CS249: SPECIAL TOPICS
MINING INFORMATION/SOCIAL NETWORKS

Overview of Networks

Instructor: Yizhou Sun

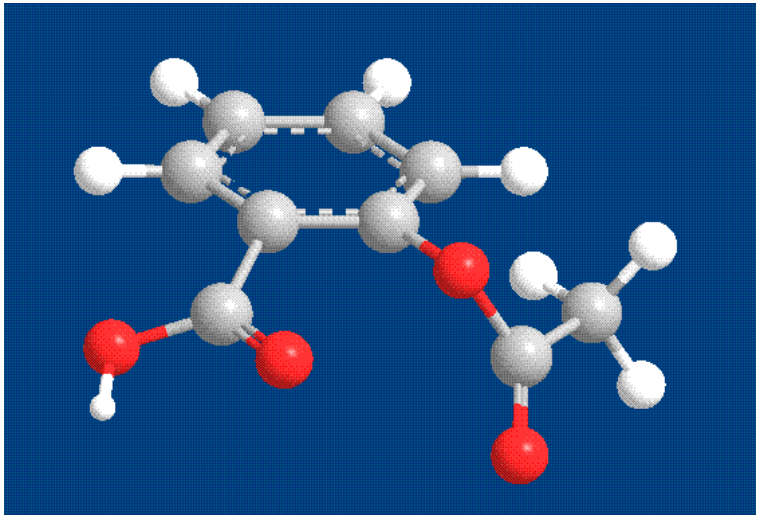
yzsun@cs.ucla.edu

January 10, 2017

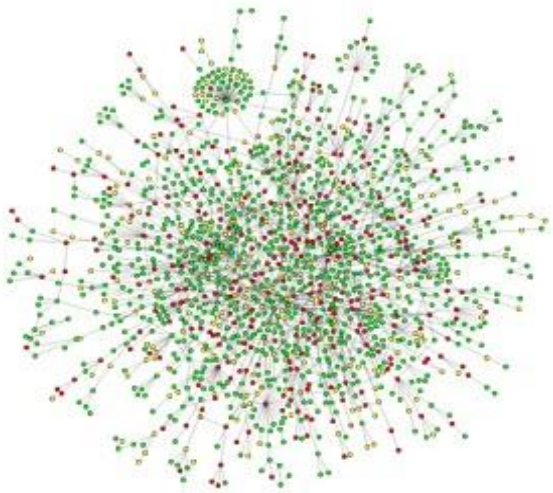
Overview of Information Network Analysis

- Network Representation
- Network Properties
- Network Generative Models
- Random Walk and Its Applications

Networks Are Everywhere

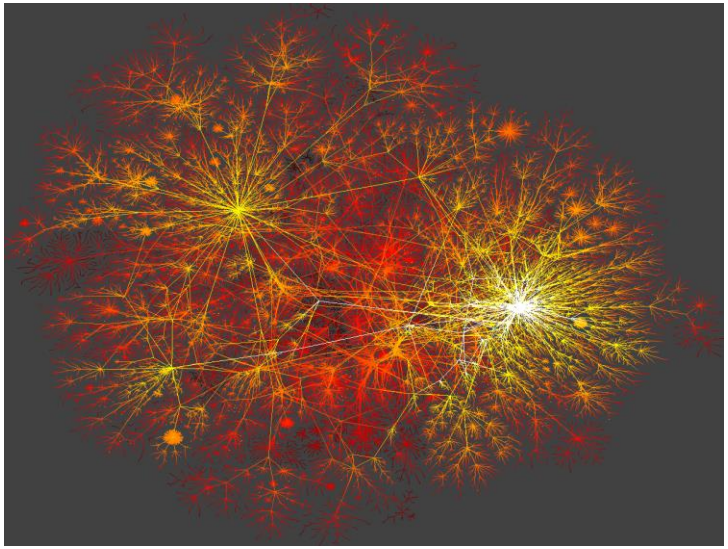


Aspirin

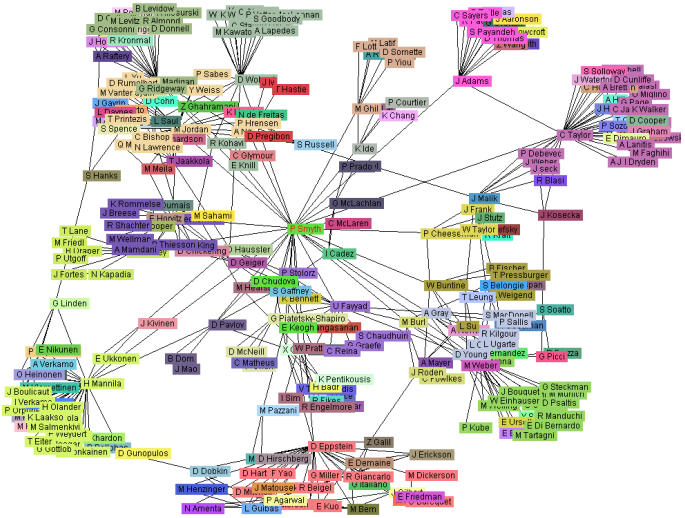


from H. Jeong et al Nature 411, 41 (2001)

Yeast protein interaction network



Internet

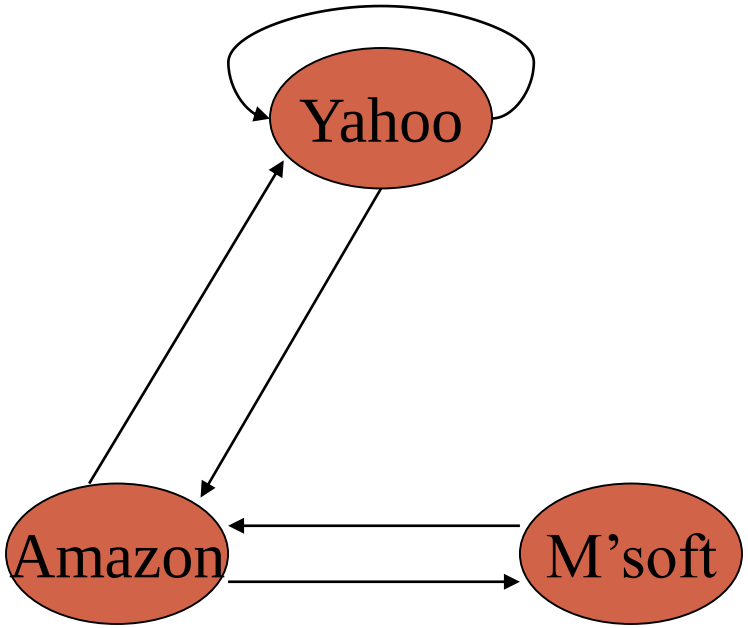


Co-author network

Representation of a Network: Graph

- $G =< V, E >$
  - $V = \{u_1, \ldots, u_n\}$: node set
  - $E \subseteq V \times V$: edge set
- Adjacency matrix
  - $A = \{a_{ij}\}, i, j = 1, \ldots, N$
    - $a_{ij} = 1, if \ < u_i, u_j > \in E$
    - $a_{ij} = 0, if \ < u_i, u_j > \notin E$
- Network types
  - Undirected graph vs. Directed graph
    - $A = A^{\mathrm{T}} \ vs. \ A \neq A^{\mathrm{T}}$
  - Binary graph Vs. Weighted graph
    - Use *W* instead of *A*, where w_{ij} represents the weight of edge $< u_i, u_j >$

