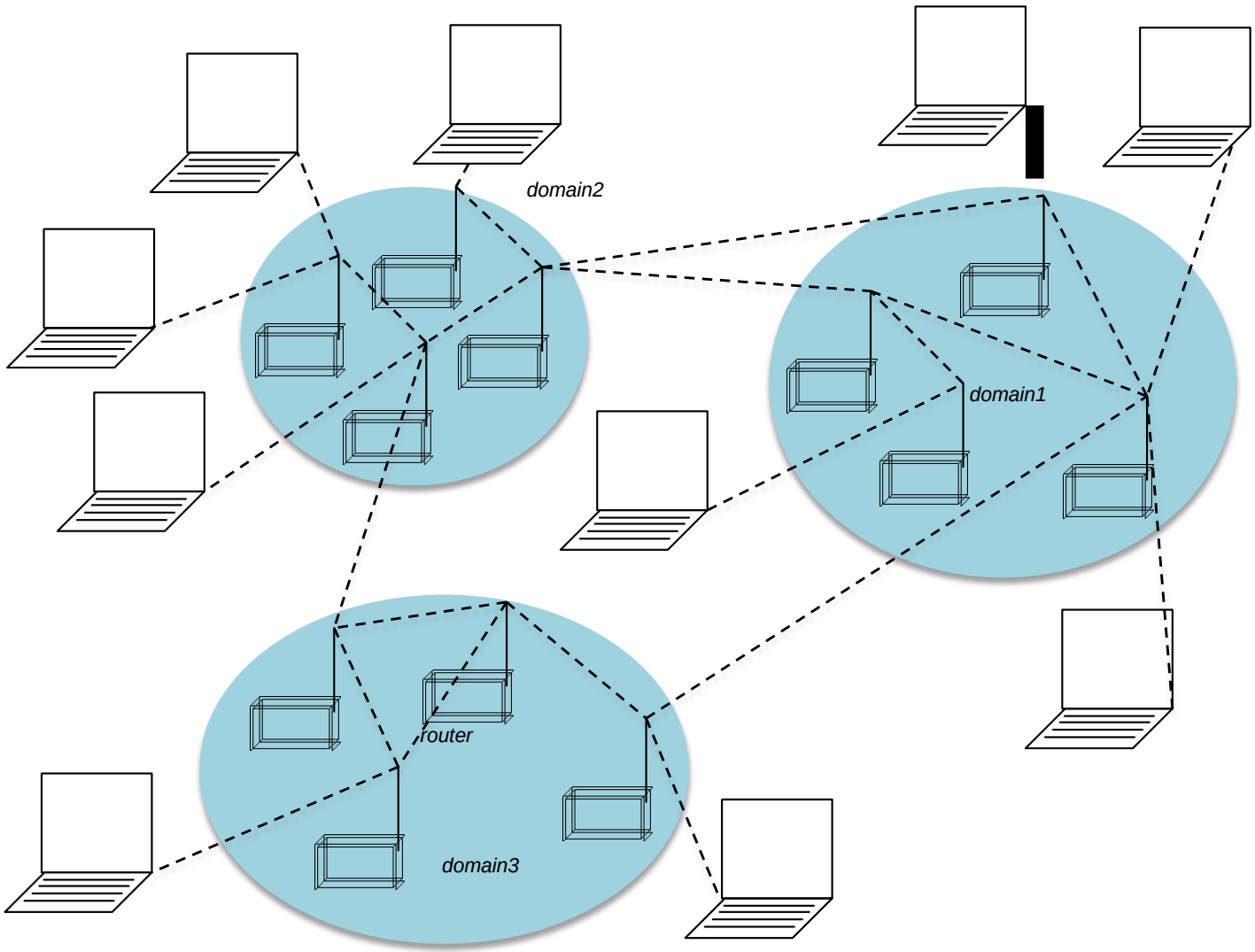
Lecture: Link Analysis

http://bicmr.pku.edu.cn/~wenzw/bigdata2016.html

Acknowledgement: this slides is based on Prof. Jure Leskovec's lecture notes

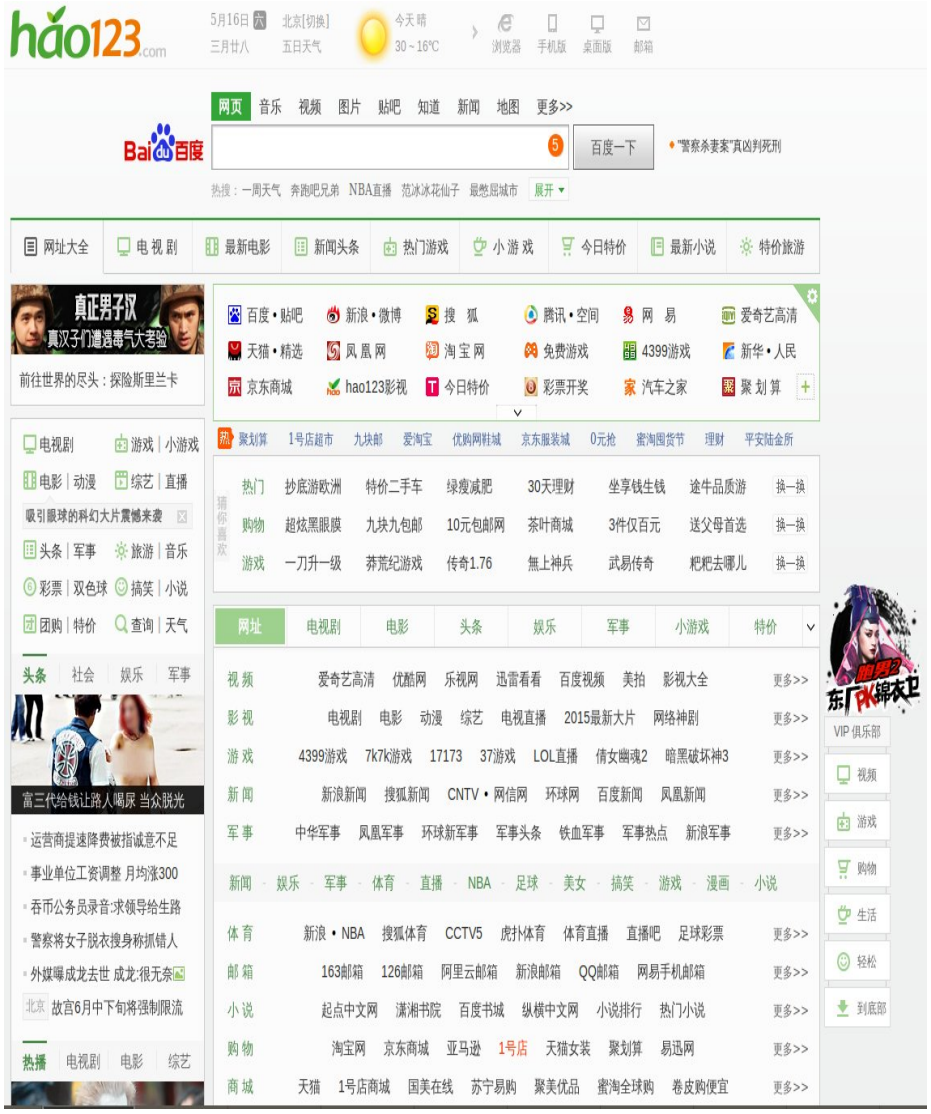
Outline

Communication networks



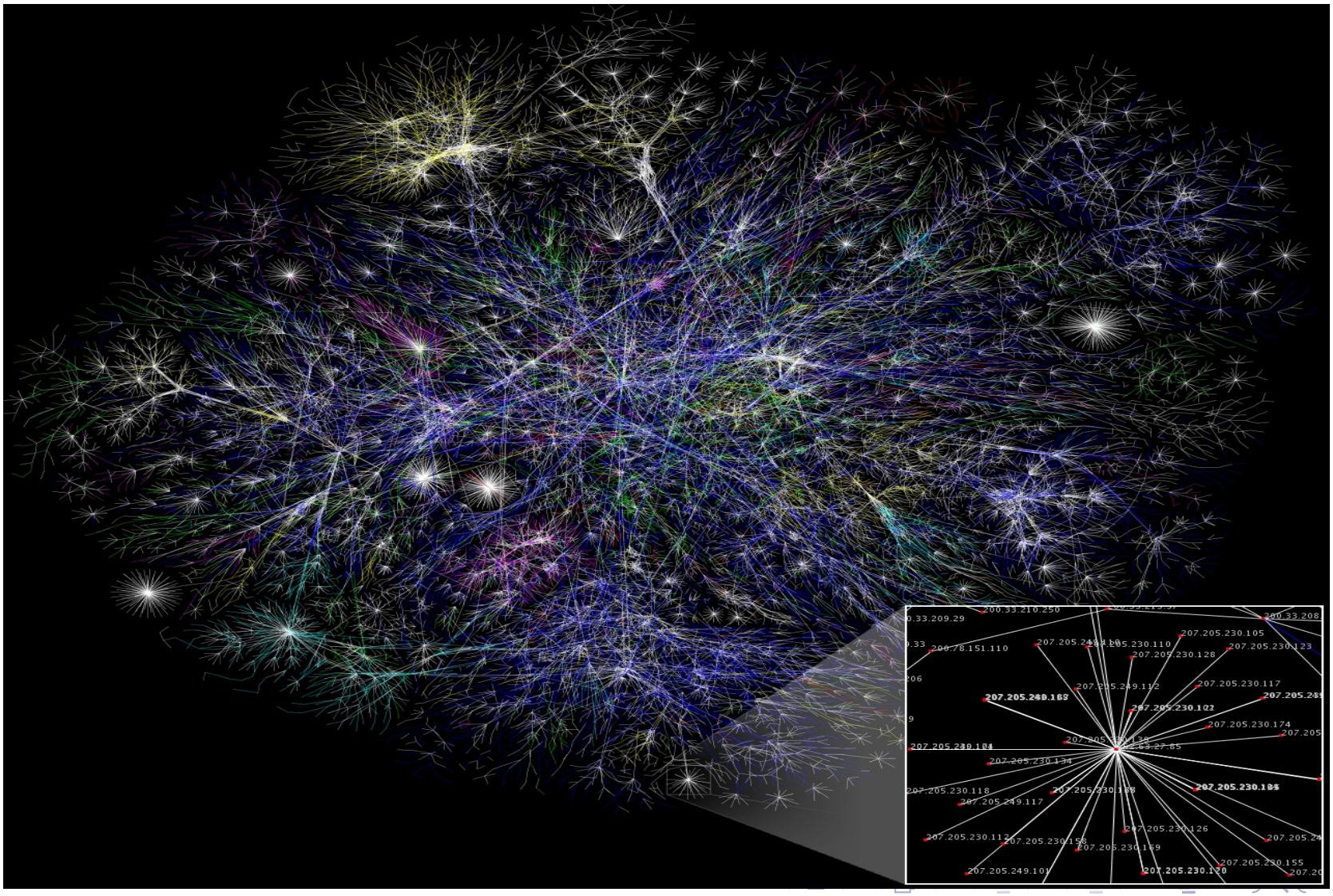
Web search

- How to organize the Web?
- First try: Human curated Web directories
  - Yahoo, baidu, hao123
- Second try: Web Search
  - Information Retrieval investigates: **Find relevant docs in a small and trusted set**
  - But: Web is huge, full of untrusted documents, random things, web spam, etc.



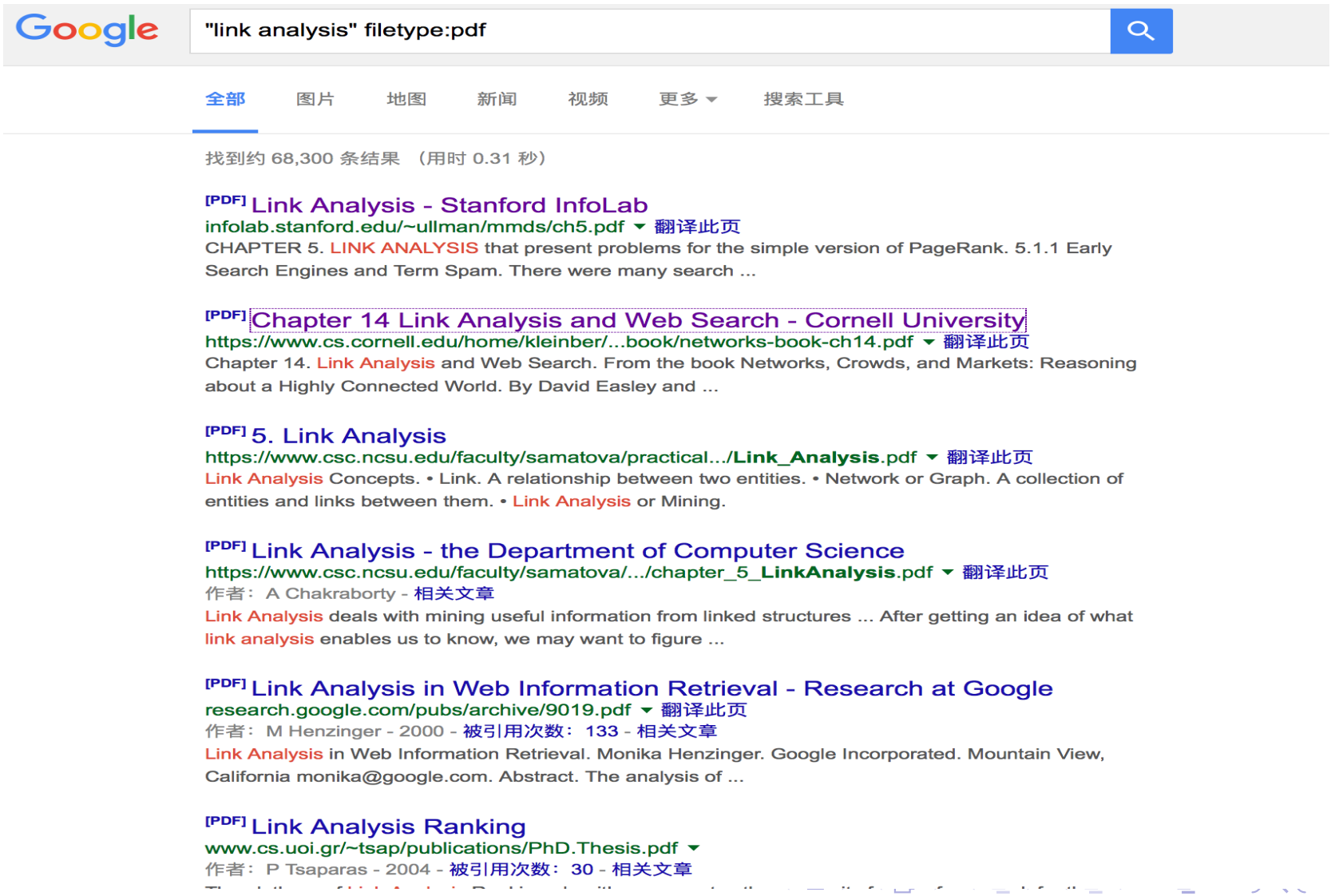
Web as a directed graph

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



Web as a directed graph

