Updating PageRank

Amy Langville
Carl Meyer

Department of Mathematics
North Carolina State University
Raleigh, NC

SCCM 11/17/2003

Google

Indexing

- Must index key terms on each page
  Robots crawl the web — software does indexing

- Inverted file structure (like book index: terms $\longrightarrow$ to pages)

$$Term_1 \rightarrow P_i,\ P_j, \ldots$$
$$Term_2 \rightarrow P_k,\ P_l, \ldots$$
$$\vdots$$

Ranking

- Determine a "PageRank" for each page $P_i, P_j, P_k, P_l, \ldots$
  Query independent — Based only on link structure

- Query matching

  $Q = Term_1, Term_2, \ldots$ produces $P_i, P_j, P_k, P_l, \ldots$

- Return $P_i, P_j, P_k, P_l, \ldots$ to user in order of PageRank

Google's PageRank Idea

(Sergey Brin & Lawrence Page 1998)

- Rankings are not query dependent

  Depend only on link structure

  Off-line calculations

- Your page P has some rank $r(P)$

- Adjust $r(P)$ higher or lower depending on ranks of pages that point to P

- Importance is not number of in-links or out-links

  One link to P from Yahoo! is important

  Many links to P from me is not

- Yahoo! points many places — value of link to P is diluted

PageRank

The Definition

$$r(P) = \sum_{P \in \mathcal{B}_P} \frac{r(P)}{|P|}$$

$\mathcal{B}_P$ = {all pages pointing to P}

$|P|$ = number of out links from P

Successive Refinement

Start with $r_0(P_i) = 1/n$ for all pages $P_1, P_2, \ldots, P_n$

Iteratively refine rankings for each page

$$r_1(P_i) = \sum_{P \in \mathcal{B}_{P_i}} \frac{r_0(P)}{|P|}$$

$$r_2(P_i) = \sum_{P \in \mathcal{B}_{P_i}} \frac{r_1(P)}{|P|}$$

$$\ddots$$

$$r_{j+1}(P_i) = \sum_{P \in \mathcal{B}_{P_i}} \frac{r_j(P)}{|P|}$$

In Matrix Notation

After Step j

$$\boldsymbol{\pi}_j^T = \left[r_j(P_1),\ r_j(P_2),\ \cdots,\ r_j(P_n)\right]$$

$$\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T \mathbf{P} \quad \text{where} \quad p_{ij} = \begin{cases} 1/|P_i| & \text{if } i \rightarrow j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{PageRank} = \lim_{j\to\infty} \boldsymbol{\pi}_j^T = \boldsymbol{\pi}^T$$ (provided limit exists)

It's A Markov Chain

$\mathbf{P} = \left[p_{ij}\right]$ is a stochastic matrix (row sums = 1)

Each $\boldsymbol{\pi}_j^T$ is a probability distribution vector $\left(\sum_i r_j(P_i) = 1\right)$

$\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T \mathbf{P}$ is random walk on the graph defined by links

$\boldsymbol{\pi}^T = \lim_{j\to\infty} \boldsymbol{\pi}_j^T$ = stationary probability distribution

Random Surfer

Web Surfer Randomly Clicks On Links (Back button not a link)

Long-run proportion of time on page P_i is $\boldsymbol{\pi}_i$

Problems

Dead end page (nothing to click on)

$\boldsymbol{\pi}^T$ not well defined

Could get trapped into a cycle $(P_i \rightarrow P_j \rightarrow P_i)$

No convergence

Convergence

Markov chain must be irreducible and aperiodic

Bored Surfer Enters Random URL

Replace $\mathbf{P}$ by $\widetilde{\mathbf{P}} = \alpha\mathbf{P} + (1-\alpha)\mathbf{E}$ $\quad e_{ij} = 1/n \quad \alpha \approx .85$

Different $\mathbf{E} = \mathbf{e}\mathbf{v}^T$ and α allow customization & speedup

Computing π^T

A Big Problem

Solve $\boldsymbol{\pi}^T = \boldsymbol{\pi}^T\mathbf{P}$ (stationary distribution vector)

$\boldsymbol{\pi}^T(\mathbf{I} - \mathbf{P}) = 0$ (too big for direct solves)

THE WORLD'S LARGEST MATRIX COMPUTATION

Google's PageRank is an eigenvector of a matrix of order 2.7 billion.

BY CLEVE MOLER

One of the reasons why Google is such an effective search engine is the PageRank™ algorithm, developed by Google's founders, Larry Page and Sergey Brin, when they were graduate students at Stanford University. PageRank is determined entirely by the link structure of the Web. It is recomputed about once a month and does not involve any of the actual content of Web pages or of any individual query. Then, for any particular query, Google finds the pages on the Web that match that query and lists those pages in the order of their PageRank.

