Reordering and Updating for the PageRank Problem

Amy Langville
Carl Meyer

Department of Mathematics
North Carolina State University
Raleigh, NC

Bertinoro, Italy

Outline

- PageRank Solution Methods
- A Reordering for PageRank
- Updating PageRank

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)

$$Term_{\mathbf{1}} \rightarrow P_i,\ P_j, \ldots$$
$$Term_{\mathbf{2}} \rightarrow P_k,\ P_l, \ldots$$
$$\vdots$$

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

  $Q = Term_{\mathbf{1}}, Term_{\mathbf{2}}, \ldots$ 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

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

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

$$\text{PageRank} = \lim_{j \to \infty} \boldsymbol{\pi}_j^T = \boldsymbol{\pi}^T$$

(provided limit exists)

It's Almost a Markov Chain

P has row sums = 1 for ND nodes, row sums = 0 for D nodes

In Matrix Notation

After Step j

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

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

$$\text{PageRank} = \lim_{j\to\infty} \boldsymbol{\pi}_j^T = \boldsymbol{\pi}^T$$

(provided limit exists)

It's Almost a Markov Chain

P has row sums = 1 for ND nodes, row sums = 0 for D nodes

Stochasticity Fix: $\bar{\mathbf{P}} = \mathbf{P} + \mathbf{a}\mathbf{v}^T$. ($a_i$=1 for $i \in D$, 0, o.w.)

In Matrix Notation

After Step j

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

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

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

It's Almost a Markov Chain

P has row sums = 1 for ND nodes, row sums = 0 for D nodes

Stochasticity Fix: $\bar{\mathbf{P}} = \mathbf{P} + \mathbf{a}\mathbf{v}^T$. ($a_i$=1 for $i \in D$, 0, o.w.)

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

$\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T \bar{\mathbf{P}}$ 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

Irreducibility Fix: $\bar{\bar{\mathbf{P}}} = \alpha\bar{\mathbf{P}} + (1-\alpha)\mathbf{E}$ $\quad e_{ij} = 1/n \quad \alpha \approx .85$

$$\bar{\bar{\mathbf{P}}} = \alpha\mathbf{P} + \alpha\,\mathbf{a}\,\mathbf{v}^T + (1-\alpha)\mathbf{E}$$

Different $\mathbf{E} = \mathbf{e}\mathbf{v}^T$ and α allow customization & speedup, yet rank-one update maintained; $\bar{\bar{\mathbf{P}}} = \alpha\mathbf{P} + (\alpha\,\mathbf{a} + (1-\alpha)\,\mathbf{e})\mathbf{v}^T$

Computing π^T

A Big Problem

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

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

Computing π^T

A Big Problem

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

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

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

Power Method to compute PageRank

$$\boldsymbol{\pi}_0^T = \mathbf{e}^T/n$$

until convergence, do

$$\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T \bar{\bar{\mathbf{P}}}$$

(dense computation)

end

Power Method to compute PageRank

$\boldsymbol{\pi}_0^T = \mathbf{e}^T/n$

until convergence, do

X $\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T \,\bar{\bar{\mathbf{P}}}$ (dense computation)

- $\boldsymbol{\pi}_{j+1}^T = \alpha\, \boldsymbol{\pi}_j^T\, \bar{\mathbf{P}} + (1-\alpha)\, \boldsymbol{\pi}_j^T\, \mathbf{e}\, \mathbf{v}^T$ (sparser computation)

end

Power Method to compute PageRank

$\boldsymbol{\pi}_0^T = \mathbf{e}^T/n$

until convergence, do

X $\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T \bar{\bar{\mathbf{P}}}$ (dense computation)

X $\boldsymbol{\pi}_{j+1}^T = \alpha\, \boldsymbol{\pi}_j^T \bar{\mathbf{P}} + (1-\alpha)\, \boldsymbol{\pi}_j^T \mathbf{e}\, \mathbf{v}^T$ (sparser computation)

- $\boldsymbol{\pi}_{j+1}^T = \alpha\, \boldsymbol{\pi}_j^T \mathbf{P} + (\alpha\, \boldsymbol{\pi}_j^T \mathbf{a} + (1-\alpha))\, \mathbf{v}^T$ (even less computation)

end

- **P** is very, very sparse with about 3-10 nonzeros per row.
- $\Rightarrow$ one vector-matrix mult. is $O(nnz(\mathbf{P})) \approx O(n)$.

Convergence

Can prove $\lambda_2(\bar{\bar{\mathbf{P}}}) = \boldsymbol{\alpha}$

($\Longrightarrow$ asymptotic rate of convergence of PageRank method is rate at which $\alpha^k \longrightarrow \mathbf{0}$)

Google

– uses $\alpha = .85$ (5/6, 1/6 interpretation)

– report 50-100 iterations til convergence

– still takes days to converge

Enhancements to the PR power method

- Kamvar et al. Extrapolation
- Kamvar et al. Adaptive PageRank
- Kamvar et al. BlockRank
- Lee et al. Lumpability of Dangling Nodes
- Langville/Meyer: Updating PageRank
- Ipsen/Kirkland: more theory for Langville/Meyer

Linear System Formulation

For $\bar{\bar{\mathbf{P}}}$

$$\pi^T(\mathbf{I} - \bar{\bar{\mathbf{P}}}) = 0^T \quad \text{and} \quad \pi^T \mathbf{e} = 1.$$

