Power-law Distribution and Graph Theoretic Methods On the Web

Yi-Wen Sun

Content

- *Graph Structure in the Web*, by A. Border, R. Kumar, et al., Computer Networks 33: 309-320, 2000 [BK00]
- *The Diameter of the World Wide Web*. R. Albert, H. Jeong, A. Barabasi, Nature 401, 130-131, 1999 [AJB99]
- *A Stochastic Model for the Evolution of the Web*, by M. Levene, T. Fenner, G. Loizou, R. Wheeldon, Computer Networks, 39:277--287, 2002. [LFLW02]

Web as a Graph

- Web <-> Directed Graph
- static pages <-> nodes
- links <-> arcs
- in-degree, out-degree, distance
- strongly connected component (SCC), weak component (WC)
- diameter, average distance

Study of Web Graph

- Observations of the power law distributions on the web
- Applying graph theoretic methods to the web
- Purpose:
  - Design crawl strategies
  - Analyze the behavior of web algorithms
  - Predict the evolution of web structure
  - Predict the emergence of new phenomena

Infrastructure [BK00]

- Connectivity Server 2 Software
- AltaVista Crawl
- Database
  - 203 million URLs and 1466 million links, May, 1999
  - 271 million URLs and 2130 million links, October, 1999

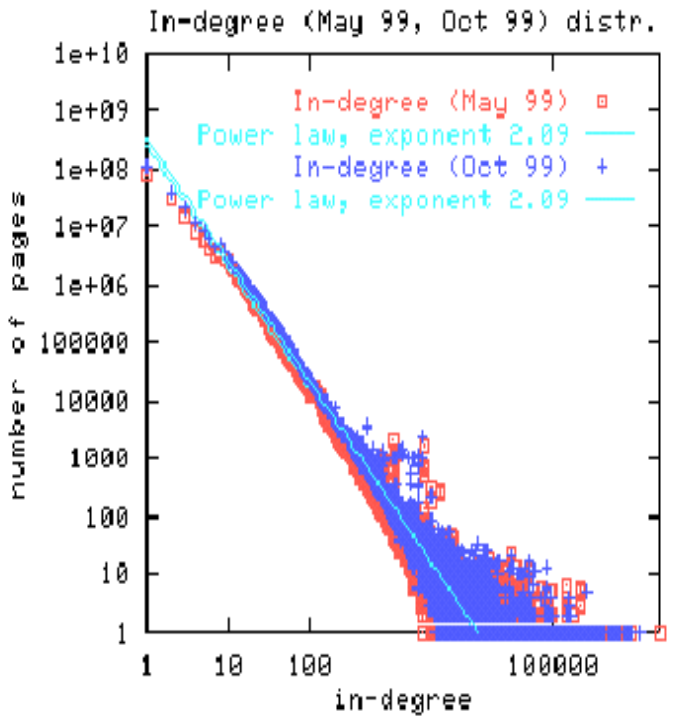
Degree Distribution

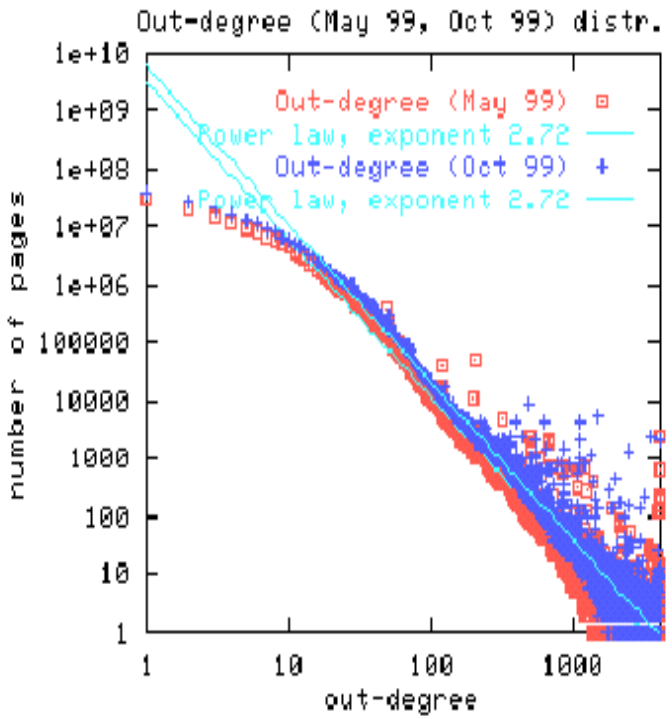
- **The power law for in-degree**

  the probability that a node has in-degree *i* is proportional to $1/i^x$, for some $x > 1$.

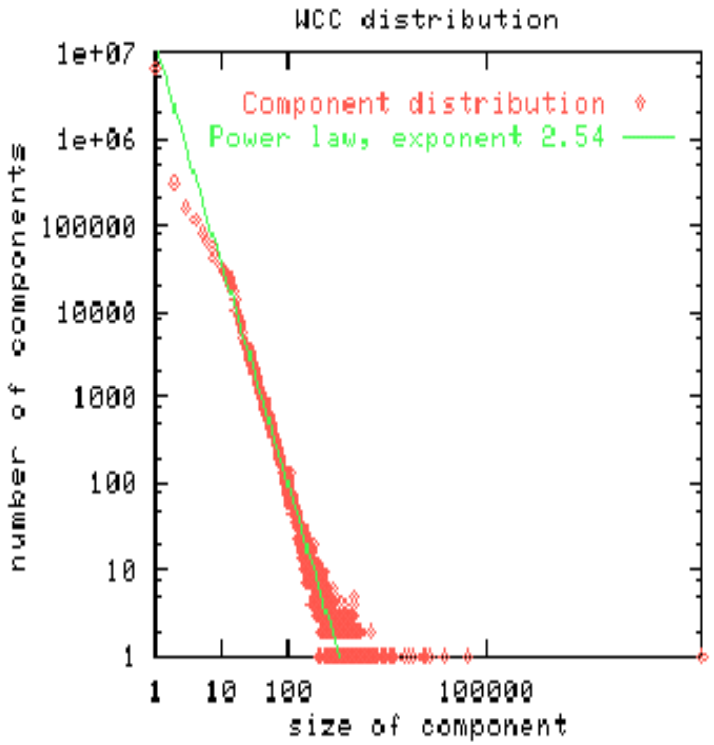
$$P_{in}(i) \sim i^{-x}$$

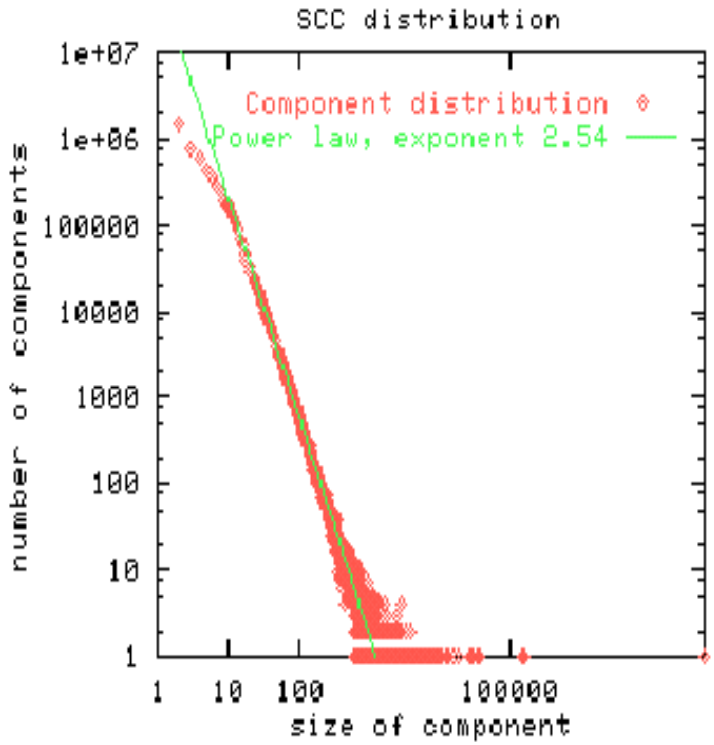
Degree Distribution (cont.)





