Updating the Stationary Vector

of an

Irreducible Markov Chain

with an eye on

Google's PageRank

Amy Langville

Carl Meyer

Department of Mathematics
North Carolina State University
Raleigh, NC

SIAM CSE 2/13/05

Outline

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

PageRank

The Hyperlink Matrix $\mathbf{H}$

$$\mathbf{H}_{ij} = 1/|O_i|$$

The Stochastic Matrix $\mathbf{S}$

$$\mathbf{S} = \mathbf{H} + \mathbf{a}\mathbf{v}^T$$

(a_i=1 for $i \in D$, 0, o.w.)

The Google Matrix $\mathbf{G}$

$$\begin{aligned}\mathbf{G} &= \alpha\mathbf{S} + (1-\alpha)\mathbf{e}\mathbf{v}^T \\ &= \alpha\mathbf{H} + (\alpha\mathbf{a} + (1-\alpha)\mathbf{e})\mathbf{v}^T\end{aligned}$$

- $\mathbf{G}$ is irreducible, aperiodic Markov chain.
- Stationary vector of $\mathbf{G}$ is PageRank vector $\boldsymbol{\pi}^T$.

  π_i is long-run proportion of time that random surfer spends on page i.

Computing π^T

A Big Problem

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

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

Computing π^T

A Big Problem

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

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

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

Power Method to compute PageRank

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

until convergence, do

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

end

Power Method to compute PageRank

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

until convergence, do

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

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

end

Power Method to compute PageRank

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

until convergence, do

- X $\boldsymbol{\pi}_{j+1}^T = \boldsymbol{\pi}_j^T \, \mathbf{G}$ (dense computation)
- X $\boldsymbol{\pi}_{j+1}^T = \alpha \, \boldsymbol{\pi}_j^T \, \mathbf{S} + (1-\alpha) \, \boldsymbol{\pi}_j^T \, \mathbf{e} \, \mathbf{v}^T$ (sparser computation)
- $\boldsymbol{\pi}_{j+1}^T = \alpha \, \boldsymbol{\pi}_j^T \, \mathbf{H} + (\alpha \, \boldsymbol{\pi}_j^T \, \mathbf{a} + (1-\alpha)) \, \mathbf{v}^T$ (even less computation)

end

- **H** 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 $\lambda_2(\mathbf{G}) \leq \alpha$

