is called a dead end
- If a transition matrix contains some columns sum to 0, then some or all of the components of the vector $M^i v$ will go to 0.



$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

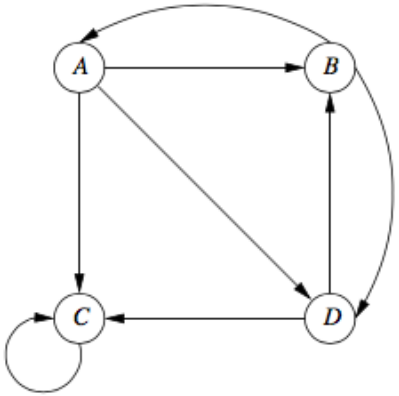
$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 3/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 5/48 \\ 7/48 \\ 7/48 \\ 7/48 \end{bmatrix}, \begin{bmatrix} 21/288 \\ 31/288 \\ 31/288 \\ 31/288 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Avoiding dead ends

- Two approaches to deal with dead ends
  - Drop the dead ends from the graph, also drop their incoming args. Doing so may create more dead ends, which also have to be dropped, recursively.
  - Taxation: modify the process by which random surfers are assumed to move about the Web. This technique can also be used to solves the problem of spider traps.

Spider Traps and Taxations

- A spider trap is a set of nodes with no dead ends but no arcs out.



$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 3/24 \\ 5/24 \\ 11/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 5/48 \\ 7/48 \\ 29/48 \\ 7/48 \end{bmatrix}, \begin{bmatrix} 21/288 \\ 31/288 \\ 205/288 \\ 31/288 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Spider Traps and Taxations

- Modify the calculation of PageRank by allowing each random surfer a small probability of **teleporting** to a random page.

$$\mathbf{v}' = \beta M \mathbf{v} + (1 - \beta)\mathbf{e}/n$$

  - **beta** is a chosen constant, often chosen in the range 0.8 to 0.9
  - **e** is a vector of all 1.

Spider Traps and Taxations

- Example: beta=0.8

$$\mathbf{v}' = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 4/5 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 1/20 \\ 1/20 \\ 1/20 \\ 1/20 \end{bmatrix}$$

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 9/60 \\ 13/60 \\ 25/60 \\ 13/60 \end{bmatrix}, \begin{bmatrix} 41/300 \\ 53/300 \\ 153/300 \\ 53/300 \end{bmatrix}, \begin{bmatrix} 543/4500 \\ 707/4500 \\ 2543/4500 \\ 707/4500 \end{bmatrix}, \cdots, \begin{bmatrix} 15/148 \\ 19/148 \\ 95/148 \\ 19/148 \end{bmatrix}$$

- The effect of the spider trap (at node C) was limited.

Additional Topics (for further reading)

- Efficient Computation of PageRank using MapReduce
- Combating Spam
  - Link Spam
  - Trust Rank
- Hubs and Authorities

Outline

- PageRank for Web Graph
  - PageRank
- Mining Social-Network Graphs
  - Social Networks as Graphs
  - Finding Communities
    - Using betweeness
    - Finding Complete Bipartie SubGraphs
  - Partitioning of Graphs
  - Similarity between nodes: SimRank

Social Networks as Graphs

- Social Network
  1. There is a collection of entities that participate in the network
  2. There is at least one relationship between entities of the network
     - “friend” relationships in Facebook
     - The relationship may have a degree, e.g.: friends, family, acquaintances, or none (discrete degree)
  3. There is an assumption of non-randomness or locality
     - The relationships tend to cluster; i.e. if entity A is related to both B and C, then there is a higher probability than average that B and C are related.

Social Networks as Graphs

- Social Networks as Graphs
  - Nodes are entities, edges represent relationships
  - Edges may have labels (degree of the relationship).
  - The Graph can be undirected (Facebook) or directed (Google+, twitter).

