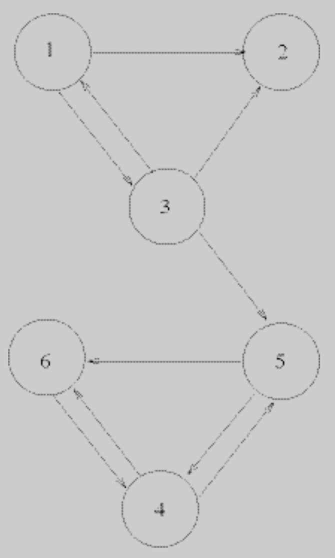
Updating PageRank by Iterative Aggregation

Amy N. Langville and Carl D. Meyer
Mathematics Department
North Carolina State University
{anlangvi,meyer}@ncsu.edu

The PageRank Problem

Solve $\pi^T = \pi^T \mathbf{P}$

π_i = importance of page i



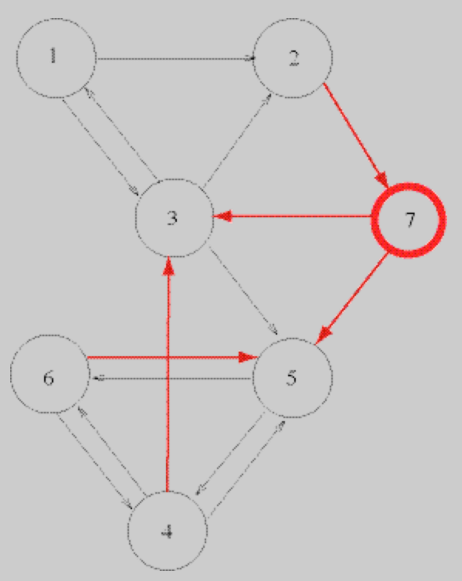
$$\mathbf{P} = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Solution: Power Method

$$\pi^{(k+1)T} = \pi^{(k)T} \mathbf{P}$$

The Updating Problem

Given π^T, $\mathbf{P}$, $\tilde{\mathbf{P}}$, find $\tilde{\pi}^T$

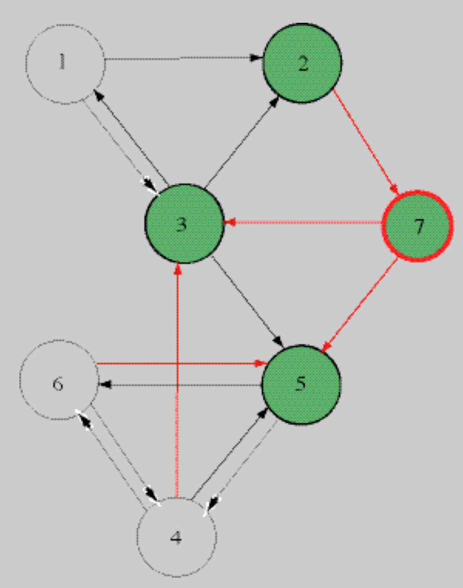


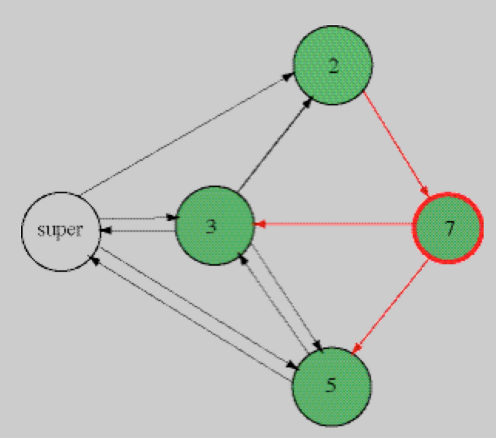
$$\tilde{\mathbf{P}} = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 & 0 & 0 \end{bmatrix}$$

Naïve Solution: full recomputation, power method on $\tilde{\mathbf{P}}$ on monthly basis