Connected Components

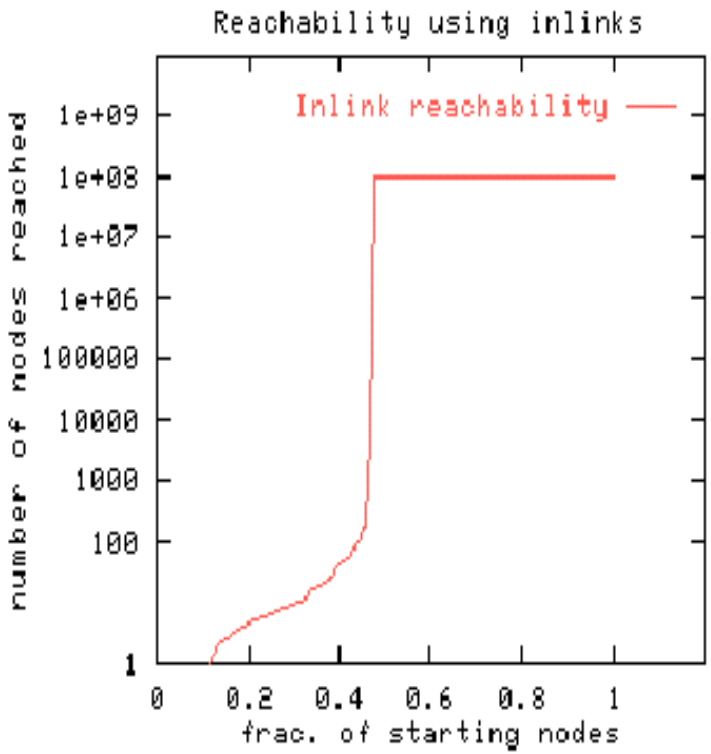


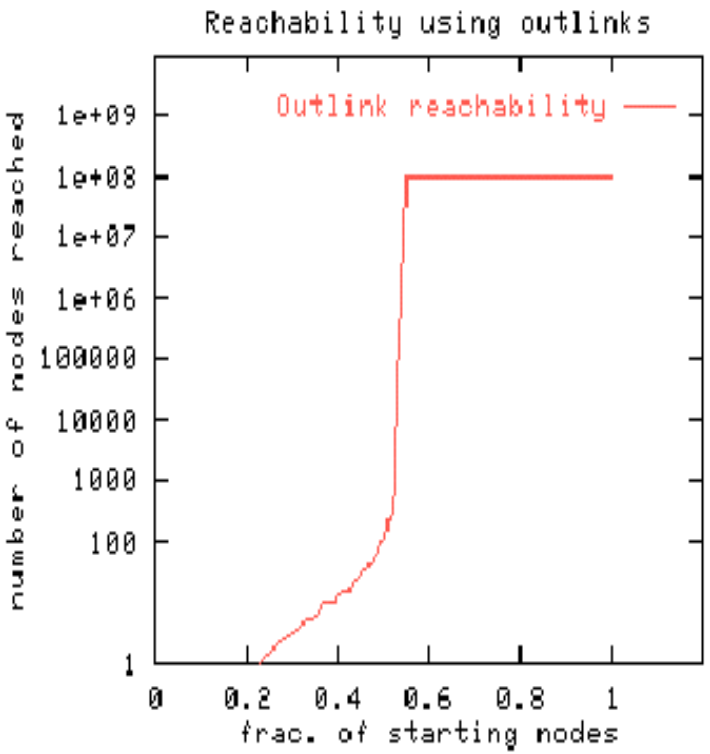


Connected Components (cont.)