Imagine surfing the Web, going from page to page by randomly choosing an outgoing link from one page to get to the next. This can lead to dead ends at pages with no outgoing links, or cycles around cliques of interconnected pages. So, a certain fraction of the time, simply choose a random page from anywhere on the Web. This theoretical random walk of the Web is a *Markov chain* or *Markov process*. The limiting probability that a dedicated random surfer visits any particular page is its PageRank. A page has high rank if it has links to and from other pages with high rank.

Let W be the set of Web pages that can reached by following a chain of hyperlinks starting from a page at Google and let n be the number of pages in W. The set W actually varies with time, but in May 2002, n was about 2.7 billion. Let G be the n-by-n connectivity matrix of

It tells us that the largest eigenvalue of A is equal to one and that the corresponding eigenvector, which satisfies the equation

$$x = Ax,$$

exists and is unique to within a scaling factor. When this scaling factor is chosen so that

$$\Sigma_i x_i = 1$$

then x is the state vector of the Markov chain. The elements of x are Google's PageRank.

If the matrix were small enough to fit in MATLAB, one way to compute the eigenvector x would be to start with a good approximate solution, such as the PageRanks from the previous month, and simply repeat the assignment statement

```
x = Ax
```

until successive vectors agree to within specified tolerance. This is known as the power method and is about the only possible approach for very large n. I'm not sure how Google actually computes PageRank, but one step of the power method would require one pass over a database of Web pages, updating weighted reference counts generated by the hyperlinks between pages.

Computing π^T

A Big Problem

Solve $\boldsymbol{\pi}^T = \boldsymbol{\pi}^T\mathbf{P}$ (stationary distribution vector)

$\boldsymbol{\pi}^T(\mathbf{I} - \mathbf{P}) = 0$ (too big for direct solves)

Start with $\boldsymbol{\pi}_0^T = \mathbf{e}/n$ and iterate $\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T\mathbf{P}$ (power method)

Computing π^T

A Big Problem

Solve $\boldsymbol{\pi}^T = \boldsymbol{\pi}^T \mathbf{P}$ (stationary distribution vector)

$\boldsymbol{\pi}^T(\mathbf{I} - \mathbf{P}) = 0$ (too big for direct solves)

Start with $\boldsymbol{\pi}_0^T = \mathbf{e}/n$ and iterate $\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T \mathbf{P}$ (power method)

