

Analysis of Large Graphs: Link Analysis, PageRank

Advanced Search Techniques for Large Scale Data Analytics

Pavel Zezula and Jan Sedmidubsky

Masaryk University

<http://disa.fi.muni.cz>

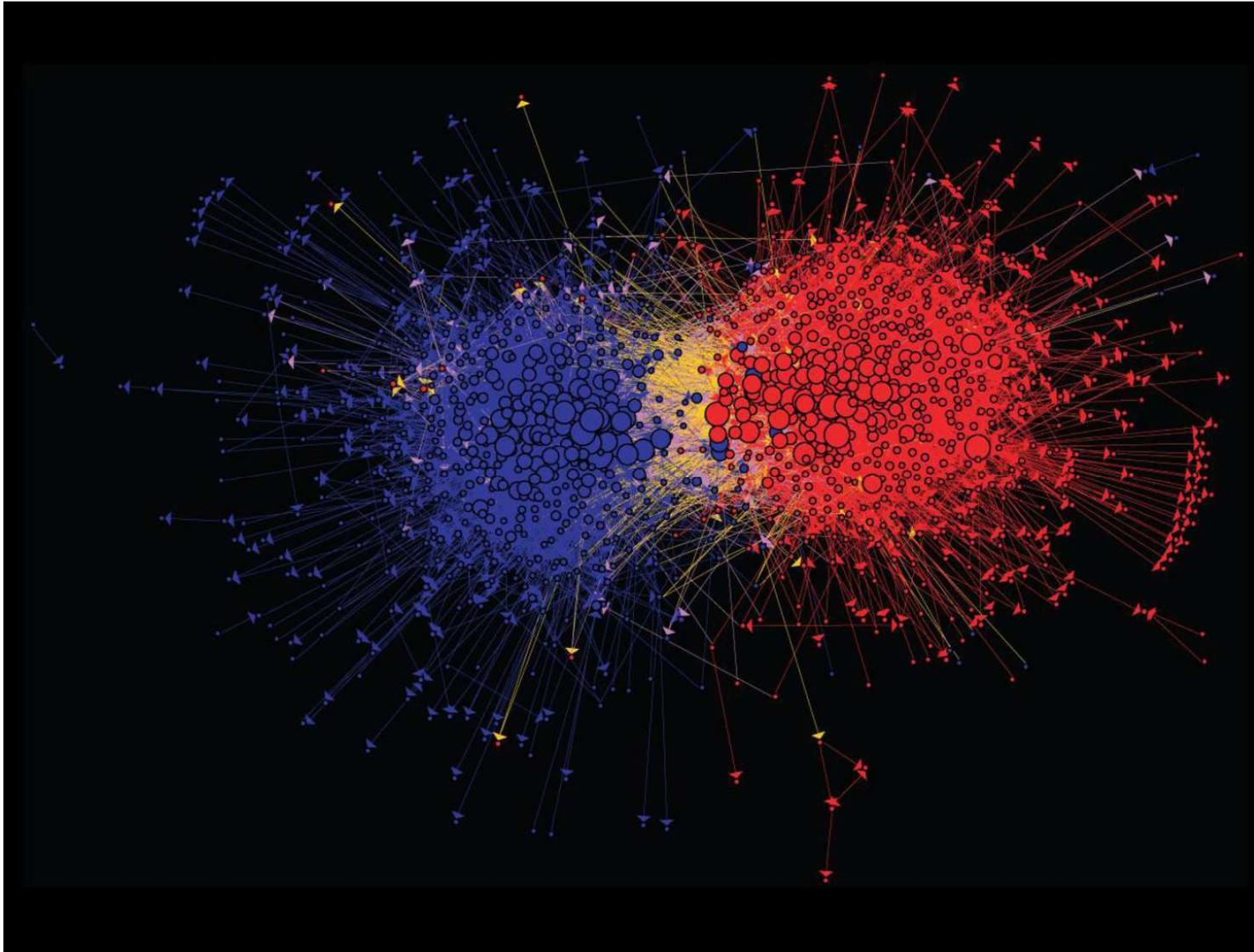
Graph Data: Social Networks



Facebook social graph

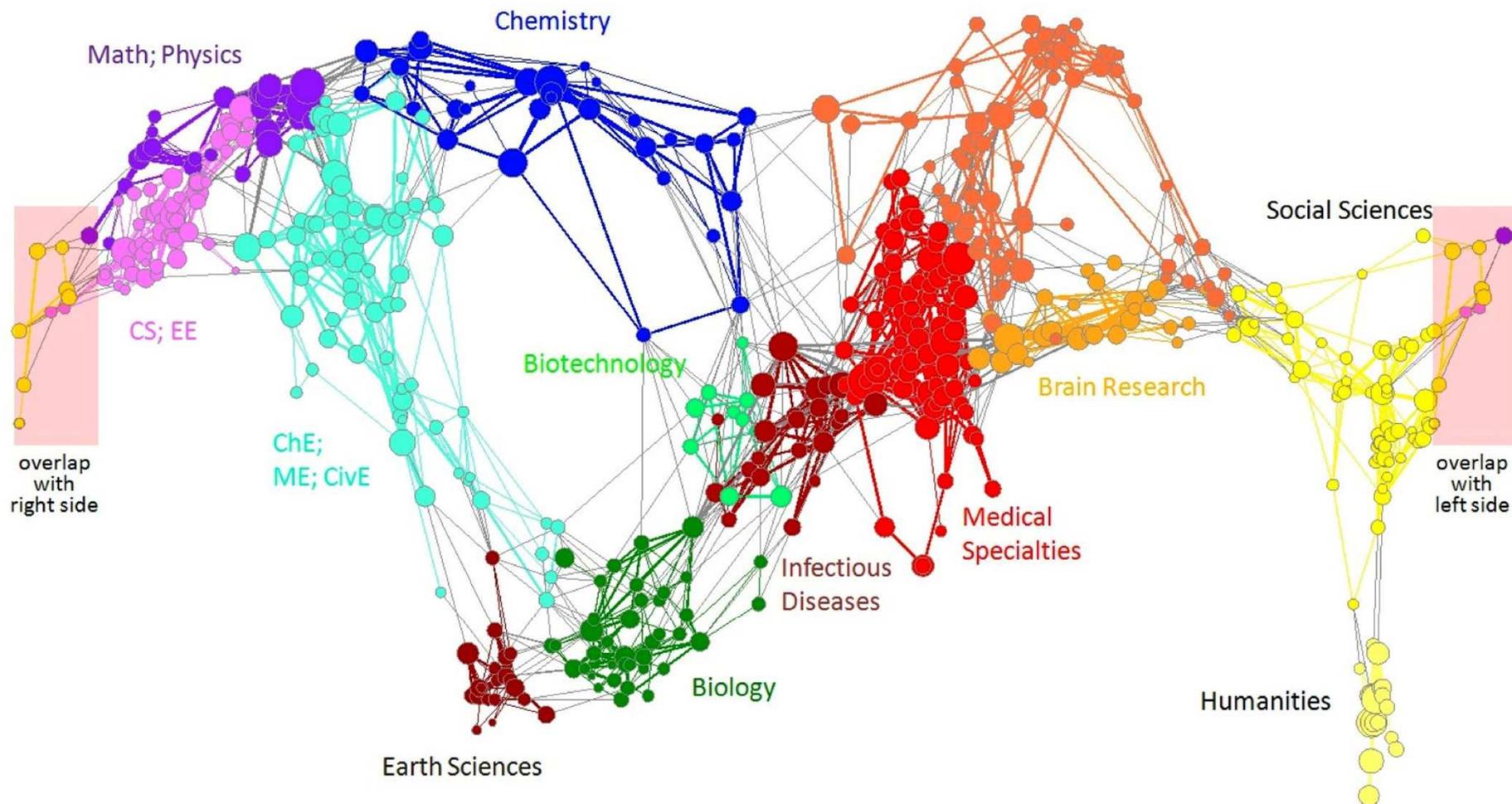
4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