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



Three basic things of search engines

- Crawl the web and locate all web pages with public access.
- Index the data from step 1, so that it can be searched efficiently for relevant keywords or phrases.
- Rate the importance of each page in the database, so that when a user does a search and the subset of pages in the database with the desired information has been found, the more important pages can be presented first.

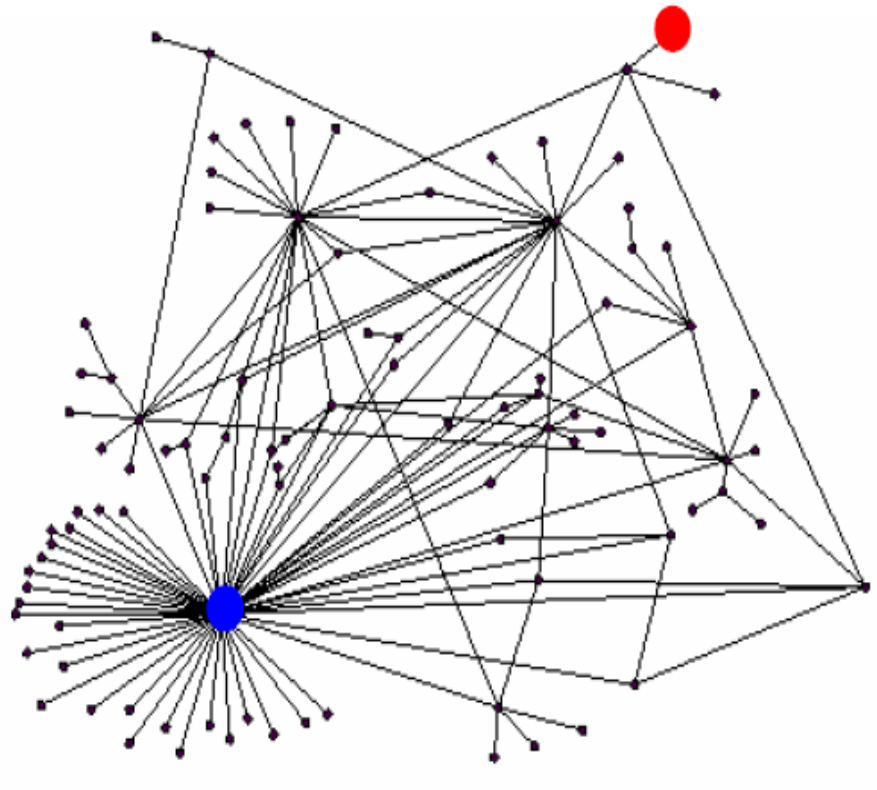
Web search: two challenges

Two 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.pku.edu.cn` vs. `www.tsinghua.edu.cn`
- There is large diversity in the web-graph node connectivity
- Let's rank the pages by the link structure!