A Bigger Problem — Updating

Link structure of web is extremely dynamic

Links on CNN change frequently

Links are added and deleted almost continuously

Google says just start from scratch every 3 to 4 weeks

Old results don't help to restart

The Updating Problem

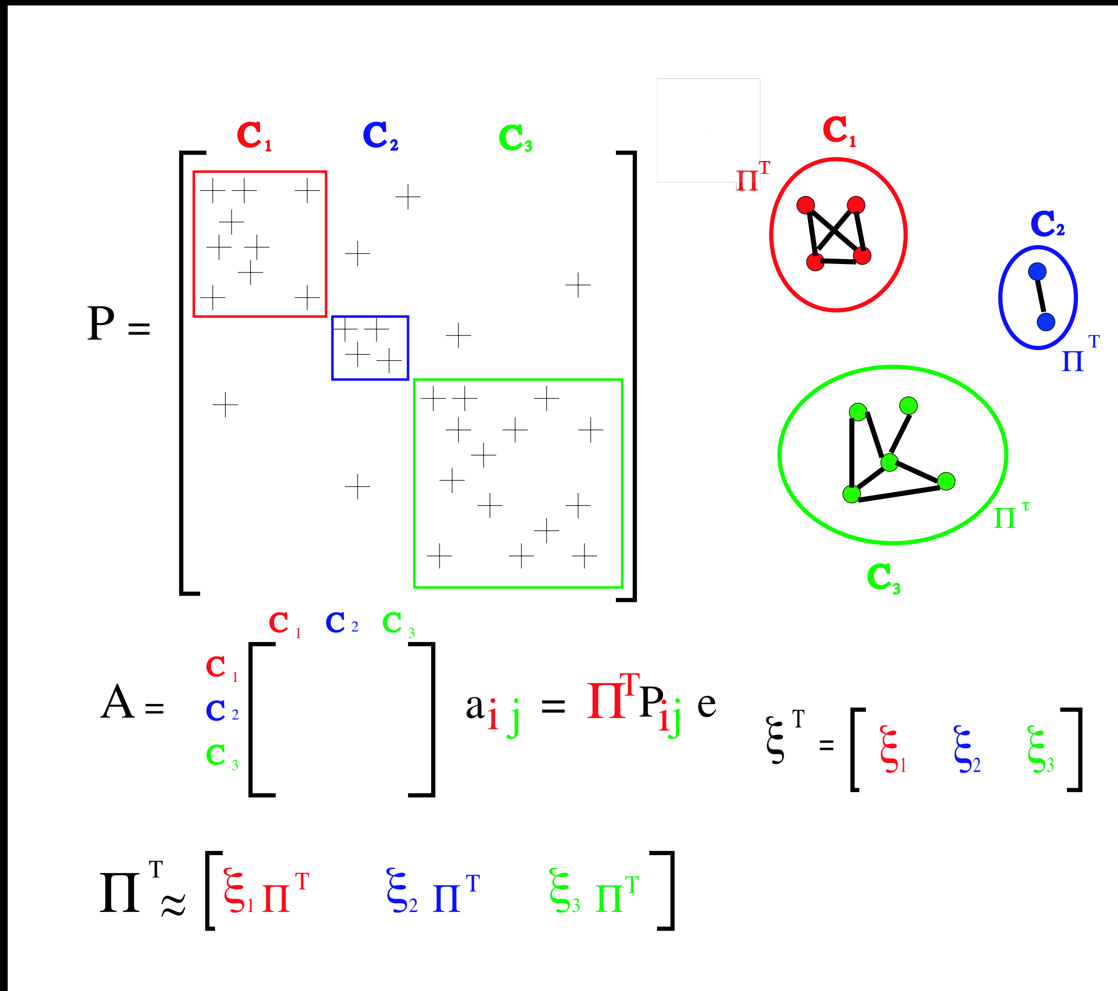
Given: Original chain's P and π^T

and new chain's $\widetilde{P}$

Find: new chain's $\widetilde{\pi}^T$

Idea behind Aggregation

Best for NCD systems (Simon and Ando (1960s), Courtois (1970s))



Pro

exploits structure to reduce work

Con

produces an approximation, quality is dependent on degree of coupling

Iterative Aggregation

- Problem: repeated aggregation leads to fixed point.
- Solution: Do a power step to move off fixed point.
- Do this iteratively. Approximations improve and approach exact solution.
- Success with NCD systems, not in general.

Input: approximation to $\boldsymbol{\pi}^T$

get censored distributions $\boldsymbol{\pi}^T$ $\boldsymbol{\pi}^T$ $\boldsymbol{\pi}^T$

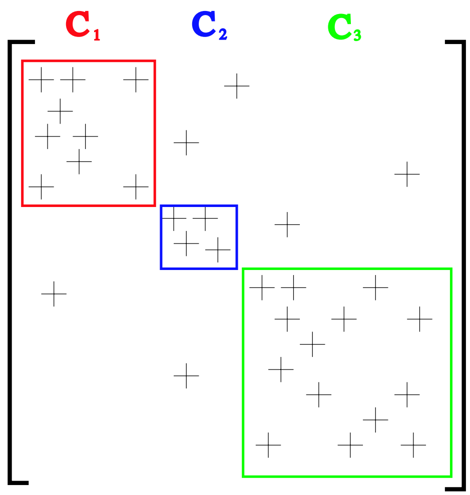
get coupling constants $\boldsymbol{\xi}_i$

Output: get approximate global stationary distribution $\boldsymbol{\pi}^T = \left[\boldsymbol{\xi}_1 \boldsymbol{\pi}^T \quad \boldsymbol{\xi}_2 \boldsymbol{\pi}^T \quad \boldsymbol{\xi}_3 \boldsymbol{\pi}^T \right]$

Output: move off fixed point with power step

Exact Aggregation

(Meyer 1989)