Graph Data: Media Networks



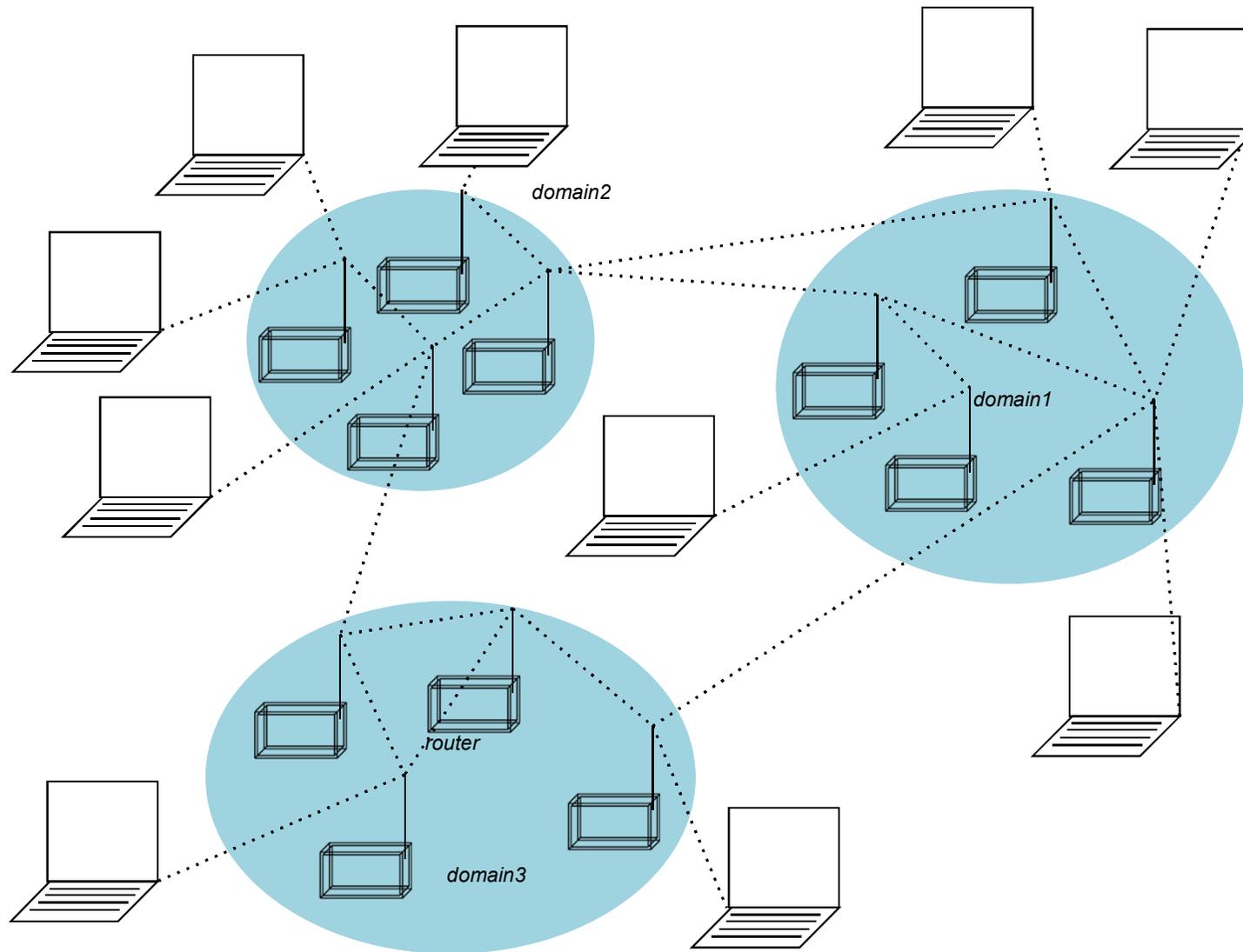
Connections between political blogs
Polarization of the network [Adamic-Glance, 2005]

Graph Data: Information Nets



Citation networks and Maps of science
[Börner et al., 2012]