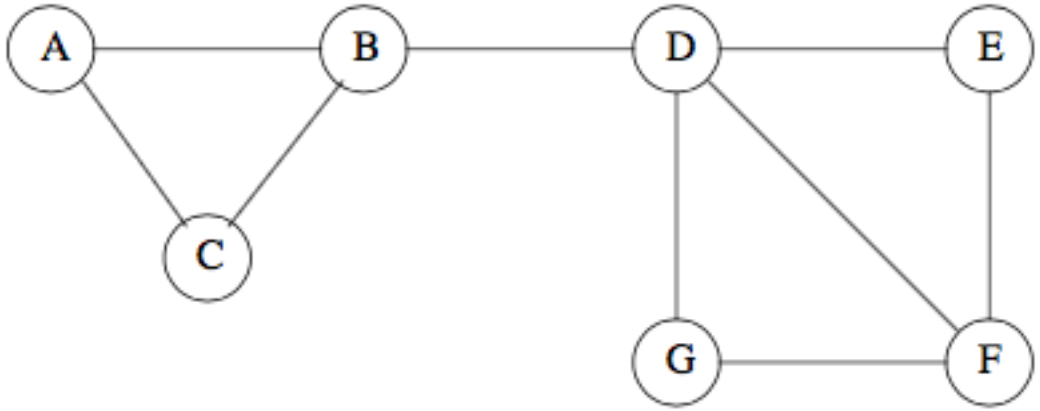
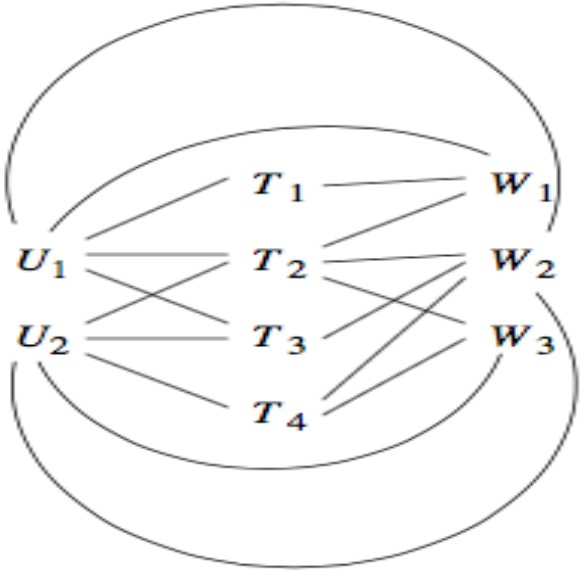


Figure 10.1: Example of a small social network

Graph with Several Node Types

- Tri-partie graph that models users, tags and Web pages at a site like deli.cio.us



- In general, a ***k-partite*** graph consists of **k** disjoint sets of nodes.

Outline

- PageRank for Web Graph
  - PageRank
- Mining Social-Network Graphs
  - Social Networks as Graphs
  - Finding Communities
    - Using Betweeness
    - Finding Complete Bipartie SubGraphs
  - Partitioning of Graphs
  - Similarity between nodes: SimRank

Finding Communities

- An important aspect of social networks is that they contain communities of entities that are connected by many edges.
  - These typically correspond to groups of friends at school, etc.

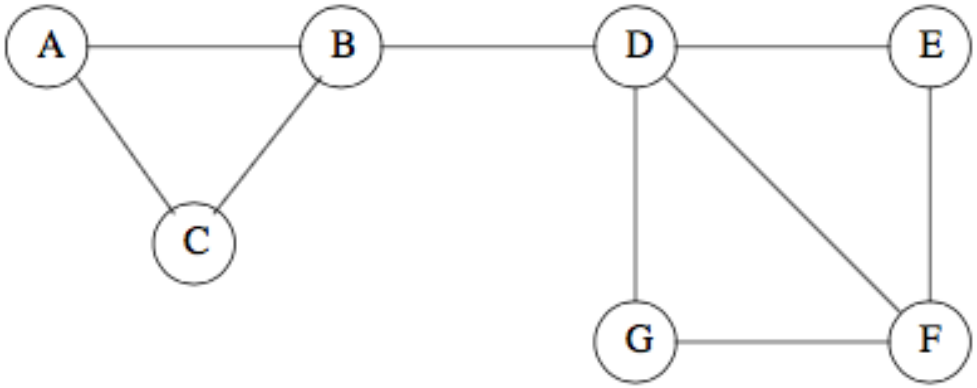
- Finding communities:
  - Clustering on the social graph
    - How to define a distance measure?
  - Using Betweeness
  - Finding complete bipartie subgraphs.