Outline

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.pku.edu.cn`: 6,649 links
    `www.alexa.com/siteinfo/www.pku.edu.cn`
  - `www.tsinghua.edu.cn`: 8579 links
    `www.alexa.com/siteinfo/www.tsinghua.edu.cn`
- Are all in-links equal?
  - Links from important pages count more
  - Recursive question!

Links as votes

What sites link to pku.edu.cn?

Total Sites Linking In 6,649

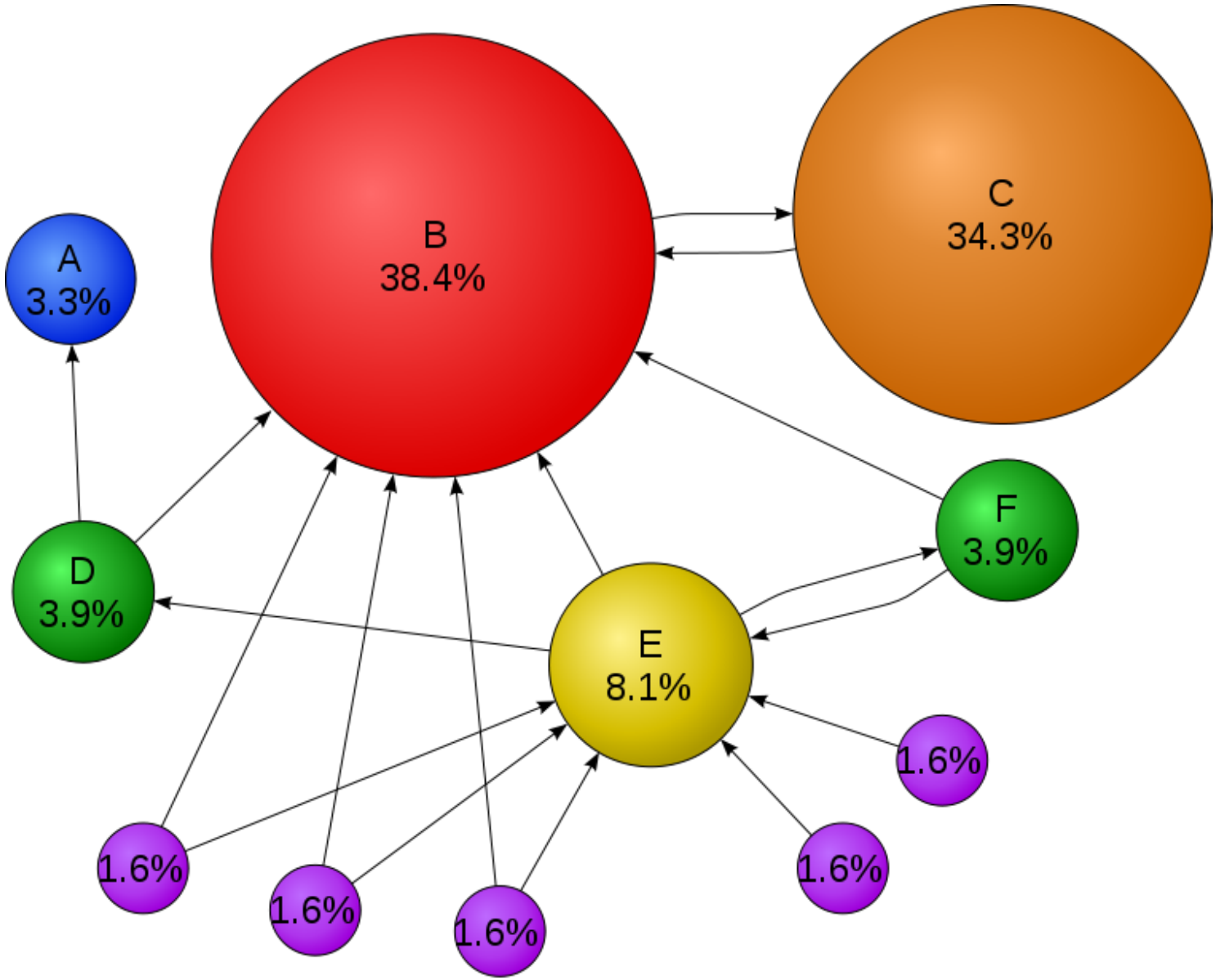
Site	Page
1. baidu.com	bdl.baidu.com/publication.html
2. msn.com	msn.com/de-at/nachrichten/wissenundtec...
3. qq.com	edu.qq.com/bschool
4. hupu.com	bbs.hupu.com/14788328.html
5. 163.com	biz.163.com

What sites link to tsinghua.edu.cn?

Total Sites Linking In 8,579

Site	Page
1. yahoo.com	travelinspirations.yahoo.com/post/:id/...
2. baidu.com	tieba.baidu.com/f?ie=utf-8&kw=清华大学
3. msn.com	msn.com/en-us/travel/tripideas/the-bes...
4. yandex.ru	ftp.yandex.ru/debian/README.mirrors.ht...
5. qq.com	city.qq.com

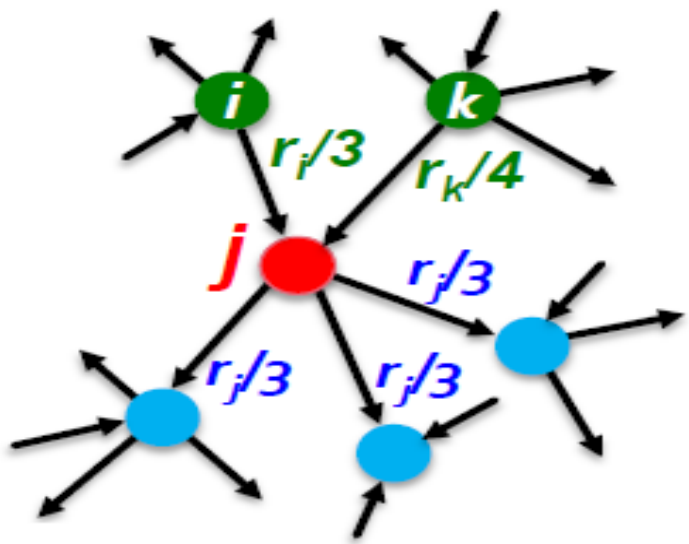
Example: PageRank scores



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

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

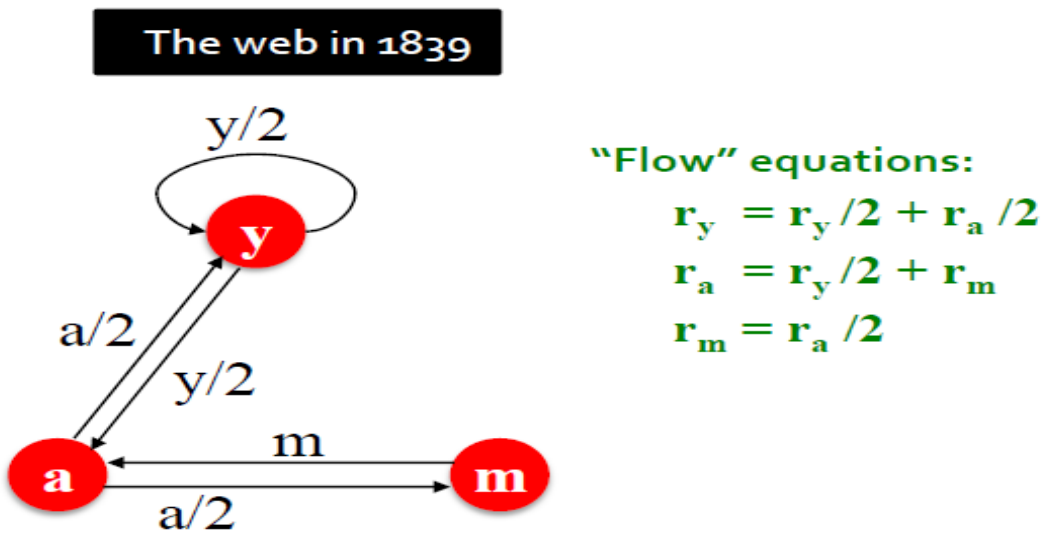


PageRank: the "flow" model

- 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

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

where d_i is the out-degree of node i



Solving the flow equations

"Flow" equations:

$$\mathbf{r_y} = \mathbf{r_y}/2 + \mathbf{r_a}/2$$
$$\mathbf{r_a} = \mathbf{r_y}/2 + \mathbf{r_m}$$
$$\mathbf{r_m} = \mathbf{r_a}/2$$

- No unique solution
- All solutions equivalent modulo the scale factor
- Additional constraint forces uniqueness:
  - $r_y + r_a + r_m = 1$
  - Solution: $r_y = 2/5, r_a = 2/5, r_m = 1/5$
- Gaussian elimination method works for small examples, but we need a better method for large web-size graphs
- We need a new formulation!