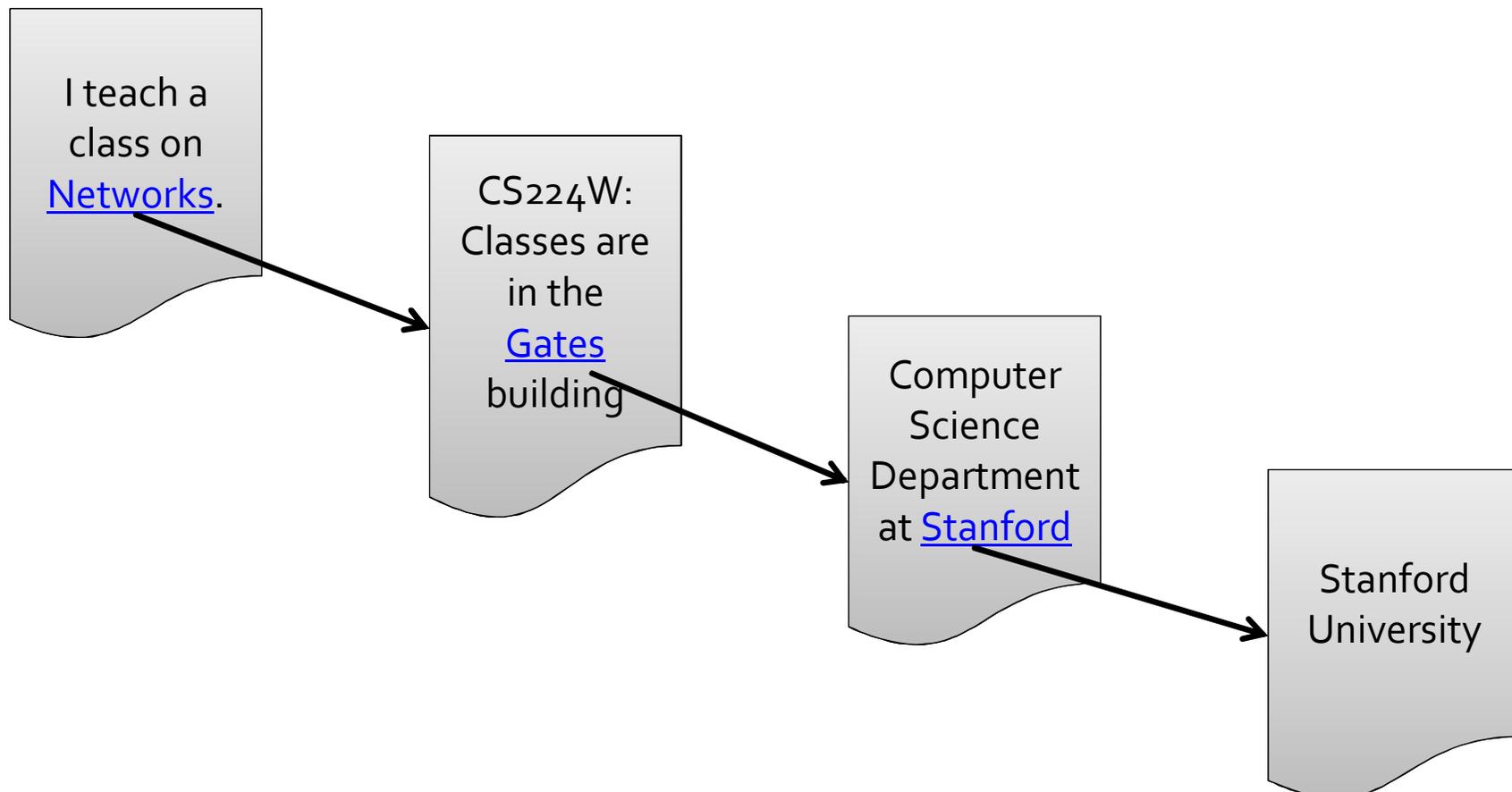
Graph Data: Communication Nets



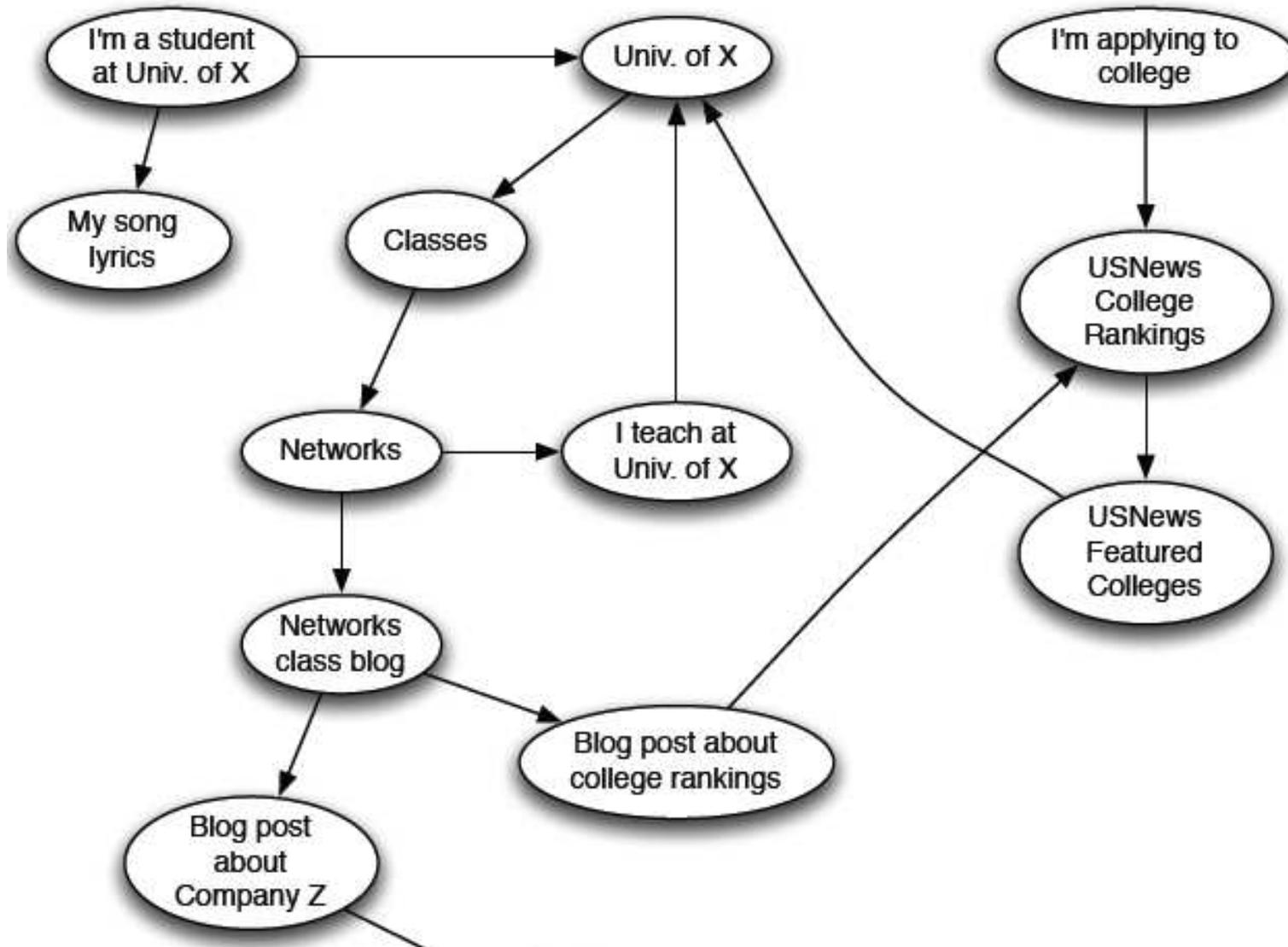
Internet

Web as a Graph

- **Web as a directed graph:**
 - **Nodes: Webpages**
 - **Edges: Hyperlinks**



Web as a Directed Graph



Broad Question

- **How to organize the Web?**

- **First try: Human curated Web directories**

- Yahoo, DMOZ, LookSmart

- **Second try: Web Search**

- **Information Retrieval** investigates:

Find relevant docs in a small and trusted set

- Newspaper articles, Patents, etc.

- **But:** Web is **huge**, full of untrusted documents, random things, web spam, etc.



Web Search: 2 Challenges

2 challenges of web search:

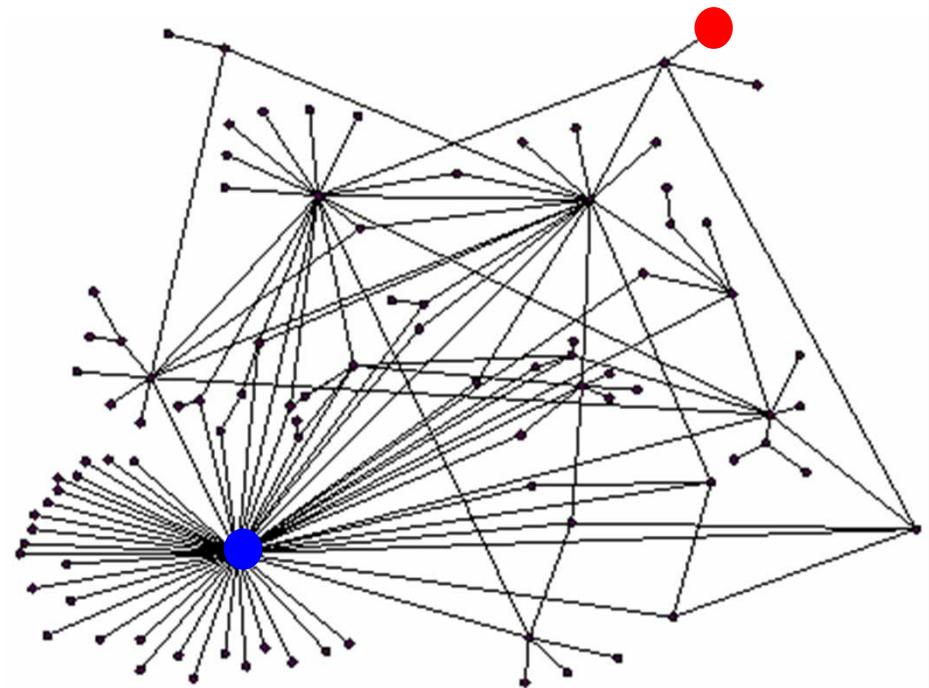
- (1) Web contains many sources of information
Who to “trust”?
 - **Trick:** Trustworthy pages may point to each other!
- (2) What is the “best” answer to query
“newspaper”?
 - No single right answer
 - **Trick:** Pages that actually know about newspapers might all be pointing to many newspapers

Ranking Nodes on the Graph

- All web pages are not equally “important”

www.joe-schmoe.com vs. www.stanford.edu

- There is large diversity in the web-graph node connectivity.
Let's rank the pages by the link structure!



Link Analysis Algorithms

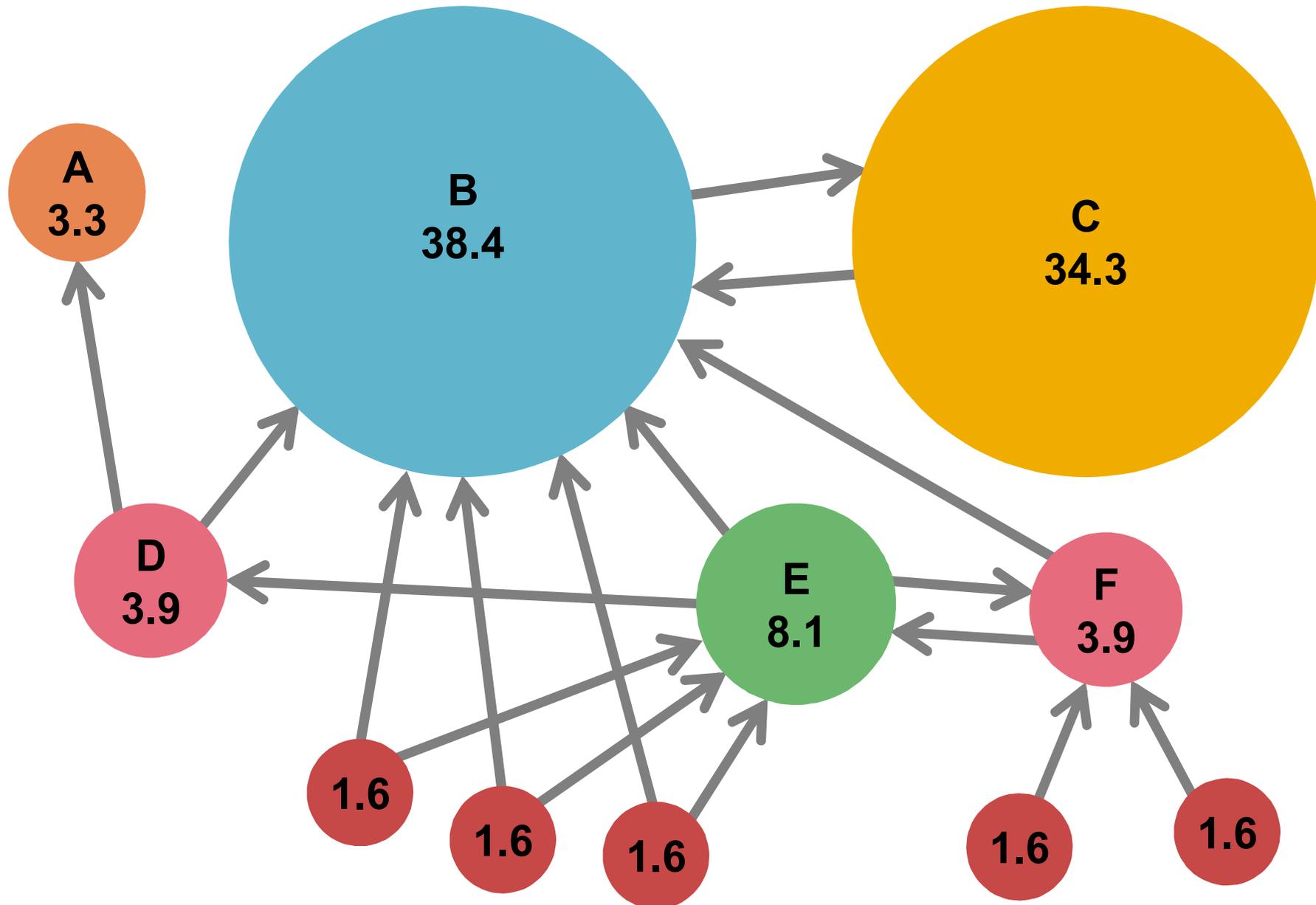
- We will cover the following **Link Analysis approaches** for computing **importances** of nodes in a graph:
 - Page Rank
 - Topic-Specific (Personalized) Page Rank
 - Web Spam Detection Algorithms

PageRank: The “Flow” Formulation

Links as Votes

- **Idea: Links as votes**
 - Page is more important if it has more links
 - In-coming links? Out-going links?
- **Think of in-links as votes:**
 - www.stanford.edu has 23,400 in-links
 - www.joe-schmoe.com has 1 in-link
- **Are all in-links are equal?**
 - Links from important pages count more
 - Recursive question!