The Iterative Aggregation Solution to Updating

- Partition Nodes into two sets, G and $\bar{\mathrm{G}}$





- Aggregate: lump nodes in $\bar{\mathrm{G}}$ into one supernode
- Solve small |G+1| × |G+1| chain called **A**
- Disaggregate to get approximation to full-sized PageRank
- Iterate

$$\mathbf{A} = \begin{bmatrix} 0 & 1/4 & 1/4 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1/3 & 1/3 & 0 & 1/3 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \end{bmatrix}$$

The Iterative Aggregation Algorithm

Initialization

Partition updated chain $\mathcal{S} = G \cup \overline{G}$

$\omega^T \longleftarrow$ components from ϕ^T corresponding to states $\overline{G}$

$\mathbf{s}^T \longleftarrow \omega^T/(\omega^T\mathbf{e})$ (Approx. censored distribution for S_2)

Iterate Until Convergence

$\mathbf{A} \longleftarrow \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12}\mathbf{e} \\ \mathbf{s}^T\mathbf{P}_{21} & 1 - \mathbf{s}^T\mathbf{P}_{21}\mathbf{e} \end{bmatrix}$ (Aggregate)

$\alpha^T \longleftarrow (\alpha_1, \ldots, \alpha_g, \alpha_{g+1})$ (Stationary distribution for A)

$\tilde{\pi}^T \longleftarrow (\alpha_1, \ldots, \alpha_g \,|\, \alpha_{g+1}\mathbf{s}^T)$ (Disaggregate)

$\pi^T \longleftarrow \tilde{\pi}^T\mathbf{P} = (\pi_1^T \,|\, \pi_2^T)$ (Smooth)

If $\|\pi^T - \tilde{\pi}^T\| < tol$, then quit

else $\mathbf{s}^T \longleftarrow \pi_2^T/\pi_2^T\mathbf{e}$ and repeat

Convergence of the Iterative Aggregation Algorithm

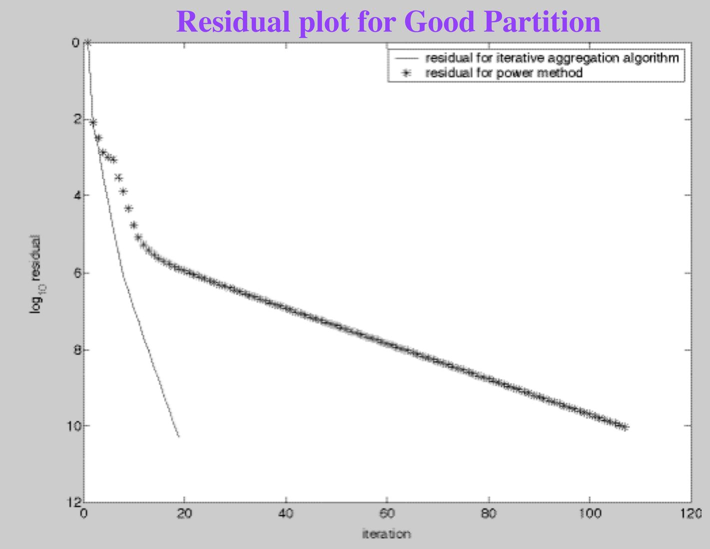
- Iterative aggregation converges to PageRank vector for all partitions S = G ∪ $\bar{\mathrm{G}}$.
- There always exists a partition such that the asymptotic rate of convergence is strictly less than the convergence rate of PageRank power method.

Performance of the Iterative Aggregation Algorithm

NCState.dat
10,000 pages
101,118 links

PageRank Power		Iterative Aggregation		
Iterations	Time	| G |	Iterations	Time
162	9.79	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