P = 

s_i^T = censored (stat.) dist. of stochastic complement $\mathbf{S}_i$

$$S_i = P_{ii} + P_{i*}(I - P_*)^{-1} P_{*i}$$

For 2-level partition,

$$S_1 = P_{11} + P_{12}(I - P_{22})^{-1} P_{21}$$

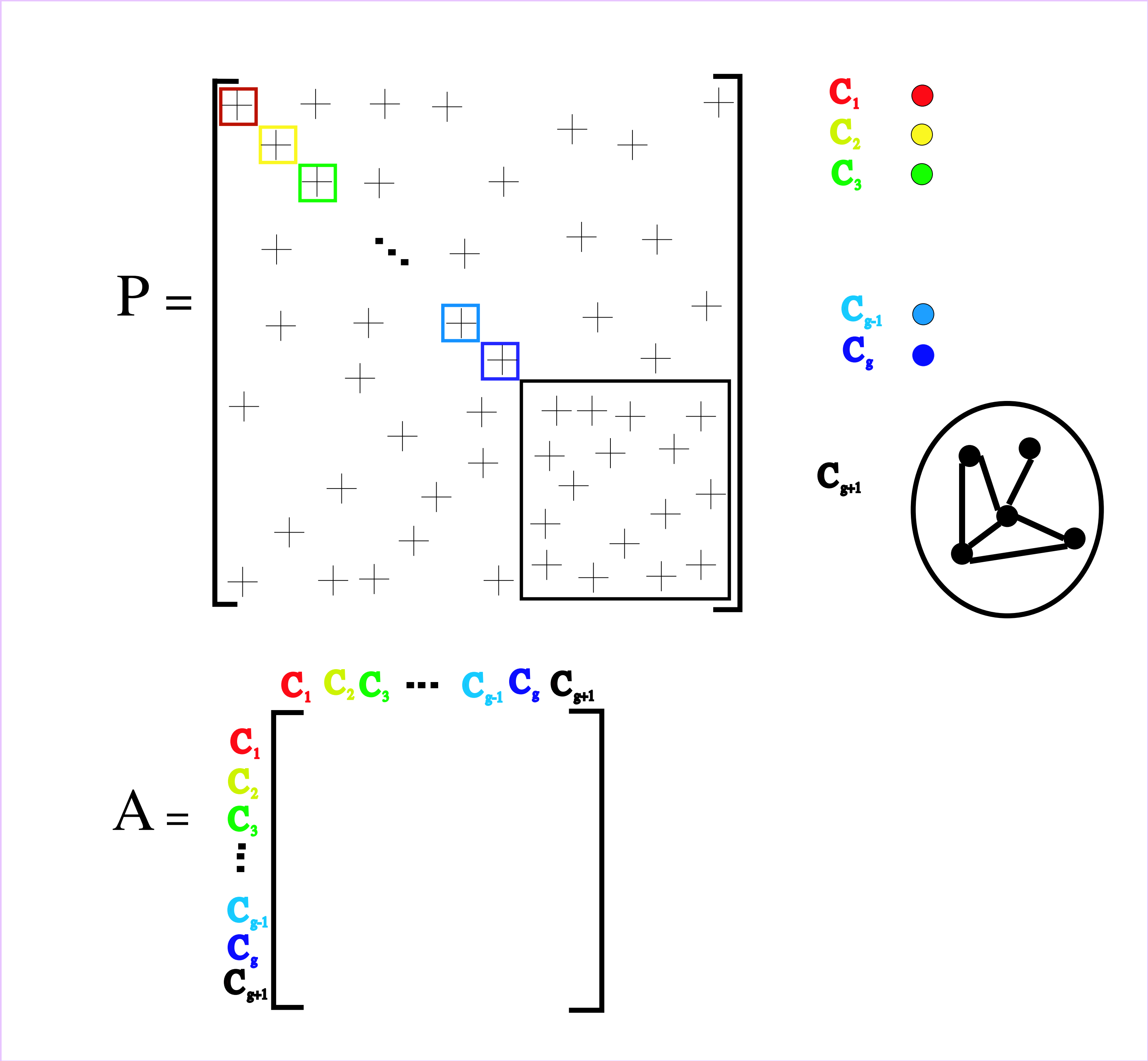
$$A = \begin{array}{c|c} & C_1 \quad C_2 \quad C_3 \\ \begin{matrix} C_1 \\ C_2 \\ C_3 \end{matrix} & \left[\begin{matrix} \quad & \quad & \quad \\ & & \\ & & \end{matrix}\right] \end{array} \qquad a_{ij} = s_i^T P_{ij} e \qquad \xi^T = [\xi_1 \quad \xi_2 \quad \xi_3]$$

$$\Pi^T = [\xi_1 s_1^T \quad \xi_2 s_2^T \quad \xi_3 s_3^T]$$

Pro only one step needed to produce exact global vector

Con SC matrices $\mathbf{S}_i$ are very expensive to compute

Back to Updating . . .



Aggregation

Partitioned Matrix

$$\mathbf{P}_{n\times n} = \begin{array}{cc} & \begin{array}{cc} G & \overline{G} \end{array} \\ \begin{array}{c} G \\ \overline{G} \end{array} & \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{pmatrix} \end{array} = \left[\begin{array}{c|c|c|c} p_{11} & \cdots & p_{1g} & \mathbf{r}_1^T \\ \hline \vdots & \ddots & \vdots & \vdots \\ \hline p_{g1} & \cdots & p_{gg} & \mathbf{r}_g^T \\ \hline \mathbf{c}_1 & \cdots & \mathbf{c}_g & \mathbf{P}_{22} \end{array}\right]$$

$$\boldsymbol{\pi}^T = (\pi_1, \ldots \pi_g \mid \pi_{g+1}, \ldots, \pi_n)$$

Advantages of this Partition

$p_{11} \cdots p_{gg}$ are 1×1 $\Longrightarrow$ Stochastic complements $= 1$

$\Longrightarrow$ censored distributions $= 1$

Only one significant complement $\quad \mathbf{S}_2 = \mathbf{P}_{22} + \mathbf{P}_{21}(\mathbf{I} - \mathbf{P}_{11})^{-1}\mathbf{P}_{12}$