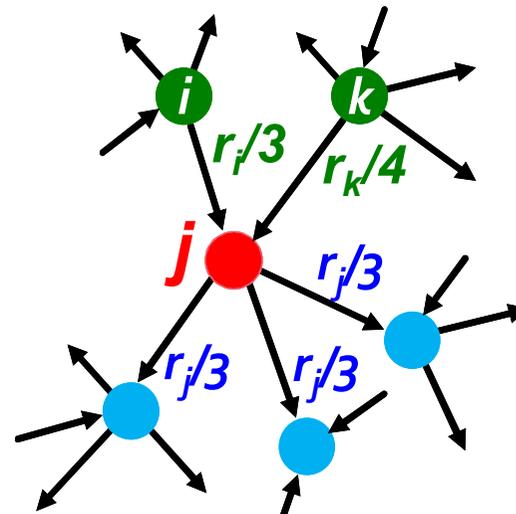
Example: PageRank Scores



Simple Recursive Formulation

- Each link's vote is proportional to the **importance** of its source page
- If page j with importance r_j has n out-links, each link gets r_j/n votes
- Page j 's own importance is the sum of the votes on its in-links

$$r_j = r_i/3 + r_k/4$$



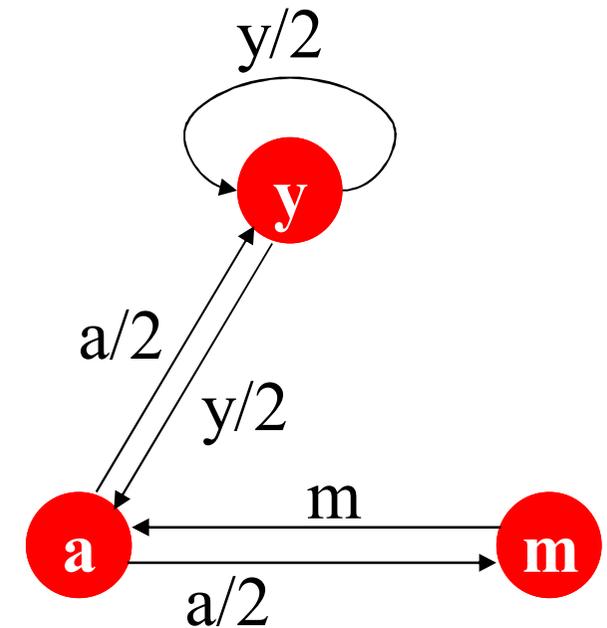
PageRank: The “Flow” Model

- A “vote” from an important page is worth more
- A page is important if it is pointed to by other important pages
- Define a “rank” r_j for page j

$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

... out-degree of node

The web in 1839



“Flow” equations:

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

Solving the Flow Equations

- **3 equations, 3 unknowns, no constants**

- No unique solution
- All solutions equivalent modulo the scale factor

- **Additional constraint forces uniqueness:**

-

- **Solution:** – – –

- **Gaussian elimination method works for small examples, but we need a better method for large web-size graphs**
- **We need a new formulation!**

Flow equations:

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

PageRank: Matrix Formulation

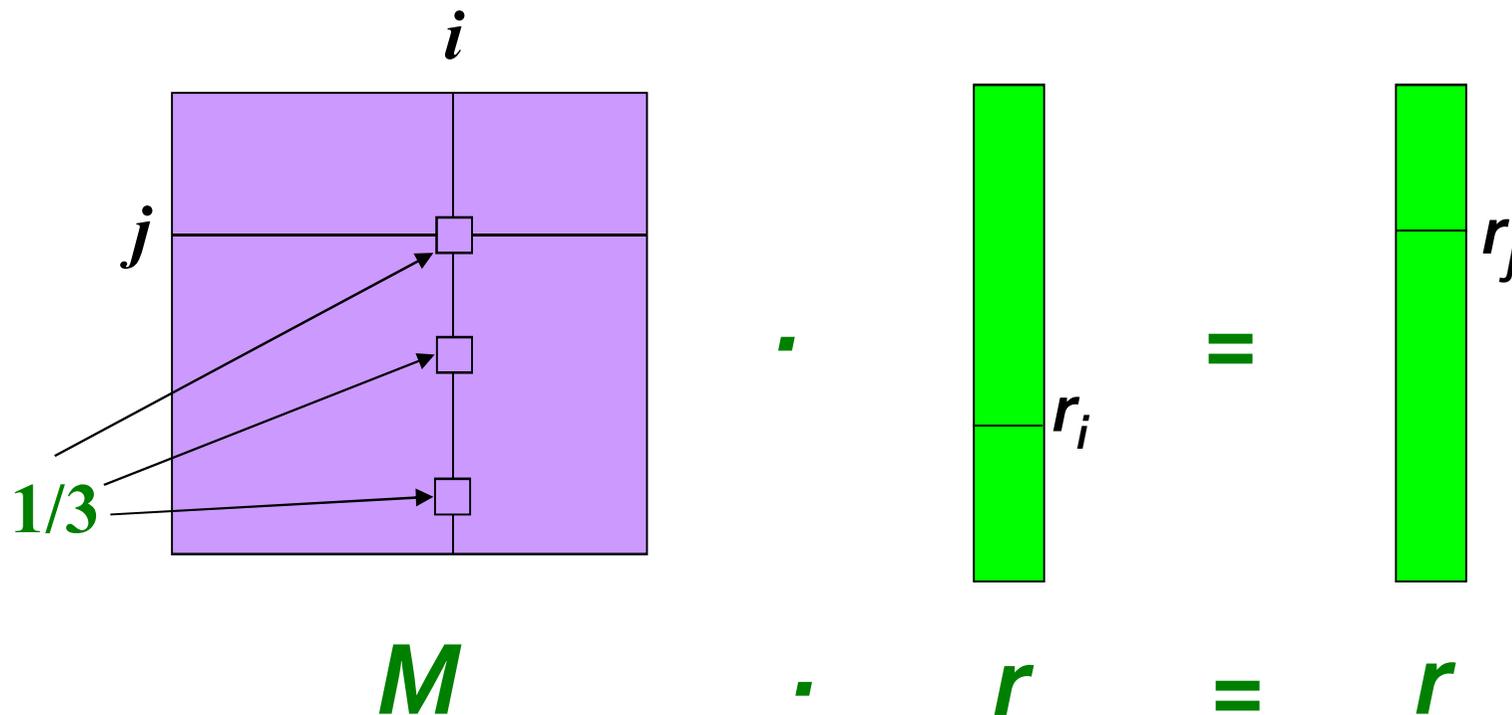
- **Stochastic adjacency matrix**
 - Let page i has d_i out-links
 - If $i \rightarrow j$, then $A_{ji} = \frac{1}{d_i}$ — else $A_{ji} = 0$
 - A is a **column stochastic matrix**
 - Columns sum to 1
- **Rank vector** r : vector with an entry per page
 - r_i is the importance score of page i
 - $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- **The flow equations can be written**

$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

Example

- Remember the flow equation: $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- Flow equation in the matrix form

- Suppose page i links to 3 pages, including j

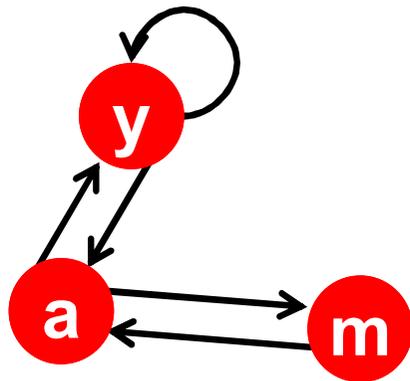


Eigenvector Formulation

- The flow equations can be written
- So the rank vector r is an eigenvector of the stochastic web matrix M
 - In fact, its first or principal eigenvector, with corresponding eigenvalue 1
 - Largest eigenvalue of M is 1 since M is column stochastic (with non-negative entries)
 - *We know r is unit length and each column of M sums to one, so*
- We can now efficiently solve for r !
The method is called Power iteration