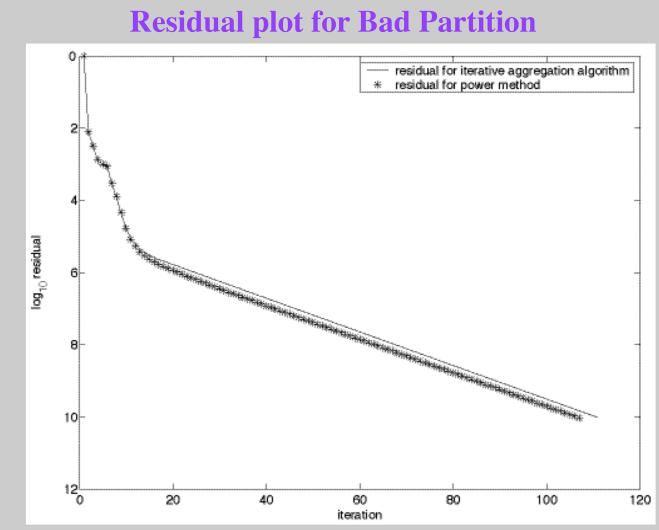
Residual plot for Good Partition



Calif.dat
9,664 pages
16,150 links

PageRank Power		Iterative Aggregation		
Iterations	Time	| G |	Iterations	Time
176	5.85	500	19	1.12
		1000	15	.92
		1250	20	1.04
		1500	**14**	**.90**
		2000	13	1.17
		5000	6	1.25

Residual plot for Bad Partition



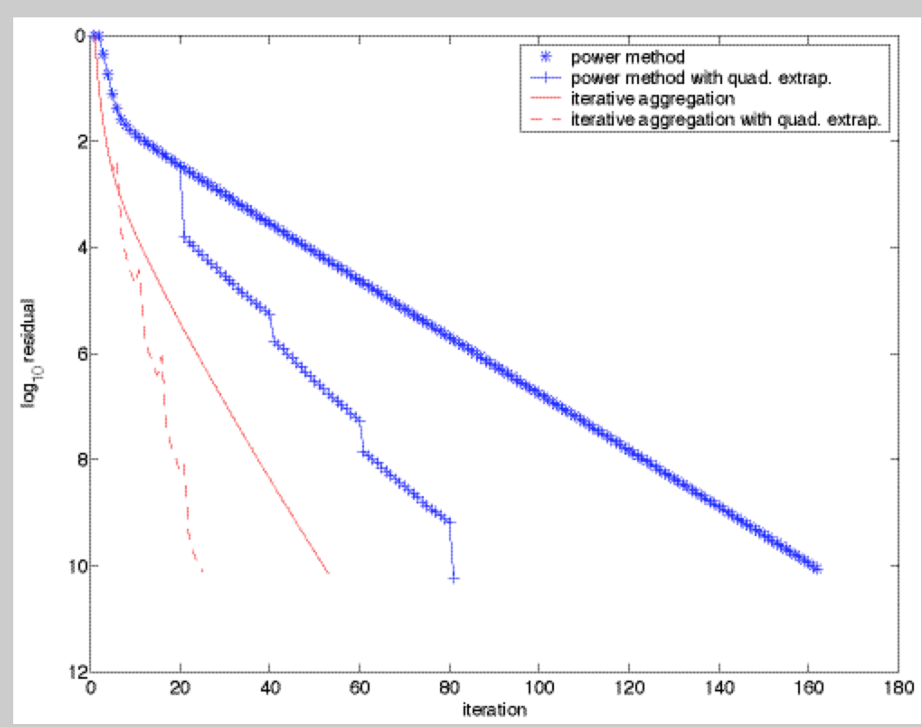
Advantage

- This iterative aggregation algorithm can be combined with other PageRank acceleration techniques to achieve even greater speedups.

PageRank Power		Power + Quad(10)		Iterative Aggregation			Iter. Aggregation + Quad(10)	
Iterations	Time	Iterations	Time	| G |	Iterations	Time	Iterations	Time
162	9.79	81	5.93	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

Residual Plot of 4 solution methods applied to NCState.dat

using Quad(10), |G|=1000



Problems

- Algorithm is very sensitive to **partition**. Much more theoretical work must be done to determine which nodes go into G.
- We need **faster machines** with **more memory** to test on larger datasets, >500K pages. Testing requires storage of more vectors and matrices, such as stochastic complements and censored vectors.
- We need **actual datasets** that **vary over time**. Currently, we are creating artificial updates to datasets.