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

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

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

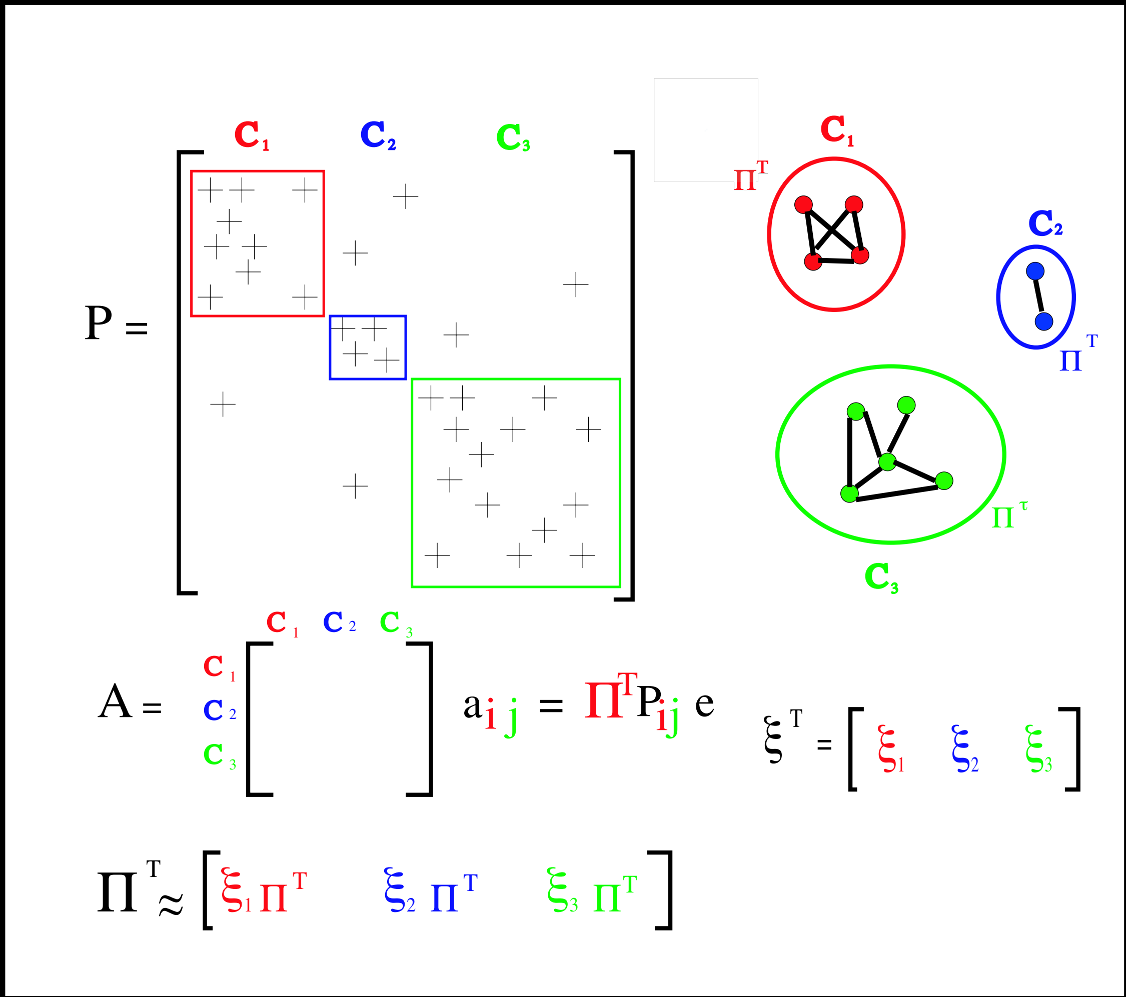
- speedup by factor of 5-10 on some datasets.

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

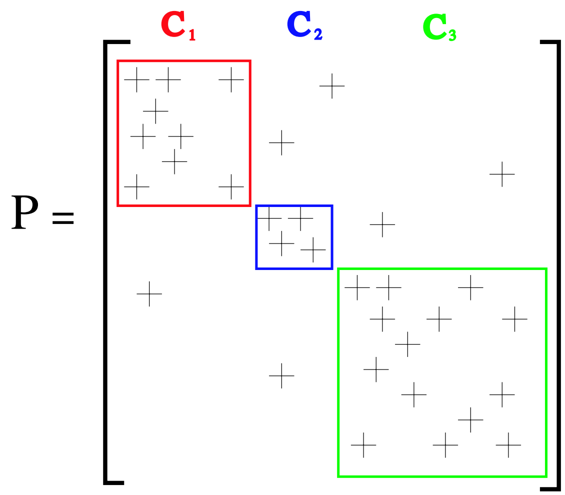
get coupling constants $\boldsymbol{\xi}_i$

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)