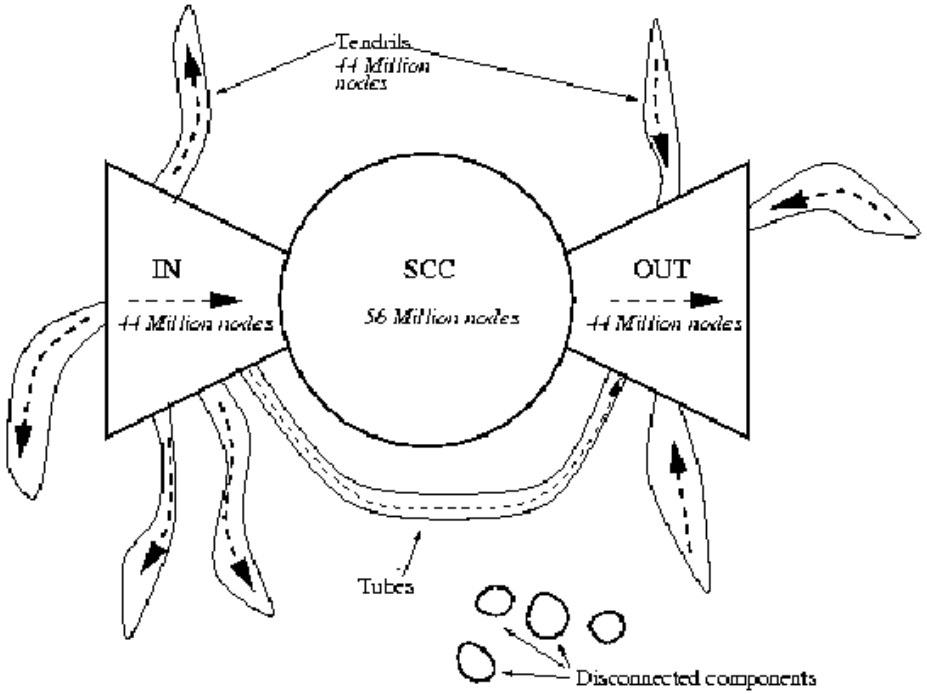
- the connectivity is extremely resilient and does not depend on the existence of nodes of high in-degree.
- such nodes, with high PageRank or considered good hubs, are embedded in a graph that is well connected without them.

Random-start BFS





Connectivity of the Web



Region	SCC	IN	OUT	TENDRILS	DISC.	Total
Size	56,463,993	43,343,168	43,166,185	43,797,944	16,777,756	203,549,046

Diameter

Measure	Minimum depth	Average depth	Maximum depth
In-links	475	482	503
Out-links	430	434	444

- Directed diameter of SCC is at least 28
- Diameter of the graph as a whole is over 500