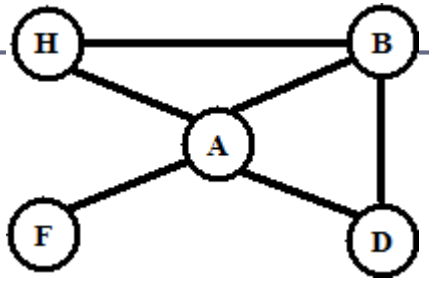
Example



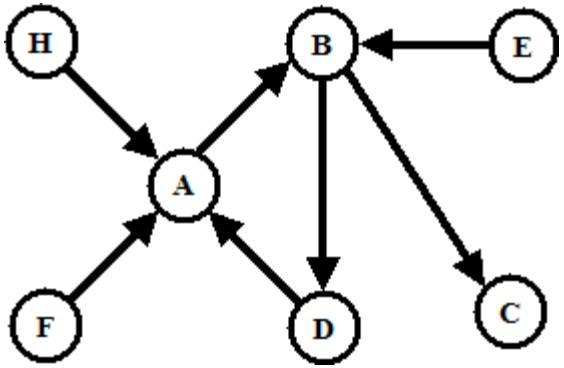
	y	a	m
y	1	1	0
a	1	0	1
m	0	1	0

Adjacency matrix A

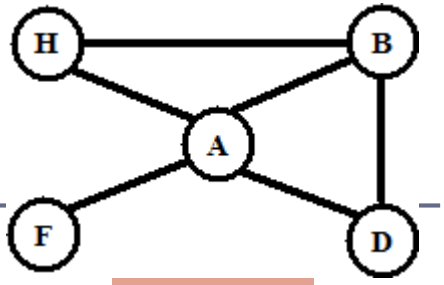
Degree of Nodes



- Let a network G = (V, E)
- Undirected Network
  - Degree (or degree centrality) of a vertex: $d(v_i)$
  - # of edges connected to it, e.g., d(A) = 4, d(H) = 2
- Directed network
  - In-degree of a vertex $d_{in}(v_i)$:
    - # of edges pointing to v_i
    - E.g., $d_{in}(A) = 3$, $d_{in}(B) = 2$
  - Out-degree of a vertex $d_{out}(v_i)$:
    - # of edges from v_i
    - E.g., $d_{out}(A) = 1$, $d_{out}(B) = 2$



Degree Distribution



Graph G_1

- **Degree sequence** of a graph: The list of degrees of the nodes sorted in non-increasing order
  - E.g., in G_1, degree sequence: (4, 3, 2, 2, 1)
- **Degree frequency distribution** of a graph: Let N_k denote the # of vertices with degree k
  - $(N_0, N_1, \ldots, N_t)$, t is max degree for a node in G
  - E.g., in G_1, degree frequency distribution: (0, 1, 2, 1, 1)
- **Degree distribution** of a graph:

  Probability mass function f for random variable X
  - (f(0), f(1), …, f(t), where $f(k) = P(X = k) = N_k/n$
  - E.g., in G_1, degree distrib.: (0, 0.2, 0.4, 0.2, 0.2)

Path

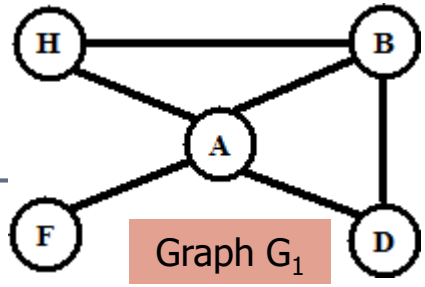
- Path: A sequence of vertices that every consecutive pair of vertices in the sequence is connected by an edge in the network
- Length of a path: # of edges traversed along the path
- Total # of path of length 2 from *j* to *i*, via any vertex in $N_{ij}^{(2)}$ is

$$N_{ij}^{(2)} = \sum_{k=1}^{n} A_{ik} A_{kj} = [A^2]_{ij}$$

- Generalizing to path of arbitrary length, we have:

$$N_{ij}^{(r)} = [A^r]_{ij}$$

Radius and Diameter



Graph G_1

- **Eccentricity**: The eccentricity of a node v_i is the maximum distance from v_i to any other nodes in the graph
  - $e(v_i) = \max_j \{d(v_i, v_j)\}$
  - E.g., $e(A) = 1$, $e(F) = e(B) = e(D) = e(H) = 2$
- **Radius** of a connected graph G: the min eccentricity of any node in G
  - $r(G) = \min_i \{e(v_i)\} = \min_i \{\max_j \{d(v_i, v_j)\}\}$
  - E.g., $r(G_1) = 1$
- **Diameter** of a connected graph G: the max eccentricity of any node in G
  - $d(G) = \max_i \{e(v_i)\} = \max_{i,j} \{d(v_i, v_j)\}$
  - E.g., $d(G_1) = 2$
- Diameter is sensitive to outliers. Effective diameter: min # of hops for which a large fraction, typically 90%, of all connected pairs of nodes can reach each other

Clustering Coefficient

- Real networks are sparse: Corresponding to a complete graph
- Clustering coefficient of a node v_i: A measure of the density of edges in the neighborhood of v_i
- Let $G_i = (V_i, E_i)$ be the subgraph induced by the neighbors of vertex v_i, $|V_i| = n_i$ (# of neighbors of v_i), and $|E_i| = m_i$ (# of edges among the neighbors of v_i)
- **Clustering coefficient** of v_i for **undirected network** is

$$C(v_i) = \frac{\#\ edges\ in\ G_i}{max\ \#\ edges\ in\ G_i} = \frac{m_i}{\binom{n_i}{2}} = \frac{2 \times m_i}{n_i(n_i - 1)}$$

  - For directed network, $$C(v_i) = \frac{\#\ edges\ in\ G_i}{max\ \#\ edges\ in\ G_i} = \frac{m_i}{n_i(n_i - 1)}$$
- Clustering coefficient of a graph G: $$C(G) = \frac{1}{n}\sum_i C(v_i)$$
  - Averaging the local clustering coefficient of all the vertices (Watts & Strogatz)

Overview of Information Network Analysis

- Network Representation
- Network Properties 
- Network Generative Models
- Random Walk and Its Applications