Finding Communities Using Betweeness

- The problem of standard clustering method
  - Measuring distances between nodes based on edges in unlabeled graphs.
- Finding communities using **betweeness**
  - Find the edges that are least likely to be inside a community, then remove those edges.
  - The **betweeness** of an edge (a,b) to be the number of pairs of nodes x and y such that the edge (a,b) lies on the shortest path between x and y.
  - High betweeness means an edge runs between two different communities

Finding Communities Using Betweeness

- Finding communities using **betweeness**
  - Find the edges that are least likely to be inside a community then remove those edges.
  - The **betweeness** of an edge (a,b) to be the number of pairs of nodes x and y such that the edge (a,b) lies on the shortest path between x and y.
  - High betweeness means an edge runs between two different communities



(B,D) has high betweeness

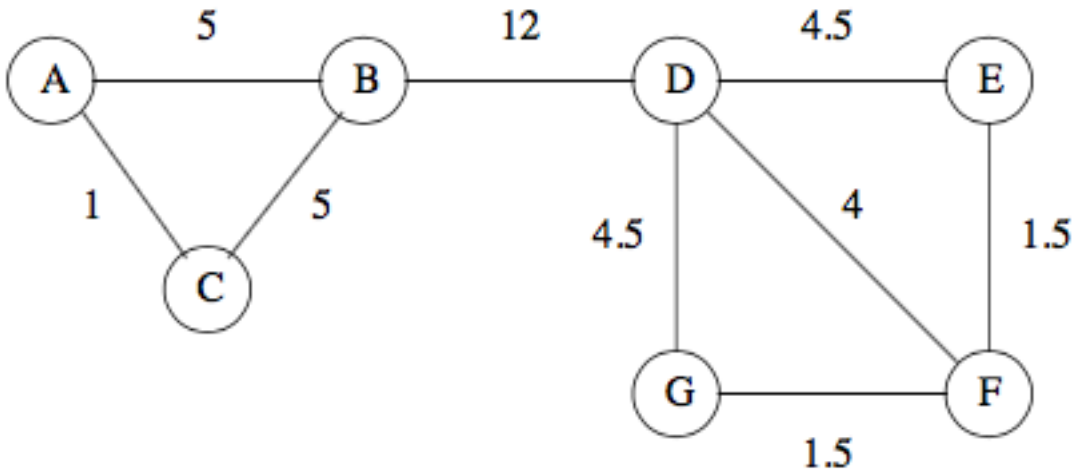
Further reading: The Girvan-Newman algorithm to find betweeness of every edges in graph

Finding Communities Using Betweeness

- The betweeness scores for the edges of a graph behave something like a distance measure on the nodes of the graph.
  - (It is not exactly a distance measure though)
  - We can cluster by taking the edges in order of increasing betweeness and add them to the graph one at a time.
  - At each step, the connected components of the graph form some clusters.
  - Another view: the process of edge removal

Finding Communities Using Betweeness

- Example:



First step: remove (B,D)
Second step: remove (A,B), and (B,C)
Third step: remove (D,E) and (D,G)
…

Finding Complete Bipartite Subgraphs for Community Detection.

- Discover communities directly by looking for subsets of the nodes that have a relatively large number of edges among them.

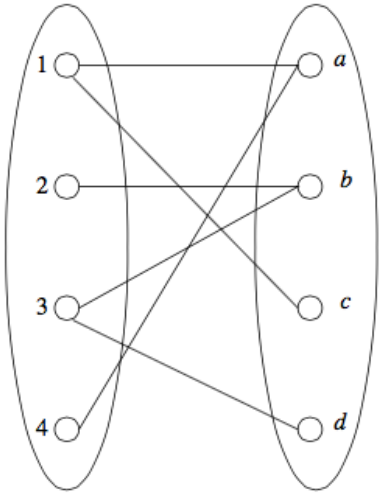
- Approaches:
  - Finding Cliques (NP-complete)
  - Finding complete bipartite graphs

Finding Complete Bipartite Subgraphs for Community Detection.