For $\bar{\mathbf{P}}$

$$\pi^T(\mathbf{I} - \alpha\bar{\mathbf{P}}) = (1-\alpha)\mathbf{v}^T \quad \text{and} \quad \pi^T \mathbf{e} = 1.$$

For $\mathbf{P}$

$$\pi^T(\mathbf{I} - \alpha\mathbf{P}) = \mathbf{v}^T \quad \text{and} \quad \pi^T \mathbf{e} = 1.$$

(**P** is very sparse, 3-10 nonzeros per row)

Properties of $(\mathbf{I} - \alpha \mathbf{P})$:

1. $(\mathbf{I} - \alpha\mathbf{P})$ is nonsingular.
2. $(\mathbf{I} - \alpha\mathbf{P})$ is an **M**-matrix.
3. The row sums of $(\mathbf{I} - \alpha\mathbf{P})$ are either $1 - \alpha$ for ND nodes or 1 for D nodes.
4. $\|\mathbf{I} - \alpha\mathbf{P}\|_\infty = 1 + \alpha$.
5. Since $(\mathbf{I} - \alpha\mathbf{P})$ is an **M**-matrix, $(\mathbf{I} - \alpha\mathbf{P})^{-1} \geq \mathbf{0}$.
6. The row sums of $(\mathbf{I} - \alpha\mathbf{P})^{-1}$ are equal to 1 for the D nodes and less than or equal to $1/(1 - \alpha)$ for the ND nodes.
7. The condition number $\kappa_\infty(\mathbf{I} - \alpha\mathbf{P}) \leq (1 + \alpha)/(1 - \alpha)$.
8. The row of $(\mathbf{I} - \alpha\mathbf{P})^{-1}$ corresponding to D node i is $\mathbf{e}_i^T$.

ND-D Reordering

$$\mathbf{P} = \begin{array}{c c} & \begin{array}{cc} ND & D \end{array} \\ \begin{array}{c} ND \\ D \end{array} & \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ 0 & 0 \end{pmatrix} \end{array}.$$

$$(\mathbf{I} - \alpha\mathbf{P}) = \begin{bmatrix} \mathbf{I} - \alpha\mathbf{P}_{11} & -\alpha\mathbf{P}_{12} \\ 0 & \mathbf{I} \end{bmatrix}.$$

$$(\mathbf{I} - \alpha\mathbf{P})^{-1} = \begin{bmatrix} (\mathbf{I} - \alpha\mathbf{P}_{11})^{-1} & \alpha(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}\mathbf{P}_{12} \\ 0 & \mathbf{I} \end{bmatrix}.$$

Algorithm 1: ND-D Reordering

Solve $\boldsymbol{\pi}^T(\mathbf{I} - \alpha\mathbf{P}) = \mathbf{v}^T$ and $\boldsymbol{\pi}^T\mathbf{e} = 1$.

Algorithm 1:

1. Solve for $\boldsymbol{\pi}_1^T$ in $\boldsymbol{\pi}_1^T(\mathbf{I} - \alpha\mathbf{P}_{11}) = \mathbf{v}_1^T$.

2. Compute $\boldsymbol{\pi}_2^T = \alpha\boldsymbol{\pi}_1^T\mathbf{P}_{12} + \mathbf{v}_2^T$.

3. Normalize $\boldsymbol{\pi}^T = [\boldsymbol{\pi}_1^T \;\; \boldsymbol{\pi}_2^T]/\|[\boldsymbol{\pi}_1^T \;\; \boldsymbol{\pi}_2^T]\|_1$.

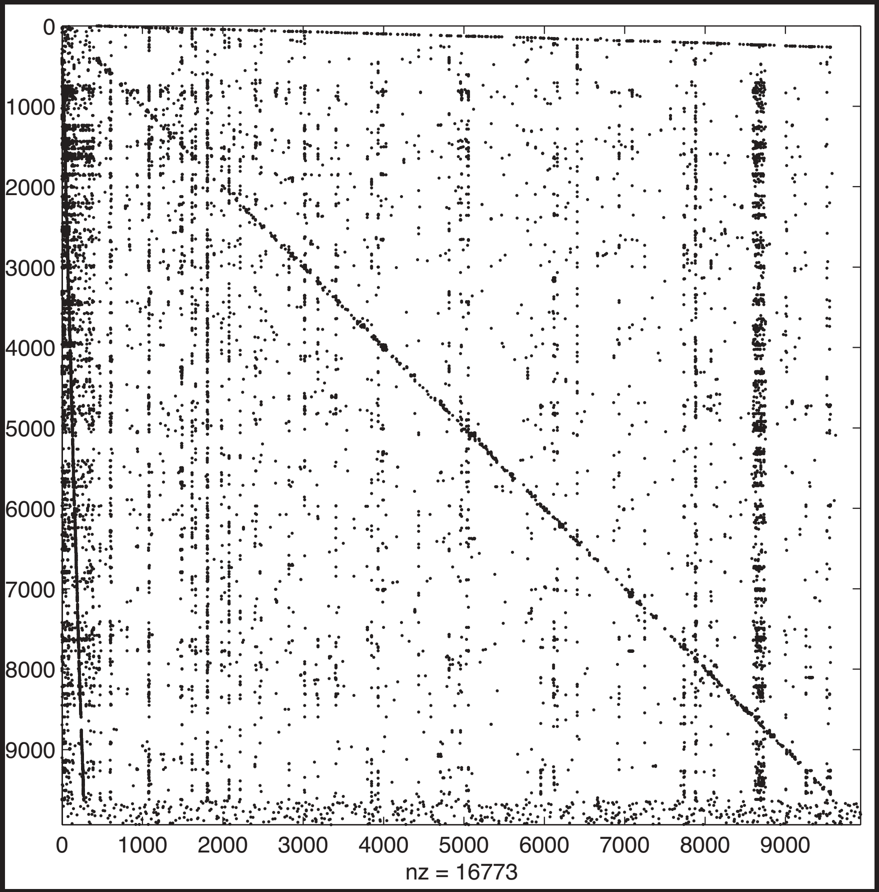
Pro: one small system solve, plus forward substitution.