PageRank: matrix formulation

- Stochastic adjacency matrix $\mathbf{M}$
  - Let page i has d_i out-links

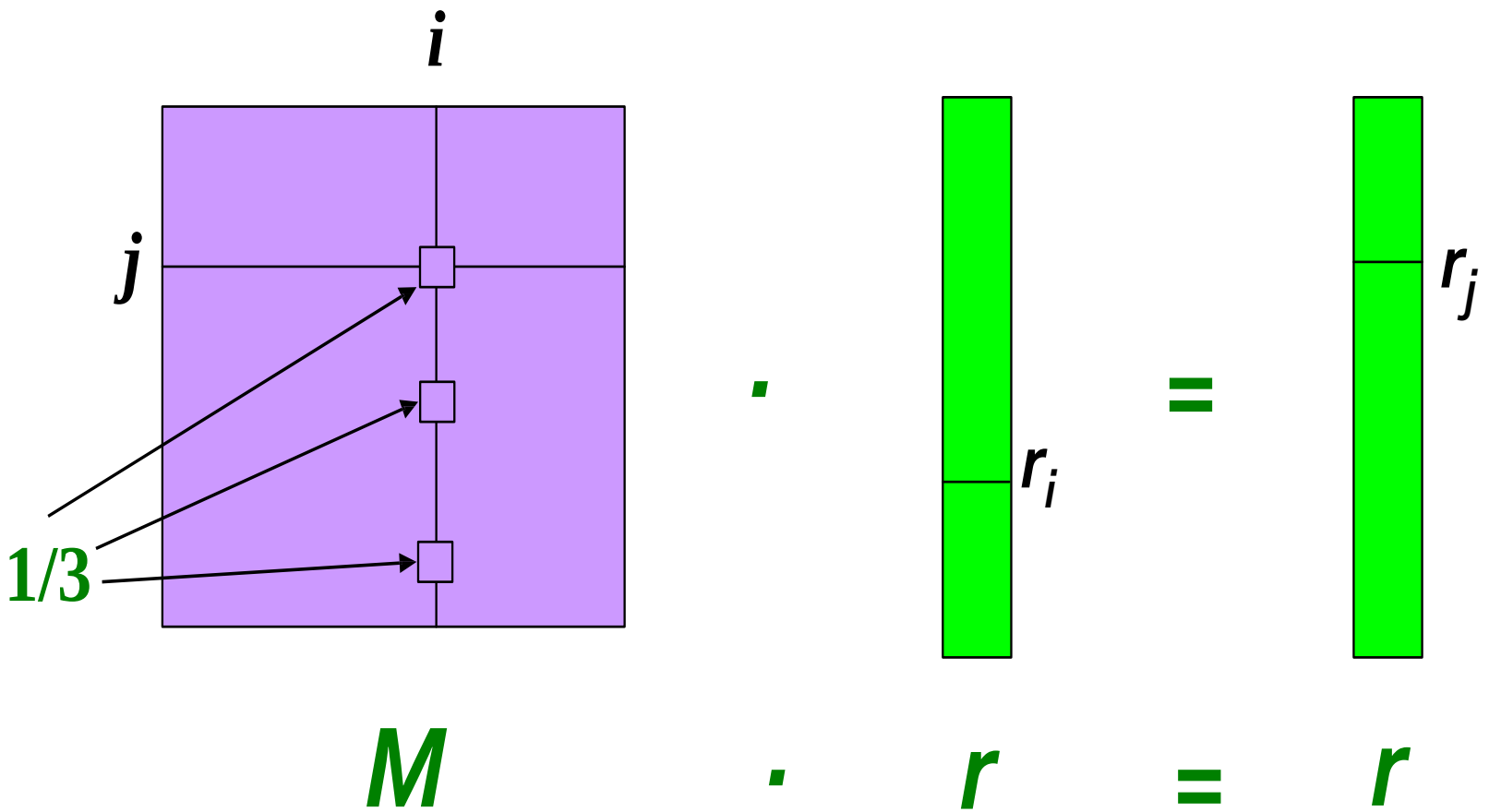
$$\mathbf{M}_{ji} = \begin{cases} \frac{1}{d_i} & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

  - $\mathbf{M}$ is a column stochastic matrix (column sum to 1)
- Rank vector $\mathbf{r}$: vector with an entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$
- The flow equation $r_j = \sum_{i \to j} \frac{r_i}{d_i}$ can be written as

$$\mathbf{r} = \mathbf{M}\mathbf{r}$$

Example

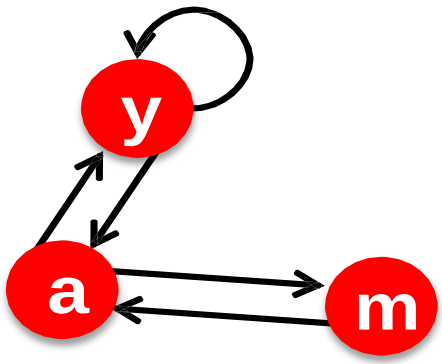
- Remember the flow equation: $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
- Flow equation in the matrix form: $\mathbf{Mr} = \mathbf{r}$
  - Suppose page i links to 3 pages, including j



Eigenvector formulation

- NOTE: x is an eigenvector of A with the corresponding eigenvalue λ if: $Ax = \lambda x$
- Flow equation in the matrix form: $\mathbf{Mr} = \mathbf{r}$
- The rank vector $\mathbf{r}$ is an eigenvector of the stochastic web matrix $\mathbf{M}$
  - In fact, its first or principal eigenvector, with corresponding eigenvalue 1
  - Largest eigenvalue of $\mathbf{M}$ is 1 since $\mathbf{M}$ is column stochastic. We know $\mathbf{r}$ is unit length and each column of $\mathbf{M}$ sums to one, so $\mathbf{Mr} \leq 1$
- We can now efficiently solve for $\mathbf{r}$ through *power iteration*

Example: flow equations



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$$r = M \cdot r$$

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

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

Power iteration

- 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: $\mathbf{r}^{(0)} = [1/N, \ldots, 1/N]^\top$
  - Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M}\mathbf{r}^{(t)}$, i.e.,

$$\mathbf{r}_j^{t+1} = \sum_{i \to j} \frac{\mathbf{r}_i^{(t)}}{d_i}, \quad d_i : \text{out-degree of node i}$$

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

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

$$\mathbf{p}_{t+1} = \mathbf{M}\mathbf{p}_t = \mathbf{p}_t$$

  then $\mathbf{p}_t$ is the **stationary distribution** of a random walk

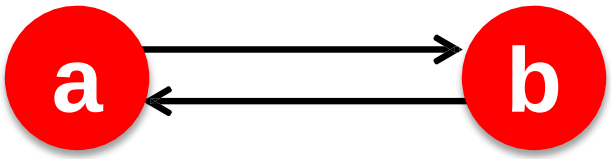
- Our original rank vector $\mathbf{r}$ satisfies $\mathbf{r} = \mathbf{M}\mathbf{r}$
- So $\mathbf{r}$ is a stationary distribution for the random walk

PageRank: three questions

$$\mathbf{r}_j^{t+1} = \sum_{i \to j} \frac{\mathbf{r}_i^{(t)}}{d_i}, \qquad \text{or equivalently} \qquad \mathbf{r} = \mathbf{M}\mathbf{r}$$

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

Does it converge?



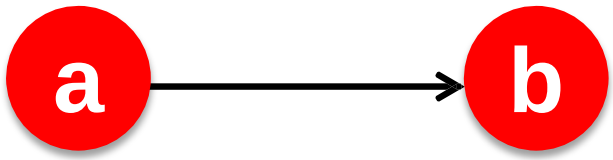
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

Example:

$$\begin{matrix} \mathrm{r_a} \\ \mathrm{r_b} \end{matrix} = \begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{matrix}$$

Iteration 0, 1, 2, …

Does it converge to what we want?