Path

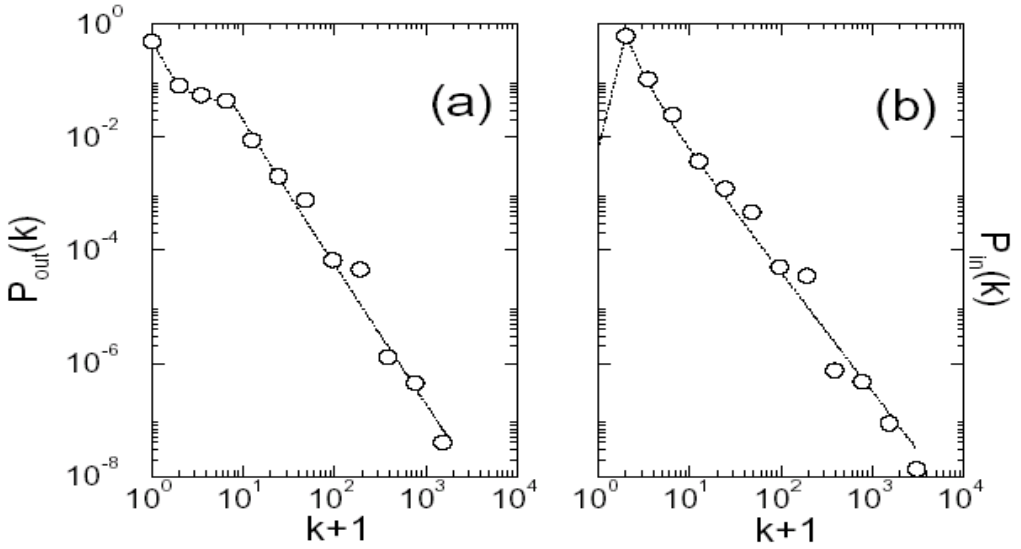
Starting Point	OUT	IN
Exploring outwards -- all nodes	3093	171
Exploring inwards -- unexpected nodes	3367	173

- The probability that a directed path exists from *u* to *v* is only 24%.
- The probability that a non-directed path exists from *u* to *v* is only 28%.

Edge type	In-links (directed)	Out-links (directed)	Undirected
Average connected distance	16.12	16.18	6.83

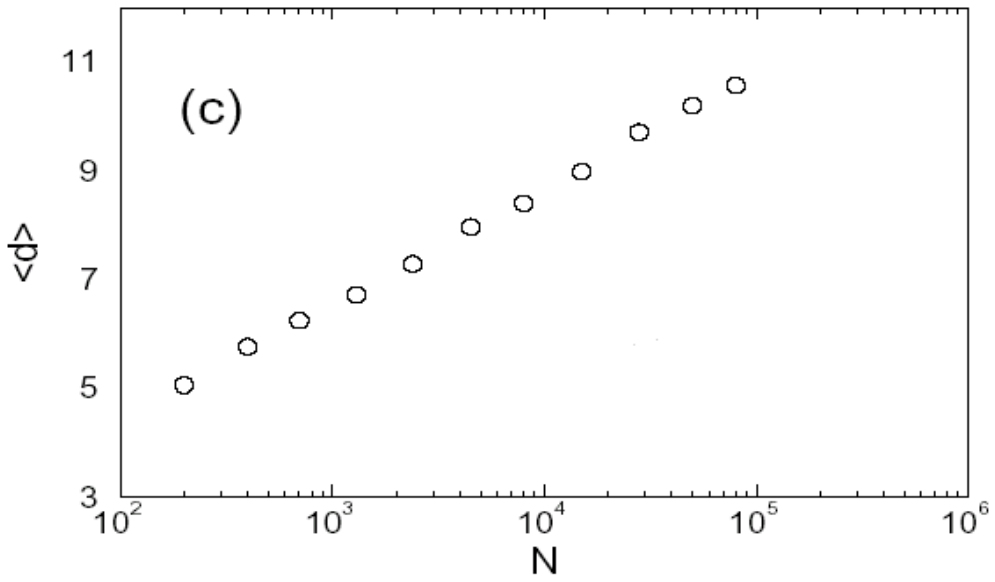
Local Connectivity [AJB99]

- The data was obtained from *nd.edu* domain, that contains 325,729 pages and 1,469, 680 links.