NOTE: x is an eigenvector with the corresponding eigenvalue λ if:

Example: Flow Equations & M



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

$$r = M \cdot r$$

$$r_y = r_y / 2 + r_a / 2$$

$$r_a = r_y / 2 + r_m$$

$$r_m = r_a / 2$$

$$\begin{array}{|c|} \hline y \\ \hline a \\ \hline m \\ \hline \end{array} = \begin{array}{|ccc|} \hline 1/2 & 1/2 & 0 \\ \hline 1/2 & 0 & 1 \\ \hline 0 & 1/2 & 0 \\ \hline \end{array} \begin{array}{|c|} \hline y \\ \hline a \\ \hline m \\ \hline \end{array}$$

Power Iteration Method

- Given a web graph with n nodes, where the nodes are pages and edges are hyperlinks
- **Power iteration:** a simple iterative scheme

- Suppose there are N web pages

- Initialize: $\mathbf{r}^{(0)} = [1/N, \dots, 1/N]^T$

- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$

- Stop when $\|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}\|_1 < \varepsilon$

$\|\mathbf{x}\|_1 = \sum_{1 \leq i \leq N} |x_i|$ is the L_1 norm

Can use any other vector norm, e.g., Euclidean

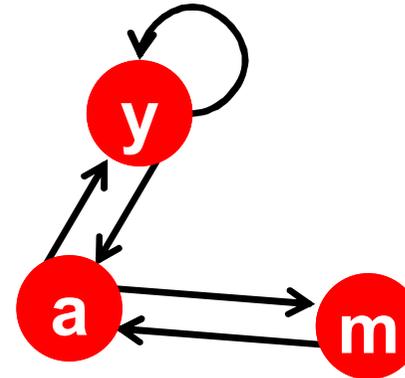
$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

d_i out-degree of node i

PageRank: How to solve?

■ Power Iteration:

- Set $\mathbf{r} = \mathbf{1}/N$
- 1: $\mathbf{r} = \mathbf{M}\mathbf{r}$
- 2: $\mathbf{r} = \mathbf{r} / \sum \mathbf{r}$
- Goto 1



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

$$\mathbf{r}_y = \mathbf{r}_y/2 + \mathbf{r}_a/2$$

$$\mathbf{r}_a = \mathbf{r}_y/2 + \mathbf{r}_m$$

$$\mathbf{r}_m = \mathbf{r}_a/2$$

■ Example:

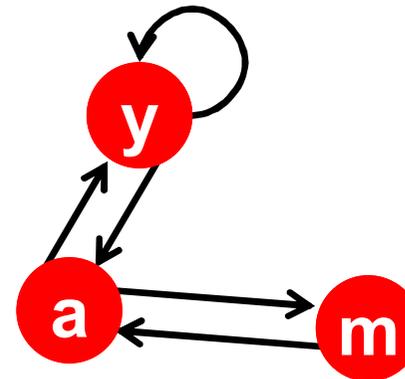
$$\begin{pmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{pmatrix} = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}$$

Iteration 0, 1, 2, ...

PageRank: How to solve?

■ Power Iteration:

- Set $r_i = 1/N$
- 1: $r_i = \sum_j r_j A_{ji}$
- 2: $r_i = r_i / \sum_j r_j$
- Goto 1



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

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

■ Example:

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{pmatrix} 1/3 & 1/3 & 5/12 & 9/24 & & 6/15 \\ 1/3 & 3/6 & 1/3 & 11/24 & \dots & 6/15 \\ 1/3 & 1/6 & 3/12 & 1/6 & & 3/15 \end{pmatrix}$$

Iteration 0, 1, 2, ...

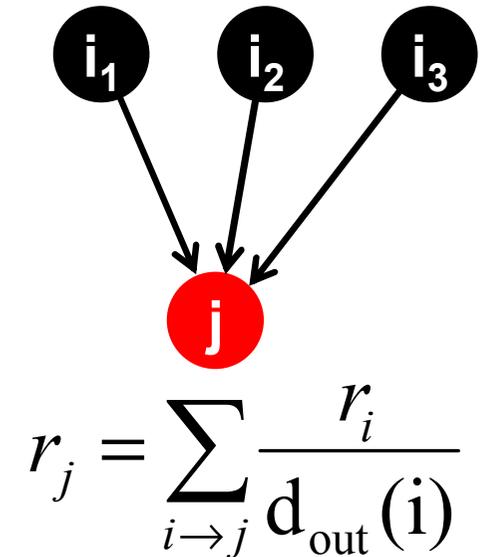
Random Walk Interpretation

- **Imagine a random web surfer:**

- At any time t , surfer is on some page i
- At time $t+1$, the surfer follows an out-link from i uniformly at random
- Ends up on some page j linked from i
- Process repeats indefinitely

- **Let:**

- \mathbf{r} ... vector whose i th coordinate is the prob. that the surfer is at page i at time t
- So, \mathbf{r} is a probability distribution over pages

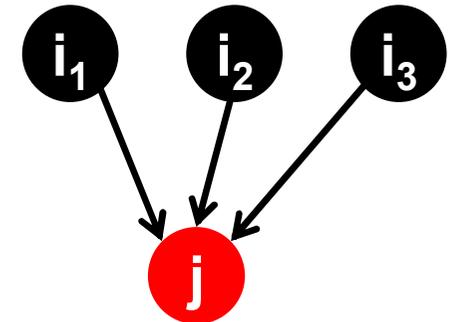


The Stationary Distribution

- **Where is the surfer at time $t+1$?**

- Follows a link uniformly at random

()



$$p(t+1) = M \cdot p(t)$$

- Suppose the random walk reaches a state

()

then () is **stationary distribution** of a random walk

- **Our original rank vector** satisfies

- **So, () is a stationary distribution for the random walk**

Existence and Uniqueness

- **A central result from the theory of random walks (a.k.a. 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 **$t = 0$**

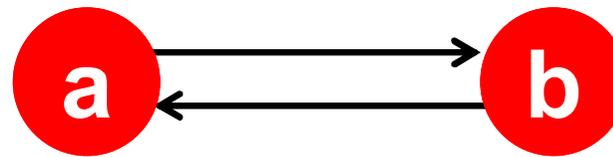
PageRank: The Google Formulation

PageRank: Three Questions

$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i} \quad \text{or equivalently} \quad \mathbf{r} = \mathbf{M}\mathbf{r}$$

- Does this converge?
- Does it converge to what we want?
- Are results reasonable?

Does this converge?



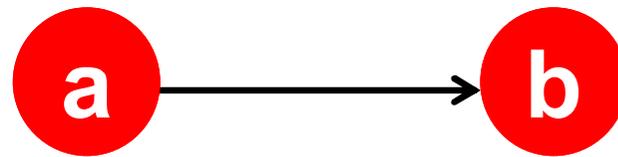
$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

■ Example:

$$\begin{array}{l} r_a \\ r_b \end{array} = \begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}$$

Iteration 0, 1, 2, ...

Does it converge to what we want?



$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

■ Example:

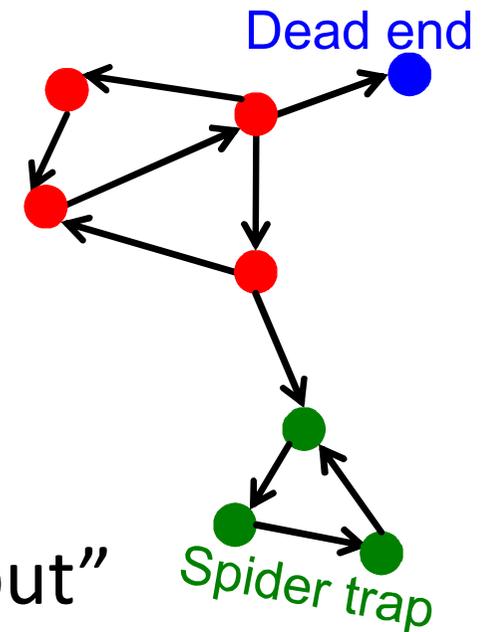
$$\begin{array}{l} r_a \\ r_b \end{array} = \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}$$

Iteration 0, 1, 2, ...