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

Example:

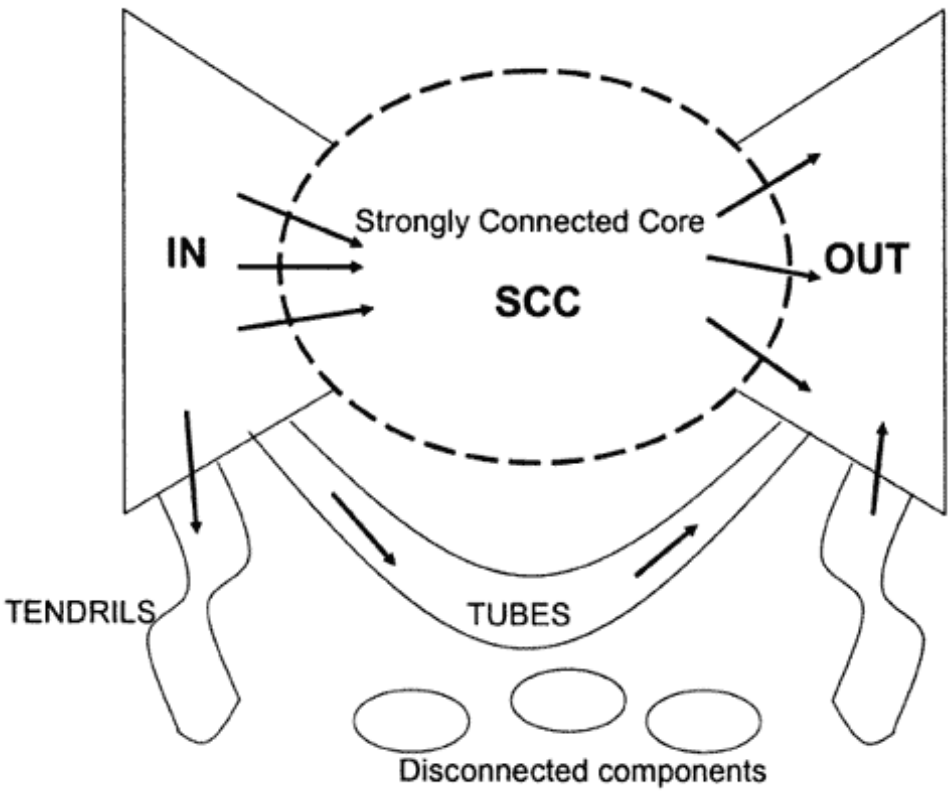
$$\begin{matrix} \mathrm{r_a} \\ \mathrm{r_b} \end{matrix} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

Iteration 0, 1, 2, …

PageRank: problems

Two problems:

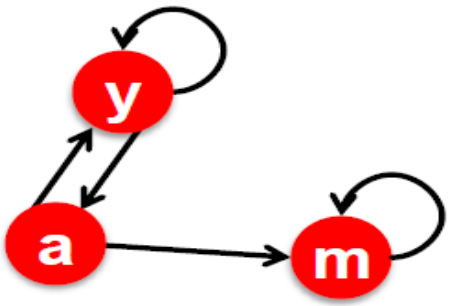
1. Some pages are dead ends (have no out-links)
   - Such pages cause importance to "leak out"
2. Spider traps (all out-links are within the group)
   - Eventually spider traps absorb all importance



Problem: spider traps

Power Iteration:

- Set $r_j = 1$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - And iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

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

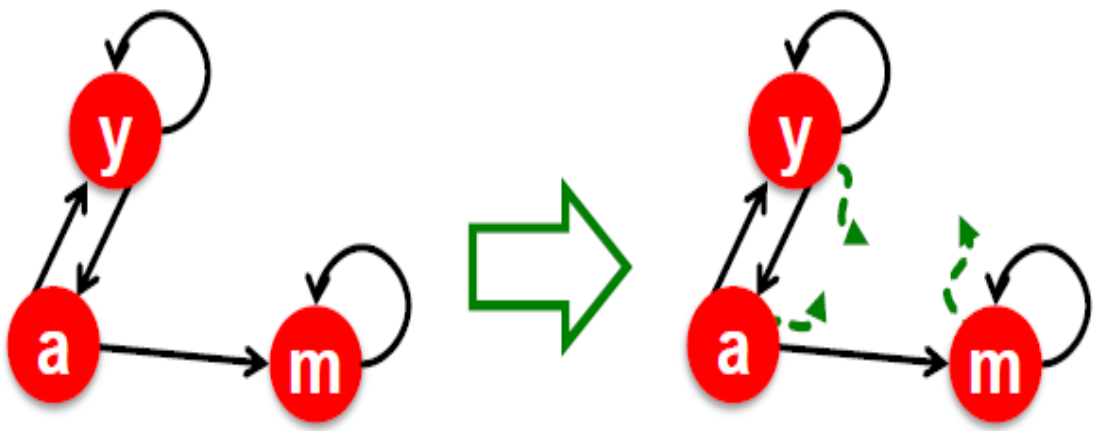
Example:

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

Iteration 0, 1, 2, …

Solution: random teleports

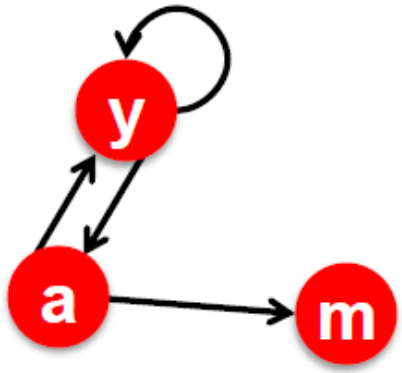
- The Google solution for spider traps: At each time step, the random surfer has two options
  - With probability β, follow a link at random
  - With probability $1 - \beta$, jump to some random page
  - Commonly $\beta \in [0.8, 0.9]$
- Surfer will teleport out of spider trap within a few time steps



Problem: dead ends

Power Iteration:

- Set $r_j = 1$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - And iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

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

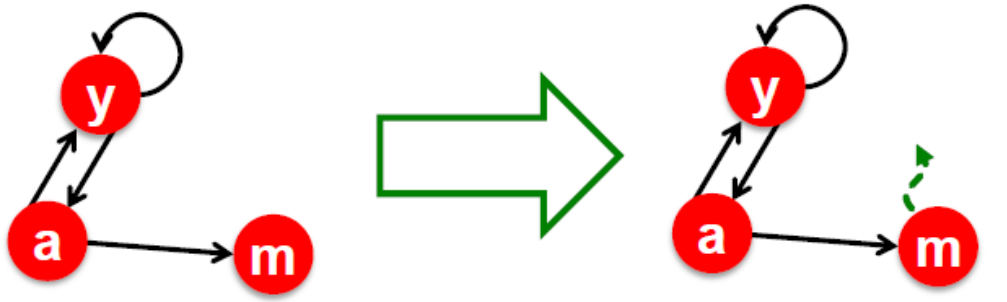
Example:

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

Iteration 0, 1, 2, …

Solution: always teleport

- Teleports: Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly
- Surfer will teleport out of spider trap within a few time steps



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

Why teleports solve the problem?

$$\mathbf{r}^{(t+1)} = \mathbf{M}\mathbf{r}^{(t)}$$

Markov chains

- Set of states X
- Transition matrix P where $P_{ij} = P(X_t = i | X_{t-1} = j)$
- π specifying the stationary probability of being at each state $x \in X$
- Goal is to find π such that $\pi = P\pi$

Why is this analogy useful?