- A complete bipartite graph consists of ***s*** nodes on 1 side and ***t*** nodes on the other side, with all ***s*t*** possible edges between the nodes of one side and the other side.
  - We denote this graph as $K_{s,t}$
- We can regard a complete bipartite subgraph as the nucleas of a community and add to it nodes with many edges to existing members of the community.
- We can extend the techniques for bipartite graphs to k-partite graph and ordinary graphs with one node type.
  - Ordinary graphs: divide the nodes into two equal groups at random.

Finding Complete Bipartite Subgraphs for Community Detection.

- Problem:
  - Given a large bipartite graph G, find instances of Ks,t within it.
  - Cast the problem of finding instances of Ks,t within G to the problem of finding frequent itemsets with the support count threshold is **s.**



Nodes on the left: items
Nodes on the right: baskets

Basket a contains items {1,4}, etc.

Further reading: why complete bipartite subgraphs must exist?

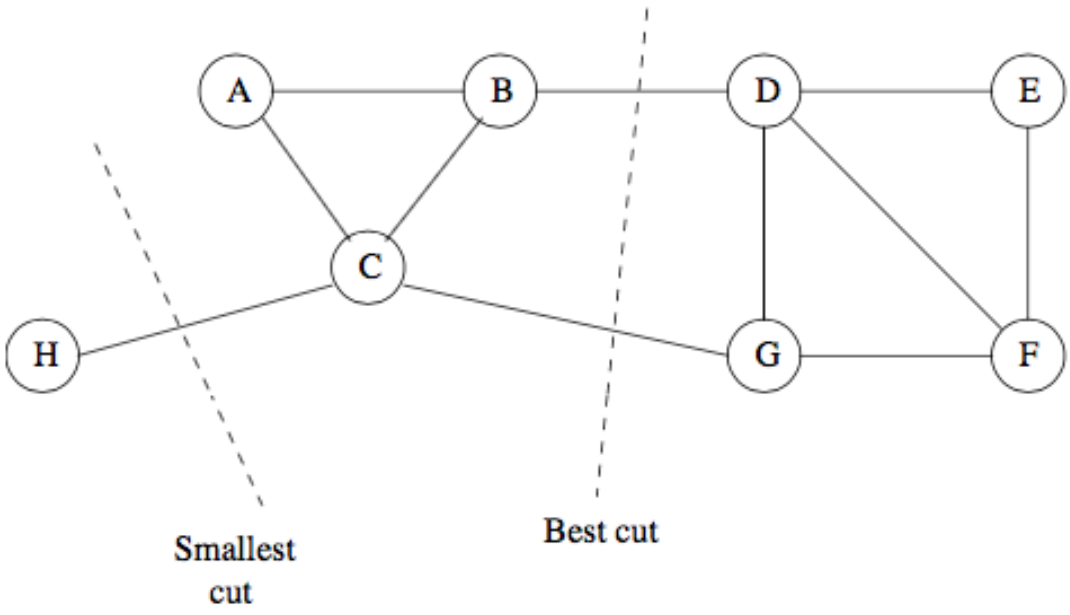
Outline

- PageRank for Web Graph
  - PageRank
- Mining Social-Network Graphs
  - Social Networks as Graphs
  - Finding Communities
    - Using Betweeness
    - Finding Complete Bipartie SubGraphs
  - Partitioning of Graphs
  - Similarity between nodes: SimRank

Partitioning of Graphs

- Problem: find a way to partition a graph to minimize the number of edges that connect different components.

- The goal is to find a good “cut”



Partitioning of Graphs

- Normalized Cuts:
  - Suppose we partition the nodes of a graph into two disjoint sets S and T.
  - Let Cut(S,T) be the number of edges that connect a node in S and a node in T.
  - The normalized cut value for S and T iss:

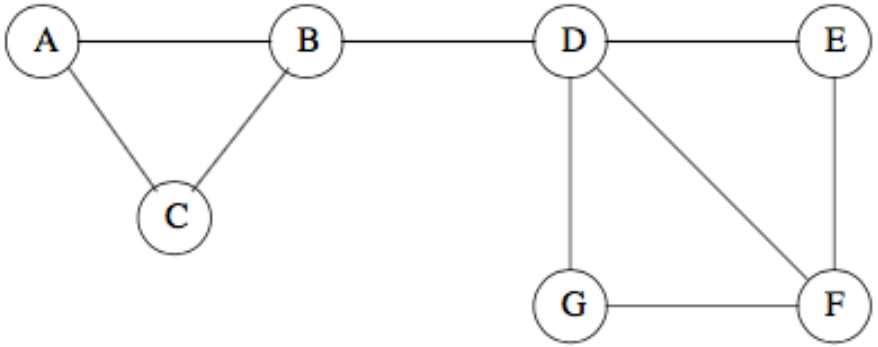
$$\frac{Cut(S,T)}{Vol(S)} + \frac{Cut(S,T)}{Vol(T)}$$

Find the cut that mimimize the normalized cut value.

Partitioning of Graphs

- Fundamental
  - Adjacency matrix (A)

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$



  - Degree matrix (D)

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Laplacian Matrix L = D - A

$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

Partitioning of Graphs

- Eigenvalues of the Laplacian Matrix
  - The smallest eigenvalue for every Laplacian matrix is 0**,** and its corresponding eigenvector is [1,1,…,1]
  - The second-smallest eigenvalue of L is the minimum of $\mathbf{x^T L x}$, where $\mathbf{x}=[x_1,x_2, \ldots, x_n]$ is a column vector with n components, and the minimum is taken under the constraints:
    - The length of **x** is 1
    - **x** is orthogonal to the eigenvector associated with the smallest eigenvalue; or $\mathbf{sum(}x_i\mathbf{)}=0$.
  - We can group the terms of $\mathbf{x^T L x}$ (**x** is the eigenvector corresponding with the second-smallest eigenvalue) in a way that distributes the terms to each pair {i,j}
    - $\mathbf{x^T L x}$ = sum of all graph edges (i,j) of $(x_i-x_j)^2$.

Partitioning of Graphs

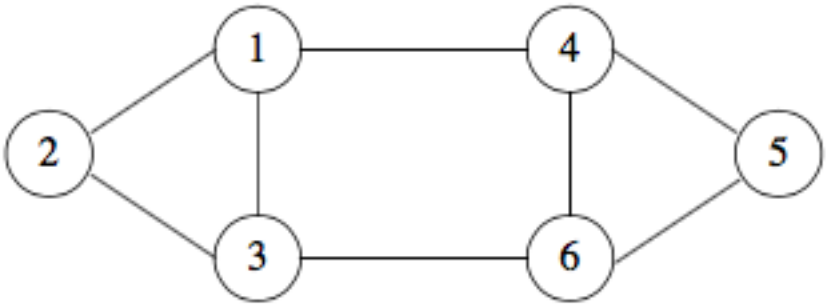
- Eigenvalues of the Laplacian Matrix
  - The second-smallest eigenvalue of L is the minimum of $\mathbf{x^T L x}$, where $\mathbf{x}=[x_1, x_2, \ldots, x_n]$ is a column vector with n components, and the minimum is taken under the constraints:
    - The length of **x** is 1
    - **sum(x_i)=0** → there some components are positive and some are negative.
    - $\mathbf{x^T L x}$ = sum of all graph edges (i,j) of $(x_i - x_j)^2$, which is likely smaller if x_i and x_j are the same sides.

→We can obtain a partition of the graph by taking one set to be the nodes i whose corresponding vector component x_i is positive and the other set to be those whose components are negative.

→Second-smallest eigenvector of L reveals a good cut.

Partitioning of Graphs

- Eigenvalues of the Laplacian Matrix



$$\begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	−5	−1	−1	−1
	1	2	4	−2	1	0
	1	1	1	3	−1	1
	1	−1	−5	−1	1	1
	1	−2	4	−2	−1	0
	1	−1	1	3	1	−1

Positive group (1,2,3), negative group (4,5,6); the partition into these two group reveal the best cut.

Outline

- PageRank for Web Graph
  - PageRank
- Mining Social-Network Graphs
  - Social Networks as Graphs
  - Finding Communities
    - Using Betweeness
    - Finding Complete Bipartie SubGraphs
  - Partitioning of Graphs
  - Similarity between nodes: SimRank

SimRank

- Apply best to graphs with several types of nodes.
  - In principle, can be applied to any graph.