More Than a Graph

- A typical network has the following common properties:
  - *Few* connected components:
    - often only 1 or a small number, independent of network size
  - *Small* diameter:
    - often a constant independent of network size (like 6)
    - growing only logarithmically with network size or even shrink?
  - A *high* degree of clustering:
    - considerably more so than for a random network
  - A *heavy-tailed* degree distribution:
    - a small but reliable number of high-degree vertices
    - often of *power law* form

Sparse

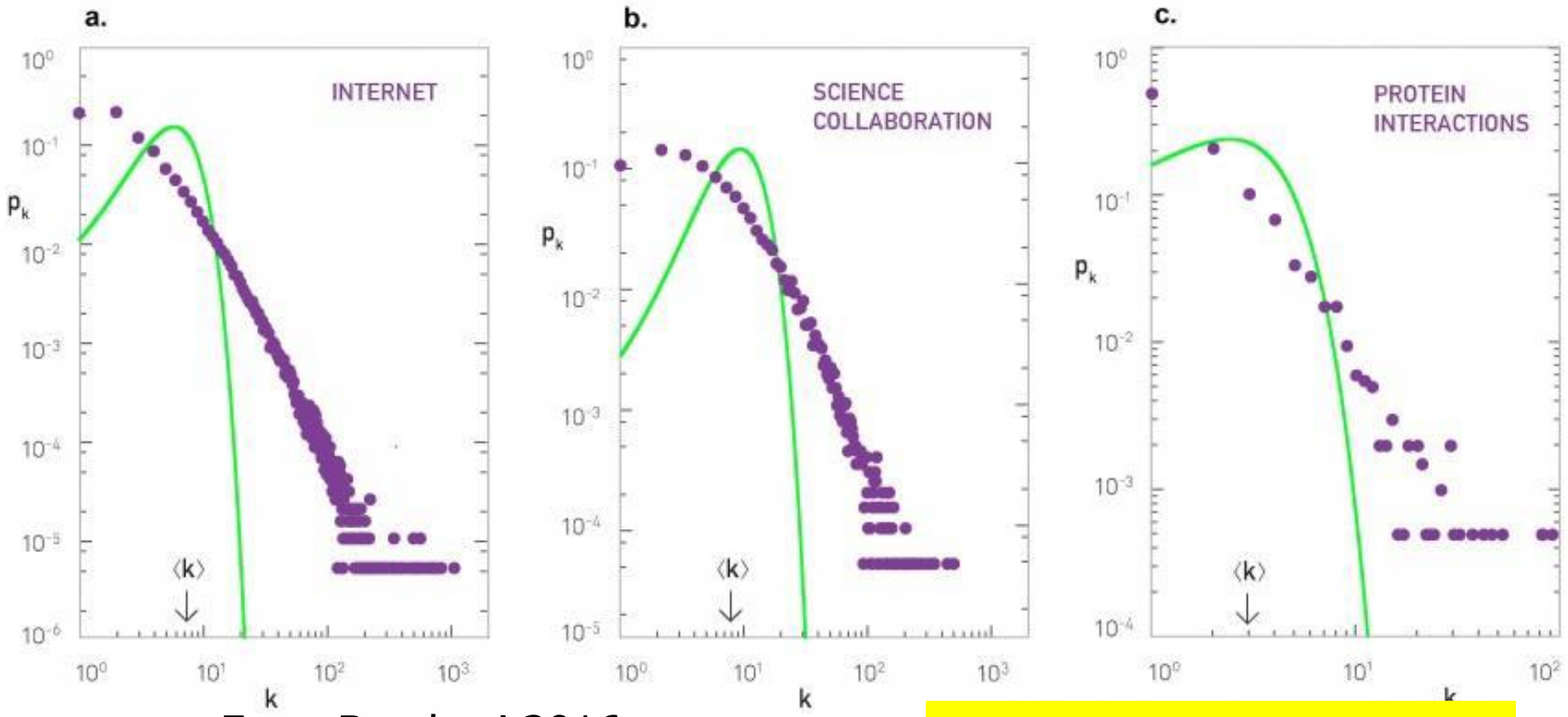
- For complete Graph
  - Average degree: N

- For real-world network
  - Average degree: $\langle k \rangle = 2E/N \ll N$

Small World Property

- Small world phenomenon (Six degrees of separation)
  - Stanley Milgram's experiments (1960s)
  - Microsoft Instant Messaging (IM) experiment: J. Leskovec & E. Horvitz (WWW'08)
    - 240 M active user accounts: Est. avg. distance 6.6 & est. mean median 7
- Why small world?
  - $N(d) \approx 1 + \langle k \rangle + \langle k \rangle^2 + \cdots + \langle k \rangle^d = \frac{\langle k \rangle^{d+1} - 1}{\langle k \rangle - 1} \approx \langle k \rangle^d$
  - E.g., $d \approx \frac{\ln N}{\ln \langle k \rangle} \approx \frac{\ln(7 \times 10^9)}{\ln(10^3)} \approx 3.28$

Degree Distribution: Power Law



From Barabasi 2016

$$f(k) \propto k^{-\gamma}$$

Typically $0 < \gamma < 2$; smaller γ gives heavier tail

The degree distribution of the (a) Internet, (b) science collaboration network, and (c) protein interaction network

High Clustering Coefficient

- Clustering effect: a high clustering coefficient for graph G
  - Friends' friends are likely friends.
  - A lot of triangles
  - C(k): avg clustering coefficient for nodes with degree k

Overview of Information Network Analysis

- Network Representation
- Network Properties
- Network Generative Models 
- Random Walk and Its Applications

Network Generative Models

- All of the network generation models we will study are *probabilistic* or *statistical* in nature
- They can generate networks of any size
- They often have various *parameters* that can be set:
  - size of network generated
  - average degree of a vertex
  - fraction of long-distance connections
- The models generate a *distribution* over networks
- Statements are always *statistical* in nature:
  - *with high probability*, diameter is small
  - on average, degree distribution has heavy tail