Only one significant censored dist $\quad \mathbf{s}_2^T\mathbf{S}_2 = \mathbf{s}_2^T$

A/D Theorem $\Longrightarrow$ $\mathbf{s}_2^T = (\pi_{g+1}, \ldots, \pi_n) / \sum_{i=g+1}^{n} \pi_i$

Aggregation Matrix

$$\mathbf{A} = \left[\begin{array}{c|c|c|c} p_{11} & \cdots & p_{1g} & \mathbf{r}_1^T\mathbf{e} \\ \hline \vdots & \ddots & \vdots & \vdots \\ \hline p_{g1} & \cdots & p_{gg} & \mathbf{r}_g^T\mathbf{e} \\ \hline \mathbf{s}_2^T\mathbf{c}_1 & \cdots & \mathbf{s}_2^T\mathbf{c}_g & \mathbf{s}_2^T\mathbf{P}_{22}\mathbf{e} \end{array}\right]_{(g+1)\times(g+1)} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12}\mathbf{e} \\ \mathbf{s}_2^T\mathbf{P}_{21} & 1 - \mathbf{s}_2^T\mathbf{P}_{21}\mathbf{e} \end{bmatrix}$$

The Aggregation/Disaggregation Theorem

If $\boldsymbol{\alpha}^T = (\alpha_1, \ldots, \alpha_g, \alpha_{g+1})$ = stationary dist for $\mathbf{A}$

Then $\boldsymbol{\pi}^T = \left(\alpha_1, \ldots, \alpha_g \,|\, \alpha_{g+1}\mathbf{s}_2^T\right)$ = stationary dist for $\mathbf{P}$

Trouble! Always A Big Problem

G small $\Rightarrow$ $\overline{G}$ big $\Rightarrow$ $\mathbf{S}_2 = \mathbf{P}_{22} + \mathbf{P}_{21}(\mathbf{I} - \mathbf{P}_{11})^{-1}\mathbf{P}_{12}$ large

G big $\Rightarrow$ $\mathbf{A}$ large

Approximate Aggregation

Assumption

Updating involves relatively few states

$$G \text{ small} \quad \Rightarrow \quad \mathbf{A} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12}\mathbf{e} \\ \mathbf{s}_2^T\mathbf{P}_{21} & 1 - \mathbf{s}_2^T\mathbf{P}_{21}\mathbf{e} \end{bmatrix}_{(g+1)\times(g+1)} \quad \text{small}$$

Approximation

$$(\pi_{g+1}, \ldots, \pi_n) \approx (\phi_{g+1}, \ldots, \phi_n),$$

where $\boldsymbol{\phi}^T$ is old PageRank vector and $\boldsymbol{\pi}^T$ is new, updated PageRank

$$\mathbf{s}_2^T = \frac{(\pi_{g+1}, \ldots, \pi_n)}{\sum_{i=g+1}^{n} \pi_i} \approx \frac{(\phi_{g+1}, \ldots, \phi_n)}{\sum_{i=g+1}^{n} \phi_i} = \widetilde{\mathbf{s}}_2^T$$

(avoids computing $\widetilde{\mathbf{s}}_2^T$ for large $\mathbf{S}_2$)

$$\mathbf{A} \approx \widetilde{\mathbf{A}} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12}\mathbf{e} \\ \widetilde{\mathbf{s}}_2^T\mathbf{P}_{21} & 1 - \widetilde{\mathbf{s}}_2^T\mathbf{P}_{21}\mathbf{e} \end{bmatrix}$$

$$\boldsymbol{\alpha}^T \approx \widetilde{\boldsymbol{\alpha}}^T = \left(\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_g, \widetilde{\alpha}_{g+1} \right)$$

$$\boldsymbol{\pi}^T \approx \widetilde{\boldsymbol{\pi}}^T = \left(\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_g \,\middle|\, \widetilde{\alpha}_{g+1}\widetilde{\mathbf{s}}_2^T \right)$$

(not bad)

Iterative Aggregation

Improve By Successive Aggregation / Disaggregation?