Analog: Lee et al. lumpable D node Markov formulation.

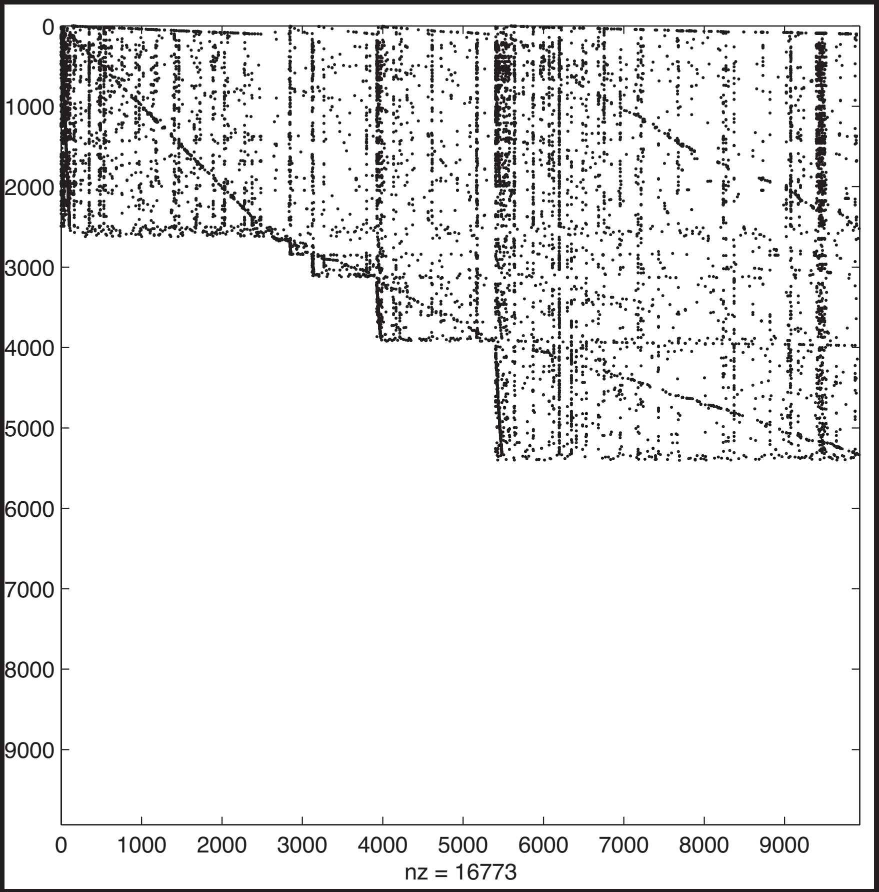
Extension of ND-D Reordering

- Continue locating 0 rows in submatrices of $(\mathbf{I} - \alpha\mathbf{P})$ until no 0 rows remain. Amounts to a reordering of indices.

Before Reordering



After Reordering



Algorithm 2: Recursive ND-D Reordering

Solve $\boldsymbol{\pi}^T(\mathbf{I} - \alpha\mathbf{P}) = \mathbf{v}^T$ and $\boldsymbol{\pi}^T\mathbf{e} = 1$.

Algorithm 2:

1. Reorder the states of the original Markov chain, so that the reordered matrix has the 0 block structure. $O(nnz(\mathbf{P})) \approx 1$ power iter.
2. Solve for $\boldsymbol{\pi}_1^T$ in $\boldsymbol{\pi}_1^T(\mathbf{I} - \alpha\mathbf{P}_{11}) = \mathbf{v}_1^T$. Jacobi method with rate of conv. $\leq \alpha$
3. Compute $\boldsymbol{\pi}_2^T = \alpha\boldsymbol{\pi}_1^T\mathbf{P}_{12} + \mathbf{v}_2^T$.
4. Compute $\boldsymbol{\pi}_3^T = \alpha\boldsymbol{\pi}_1^T\mathbf{P}_{13} + \alpha\boldsymbol{\pi}_2^T\mathbf{P}_{23} + \mathbf{v}_3^T$.
5. Compute $\boldsymbol{\pi}_b^T = \alpha\boldsymbol{\pi}_1^T\mathbf{P}_{1b} + \alpha\boldsymbol{\pi}_2^T\mathbf{P}_{2b} + \cdots + \alpha\boldsymbol{\pi}_{b-1}^T\mathbf{P}_{b-1,b} + \mathbf{v}_b^T$. $O(nnz(\mathbf{P}))$
6. Normalize $\boldsymbol{\pi}^T = [\boldsymbol{\pi}_1^T \ \ \boldsymbol{\pi}_2^T \ \ \cdots \ \ \boldsymbol{\pi}_b^T] / \|[\boldsymbol{\pi}_1^T \ \ \boldsymbol{\pi}_2^T \ \ \cdots \ \ \boldsymbol{\pi}_b^T]\|_1$.