Examples

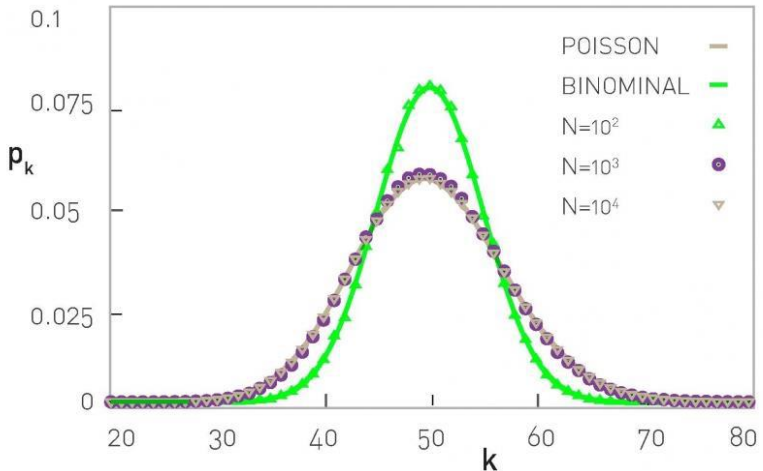
- ***Erdös-Rényi Random graph model:***
  - Gives few components and small diameter
  - does not give high clustering and heavy-tailed degree distributions
  - is the mathematically most well-studied and understood model
- ***Watts-Strogatz small world graph model:***
  - gives few components, small diameter and high clustering
  - does not give heavy-tailed degree distributions
- ***Barabási-Albert Scale-free model:***
  - gives few components, small diameter and heavy-tailed distribution
  - does not give high clustering
- ***Stochastic Block Model***
- ***…***

Erdös-Rényi (ER) Random Graph Model

- **Every possible edge occurs independently with probability *p***
  - *G*(*N, p*): a network of **N** nodes, each node pair is connected with probability of *p*
    - Paul Erdős and Alfréd Rényi: "On Random Graphs" (1959)
    - E. N. Gilbert: "Random Graphs" (1959) (proposed independently)
  - Usually, *N is large* and *p ~ 1/N*
    - Choices: p = 1/2N, p = 1/N, p = 2/N, p = 10/N, p = log(N)/N, etc.

Degree Distribution

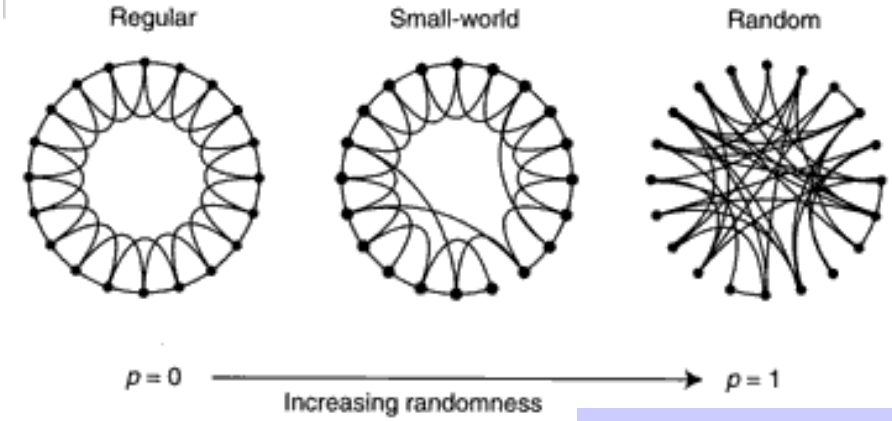
- The degree distribution of a random (small) network follows binomial distribution
  - $$p_k = \binom{N-1}{k} p^k (1-p)^{N-1-k}$$
  - When N is large and Np is fixed, approximated by Poisson distributi $$p_k = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!}$$

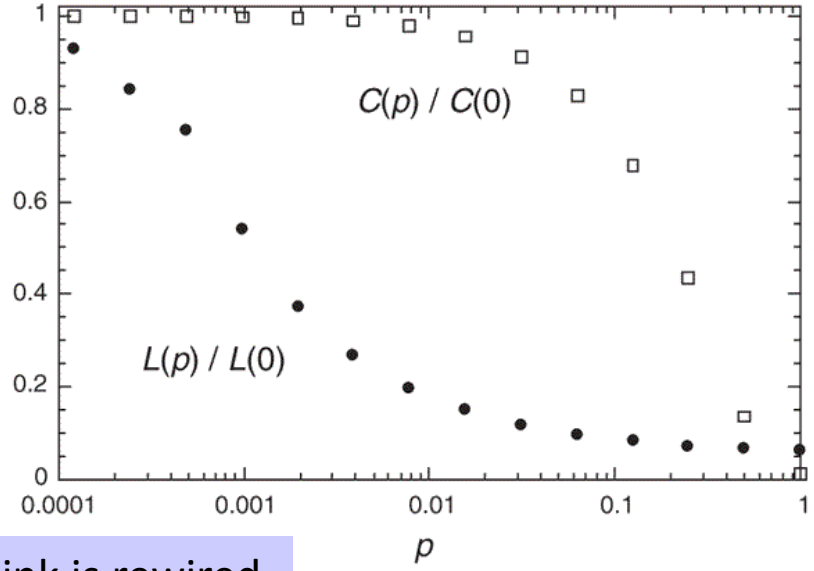


From Barabasi 2016

Watts–Strogatz small world model

- Interpolates between regular lattice and a random network to generate graphs with
  - **Small-world**: short average path lengths
  - **High clustering coefficient:**





p: the prob. each link is rewired to a randomly chosen node

C(p) : clustering coeff.

L(p) : average path length

Barabási-Albert Model: Preferential Attachment

- Major limitation of the Watts-Strogatz model
  - It produces graphs that are homogeneous in degree
  - Real networks are often inhomogeneous in degree, having hubs and a scale-free degree distribution (*scale-free networks*)
- **Scale-free networks** are better described by the ***preferential attachment*** family of models, e.g., the *Barabási–Albert (BA) model*
  - *"rich-get-richer":* New edges are more likely to link to nodes with higher degrees
  - **Preferential attachment:** The probability of connecting to a node is proportional to the current degree of that node
- This leads to the proposal of a new model: ***scale-free network***, a network whose degree distribution follows a ***power law***, at least asymptotically

Overview of Information Network Analysis

- Network Representation
- Network Properties
- Network Generative Models
- Random Walk and Its Applications 

The History of PageRank

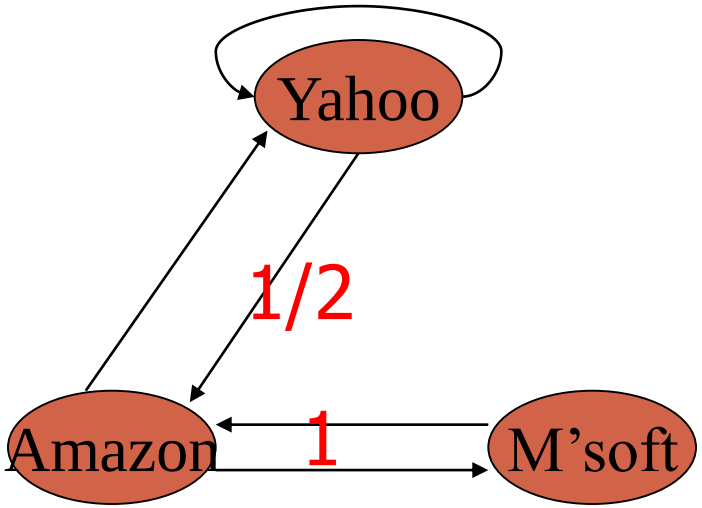
- PageRank was developed by Larry Page (hence the name *Page*-Rank) and Sergey Brin.
- It is first as part of a research project about a new kind of search engine. That project started in 1995 and led to a functional prototype in 1998.
- Shortly after, Page and Brin founded Google.