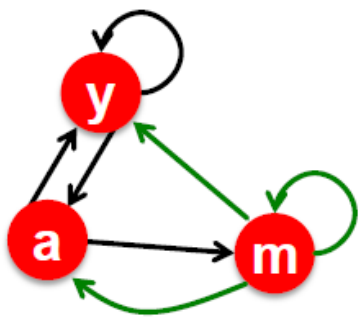
- Theory of Markov chains
- Fact: For *any start vector*, the power method applied to a Markov transition matrix P will converge to a unique positive stationary vector as long as P is **stochastic**, **irreducible** and **aperiodic**
- (By the Perron-Frobenius theorem, an irreducible and aperiodic Markov chain is guaranteed to converge to a unique stationary distribution)

Make $\mathbf{M}$ stochastic

- **(column)-stochastic**: every column sums to 1
- A possible solution: add green links

$$\mathbf{A} = \mathbf{M} + \mathbf{a}^{\top}(\frac{1}{n}\mathbf{1})$$

where $a_i = 1$ if node i has out deg 0, otherwise $a_i = 0$

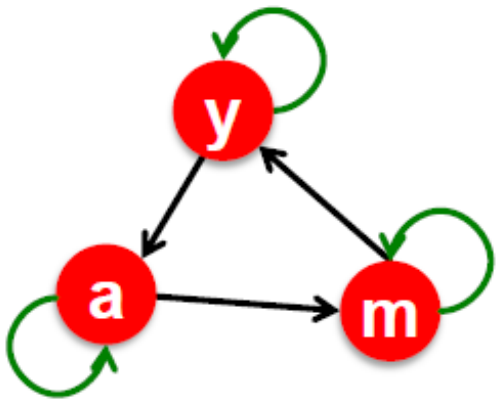


	y	a	m
y	½	½	1/3
a	½	0	1/3
m	0	½	1/3

$$\mathbf{r}_y = \mathbf{r}_y/2 + \mathbf{r}_a/2 + \mathbf{r}_m/3$$
$$\mathbf{r}_a = \mathbf{r}_y/2 + \mathbf{r}_m/3$$
$$\mathbf{r}_m = \mathbf{r}_a/2 + \mathbf{r}_m/3$$

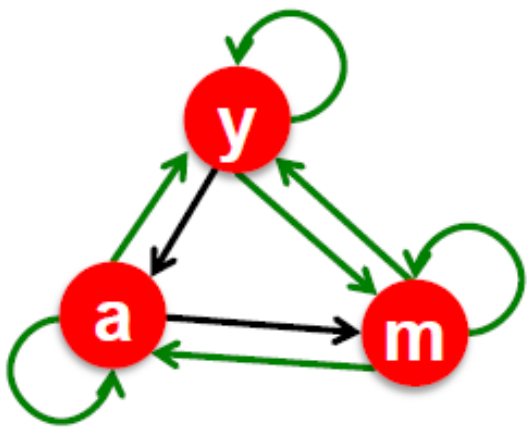
Make $\mathbf{M}$ aperiodic

- A chain is **periodic** if there exists $k > 1$ such that the interval between two visits to some state s is always a multiple of k
- A possible solution: add green links



Make $\mathbf{M}$ irreducible

- From any state, there is a non-zero probability of going from any one state to any another
- A possible solution: add green links



Google's solution: random jumps

- Google's solution that does it all:
  - Makes M **stochastic, aperiodic, irreducible**
- 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]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n}$$

- This formulation assumes that $\mathbf{M}$ has no dead ends
- We can either preprocess matrix $\mathbf{M}$ to remove all dead ends or explicitly follow random teleport links with probability 1.0 from dead-ends

Google's solution: random jumps

- PageRank equation [Brin-Page, 98]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta)\frac{1}{n}$$

- Since $\mathbf{1}^\top \mathbf{r} = 1$, the Google matrix $\mathbf{A}$:

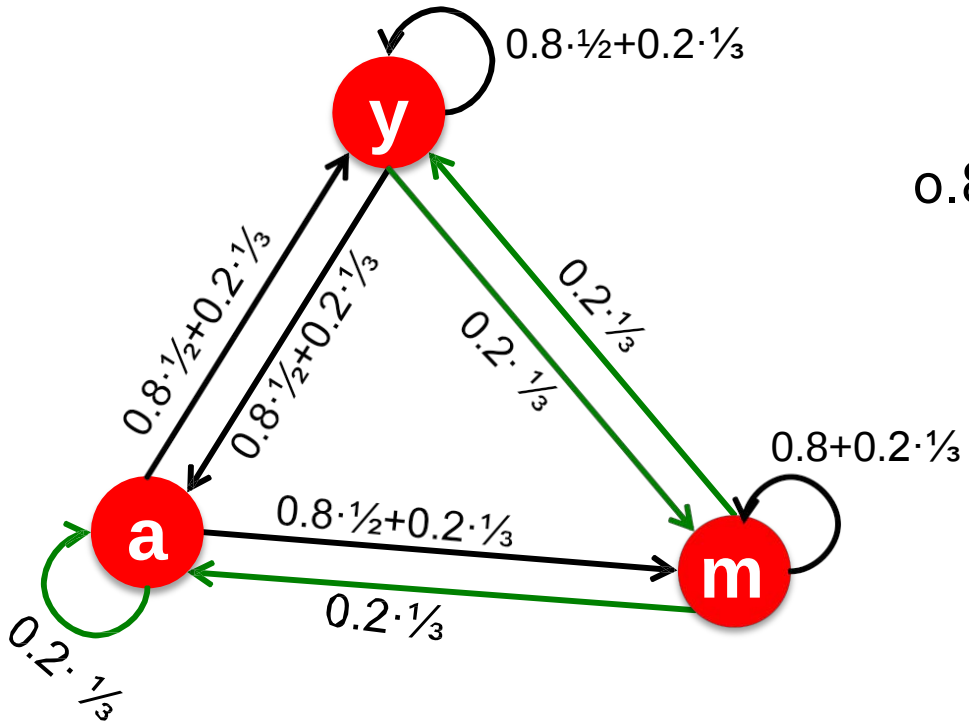
$$\mathbf{A} = \beta \mathbf{M} + (1 - \beta)\frac{1}{n}\mathbf{1}\mathbf{1}^\top$$

- $\mathbf{A}$ is stochastic, aperiodic and irreducible, so

$$\mathbf{r}^{(t+1)} = \mathbf{A}\mathbf{r}^{(t)}$$

- In practice $\beta \in [0.8, 0.9]$ (make around 5 steps and jump)

Random teleports ($\beta = 0.8$)



M

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

$1/n \cdot 1 \cdot 1^T$

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

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

A

$$\begin{array}{c} y \\ a \\ m \end{array} = \begin{array}{ccccc} 1/3 & 0.33 & 0.24 & 0.26 & \\ 1/3 & 0.20 & 0.20 & 0.18 & \dots \\ 1/3 & 0.46 & 0.52 & 0.56 & \end{array} \begin{array}{c} 7/33 \\ 5/33 \\ 21/33 \end{array}$$

Simple proof using linear algebra

- Every stochastic matrix has 1 as an eigenvalue.
- $V_1(\mathbf{A})$: eigenspace for eigenvalue 1 of a stochastic matrix $\mathbf{A}$.
- Fact 1: If $\mathbf{M}$ is positive and stochastic, then any eigenvector in $V_1(\mathbf{M})$ has all positive or all negative components.
- Fact 2: If $\mathbf{M}$ is positive and stochastic, then $V_1(\mathbf{M})$ has dimension 1.

Proof of Fact 1

- Suppose $x \in V_1(\mathbf{M})$ contains elements of mixed sign.
- Since $M_{ij} > 0$, each $M_{ij}x_j$ are of mixed sign. Then

$$|x_i| = |\sum_{j=1}^{n} M_{ij}x_j| < \sum_{j=1}^{n} M_{ij}|x_j|$$

- Since $\mathbf{M}$ is stochastic, we can obtain a contradition

$$\sum_{i=1}^{n} |x_i| < \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}|x_j| = \sum_{j=1}^{n}\left(\sum_{i=1}^{n} M_{ij}\right)|x_j| = \sum_{j=1}^{n} |x_j|$$

- If $x_j \geq 0$ for all j, then $x_i > 0$ since $M_{ij} > 0$ and not all x_j are zero.

Proof of Fact 2

- **Claim**: Let $v, w \in \mathbb{R}^m$ with $m \geq 2$ and linearly independent. Then for some s and t that are not both zero, the vector $x = sv + tw$ has both positive and negative components.
  - Linear independence implies neither v nor w is zero. Let $d = \sum_i v_i$.
  - If $d = 0$, then v must contain mixed sign, and taking $s = 1$, $t = 0$.
  - If $d \neq 0$, set $s = -\sum_i w_i/d$, $t = 1$ and $x = sv + w$. Then $x \neq 0$ but $\sum_i x_i = 0$.
- **Fact 2: Proof by contradiction**. Suppose there are two linearly independent eigenvectors v and w in the subspace $V_1(\mathbf{M})$. For any real numbers s and t that are not both zero, the nonzero vector $x = sv + tw$ must be in $V_1(\mathbf{M})$, and so have components that are all negative or all positive. But by the above claim, for some choice of s and t the vector x must contain components of mixed sign.

Convergence of Power Iteration

Claim 1: Let $\mathbf{M}$ be positive and stochastic. Let V be a subspace of v such that $\sum_i v_i = 0$. Then $\mathbf{M}v \in V$ and $\|\mathbf{M}v\|_1 \leq c\|v\|_1$ for any $v \in V$ and $0 < c < 1$.