Pro: even smaller system solve, plus forward substitution.

Speedup: by factor of $nnz(\mathbf{P})/nnz(\mathbf{P}_{11})$ (estimated)

Results of Reordered PageRank

		EPA.dat	CA.dat	NCS.dat	ND.dat	SU450k.dat
PR	$Time$	3.80	6.63	13.17	177.16	237.37
	$Iter.$	159	176	162	166	164
	$n(\mathbf{P})$	5,042	9,664	10,000	325,729	451,237
	$nz(\mathbf{P})$	9,563	16,873	101,118	1,497,134	1,082,604
$RePR$	$Time$	.59	1.42	7.65	130.54	52.84
	$Iter.$	155	169	160	170	145
	b	10	9	5	18	12
	$n(\mathbf{P}_{11})$	704	2,622	7,136	127,472	84,861
	$nz(\mathbf{P}_{11})$	1,330	5,238	79,230	1,191,761	267,566
$Speed$	$Est.$	7.2	3.2	1.3	1.3	4.0
Up	$Act.$	6.4	4.7	1.7	1.4	4.5

- can do no worse than original PR power method
- Speedup is dataset-dependent

Langville/Meyer Updating

Motivation

- Updating PR is huge problem. Currently done monthly, but web changes hourly.
- Chien et al. use aggregation to focus on pages whose PR is most likely to change.

Idea

- Use iterative aggregation to extend Chien idea.
- Focus on bad states, aggregate good, fast-converging states into one superstate.
- $\Rightarrow$ only work on much smaller aggregated chain.

Results

- speedup by factor of 5-10 on some datasets.

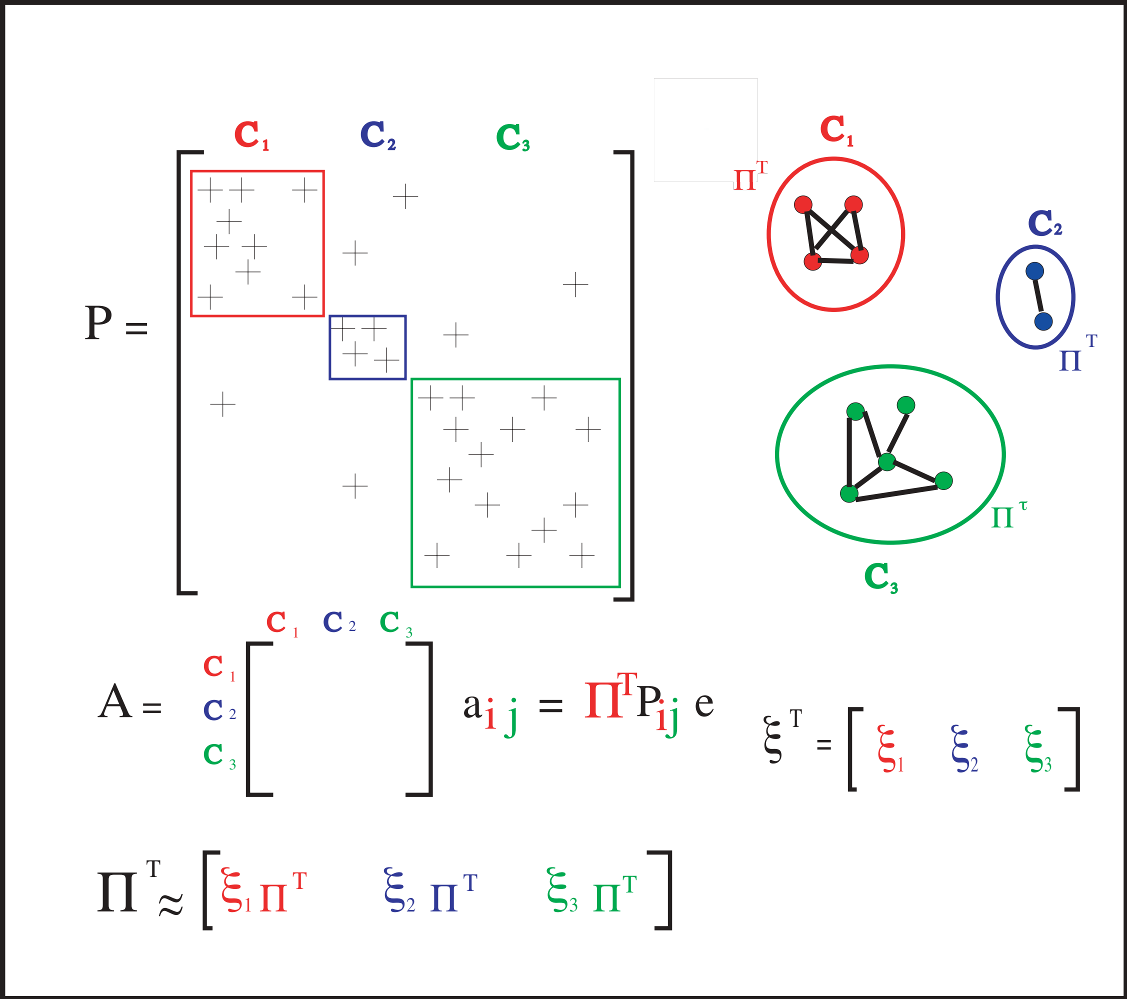
Issue

- Partitioning into good and bad states is hard, and IAD is very sensitive to partition.