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,

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

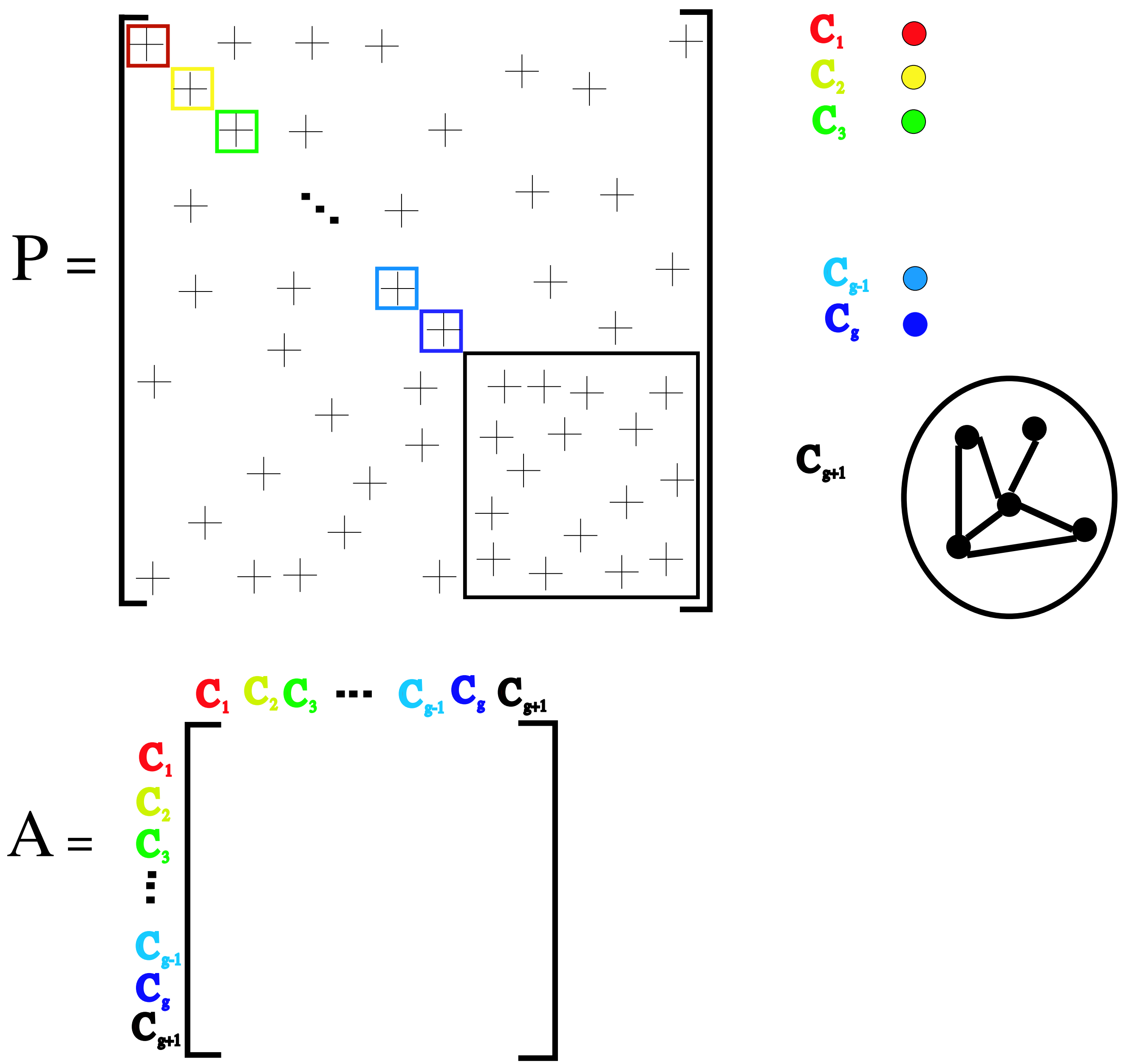
$$A = \begin{array}{cc} & \begin{array}{ccc} C_1 & C_2 & C_3 \end{array} \\ \begin{array}{c} C_1 \\ C_2 \\ C_3 \end{array} & \begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix} \end{array} \qquad a_{ij} = s_i^T P_{ij}\, e \qquad \xi^T = \begin{bmatrix} \xi_1 & \xi_2 & \xi_3 \end{bmatrix}$$

$$\Pi^T = \begin{bmatrix} \xi_1 s_1^T & \xi_2 s_2^T & \xi_3 s_3^T \end{bmatrix}$$

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

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

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

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

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

The Aggregation/Disaggregation Theorem

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

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

$\widetilde{\widetilde{\pi}}^T = \widetilde{\pi}^T \mathbf{P}$ (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 $\widetilde{\pi}^T$

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

Use $\widetilde{\widetilde{\pi}}^T$ 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 $\lambda_2^k \to 0$

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 $\mathbf{y}_2^T$)