PageRank: Problems

2 problems:

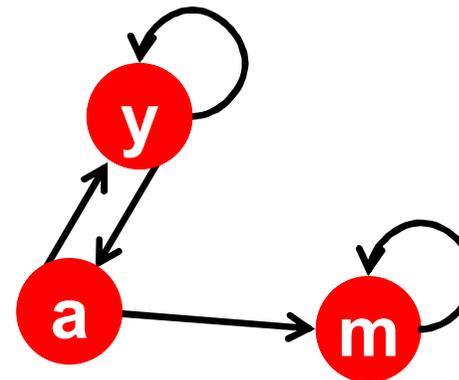
- **(1) Some pages are dead ends** (have no out-links)
 - Random walk has “nowhere” to go to
 - Such pages cause importance to “leak out”
- **(2) Spider traps:** (all out-links are within the group)
 - Random walked gets “stuck” in a trap
 - And eventually spider traps absorb all importance



Problem: Spider Traps

Power Iteration:

- Set
- —
- And iterate



m is a spider trap

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

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2$$

$$r_m = r_a/2 + r_m$$

Example:

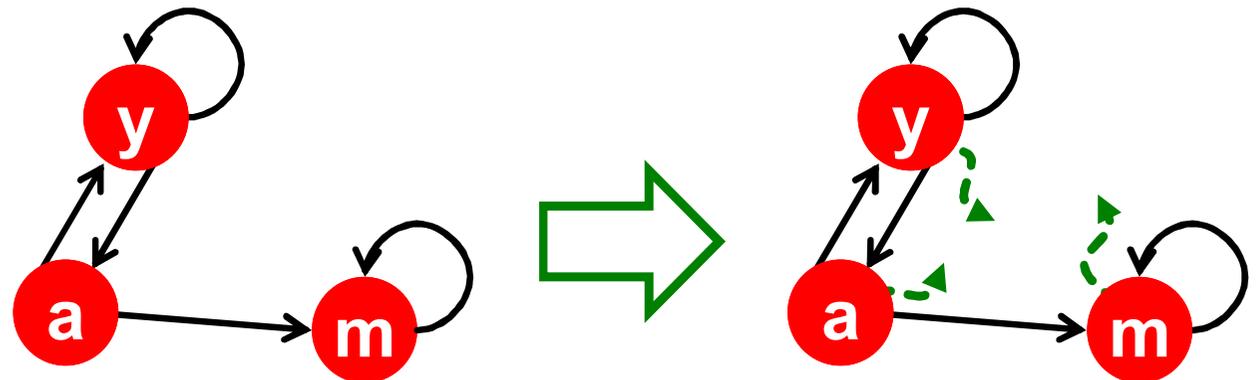
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{pmatrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{pmatrix}$$

Iteration 0, 1, 2, ...

All the PageRank score gets “trapped” in node m.

Solution: Teleports!

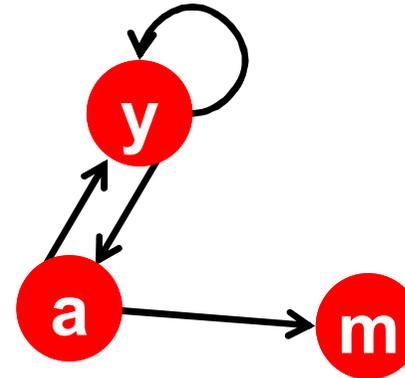
- **The Google solution for spider traps: At each time step, the random surfer has two options**
 - With prob. β , follow a link at random
 - With prob. $1-\beta$, jump to some random page
 - Common values for β are in the range 0.8 to 0.9
- **Surfer will teleport out of spider trap within a few time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



Problem: Dead Ends

Power Iteration:

- Set
- —
- And iterate



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

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2$$

$$r_m = r_a/2$$

Example:

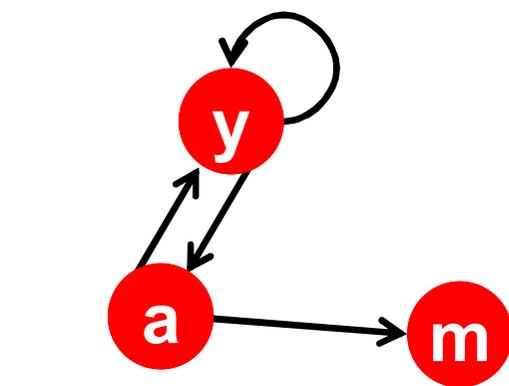
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{pmatrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{pmatrix}$$

Iteration 0, 1, 2, ...

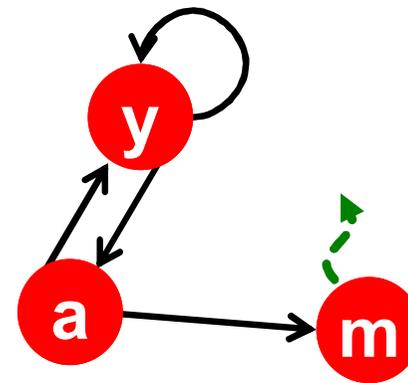
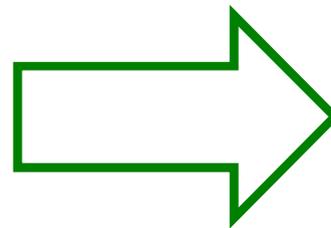
Here the PageRank "leaks" out since the matrix is not stochastic.

Solution: Always Teleport!

- **Teleports:** Follow random teleport links with probability 1.0 from dead-ends
 - Adjust matrix accordingly



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
a	$\frac{1}{2}$	0	$\frac{1}{3}$
m	0	$\frac{1}{2}$	$\frac{1}{3}$

Why Teleports Solve the Problem?

Why are dead-ends and spider traps a problem and **why do teleports solve the problem?**

- **Spider-traps** are not a problem, but with traps PageRank scores are **not** what we want
 - **Solution:** Never get stuck in a spider trap by teleporting out of it in a finite number of steps
- **Dead-ends** are a problem
 - The matrix is not column stochastic so our initial assumptions are not met
 - **Solution:** Make matrix column stochastic by always teleporting when there is nowhere else to go

Solution: Random Teleports

- Google's solution that does it all:

At each step, random surfer has two options:

- With probability β , follow a link at random
- With probability $1-\beta$, jump to some random page

- **PageRank equation** [Brin-Page, 98]

d_i ... out-degree
of node i

This formulation assumes that A has no dead ends. We can either preprocess matrix A to remove all dead ends or explicitly follow random teleport links with probability 1.0 from dead-ends.

The Google Matrix

- **PageRank equation** [Brin-Page, '98]

— —

- **The Google Matrix A:**

$[1/N]_{N \times N}$... N by N matrix
where all entries are $1/N$

$$\left(\quad \right) \begin{bmatrix} - \\ - \end{bmatrix}$$

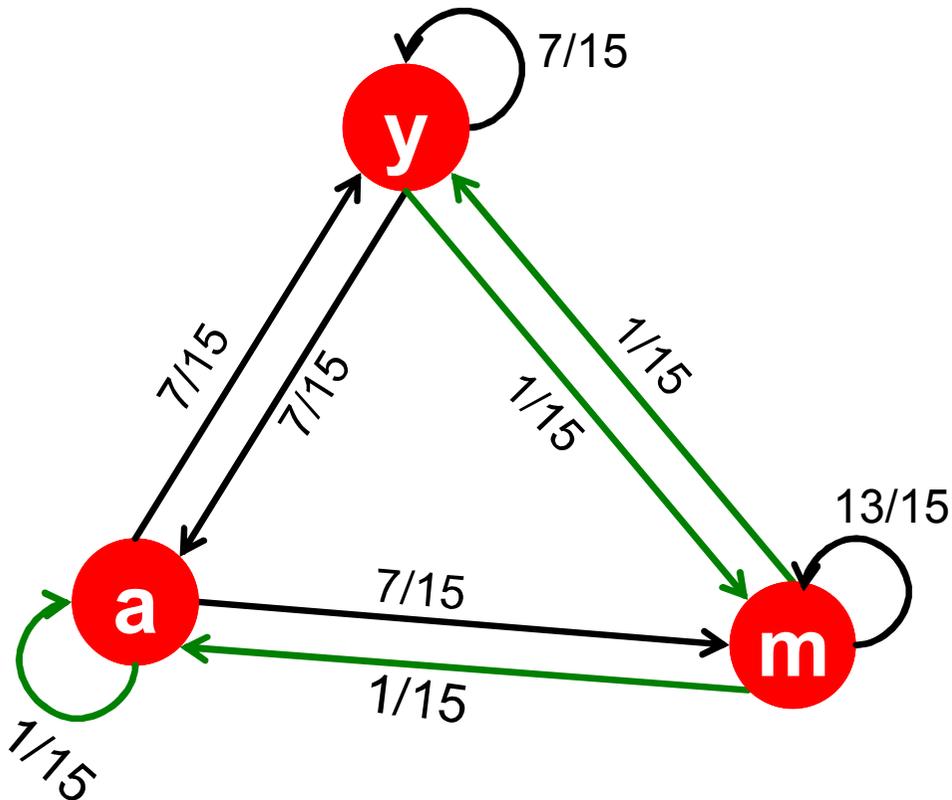
- **We have a recursive problem:**

And the Power method still works!