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 $\boldsymbol{\pi}^T \; \boldsymbol{\pi}^T \; \boldsymbol{\pi}^T$

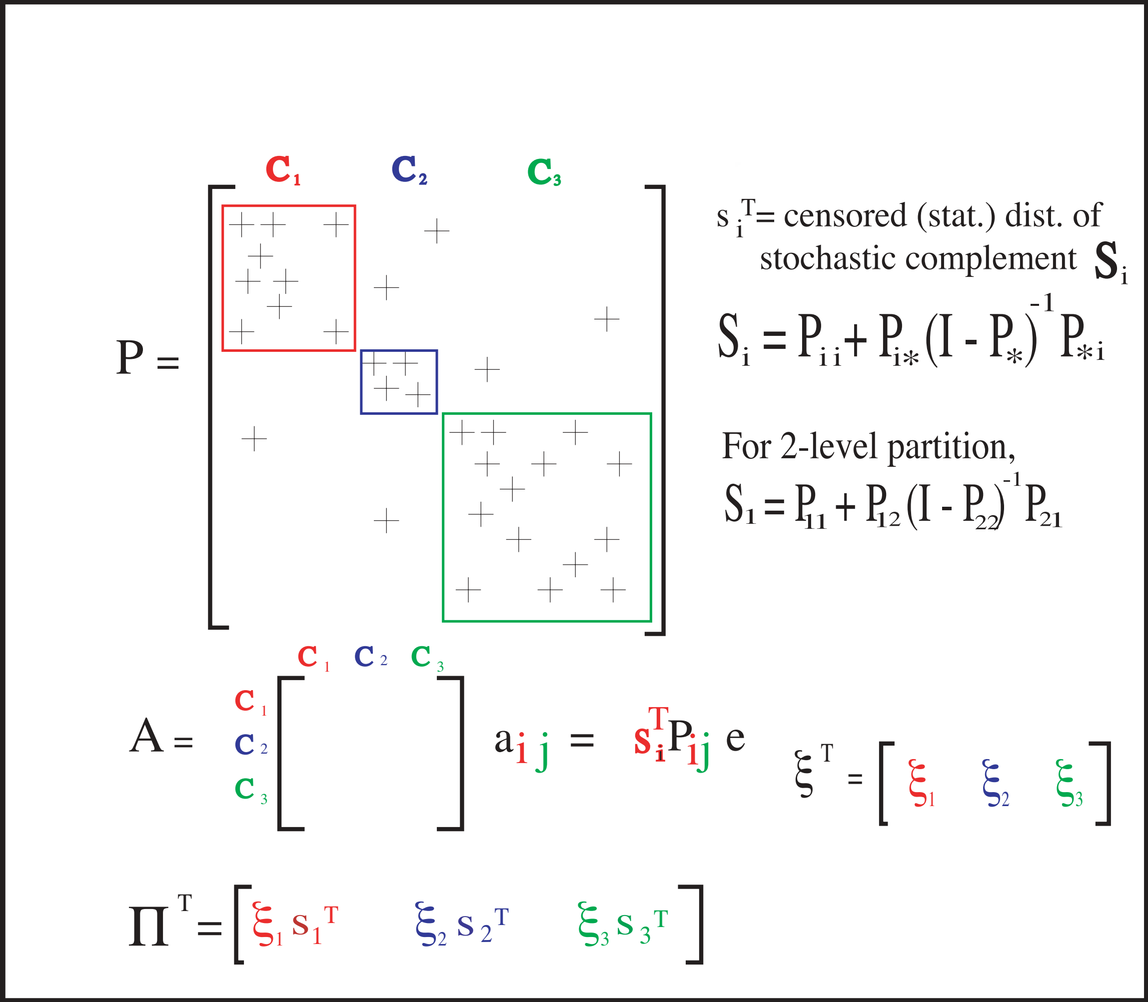
get coupling constants $\boldsymbol{\xi}_i$

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)