Ranking web pages

- Web pages are not equally “important”
  - www.cnn.com vs. a personal webpage
- Inlinks as votes
  - The more inlinks, the more important
- Are all inlinks equal?
  - Higher ranked inlink should play a more important role
  - Recursive question!

Simple recursive formulation

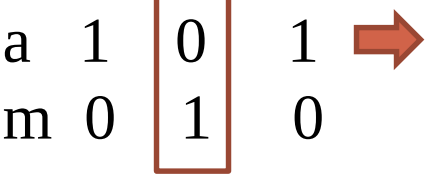
- Each link's vote is proportional to the importance of its source page
- If page P with importance x has n outlinks, each link gets x/n votes
- Page P's own importance is the sum of the votes on its inlinks



Matrix formulation

- Matrix **M** has one row and one column for each web page
- Suppose page j has n outlinks
  - If j -> i, then M_{ij}=1/n
  - Else M_{ij}=0
- **M** is a column stochastic matrix
  - Columns sum to 1
- Suppose **r** is a vector with one entry per web page
  - r_i is the importance score of page i
  - Call it the rank vector
  - |**r**| = 1 (i.e., $r_1 + r_2 + \cdots + r_N = 1$)

	y	a	m
y	1	1	0
a	1	0	1
m	0	1	0



½, 0, 1

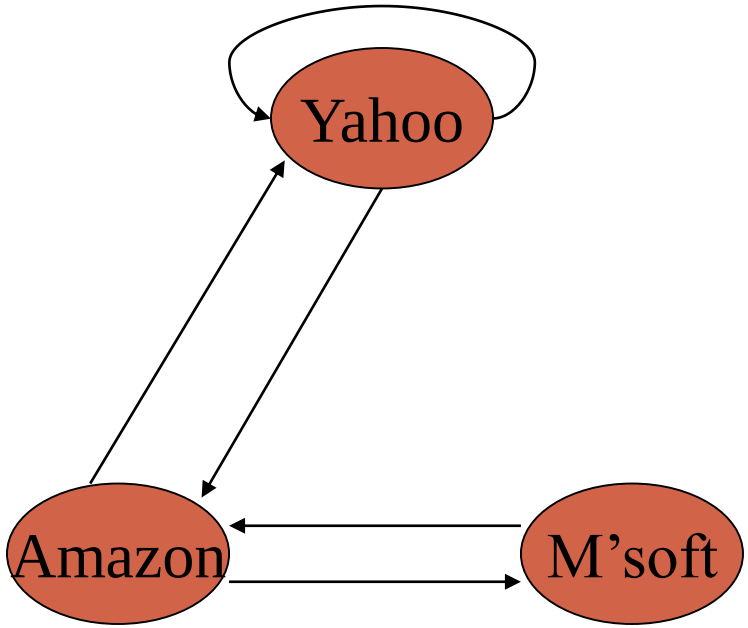
Eigenvector formulation

- The flow equations can be written

$$r = Mr$$

- So the rank vector is an eigenvector of the stochastic web matrix
  - In fact, its first or principal eigenvector, with corresponding eigenvalue 1

Example



$$y = y/2 + a/2$$
$$a = y/2 + m$$
$$m = a/2$$

	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

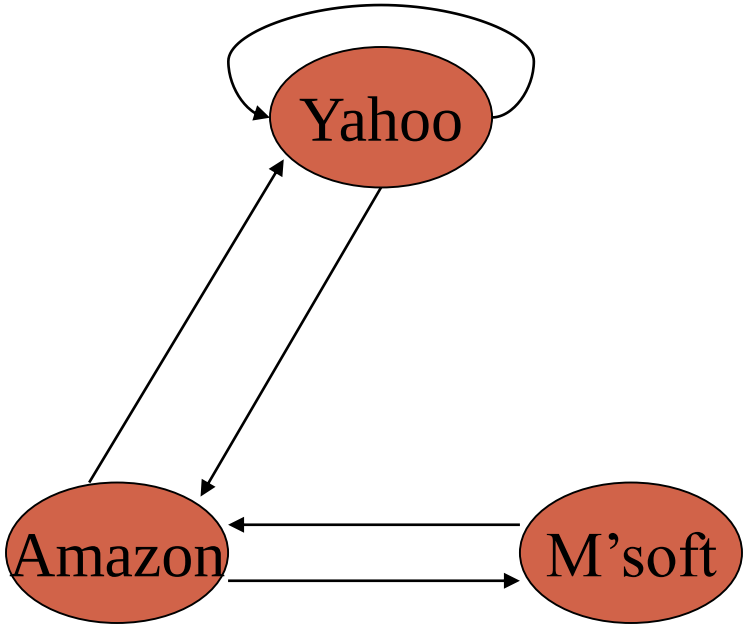
r = M * r

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

Power Iteration method

- Simple iterative scheme
- Suppose there are N web pages
  - Initialize: $\mathbf{r}^0 = [1/N,....,1/N]^T$
  - Iterate: $\mathbf{r}^{k+1} = \mathbf{M}\mathbf{r}^k$
  - Stop when $|\mathbf{r}^{k+1} - \mathbf{r}^k|_1 < \varepsilon$
    - $|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm
    - Can use any other vector norm e.g., Euclidean

Power Iteration Example



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

y		1/3	1/3	5/12	3/8		2/5
a	=	1/3	1/2	1/3	11/24	. . .	2/5
m		1/3	1/6	1/4	1/6		1/5
		r_0	r_1	r_2	r_3	...	r^*

Random Walk Interpretation

- Imagine a random web surfer
  - At any time t, surfer is on some page P
  - At time t+1, the surfer follows an outlink from P uniformly at random
  - Ends up on some page Q linked from P
  - Process repeats indefinitely
- Let **p**(t) be a vector whose i^{th} component is the probability that the surfer is at page i at time t
  - **p**(t) is a probability distribution on pages

The stationary distribution

- Where is the surfer at time t+1?
  - Follows a link uniformly at random
  - $\mathbf{p}(t+1) = \mathbf{M}\mathbf{p}(t)$
- Suppose the random walk reaches a state such that $\mathbf{p}(t+1) = \mathbf{M}\mathbf{p}(t) = \mathbf{p}(t)$
  - Then $\mathbf{p}(t)$ is called a stationary distribution for the random walk
- Our rank vector $\mathbf{r}$ satisfies $\mathbf{r} = \mathbf{M}\mathbf{r}$
  - So it is a stationary distribution for the random surfer