- **What is β ?**

- In practice $\beta = 0.8, 0.9$ (make 5 steps on avg., jump)

Random Teleports ($\beta = 0.8$)



$$0.8 \begin{matrix} \mathbf{M} \\ \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \end{matrix} + 0.2 \begin{matrix} [1/N]_{N \times N} \\ \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} \end{matrix}$$

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

y	=	1/3	0.33	0.24	0.26	7/33
a	=	1/3	0.20	0.20	0.18	...
m	=	1/3	0.46	0.52	0.56	21/33

**How do we actually compute
the PageRank?**

Computing Page Rank

- **Key step is matrix-vector multiplication**

- $r^{\text{new}} = \mathbf{A} \cdot r^{\text{old}}$

- Easy if we have enough main memory to hold \mathbf{A} , r^{old} , r^{new}

- **Say $N = 1$ billion pages**

- We need 4 bytes for each entry (say)

- 2 billion entries for vectors, approx 8GB

- **Matrix \mathbf{A} has N^2 entries**

- 10^{18} is a large number!

$$\mathbf{A} = \beta \cdot \mathbf{M} + (1-\beta) [\mathbf{1}/N]_{N \times N}$$

$$\mathbf{A} = 0.8 \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix} + 0.2 \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{7}{15} & \frac{7}{15} & \frac{1}{15} \\ \frac{7}{15} & \frac{1}{15} & \frac{1}{15} \\ \frac{1}{15} & \frac{7}{15} & \frac{13}{15} \end{bmatrix}$$

Matrix Formulation

- Suppose there are N pages
- Consider page i , with d_i out-links
- We have $M_{ji} = 1/|d_i|$ when $i \rightarrow j$
and $M_{ji} = 0$ otherwise
- **The random teleport is equivalent to:**
 - Adding a **teleport link** from i to every other page and setting transition probability to $(1-\beta)/N$
 - Reducing the probability of following each out-link from $1/|d_i|$ to $\beta/|d_i|$
 - **Equivalent:** Tax each page a fraction $(1-\beta)$ of its score and redistribute evenly

Rearranging the Equation

■ $\mathbf{M} \mathbf{x} = \mathbf{b}$, where $\mathbf{M} = \mathbf{M} - \mathbf{M} \mathbf{D}^{-1} \mathbf{D}$

■ $\mathbf{M} = \mathbf{M} - \mathbf{M} \mathbf{D}^{-1} \mathbf{D}$

■ $\mathbf{M} = \mathbf{M} - \mathbf{M} \mathbf{D}^{-1} \mathbf{D}$

$\mathbf{D}^{-1} \mathbf{D}$ since

■ So we get: $\mathbf{M} \mathbf{x} = \mathbf{b}$

Note: Here we assumed \mathbf{M} has no dead-ends

$[x]_N \dots$ a vector of length N with all entries x

Sparse Matrix Formulation

- We just rearranged the **PageRank equation**

$$\left[\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \right]$$

- where $[(1-\beta)/N]_N$ is a vector with all N entries $(1-\beta)/N$
- M is a **sparse matrix!** (with no dead-ends)
 - 10 links per node, approx $10N$ entries
- So in each iteration, we need to:
 - Compute $r^{\text{new}} = \beta M \cdot r^{\text{old}}$
 - Add a constant value $(1-\beta)/N$ to each entry in r^{new}
 - **Note if M contains dead-ends then r^{new} and we also have to renormalize r^{new} so that it sums to 1**

PageRank: The Complete Algorithm

- **Input: Graph and parameter**
 - Directed graph (can have **spider traps** and **dead ends**)
 - Parameter
- **Output: PageRank vector**

- **Set:** $\mathbf{S} = \{s \in V \mid \text{in-degree}(s) = 0\}$
- **repeat until convergence:** $\mathbf{P} = \mathbf{D}^{-1} \mathbf{A} \mathbf{P}$
 - $\mathbf{D} = \text{diag}(d_1, \dots, d_n)$ where $d_i = \text{in-degree}(i)$ if in-degree of i is 0
 - **Now re-insert the leaked PageRank:** $\mathbf{P} = \beta \mathbf{D}^{-1} \mathbf{A} \mathbf{P} + (1 - \beta) \mathbf{S}$
 - **where:** $\mathbf{S} = \text{diag}(s_1, \dots, s_n)$ where $s_i = 1$ if $i \in S$ and $s_i = 0$ otherwise

If the graph has no dead-ends then the amount of leaked PageRank is $1 - \beta$. But since we have dead-ends the amount of leaked PageRank may be larger. We have to explicitly account for it by computing \mathbf{S} .

Sparse Matrix Encoding

- **Encode sparse matrix using only nonzero entries**
 - Space proportional roughly to number of links
 - Say 10N, or $4 \cdot 10^1$ 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

Some Problems with PageRank

- **Measures generic popularity of a page**
 - Will ignore/miss topic-specific authorities
 - **Solution:** Topic-Specific PageRank (**next**)
- **Uses a single measure of importance**
 - Other models of importance
 - **Solution:** Hubs-and-Authorities
- **Susceptible to Link spam**
 - Artificial link topographies created in order to boost page rank
 - **Solution:** TrustRank

Topic-Specific PageRank

Topic-Specific PageRank

- **Instead of generic popularity, can we measure popularity within a topic?**
- **Goal:** Evaluate Web pages not just according to their popularity, but by how close they are to a particular topic, e.g. “sports” or “history”
- **Allows search queries to be answered based on interests of the user**
 - **Example:** Query “Trojan” wants different pages depending on whether you are interested in sports, history and computer security

Topic-Specific PageRank

- Random walker has a small probability of teleporting at any step
- **Teleport can go to:**
 - **Standard PageRank:** Any page with equal probability
 - To avoid dead-end and spider-trap problems
 - **Topic Specific PageRank:** A topic-specific set of “relevant” pages (**teleport set**)
- **Idea: Bias the random walk**
 - When walker teleports, she pick a page from a set S
 - S contains only pages that are relevant to the topic
 - E.g., Open Directory (DMOZ) pages for a given topic/query
 - For each teleport set S , we get a different vector r_S

Matrix Formulation

- To make this work all we need is to update the teleportation part of the PageRank formulation:

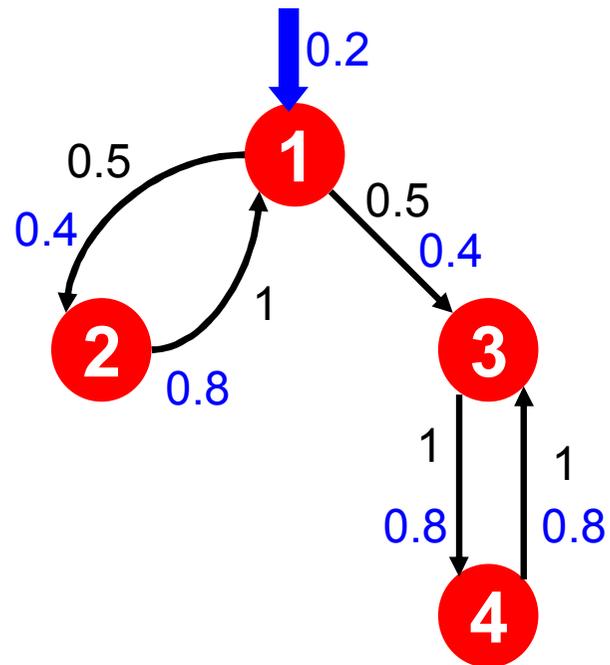
{

if

otherwise

- A is stochastic!
- We weighted all pages in the teleport set S equally
 - Could also assign different weights to pages!
- Compute as for regular PageRank:
 - Multiply by M , then add a vector
 - Maintains sparseness

Example: Topic-Specific PageRank



Suppose $S = \{1\}$, $\beta = 0.8$

Node	Iteration				
	0	1	2	...	stable
1	0.25	0.4	0.28		0.294
2	0.25	0.1	0.16		0.118
3	0.25	0.3	0.32		0.327
4	0.25	0.2	0.24		0.261

$S = \{1\}$, $\beta = 0.90$:

$r = [0.17, 0.07, 0.40, 0.36]$

$S = \{1\}$, $\beta = 0.8$:

$r = [0.29, 0.11, 0.32, 0.26]$

$S = \{1\}$, $\beta = 0.70$:

$r = [0.39, 0.14, 0.27, 0.19]$

$S = \{1, 2, 3, 4\}$, $\beta = 0.8$:

$r = [0.13, 0.10, 0.39, 0.36]$

$S = \{1, 2, 3\}$, $\beta = 0.8$:

$r = [0.17, 0.13, 0.38, 0.30]$

$S = \{1, 2\}$, $\beta = 0.8$:

$r = [0.26, 0.20, 0.29, 0.23]$

$S = \{1\}$, $\beta = 0.8$:

$r = [0.29, 0.11, 0.32, 0.26]$

TrustRank: Combating the Web Spam

What is Web Spam?