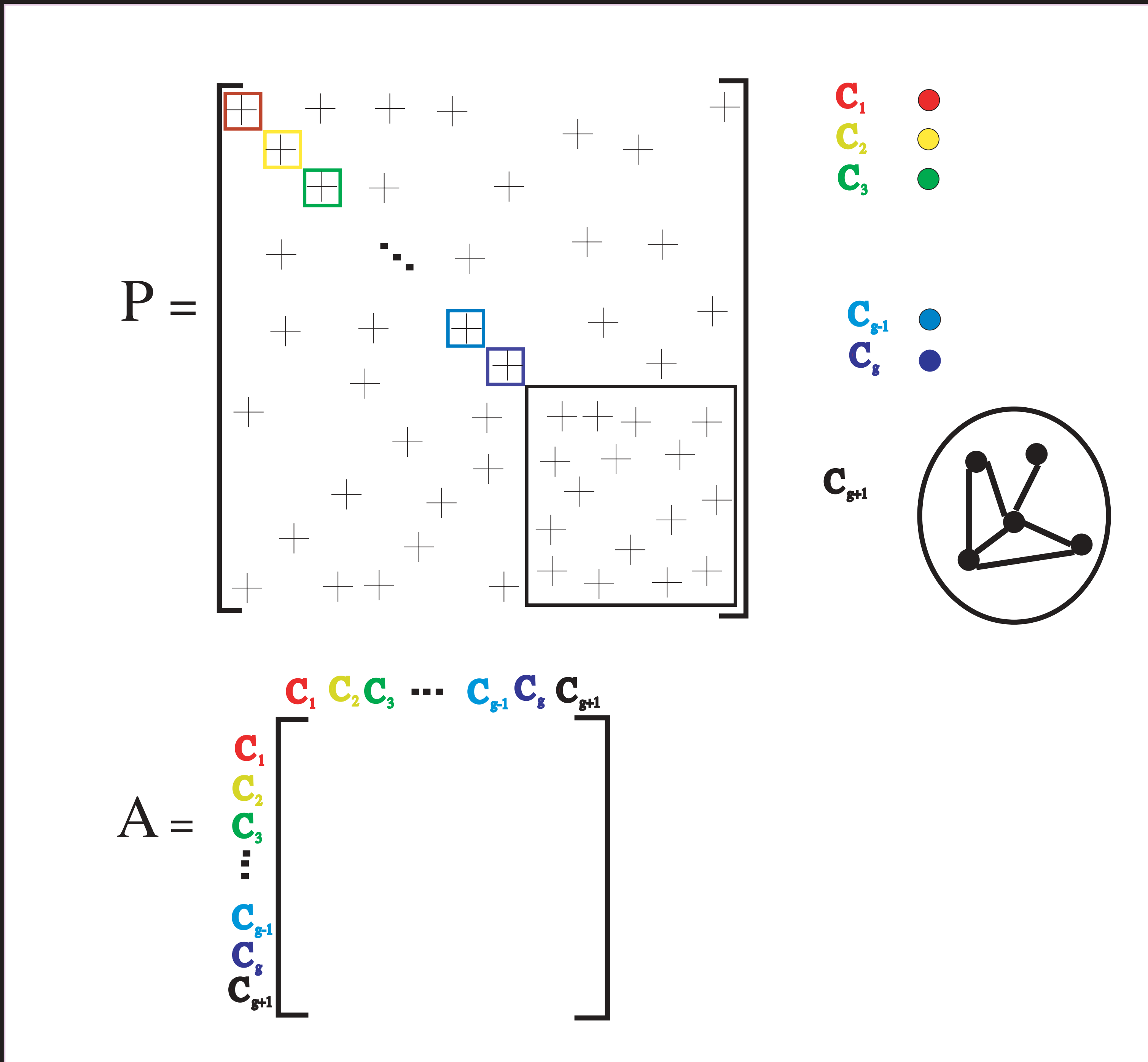
Pro

Con

only one step needed to produce exact global vector

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

Back to Updating . . .



Aggregation

Partitioned Matrix

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

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

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

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

The Aggregation/Disaggregation Theorem

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

Then $\boldsymbol{\pi}^T = \left(\alpha_1, \ldots, \alpha_g \,\middle|\, \alpha_{g+1}\mathbf{s}_2^T\right)$ = 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 $\boldsymbol{\pi}_i$ are slow-converging. $\Rightarrow$ inexpensive soln = use old $\boldsymbol{\pi}^T$ to determine G. (adaptively approximate $\mathbf{y}_2^T$)

Implications of Web's scale-free nature

Facts:

(1) π^T follows power law since WWW is scale-free

(experimental and theoretical justification)

(2) not all pages converge to their PageRanks at same rate

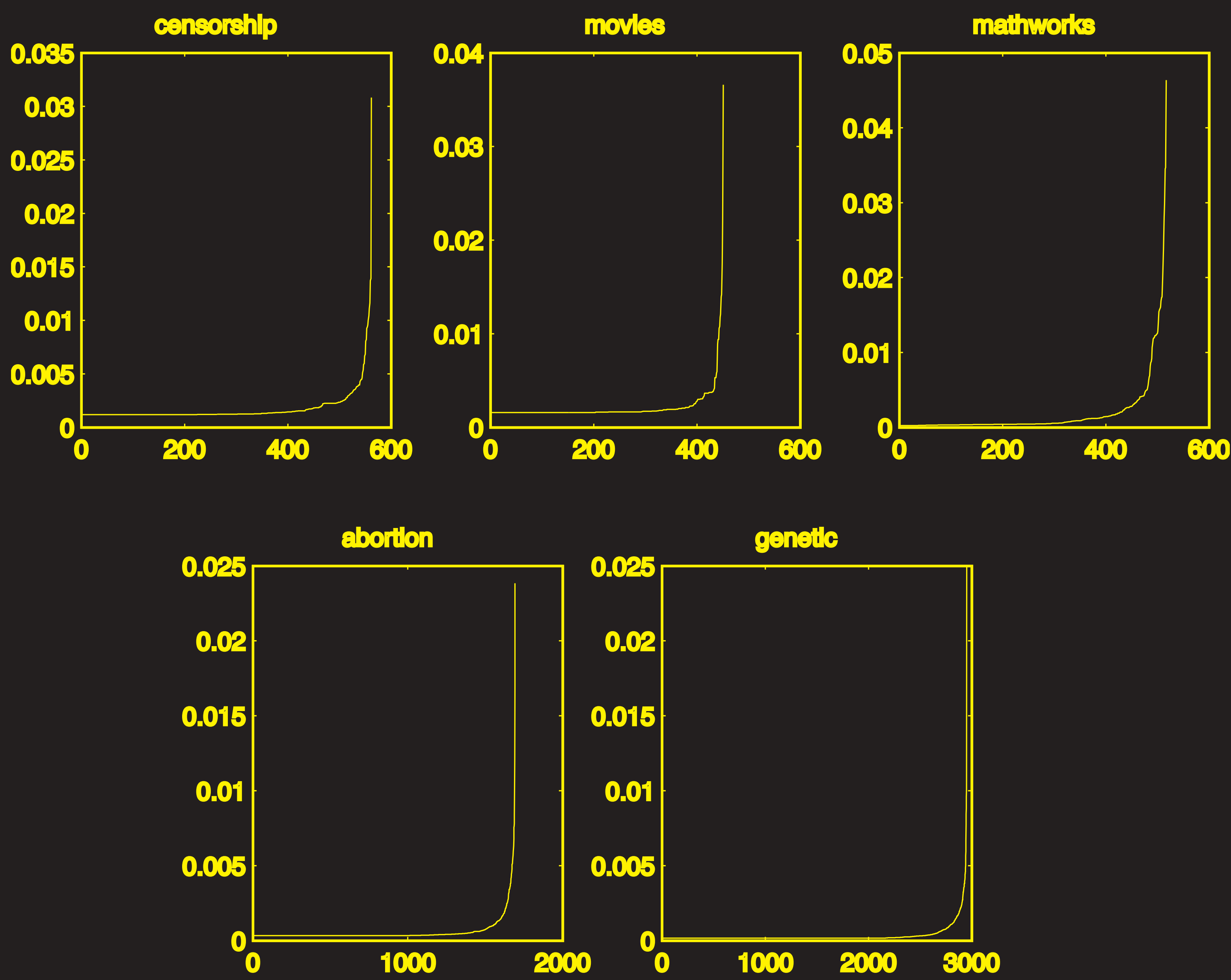
(3) pages with high PR are slow-converging