Existence and Uniqueness

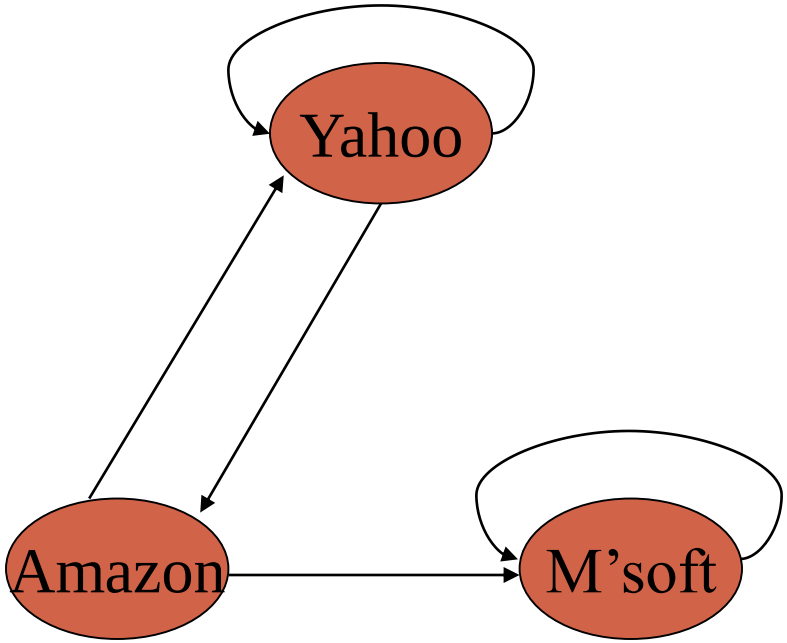
A central result from the theory of random walks (aka Markov processes):

For graphs that satisfy certain conditions, the stationary distribution is unique and eventually will be reached no matter what the initial probability distribution at time t = 0.

Spider traps

- A group of pages is a spider trap if there are no links from within the group to outside the group
  - Random surfer gets trapped
- Spider traps violate the conditions needed for the random walk theorem

Microsoft becomes a spider trap



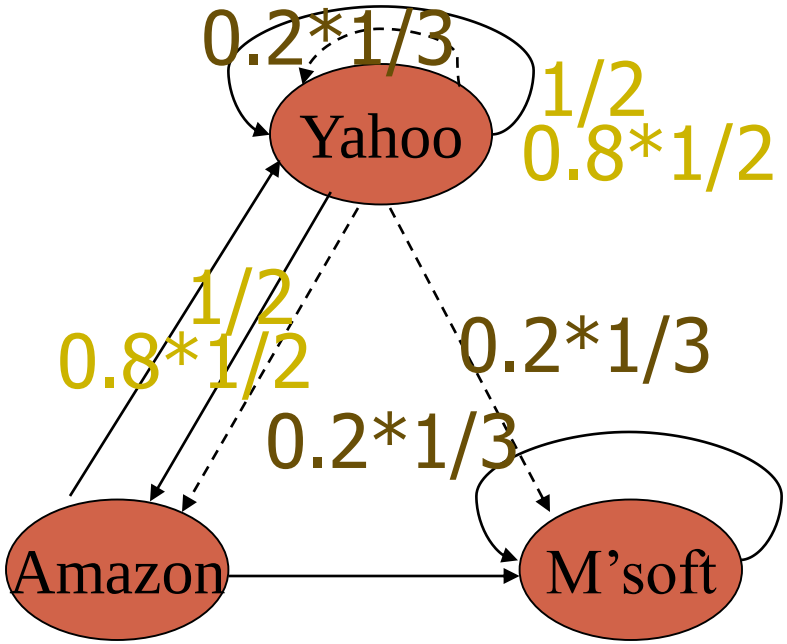
	y	a	m
y	1/2	1/2	0
a	1/2	0	0
m	0	1/2	1

y		1/3	1/3	1/4	5/24		0
a	=	1/3	1/6	1/6	1/8	. . .	0
m		1/3	1/2	7/12	2/3		1

Random teleports

- The Google solution for spider traps
- At each time step, the random surfer has two options:
  - With probability β, follow a link at random
  - With probability 1-β, jump to some page uniformly at random
  - Common values for β are in the range 0.8 to 0.9
- Surfer will teleport out of spider trap within a few time steps

Random teleports ($\beta = 0.8$)



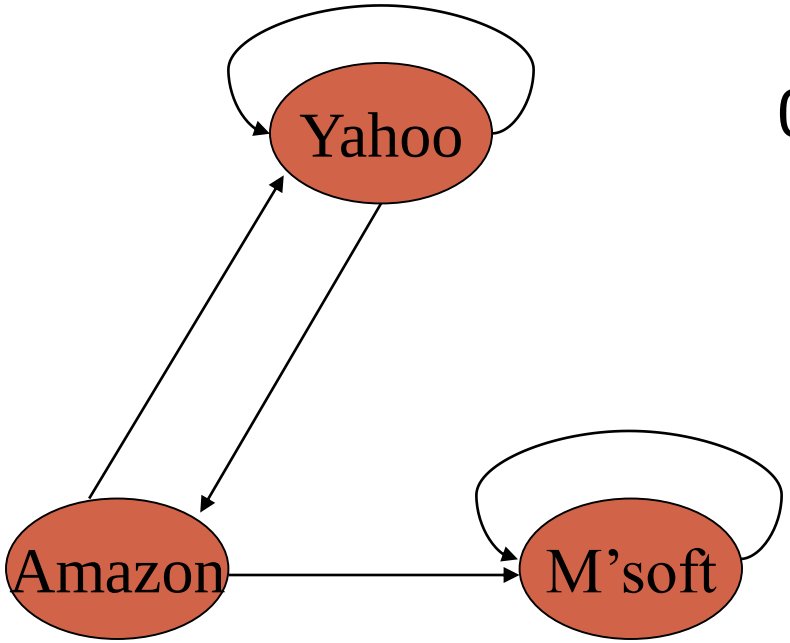
$$\begin{array}{c} \\ y \\ a \\ m \end{array}\begin{array}{c} y \\ \boxed{\begin{array}{c} 1/2 \\ 1/2 \\ 0 \end{array}} \end{array} \quad 0.8\begin{array}{c} y \\ \boxed{\begin{array}{c} 1/2 \\ 1/2 \\ 0 \end{array}} \end{array} \quad + 0.2\begin{array}{c} y \\ \boxed{\begin{array}{c} 1/3 \\ 1/3 \\ 1/3 \end{array}} \end{array}$$

$$0.8\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \quad + 0.2\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

⇢ : teleport links from "Yahoo"

$$\begin{array}{c} \\ y \\ a \\ m \end{array}\begin{array}{c} \\ \boxed{\begin{array}{ccc} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{array}} \end{array}$$

Random teleports ($\beta = 0.8$)