NO

Can’t do A/D twice — a fixed point emerges

Solution

Perturb A/D output to move off of fixed point

Move it in direction of solution

$\widetilde{\widetilde{\boldsymbol{\pi}}}^T = \widetilde{\boldsymbol{\pi}}^T \mathbf{P}$ (a smoothing step)

The Iterative A/D Updating Algorithm

Determine the “G-set” partition $\mathcal{S} = G \cup \overline{G}$

Approximate A/D step generates approximation $\widetilde{\boldsymbol{\pi}}^T$

Smooth the result $\widetilde{\widetilde{\boldsymbol{\pi}}}^T = \widetilde{\boldsymbol{\pi}}^T \mathbf{P}$

Use $\widetilde{\widetilde{\boldsymbol{\pi}}}^T$ as input to another approximate aggregation step

$\vdots$

How to Partition for Updating Problem?

Intuition

- There are some bad states (G) and some good states ($\overline{G}$).
- Give more attention to bad states. Each state in G forms a partitioning level. Much progress toward correct PageRank is made during aggregation step.
- Lump good states in $\overline{G}$ into 1 superstate. Progress toward correct PageRank is made during smoothing step (power iteration).

Definitions for "Good" and "Bad"

1. Good = states least likely to have π_i change
   Bad = states most likely to have π_i change

2. Good = states with smallest π_i after k transient steps
   Bad = states "nearby", with largest π_i after k transient steps

3. Good = smallest π_i from old PageRank vector
   Bad = largest π_i from old PageRank vector

4. Good = fast–converging states
   Bad = slow–converging states

Determining "Fast" and "Slow"

Consider power method and its rate of convergence

$$\boldsymbol{\pi}_{k+1}^T = \boldsymbol{\pi}_k^T \mathbf{P} = \boldsymbol{\pi}_k^T \mathbf{e}\boldsymbol{\pi}^T + \lambda_2^k \boldsymbol{\pi}_k^T \mathbf{x}_2 \mathbf{y}_2^T + \lambda_3^k \boldsymbol{\pi}_k^T \mathbf{x}_3 \mathbf{y}_3^T + \cdots + \lambda_n^k \boldsymbol{\pi}_k^T \mathbf{x}_n \mathbf{y}_n^T$$

Asymptotic rate of convergence is rate at which $\lambda_2^k \to \mathbf{0}$

Consider convergence of elements

Some states converge to stationary value faster than λ_2–rate, due to LH e–vector $\mathbf{y}_2^T$.

Partitioning Rule

Put states with largest $|\mathbf{y}_2^T|_i$ values in bad group G, where they receive more individual attention in aggregation method.