- **Spamming:**
 - Any deliberate action to boost a web page's position in search engine results, incommensurate with page's real value
- **Spam:**
 - Web pages that are the result of spamming
- This is a very broad definition
 - **SEO** industry might disagree!
 - SEO = search engine optimization
- Approximately **10-15%** of web pages are spam

Web Search

- **Early search engines:**
 - Crawl the Web
 - Index pages by the words they contained
 - Respond to search queries (lists of words) with the pages containing those words
- **Early page ranking:**
 - Attempt to order pages matching a search query by “importance”
 - **First search engines considered:**
 - (1) Number of times query words appeared
 - (2) Prominence of word position, e.g. title, header

First Spammers

- As people began to use search engines to find things on the Web, those with commercial interests tried to **exploit search engines** to bring people to their own site – whether they wanted to be there or not
- **Example:**
 - Shirt-seller might pretend to be about “movies”
- **Techniques for achieving high relevance/importance for a web page**

First Spammers: Term Spam

- **How do you make your page appear to be about movies?**
 - **(1)** Add the word movie 1,000 times to your page
 - Set text color to the background color, so only search engines would see it
 - **(2)** Or, run the query “movie” on your target search engine
 - See what page came first in the listings
 - Copy it into your page, make it “invisible”
- **These and similar techniques are term spam**

Google's Solution to Term Spam

- **Believe what people say about you, rather than what you say about yourself**
 - Use words in the anchor text (words that appear underlined to represent the link) and its surrounding text
- PageRank as a tool to measure the “importance” of Web pages

Why It Works?

- **Our hypothetical shirt-seller looses**
 - Saying he is about movies doesn't help, because others don't say he is about movies
 - His page isn't very important, so it won't be ranked high for shirts or movies
- **Example:**
 - Shirt-seller creates 1,000 pages, each links to his with "movie" in the anchor text
 - These pages have no links in, so they get little PageRank
 - So the shirt-seller can't beat truly important movie pages, like IMDB

Google vs. Spammers: Round 2!

- Once Google became the dominant search engine, spammers began to work out ways to fool Google
- **Spam farms** were developed to concentrate PageRank on a single page
- **Link spam:**
 - Creating link structures that boost PageRank of a particular page



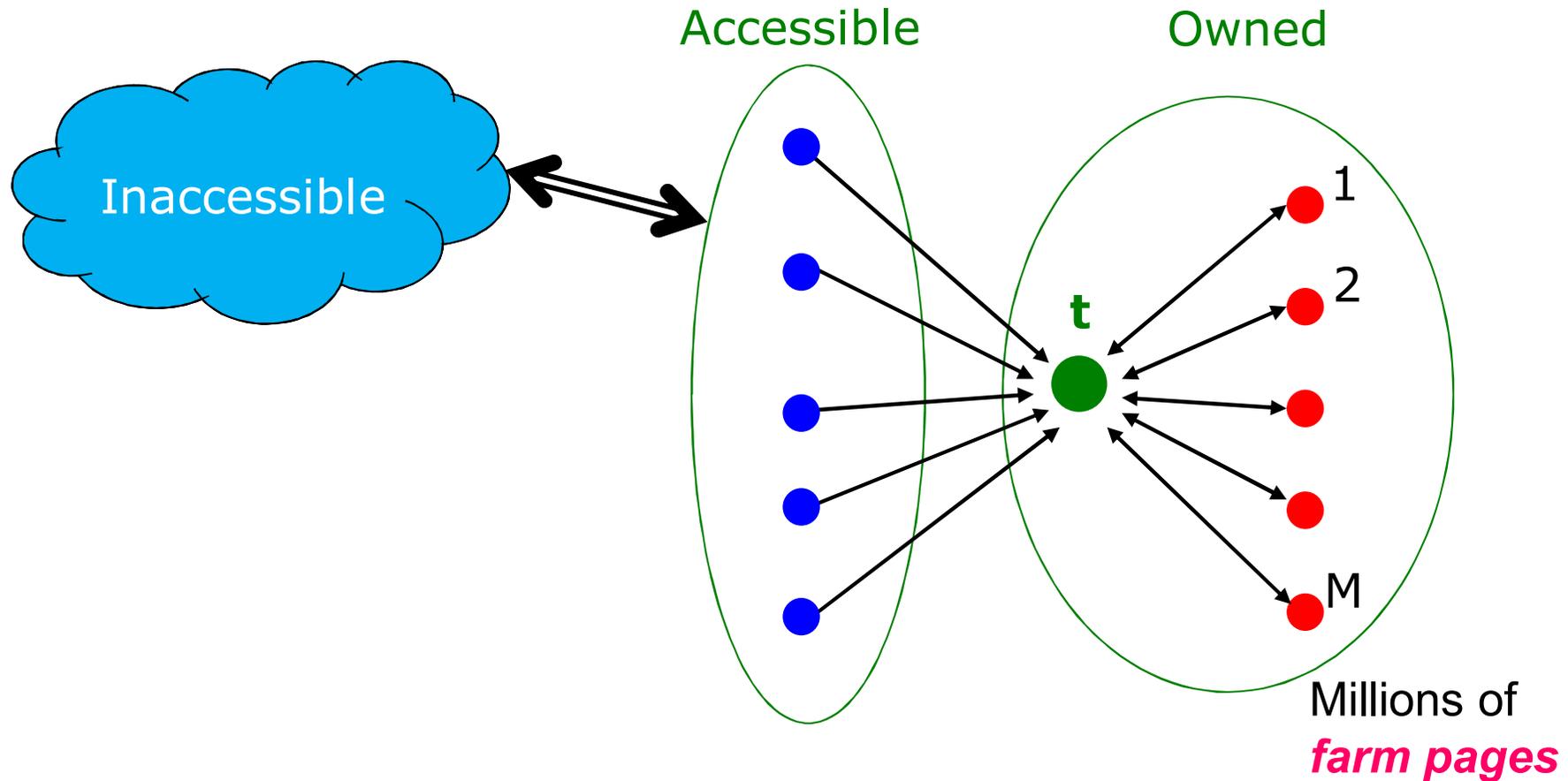
Link Spamming

- **Three kinds of web pages from a spammer's point of view**
 - **Inaccessible pages**
 - **Accessible pages**
 - e.g., blog comments pages
 - spammer can post links to his pages
 - **Owned pages**
 - Completely controlled by spammer
 - May span multiple domain names

Link Farms

- **Spammer's goal:**
 - Maximize the PageRank of target page t
- **Technique:**
 - Get as many links from accessible pages as possible to target page t
 - Construct “link farm” to get PageRank multiplier effect

Link Farms



One of the most common and effective organizations for a link farm

TrustRank: Combating the Web Spam

Combating Spam

■ Combating term spam

- Analyze text using statistical methods
- Similar to email spam filtering
- Also useful: Detecting approximate duplicate pages

■ Combating link spam

- **Detection and blacklisting of structures that look like spam farms**
 - Leads to another war – hiding and detecting spam farms
- **TrustRank** = topic-specific PageRank with a teleport set of **trusted pages**
 - **Example:** .edu domains, similar domains for non-US schools

TrustRank: Idea

- **Basic principle: Approximate isolation**
 - It is rare for a “good” page to point to a “bad” (spam) page
- Sample a set of **seed pages** from the web
- Have an **oracle (human)** to identify the good pages and the spam pages in the seed set
 - **Expensive task**, so we must make seed set as small as possible

Why is it a good idea?

- **Trust attenuation:**

- The degree of trust conferred by a trusted page decreases with the distance in the graph

- **Trust splitting:**

- The larger the number of out-links from a page, the less scrutiny the page author gives each out-link
- Trust is **split** across out-links

Hubs and Authorities

- **HITS (Hypertext-Induced Topic Selection)**
 - Is a measure of importance of pages or documents, similar to PageRank
 - Proposed at around same time as PageRank ('98)
- **Goal:** Say we want to find good newspapers
 - Don't just find newspapers. Find "experts" – people who link in a coordinated way to good newspapers
- **Idea: Links as votes**
 - Page is more important if it has more links
 - In-coming links? Out-going links?

PageRank and HITS

- PageRank and HITS are two solutions to the same problem:
 - What is the value of an in-link from u to v ?
 - In the PageRank model, the value of the link depends on the links into u
 - In the HITS model, it depends on the value of the other links out of u
- The destinies of PageRank and HITS post-1998 were very different