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

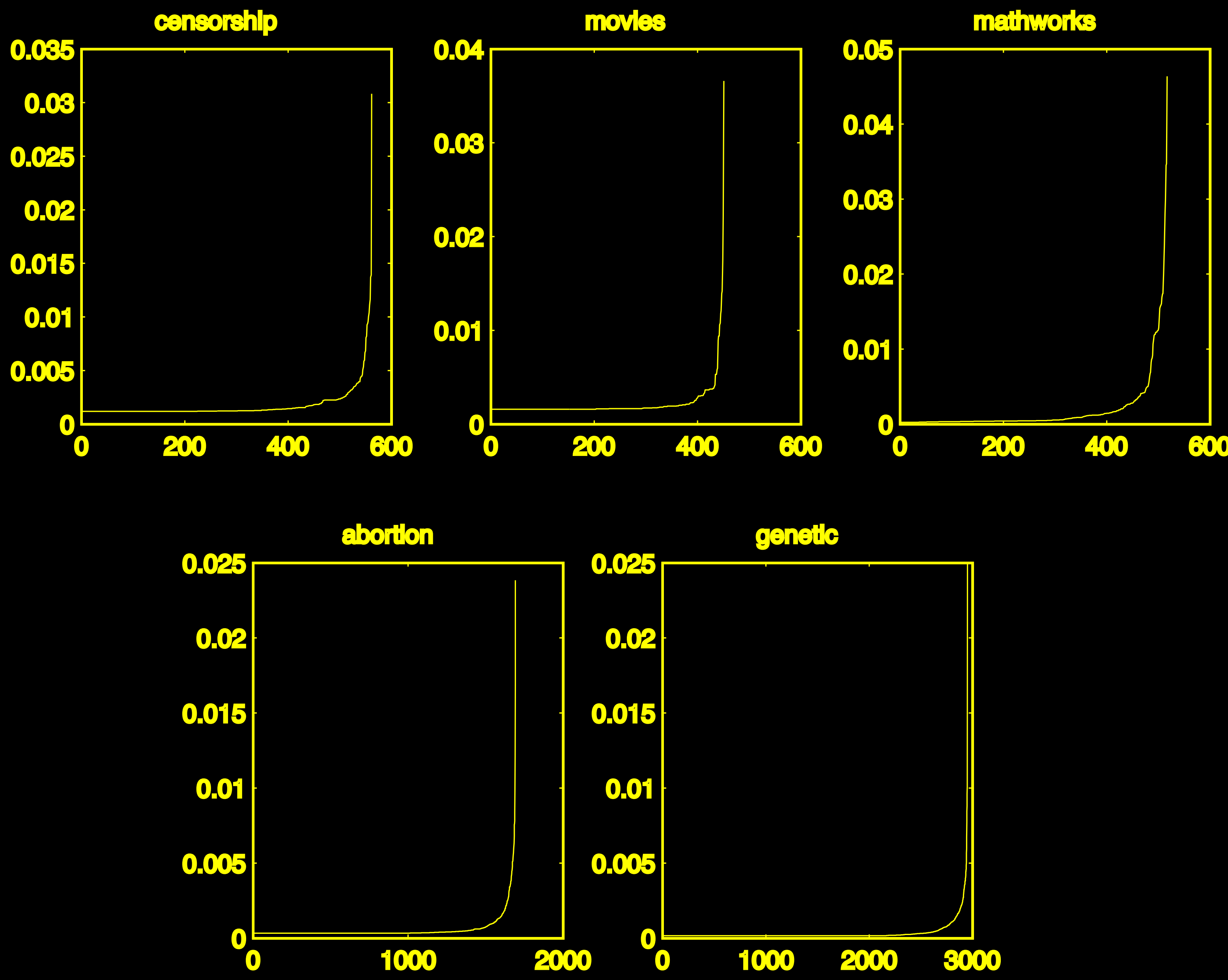
(3) pages with high PR are slow-converging

$\Rightarrow$ 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 $\Longrightarrow \boldsymbol{\pi}_k^T \to \boldsymbol{\pi}^T$ at the rate at which $\lambda_2^n \to 0$

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 **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

50

Number Of Nodes (States) Removed

30

Number Of Links Added

(Different values have little effect on results)

300

Number Of Links Removed

200

Stopping Criterion

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

NC State

Power Method

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

Iterative Aggregation

$\lvert G \rvert$	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

$\|G\|$	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
$\mathbf{176}$	$\mathbf{5.85}$