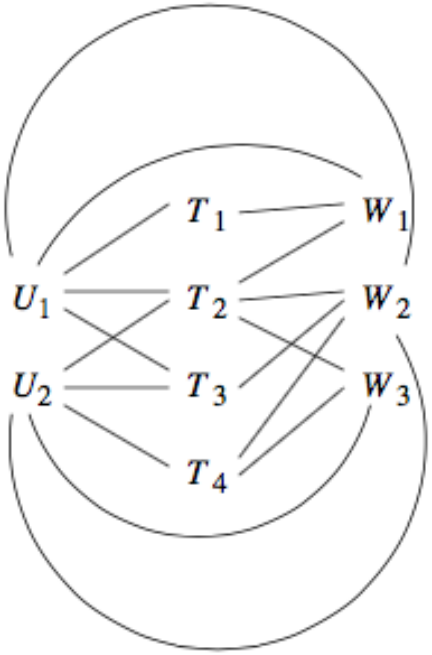
- The purpose of simrank is to measure the similarity between nodes of the same type.
  - Main idea: see where random walkers on the graph wind up when starting at a particular node.

Random Walkers on Social Graph

- A walker at a node N of an undirected graph will move with equal probability to any of the **neighbors of N**

A walker starts at T1, the chance that the walker will visit T2 is better than the chance he will visit T3, T4.

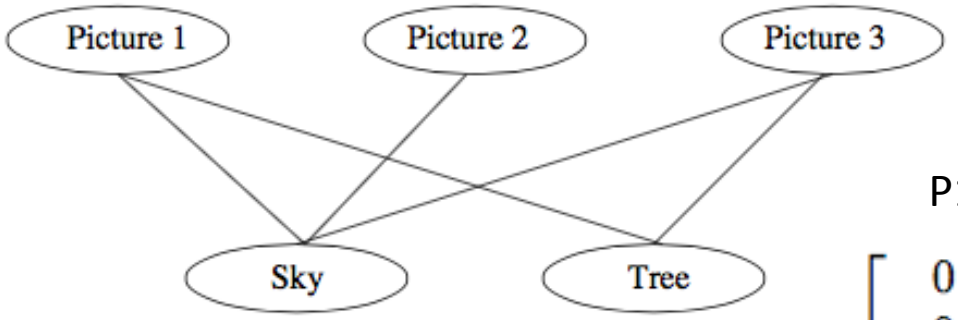
We could infer that tags T1 and T2 are therefore related in some way.



A graph of users, tags, and web pages.

Random Walks with Restart

- As we seen in PageRank, it is beneficial to consider a small probability that the walker will stop walking at random; also we should select a subset of Web pages as the teleport set.
- We also use the idea of "teleporting" here, but the teleport set contains only one node, the starting node.
- Consider the following example:



	P1	P2	P3	Sky	Tree
	0	0	0	1/3	1/2
	0	0	0	1/3	0
	0	0	0	1/3	1/2
	1/2	1	1/2	0	0
	1/2	0	1/2	0	0

Random Walks with Restart

- Let beta as the probability that the walker continues at random; (1-beta) is the probability that the walker will teleport to the initial node N.

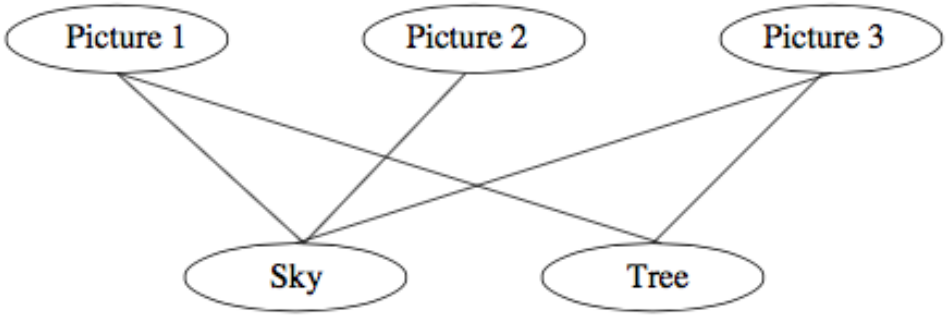
- Let v, v' be the column vectors that reflect the probability that the walker is at every node at a particular round; and its next round.

$$\mathbf{v}' = \beta M \mathbf{v} + (1 - \beta)\mathbf{e}_N$$

where e_N is a column vector with 1 is set at node N and 0 elsewhere.