$\Rightarrow$ very few pages are slow-converging, but these are the pages that cause power method to drag on

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 $\lambda_2(\mathbf{S}_2)$

Ipsen/Kirkland Updating Theory

Motivation

- L/M prove updating method converges at rate $(\lambda_2(\mathbf{S_2}))^k \to \mathbf{0}$.
- Ipsen/Kirkand wonder: can $\lambda_2(\mathbf{S_2}) > \alpha$?

Results

- $\lambda_2(\mathbf{S_2}) \leq \alpha$ for all partitions.
- $\lambda_2(\mathbf{S_2}) < \alpha$ under two trivial assumptions on $\mathbf{P}$.

  (**P** is reducible, and at least one page in each essential class does not self-link)

Ipsen/Kirkland Updating Theory

Motivation

- L/M prove updating method converges at rate $(\lambda_2(\mathbf{S_2}))^k \rightarrow \mathbf{0}$.
- Ipsen/Kirkand wonder: can $\lambda_2(\mathbf{S_2}) > \alpha$?

Results

- $\lambda_2(\mathbf{S_2}) \leq \alpha$ for all partitions.
- $\lambda_2(\mathbf{S_2}) < \alpha$ under two trivial assumptions on **P**.

  (**P** is reducible, and at least one page in each essential class does not self-link)

But ... how do we find partition so that $\lambda_2(\mathbf{S_2}) << \alpha$?

Experiments

Test Networks From Crawl Of Web

NCState (NCSU internal crawl)

10,000 nodes 101,118 links

California (Sites concerning “california” query)

9,664 nodes 16,150 links

Parameters

Number Of Nodes (States) Added

$\mathbf{50}$

Number Of Nodes (States) Removed

$\mathbf{30}$

Number Of Links Added

(Different values have little effect on results)

$\mathbf{300}$

Number Of Links Removed

$\mathbf{200}$

Stopping Criterion

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

NC State

Power Method

Iterations	Time
$\mathbf{162}$	$\mathbf{9.79}$

Iterative Aggregation

$\|G\|$	Iterations	Time
$\mathbf{500}$	$\mathbf{160}$	$\mathbf{10.18}$
$\mathbf{1000}$	$\mathbf{51}$	$\mathbf{3.92}$
$\mathbf{1500}$	$\mathbf{33}$	$\mathbf{2.82}$
$\mathbf{2500}$	$\mathbf{16}$	$\mathbf{2.15}$
$\mathbf{3000}$	$\mathbf{13}$	$\mathbf{1.99}$
$\mathbf{5000}$	$\mathbf{7}$	$\mathbf{1.77}$

$nodes = \mathbf{10,000} \quad links = \mathbf{101,118}$

NC State

Power Method

Iterations	Time
162	9.79

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
500	160	10.18
1000	51	3.92
1500	33	2.82
2000	21	2.22
2500	16	2.15
3000	13	1.99
5000	7	1.77

$nodes = 10,000 \quad links = 101,118$

California

Power Method

Iterations	Time
$\mathbf{176}$	$\mathbf{5.85}$

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
$\mathbf{500}$	$\mathbf{19}$	$\mathbf{1.12}$
$\mathbf{1000}$	$\mathbf{15}$	$\mathbf{.92}$
$\mathbf{1250}$	$\mathbf{20}$	$\mathbf{1.04}$
$\mathbf{2000}$	$\mathbf{13}$	$\mathbf{1.17}$
$\mathbf{5000}$	$\mathbf{6}$	$\mathbf{1.25}$

$$nodes = \mathbf{9,664} \quad links = \mathbf{16,150}$$

California

Power Method

Iterations	Time
176	5.85

Iterative Aggregation

$\|G\|$	Iterations	Time
500	19	1.12
1000	15	.92
1250	20	1.04
1500	14	.90
2000	13	1.17
5000	6	1.25