Iterative Aggregation

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

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

Advantage

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

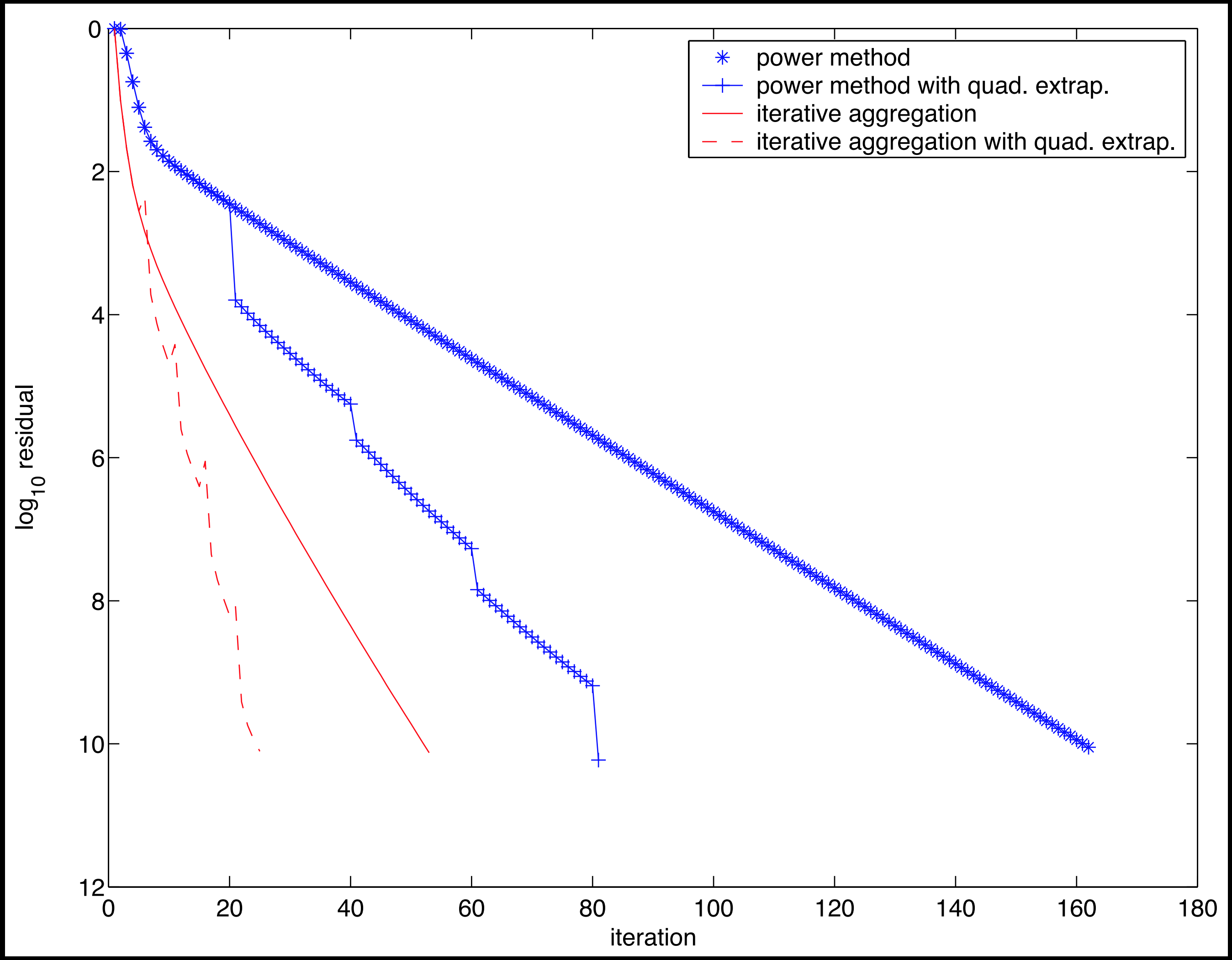
Power | **Power+Quad(10)** | **Iter. Agg.** | **Iter.Agg.+Quad(10)**

Iter.	Time	Iter.	Time
162	9.69	81	5.93

$\lvert G\rvert$	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

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

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

- Aggregation methods reduce PageRank computation for the updating problem. However, partitioning is a difficult, unresolved issue.
- Many of the acceleration methods can be combined to achieve even greater speedups.
- We are moving closer to lofty goal of computing real-time personalized PageRank.