- Let $w = Mv$. Then $w \in V$ since

$$\sum_{i=1}^{n} w_i = \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}v_j = \sum_{j=1}^{n} v_j \left(\sum_{i=1}^{n} M_{ij}\right) = \sum_{j=1}^{n} v_j = 0$$

- Let $e_i = \text{sgn}(w_i)$ and $a_j = \sum_{i=1}^{n} e_i M_{ij}$, then e_i are of mixed sign

$$\|w\|_1 = \sum_{i=1}^{n} e_i w_i = \sum_{i=1}^{n} e_i \left(\sum_{j=1}^{n} M_{ij}v_j\right) = \sum_{j=1}^{n} a_j v_j$$

- Since $\sum_{i=1}^{n} M_{ij} = 1$ with $0 < M_{ij} < 1$, there exists $0 < c < 1$ such that $|a_j| < c$.

Convergence of Power Iteration

Claim 2: Let $\mathbf{M}$ be positive and stochastic. Then it has a unique $q > 0$ such that $\mathbf{M}q = q$ with $\|q\|_1 = 1$. The vector q can be computed as $q = \lim_{k\to\infty} \mathbf{M}^k x_0$ with $x_0 > 0$ and $\|x_0\|_1 = 1$.

- The existence of q has been proved.
- We can write $x_0 = q + v$ where $v \in V$ defined in Claim 1. We have

$$\mathbf{M}^k x_0 = \mathbf{M}^k q + \mathbf{M}^k v = q + \mathbf{M}^k v$$

- Since $\|\mathbf{M}^k v\|_1 \leq c^k \|v\|_1$ for $0 < c < 1$, then $\lim_{k\to\infty} \mathbf{M}^k x_0 = q$.

Convergence rate of Power Iteration

- $\mathbf{r}^{(1)} = \mathbf{M}\mathbf{r}^{(0)}$, $\mathbf{r}^{(2)} = \mathbf{M}\mathbf{r}^{(1)} = \mathbf{M}^2\mathbf{r}^{(0)}$, ...
- **Claim**: The sequence $\mathbf{M}\mathbf{r}^{(0)}, \ldots, \mathbf{M}^k\mathbf{r}^{(0)}, \ldots$ approaches the dominant eigenvector of $\mathbf{M}$.
- Proof: Assume $\mathbf{M}$ has n linearly independent eigenvectors, $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ such that $\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n$
- Since $x_1, x_2, \ldots, x_n$ is a basis in $\mathbb{R}^n$, we can write

$$\mathbf{r}^{(0)} = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$

- Using $\mathbf{M}x_i = \lambda_i x_i$, we have

$$\begin{aligned} \mathbf{M}\mathbf{r}^{(0)} &= \mathbf{M}(c_1 x_1 + c_2 x_2 + \ldots + c_n x_n) \\ &= \sum_{i=1}^{n} c_i \lambda_i x_i \end{aligned}$$

Convergence rate of Power Iteration

- Repeated multiplication on both sides produces

$$\begin{aligned}\mathbf{M}^k\mathbf{r}^{(0)} &= \sum_{i=1}^{n} c_i\lambda_i^k x_i \\ &= \lambda_1^k\left(\sum_{i=1}^{n} c_i\left(\frac{\lambda_i}{\lambda_1}\right)^k x_i\right)\end{aligned}$$

- Since $\lambda_1 > \lambda_i$, $i = 2, \ldots, n$. Then $\frac{\lambda_i}{\lambda_1} < 1$, and $\lim_{k\to\infty}\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0$, $i = 2, \ldots, n$.
- Therefore,

$$\mathbf{M}^k\mathbf{r}^{(0)} \approx c_1\lambda_1^k x_1$$

Note if $c_1 = 0$, then the method won't converge.

Outline

Computing the PageRank

- The matrix $\mathbf{A} = \beta\mathbf{M} + (1-\beta)\frac{1}{N}\mathbf{1}\mathbf{1}^\top$
- Key step is matrix-vector multiplication

$$\mathbf{r}^{new} = \mathbf{A}\mathbf{r}^{old}$$

- Easy if we have enough main memory to hold $\mathbf{A}, \mathbf{r}^{old}, \mathbf{r}^{new}$
- Suppose there are $N = 1$ billion pages
  - Suppose we need 4 bytes for each entry
  - 2 billion entries for vectors, approx 8GB
  - Matrix $\mathbf{A}$ has N^2 entries - 10^{18} is huge!

Sparse matrix formulation

- We just rearranged the PageRank equation

$$\mathbf{r} = \beta \mathbf{M} \mathbf{r} + \frac{1-\beta}{N} \mathbf{1}_N$$

- $\mathbf{M}$ is a sparse matrix! (with no dead-ends)
  - 10 links per node, approximately $10N$ entries
- So in each iteration, we need to
  - Compute $\mathbf{r}^{new} = \mathbf{A}\mathbf{r}^{old}$
  - Add a constant value $(1-\beta)/N$ to each entry in $\mathbf{r}^{new}$
  - Note if $\mathbf{M}$ contains dead-ends then $\sum_i r_i^{new} < 1$ and we also have to renormalize $\mathbf{r}^{new}$ so that it sums to 1

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: update step

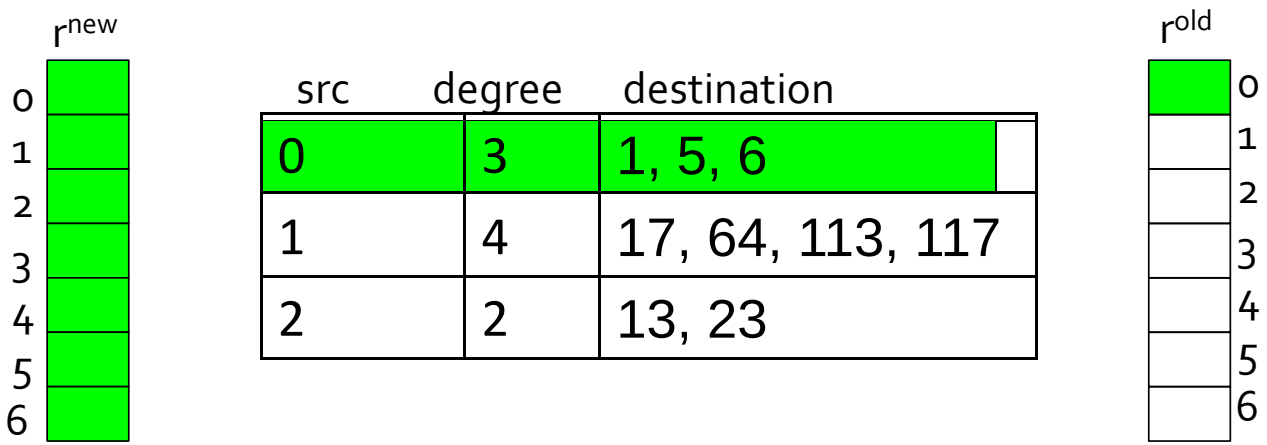
- Assume enough RAM to fit rnew into memory
  - Store $\mathbf{r}^{old}$ and matrix $\mathbf{M}$ on disk
- Then 1 step of power-iteration is

  Initialize all entries of $\mathbf{r}^{new}$ to $(1-\beta)/N$

  For each page p (of out-degree n):

  Read into memory: $p, n, dest_1, , dest_n, r^{old}(p)$

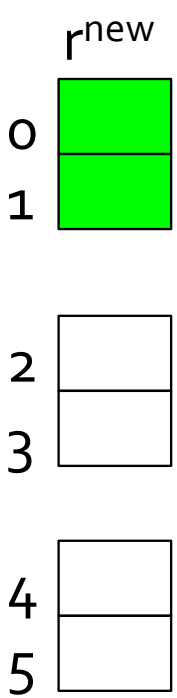
  for $j = 1$ to n: $r^{new}(dest_j)$ += $\beta r^{old}(p)/n$



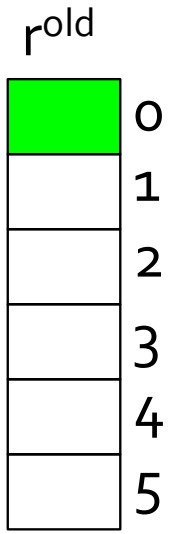
Analysis

- Assume enough RAM to fit $\mathbf{r}^{new}$ into memory
  - Store $\mathbf{r}^{old}$ and matrix $\mathbf{M}$ on disk
- In each iteration, we have to
  - Read $\mathbf{r}^{old}$ and $\mathbf{M}$
  - Write $\mathbf{r}^{new}$ back to disk
  - IO cost: $2|\mathbf{r}| + |\mathbf{M}|$
- Question: What if we could not even fit $\mathbf{r}^{new}$ in memory?