$nodes = 9,664 \quad links = 16,150$

Advantage

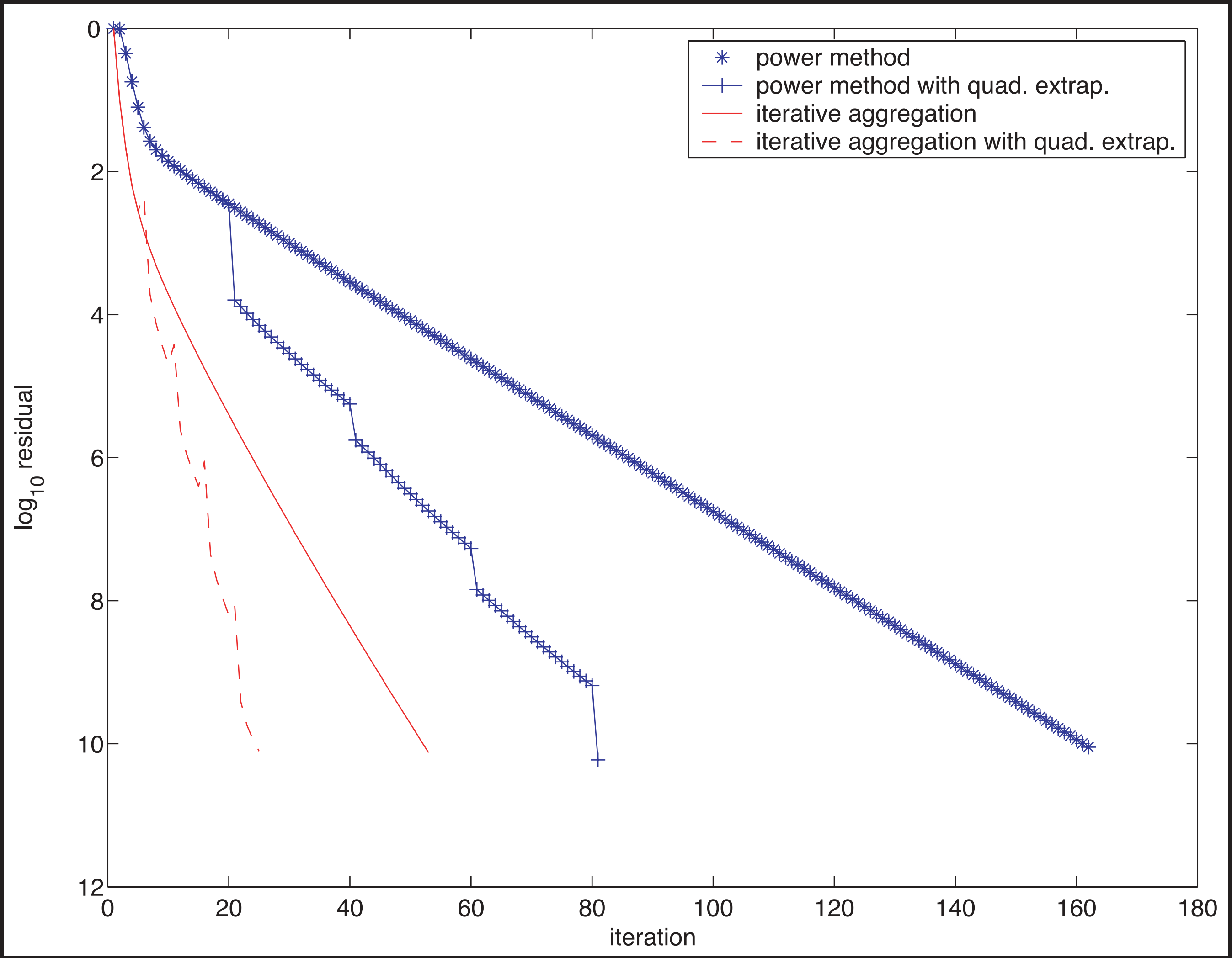
— updating algorithm can be combined with other PR acceleration methods.

Power		Power+Quad(10)	
Iter.	Time	Iter.	Time
162	9.69	81	5.93

	Iter. Agg.		Iter.Agg.+Quad(10)	
$\|G\|$	Iter.	Time	Iter.	Time
500	160	10.18	57	5.25
1000	51	3.92	31	2.87
1500	33	2.82	23	2.38
2000	21	2.22	16	1.85
2500	16	2.15	12	1.88
3000	13	1.99	11	1.91
5000	7	1.77	6	1.86

$nodes = 10,000$ $links = 101,118$

Residual Plot for NC State



Large-Scale Implementation

Partitioning

— need more theoretical work on good partitioning.

IAD's Aggregated System Solve

— direct vs. sparse methods

Simulating updates to Web

— how to do this accurately, and keep scale-free properties of web

— need collections of the web over time.

Conclusions

- An appropriate reordering of the pages of the web can greatly speed the PageRank computation.
- Aggregation methods reduce PageRank computation for the updating problem. However, partitioning is a difficult, unresolved issue.
- many of these methods can be combined to achieve even greater speedups.
- We are moving closer to lofty goal of computing real-time personalized PageRank.