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$\begin{matrix} y \\ a \\ m \end{matrix} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

$$\begin{matrix} y \\ a \\ m \end{matrix} = \begin{bmatrix} 0.333 \\ 0.333 \\ 0.333 \end{bmatrix} \begin{bmatrix} 0.333 \\ 0.200 \\ 0.467 \end{bmatrix} \begin{bmatrix} 0.280 \\ 0.200 \\ 0.520 \end{bmatrix} \begin{bmatrix} 0.259 \\ 0.179 \\ 0.563 \end{bmatrix} \ldots \begin{bmatrix} 7/33 \\ 5/33 \\ 21/33 \end{bmatrix}$$

Matrix formulation

- Suppose there are N pages
  - Consider a page j, with set of outlinks O(j)
  - We have $M_{ij} = 1/|O(j)|$ when j->i and $M_{ij} = 0$ otherwise
  - The random teleport is equivalent to
    - adding a teleport link from j to every other page with probability $(1-\beta)/N$
    - reducing the probability of following each outlink from $1/|O(j)|$ to $\beta/|O(j)|$
    - Equivalent: tax each page a fraction $(1-\beta)$ of its score and redistribute evenly

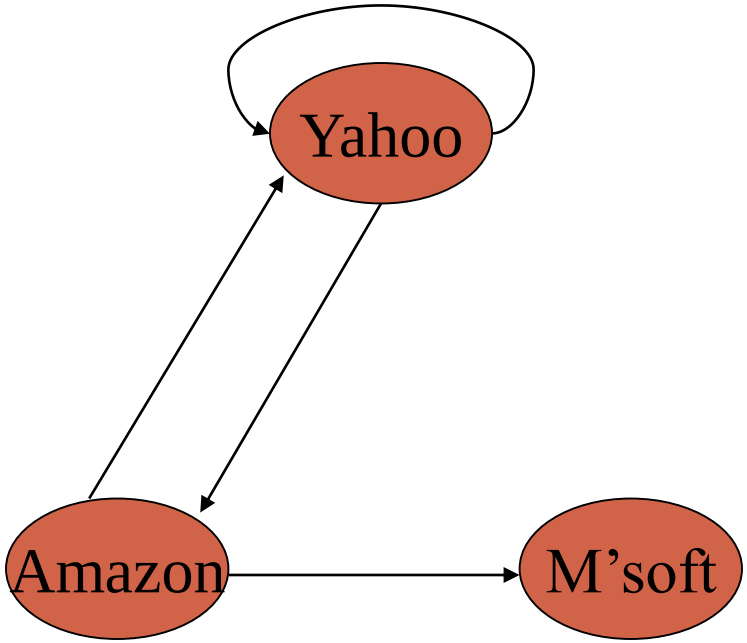
PageRank

- Construct the N-by-N matrix **A** as follows
  - $A_{ij} = \beta M_{ij} + (1-\beta)/N$
- Verify that **A** is a stochastic matrix
- The page rank vector **r** is the principal eigenvector of this matrix
  - satisfying $\mathbf{r} = \mathbf{A}\mathbf{r}$
- Equivalently, **r** is the stationary distribution of the random walk with teleports

Dead ends

- Pages with no outlinks are "dead ends" for the random surfer
  - Nowhere to go on next step

Microsoft becomes a dead end



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$\begin{matrix} y \\ a \\ m \end{matrix} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 1/15 \end{bmatrix}$$

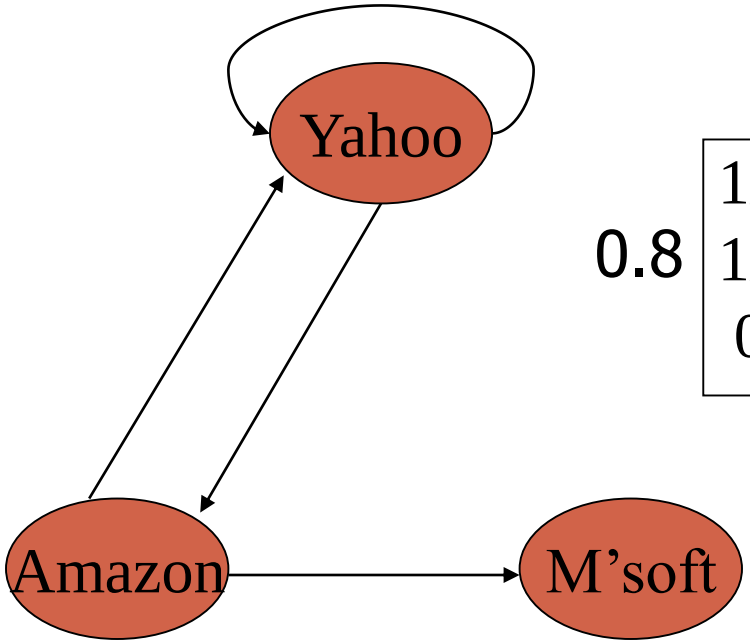
Non-stochastic!

$$\begin{matrix} y \\ a \\ m \end{matrix} = \begin{matrix} 1/3 & 1/3 \\ 1/3 & 0.2 \\ 1/3 & 0.2 \end{matrix} \quad \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \quad \ldots$$

Dealing with dead-ends

- Teleport
  - Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly
- Prune and propagate
  - Preprocess the graph to eliminate dead-ends
    - Might require multiple passes
  - Compute page rank on reduced graph
  - Approximate values for deadends by propagating values from reduced graph