Random Walks with Restart

Graph



Transition Matrix

$$\begin{bmatrix} 0 & 0 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/2 \\ 1/2 & 1 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \end{bmatrix}$$

Beta = 0.8, starting at node Picture 1

$$\mathbf{v}' = \begin{bmatrix} 0 & 0 & 0 & 4/15 & 2/5 \\ 0 & 0 & 0 & 4/15 & 0 \\ 0 & 0 & 0 & 4/15 & 2/5 \\ 2/5 & 4/5 & 2/5 & 0 & 0 \\ 2/5 & 0 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Random Walks with Restart

- Example (cont.): The original matrix is stochastic (since the graph is connected), then if the initial v has components that sum to 1, then v' will also have components that sum to 1.
  - We can simplify the iteration equation by adding 1/5 to each of the entries in the first row of the matrix.

$$\mathbf{v}' = \begin{bmatrix} 1/5 & 1/5 & 1/5 & 7/15 & 3/5 \\ 0 & 0 & 0 & 4/15 & 0 \\ 0 & 0 & 0 & 4/15 & 2/5 \\ 2/5 & 4/5 & 2/5 & 0 & 0 \\ 2/5 & 0 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v}$$

  - The iteration sequence of estimates of the distribution of the walker is

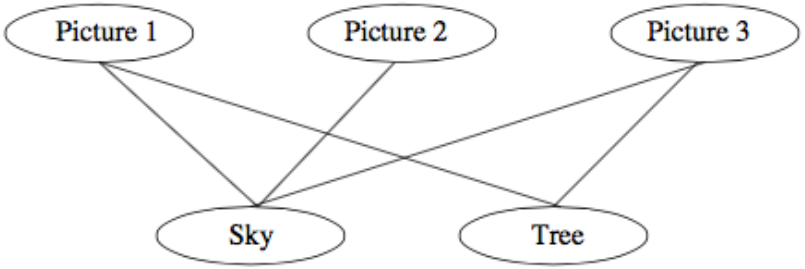
$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 2/5 \\ 2/5 \end{bmatrix}, \begin{bmatrix} 35/75 \\ 8/75 \\ 20/75 \\ 6/75 \\ 6/75 \end{bmatrix}, \begin{bmatrix} 95/375 \\ 8/375 \\ 20/375 \\ 142/375 \\ 110/375 \end{bmatrix}, \begin{bmatrix} 2353/5625 \\ 568/5625 \\ 1228/5625 \\ 786/5625 \\ 690/5625 \end{bmatrix}, \ldots, \begin{bmatrix} .345 \\ .066 \\ .145 \\ .249 \\ .196 \end{bmatrix}$$

Random Walks with Restart

- Example (cont.)

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 2/5 \\ 2/5 \end{bmatrix}, \begin{bmatrix} 35/75 \\ 8/75 \\ 20/75 \\ 6/75 \\ 6/75 \end{bmatrix}, \begin{bmatrix} 95/375 \\ 8/375 \\ 20/375 \\ 142/375 \\ 110/375 \end{bmatrix}, \begin{bmatrix} 2353/5625 \\ 568/5625 \\ 1228/5625 \\ 786/5625 \\ 690/5625 \end{bmatrix}, \ldots, \begin{bmatrix} .345 \\ .066 \\ .145 \\ .249 \\ .196 \end{bmatrix}$$

  - Starts from Picture 1, the walker is more than twice as likely to be at Picture 3 (with probability .145) than Picture 2 (with probability .066).



Additional Topics (for further reading)

- Mining Social-Network Graphs
  - Discover overlapping communities
  - Count triangles
  - Neighborhood properties of graphs.
  - And so on…

Summary

- PageRank for Web Graph
  - PageRank
- Mining Social-Network Graphs
  - Social Networks as Graphs
  - Finding Communities
    - Using Betweeness
    - Finding Complete Bipartie SubGraphs
  - Partitioning of Graphs
  - Similarity between nodes: SimRank