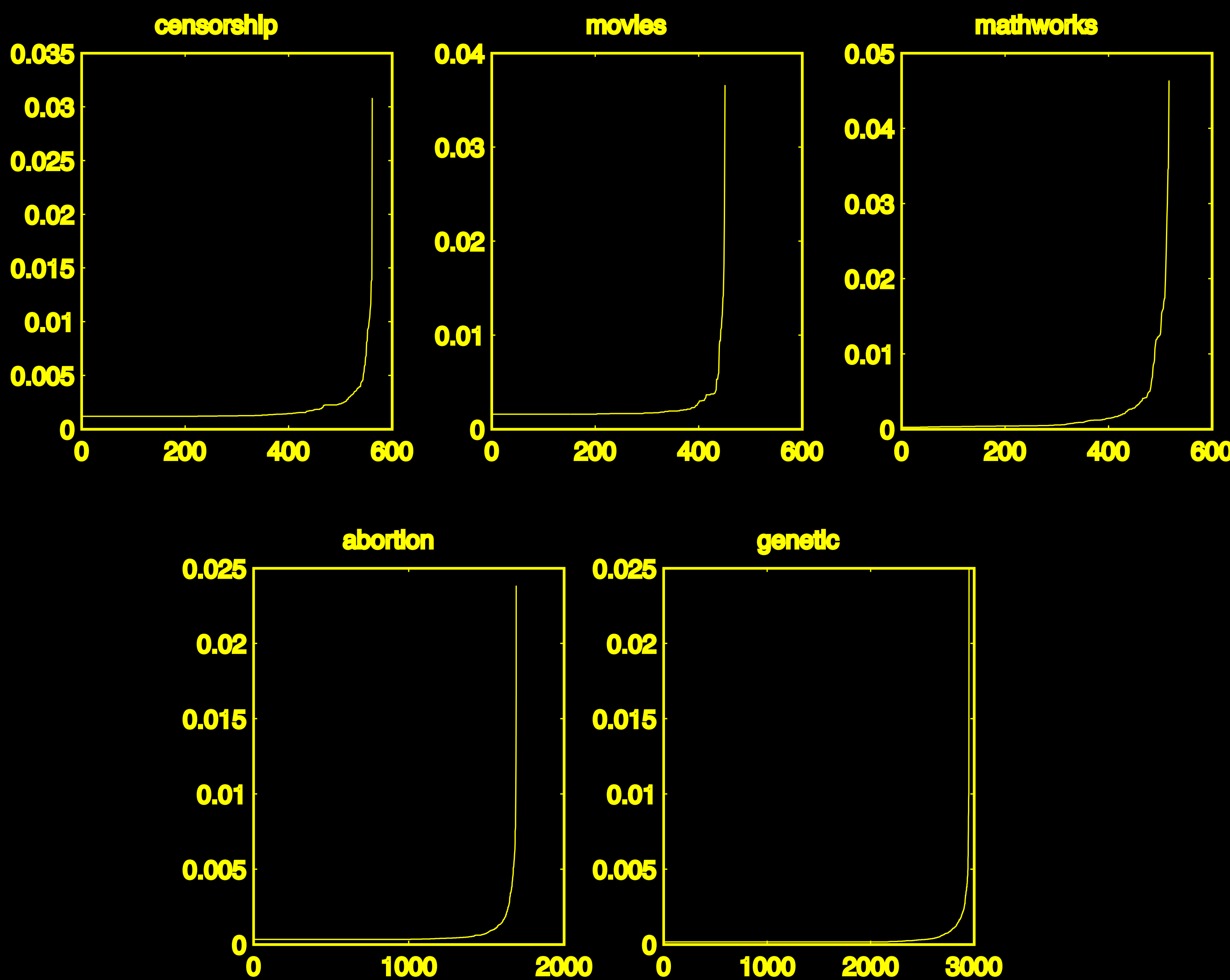
Practicality

$\mathbf{y}_2^T$ expensive, but for PageRank problem, Kamvar et al. show states with large π_i are slow-converging. $\Rightarrow$ inexpensive soln = use old $\boldsymbol{\pi}^T$ to determine G. (adaptively approximate $\mathbf{y}_2^T$)

Power law for PageRank

Scale-free Model of Web network creates power laws

(Kamvar, Barabasi, Raghavan)



Convergence

Theorem

Always converges to stationary dist $\boldsymbol{\pi}^T$ for **P**

Converges for all partitions $\mathcal{S} = G \cup \overline{G}$

Rate of convergence is rate at which $\mathbf{S}_2^n$ converges

$$\mathbf{S}_2 = \mathbf{P}_{22} + \mathbf{P}_{21}(\mathbf{I} - \mathbf{P}_{11})^{-1}\mathbf{P}_{12}$$

Dictated by Jordan structure of $\lambda_2(\mathbf{S}_2)$

$\lambda_2(\mathbf{S}_2)$ simple $\Longrightarrow \boldsymbol{\pi}_k^T \rightarrow \boldsymbol{\pi}^T$ at the rate at which $\lambda_2^n \rightarrow 0$

The Game

Goal now is to find a relatively small G that minimizes $\lambda_2(\mathbf{S}_2)$

Experiments

Test Networks From Crawl Of Web (Supplied by Ronny Lempel)

Censorship (Sites concerning "censorship on the net")
562 nodes 736 links

Movies (Sites concerning "movies")
451 nodes 713 links

MathWorks (Supplied by Cleve Moler)
517 nodes 13,531 links

Abortion (Sites concerning "abortion")
1,693 nodes 4,325 links

Genetics (Sites concerning "genetics")
2,952 nodes 6,485 links

Parameters

Number Of Nodes (States) Added

$\mathbf{3}$

Number Of Nodes (States) Removed

$\mathbf{5}$

Number Of Links Added

(Different values have little effect on results)

$\mathbf{10}$

Number Of Links Removed

$\mathbf{20}$

Stopping Criterion

1-norm of residual $< 10^{-10}$

Censorship

Power Method

Iterations	Time
38	1.40

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
5	38	1.68
10	38	1.66
15	38	1.56
20	20	1.06
25	20	1.05
50	10	.69
100	8	.55
300	6	.65
400	5	.70

$nodes = \mathbf{562} \quad links = \mathbf{736}$

Censorship

Power Method

Iterations	Time
38	1.40

Iterative Aggregation

$\|G\|$	Iterations	Time
5	38	1.68
10	38	1.66
15	38	1.56
20	20	1.06
25	20	1.05
50	10	.69
100	8	.55
200	6	.53
300	6	.65
400	5	.70

$nodes$ = **562** $links$ = **736**

Movies

Power Method

Iterations	Time
17	.40

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
5	12	.39
10	12	.37
15	11	.36
20	11	.35
100	9	.33
200	8	.35
300	7	.39
400	6	.47

$nodes = 451 \quad links = 713$

Movies

Power Method

Iterations	Time
17	.40

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
5	12	.39
10	12	.37
15	11	.36
20	11	.35
25	11	.31
50	9	.31
100	9	.33
200	8	.35
300	7	.39
400	6	.47