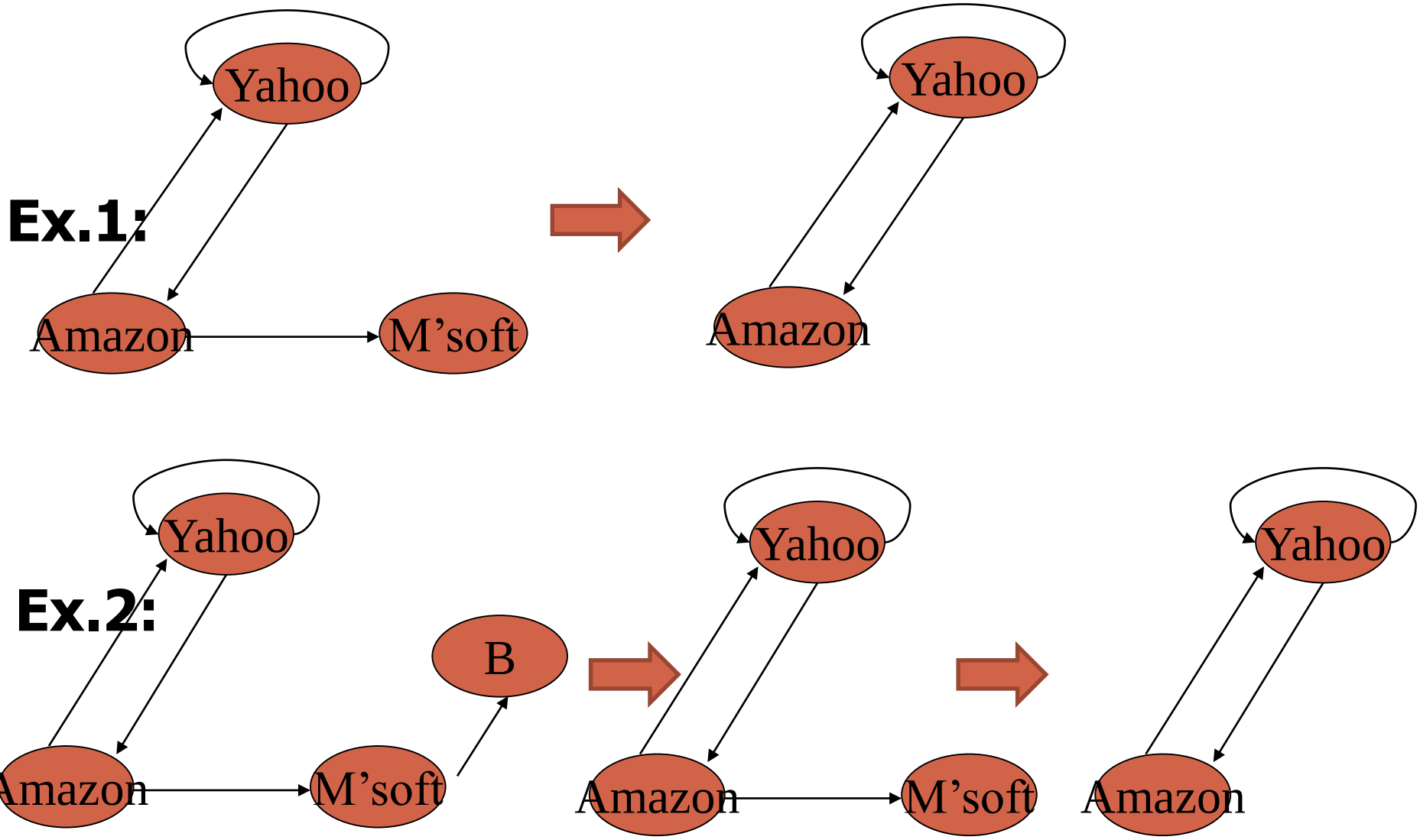
Dealing dead end: teleport



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} + \begin{bmatrix} 0.2*1/3 & 0.2*1/3 & 1*1/3 \\ 0.2*1/3 & 0.2*1/3 & 1*1/3 \\ 0.2*1/3 & 0.2*1/3 & 1*1/3 \end{bmatrix}$$

$$\begin{matrix} y \\ a \\ m \end{matrix} \begin{bmatrix} 7/15 & 7/15 & 1/3 \\ 7/15 & 1/15 & 1/3 \\ 1/15 & 7/15 & 1/3 \end{bmatrix}$$

Dealing dead end: reduce graph



Computing PageRank

- Key step is matrix-vector multiplication
  - $\mathbf{r}^{new} = \mathbf{A}\mathbf{r}^{old}$
- Easy if we have enough main memory to hold $\mathbf{A}$, $\mathbf{r}^{old}$, $\mathbf{r}^{new}$
- Say N = 1 billion pages
  - We need 4 bytes for each entry (say)
  - 2 billion entries for vectors, approx 8GB
  - Matrix A has N^2 entries
    - 10^{18} is a large number!

Rearranging the equation

$\mathbf{r} = \mathbf{A}\mathbf{r}$, where

$A_{ij} = \beta M_{ij} + (1-\beta)/N$

$r_i = \sum_{1 \le j \le N} A_{ij} r_j$

$r_i = \sum_{1 \le j \le N} [\beta M_{ij} + (1-\beta)/N] r_j$

$= \beta \sum_{1 \le j \le N} M_{ij} r_j + (1-\beta)/N \sum_{1 \le j \le N} r_j$

$= \beta \sum_{1 \le j \le N} M_{ij} r_j + (1-\beta)/N$, since $|\mathbf{r}| = 1$

$\mathbf{r} = \beta \mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$

where $[x]_N$ is an N-vector with all entries x

Sparse matrix formulation

- We can rearrange the page rank equation:
  - $\mathbf{r} = \beta\mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$
  - $[(1-\beta)/N]_N$ is an N-vector with all entries $(1-\beta)/N$
- **M** is a sparse matrix!
  - 10 links per node, approx 10N entries
- So in each iteration, we need to:
  - Compute $\mathbf{r}^{new} = \beta\mathbf{M}\mathbf{r}^{old}$
  - Add a constant value $(1-\beta)/N$ to each entry in $\mathbf{r}^{new}$

Sparse matrix encoding

- Encode sparse matrix using only nonzero entries
  - Space proportional roughly to number of links
  - say 10N, or 4*10*1 billion = 40GB
  - still won't fit in memory, but will fit on disk

source node	degree	destination nodes
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

Basic Algorithm

- Assume we have enough RAM to fit $\mathbf{r}^{new}$, plus some working memory
  - Store $\mathbf{r}^{old}$ and matrix $\mathbf{M}$ on disk

Basic Algorithm:

- Initialize: $\mathbf{r}^{old} = [1/N]_N$
- Iterate:
  - Update: Perform a sequential scan of $\mathbf{M}$ and $\mathbf{r}^{old}$ to update $\mathbf{r}^{new}$
  - Write out $\mathbf{r}^{new}$ to disk as $\mathbf{r}^{old}$ for next iteration
  - Every few iterations, compute $|\mathbf{r}^{new}-\mathbf{r}^{old}|$ and stop if it is below threshold
    - Need to read in both vectors into memory

Summary

- Network Representation
- Network Properties
- Network Generative Models
- Random Walk and Its Applications

Paper Sign-Up

- https://docs.google.com/spreadsheets/d/1SaoPGP2SsYyaycX82T7mF_efbiueOI53bnZtZS04BtQ/edit?usp=sharing
- If you are still on waiting list
  - Sign-up for Presenter 4 only

Credits

- This is 4-credit course, please change it if you are current enrolled with 2-credit

Course Project Examples

- Citation graph summary
  - Find k papers that can tell the main structure evolution of a certain field
- Name disambiguation problem in DBLP
  - Different people may share the same name, e.g., distinguish "Wei Wang"'s;
  - Same person may have different forms of names, e.g., initials, middle names, typos

- User profile prediction in heterogeneous information networks
  - Suppose we only know small number of labels for people's ideology, profession, education, can we predict the remaining?

- Sentence embedding
  - Can we find the most similar sentences or S-V-O (subject-verb-object) triplets to the given one, by converting the text into a network?