Block based update algorithm



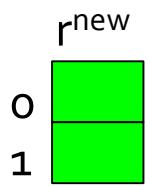
src	degree	destination
0	4	0, 1, 3, 5
1	2	0, 5
2	2	3, 4



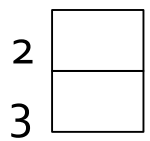
Analysis of block update

- Similar to nested-loop join in databases
  - Break $\mathbf{r}^{new}$ into k blocks that fit in memory
  - Scan $\mathbf{M}$ and $\mathbf{r}^{old}$ once for each block
- k scans of $\mathbf{M}$ and $\mathbf{r}^{old}$
  - $k(|\mathbf{r}| + |\mathbf{M}|) + |\mathbf{r}| = k|\mathbf{M}| + (k+1)|\mathbf{r}|$
- Can we do better?
  - Hint: $\mathbf{M}$ is much bigger than $\mathbf{r}$ (approx 10-20x), so we must avoid reading it k times per iteration

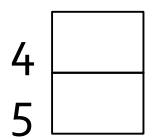
Block stripe update algorithm



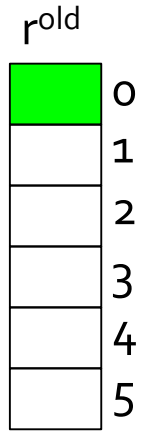
src	degree	destination
0	4	0, 1
1	3	0
2	2	1



0	4	3
2	2	3



0	4	5
1	3	5
2	2	4



Analysis of block stripe update

- Break $\mathbf{M}$ into stripes
  - Each stripe contains only destination nodes in the corresponding block of $\mathbf{r}^{new}$
- Some additional overhead per stripe
  - But it is usually worth it
- Cost per iteration: $|\mathbf{M}|(1+\epsilon) + (k+1)|\mathbf{r}|$

Some problems with PageRank

- Measures generic popularity of a page
  - Biased against topic-specific authorities
  - Solution: Topic-Specific PageRank
- Uses a single measure of importance
  - Other models e.g., hubs-and-authorities
  - Solution: Hubs-and-Authorities (HITS)
- Susceptible to Link spam
  - Artificial link topographies created in order to boost page rank
  - Solution: TrustRank

Outline

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
  - 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
  - For each teleport set S, we get a different vector $\mathbf{r}_S$

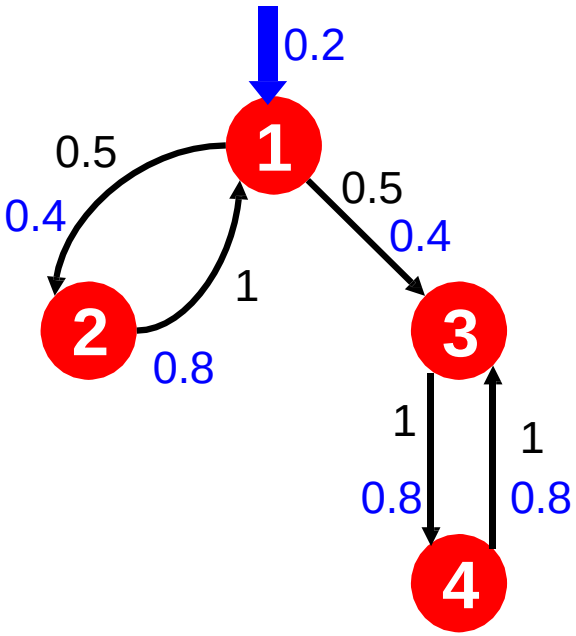
Matrix formulation

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

$$\mathbf{A}_{ij} = \begin{cases} \beta \mathbf{M}_{ij} + (1-\beta)/|S| & \text{if } i \in S \\ \beta \mathbf{M}_{ij} & \text{otherwise} \end{cases}$$

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

Example



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}, β=0.90:
r=[0.17, 0.07, 0.40, 0.36]
S={1} , β=0.8:
r=[0.29, 0.11, 0.32, 0.26]
S={1}, β=0.70:
r=[0.39, 0.14, 0.27, 0.19]

S={1,2,3,4}, β=0.8:
r=[0.13, 0.10, 0.39, 0.36]
S={1,2,3} , β=0.8:
r=[0.17, 0.13, 0.38, 0.30]
S={1,2} , β=0.8:
r=[0.26, 0.20, 0.29, 0.23]
S={1} , β=0.8:
r=[0.29, 0.11, 0.32, 0.26]

Discovering the topic

- Create different PageRanks for different topics
- Which topic ranking to use?
  - User can pick from a menu
  - Classify query into a topic
  - Can use the context of the query
    - E.g., query is launched from a web page talking about a known topic
    - History of queries e.g., "basketball" followed by "Jordan"
  - User context, e.g., user's bookmarks, ...

TrustRank: combating the web spam

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 (Search Engine Optimization) industry might disagree!
- 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-sellers might pretend to be about "movies"
  - Add the word movie 1,000 times to your page and set text color to the background color
  - Or, run the query "movie" on your target search engine, copy the first result into your page and make it "invisible"
- Techniques for achieving high relevance/importance for a web page
- 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

Spam farming

- 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 farm: creating link structures that boost PageRank of a particular page

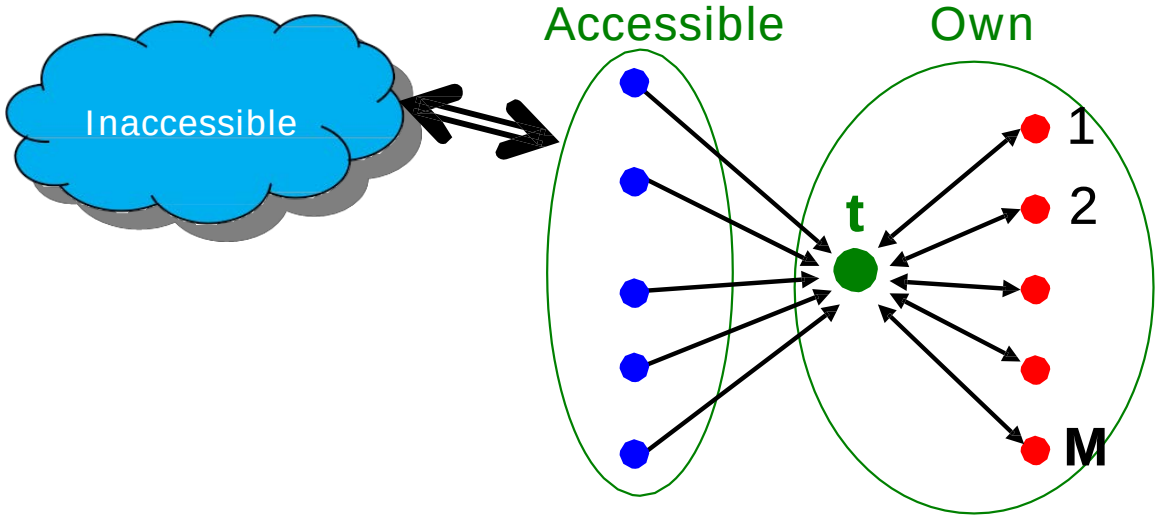
Link spamming

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
- Own 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



N…# pages on the web
M…# of pages spammer owns

Analysis

- x: PageRank contributed by accessible pages
- y: PageRank of target page t
- Rank of each "farm" page $= \frac{\beta y}{M} + \frac{1-\beta}{N}$

$$\begin{aligned} y &= x + \beta M[\frac{\beta y}{M} + \frac{1-\beta}{N}] + \frac{1-\beta}{N} \\ &= x + \beta^2 y + \frac{\beta(1-\beta)M}{N} + \frac{1-\beta}{N} \end{aligned}$$

- Ignore the last term (very small) and solve for y:

$$y = \frac{x}{1-\beta^2} + c\frac{M}{N}$$

  where $c = \frac{\beta}{1+\beta}$
- For $\beta = 0.85$, $1/(1-\beta^2) = 3.6$
- Multiplier effect for "acquired" PageRank
- By making M large, we can make y as large as we want

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
  - TrustRank = topic-specific PageRank with a teleport set of "trusted" pages

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

Trust propagation

- Call the subset of seed pages that are identified as good the trusted pages
- Perform a topic-sensitive PageRank with teleport set = trusted pages
  - Propagate trust through links: each page gets a trust value between 0 and 1
- Use a threshold value and mark all pages below the trust threshold as spam

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

Picking the seed set

- Two conflicting considerations
  - Human has to inspect each seed page, so seed set must be as small as possible
  - Must ensure every good page gets adequate trust rank, so need make all good pages reachable from seed set by short paths
- Suppose we want to pick a seed set of k pages, how?
  1. PageRank: pick the top-k pages by PageRank
  2. Use trusted domains, e.g. .edu, .mil, .gov