$nodes = 451 \quad links = 713$

MathWorks

Power Method

Iterations	Time
$\mathbf{54}$	$\mathbf{1.25}$

Iterative Aggregation

$\|G\|$	Iterations	Time
$\mathbf{5}$	$\mathbf{53}$	$\mathbf{1.18}$
$\mathbf{10}$	$\mathbf{52}$	$\mathbf{1.29}$
$\mathbf{15}$	$\mathbf{52}$	$\mathbf{1.23}$
$\mathbf{20}$	$\mathbf{42}$	$\mathbf{1.05}$
$\mathbf{25}$	$\mathbf{20}$	$\mathbf{1.13}$
$\mathbf{300}$	$\mathbf{11}$	$\mathbf{.83}$
$\mathbf{400}$	$\mathbf{10}$	$\mathbf{1.01}$

$nodes = \mathbf{517} \quad links = \mathbf{13,531}$

MathWorks

Power Method

Iterations	Time
54	1.25

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
5	53	1.18
10	52	1.29
15	52	1.23
20	42	1.05
25	20	1.13
50	18	.70
100	16	.70
200	13	.70
300	11	.83
400	10	1.01

$nodes = \mathbf{517} \quad links = \mathbf{13,531}$

Abortion

Power Method

Iterations	Time
106	37.08

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
5	109	38.56
10	105	36.02
15	107	38.05
20	107	38.45
25	97	34.81
50	53	18.80
250	12	5.62
500	6	5.21
750	5	10.22
1000	5	14.61

$nodes = \mathbf{1,693} \quad links = \mathbf{4,325}$

Abortion

Power Method

Iterations	Time
106	37.08

Iterative Aggregation

$\|G\|$	Iterations	Time
5	109	38.56
10	105	36.02
15	107	38.05
20	107	38.45
25	97	34.81
50	53	18.80
100	13	5.18
250	12	5.62
500	6	5.21
750	5	10.22
1000	5	14.61

$nodes = \mathbf{1,693} \quad links = \mathbf{4,325}$

Genetics

Power Method

Iterations	Time
$\mathbf{92}$	$\mathbf{91.78}$

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
$\mathbf{5}$	$\mathbf{91}$	$\mathbf{88.22}$
$\mathbf{10}$	$\mathbf{92}$	$\mathbf{92.12}$
$\mathbf{20}$	$\mathbf{71}$	$\mathbf{72.53}$
$\mathbf{50}$	$\mathbf{25}$	$\mathbf{25.42}$
$\mathbf{100}$	$\mathbf{19}$	$\mathbf{20.72}$
$\mathbf{250}$	$\mathbf{13}$	$\mathbf{14.97}$
$\mathbf{1000}$	$\mathbf{5}$	$\mathbf{17.76}$
$\mathbf{1500}$	$\mathbf{5}$	$\mathbf{31.84}$

$nodes = \mathbf{2,952} \quad links = \mathbf{6,485}$

Genetics

Power Method

Iterations	Time
$\mathbf{92}$	$\mathbf{91.78}$

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
$\mathbf{5}$	$\mathbf{91}$	$\mathbf{88.22}$
$\mathbf{10}$	$\mathbf{92}$	$\mathbf{92.12}$
$\mathbf{20}$	$\mathbf{71}$	$\mathbf{72.53}$
$\mathbf{50}$	$\mathbf{25}$	$\mathbf{25.42}$
$\mathbf{100}$	$\mathbf{19}$	$\mathbf{20.72}$
$\mathbf{250}$	$\mathbf{13}$	$\mathbf{14.97}$
$\mathbf{500}$	$\mathbf{7}$	$\mathbf{11.14}$
$\mathbf{1000}$	$\mathbf{5}$	$\mathbf{17.76}$
$\mathbf{1500}$	$\mathbf{5}$	$\mathbf{31.84}$

$nodes = \mathbf{2,952} \quad links = \mathbf{6,485}$

Conclusions

- First updating algorithm to handle both element– and state–updates.
- Algorithm is very sensitive to partition.
- For PageRank problem, partition can be determined cheaply from old PageRanks.
- For general Markov updating, use $\mathbf{y}_2^T$ to determine partition. When too expensive, approximate adaptively with Aitken's δ^2 or difference of successive iterates.
- Improvements
  - Practical
    - Optimize G-set
    - Accelerate Smoothing
  - Theoretical
    - Relationship between partitioning by $\mathbf{y}_2^T$ and $\lambda_2(\mathbf{S}_2)$ not well-understood.
- Predict algorithm and partitioning by old $\boldsymbol{\pi}^T$ will work very well on other scale-free networks.