- Both $P_{out}(k)$ and $P_{in}(k)$ follow a power-law distribution

Shortest Path



- Average distance <d> = 0.35+2.06log(N)
- $\langle d_{nd.edu} \rangle = 11.2$

Topology of WWW Model

- Using $N = 8 \times 10^8$, <d_{www}>=18.59
- N increases 1000%, <d_{www}> changes from 19 to 21
- WWW – a highly connected graph

Intelligent agent vs robot

- Intelligent agent
  - interpret the links and follow only relevant one
  - In a short time find desired info by WWW
- Robot
  - Locate the info based on matching strings
  - Need $M(\langle d \rangle) \simeq 0.53 N^{0.92}$ search, to find a page at distance <d>

    $N = 8 \times 10^8$ [1], leads to $M = 8 \times 10^7$

Stochastic process [LFLW02]

- Simon's stochastic process – a birth process
  - There is a constant probability p that the next word is a new word
  - Given that the next word has already appeared, its probability of occurrence is proportional to the previous number of occurrences of that word.
- Rich get richer

Previous Work

- A power-law distribution is a function of the form

$$f(i) = Ci^{-\tau}$$

- Albert et al. predict τ=3
- Dorogovtsev et al. predict τ=2+A/m

An Urn Transfer Model

- Extension of Simon's stochastic process
- urn_i
- balls – web pages
- pins – links
- $F_i(k)$ – # of balls in urn_i after k steps
- $B(k) = \sum_i F_i(k)$ -- total # of balls in all urns.

An Urn Transfer Model (cont.)

- At step k+1, either:
  - A new ball is added to urn_1 with

$$p_{k+1} = 1 - \frac{(1-p)\sum_{i=1}^{k}(i+\alpha)F_i(k)}{k(1+\alpha p)+\alpha(1-p)}, \quad (1)$$

  - An urn is selected with $1-p_{k+1}$, urn_i is chosen with

$$\frac{(1-p)(i+\alpha)F_i(k)}{k(1+\alpha p)+\alpha(1-p)}, \quad (2)$$

Then one ball from urn_i is transferred to urn_{i+1}.

An Urn Transfer Model (cont.)

- To make pk+1 well defined, we must have

$$(1-p)(k+\alpha B(k)) \leq k(1+\alpha p)+\alpha(1-p).$$

- Then
  - P_{k+1} is always well defined, when p>=1/2
  - P_{k+1} is well defined only if $\alpha \leq \frac{p}{1-2p}$ when p<1/2

An Urn Transfer Model (cont.)

- Expected # of balls in *urni* is stated as

$$E_k(F_i(k+1)) = F_i(k) + \beta_k\Big((i-1+\alpha)F_{i-1}(k) - (i+\alpha)F_i(k)\Big) \quad \text{for i>1,}$$

and

$$E_k(F_1(k+1)) = F_1(k) + p_{k+1} - \beta_k(1+\alpha)F_1(k), \quad \text{for i=1}$$

Where

$$\beta_k = \frac{1-p}{k(1+\alpha p) + \alpha(1-p)},$$

is the normalising constant used in (2).

An Urn Transfer Model (cont.)

- Assume k tends to infinity,

  $E(F_i(k+1)) - E(F_i(k))$ tends to f_i and $\beta_k E(F_i(k))$ tends to βf_i;

- Then

$$f_i = \beta\Big((i-1+\alpha)f_{i-1} - (i+\alpha)f_i\Big)$$

- for recurrence equation

$$f_i = \frac{\beta(i-1+\alpha)}{1+\beta(i+\alpha)}\, f_{i-1}$$

with $$f_1 = \frac{p}{1+\beta(1+\alpha)},$$

An Urn Transfer Model (cont.)

- Using Stirling's approximation, we have

$$f_i \sim C\ i^{-(1+\rho)},$$

- A general power-law distribution for f_i, with exponent $1+\rho$.

An Evolution Model of Web

- Web is a directed graph G=(N,E)
- Fi(k), i>=1, is the number of nodes in the Web graph having i incoming links;
- Initially G contains just a single node,
- At each step, either:
  - With probability p a new node is added to G having one incoming link.
  - With probability 1-p a node is chosen with probability proportional to (i+), and then an additional incoming link is added to this node.

Simulation Results

Interpretation	Empirical	p_k-model	p-model
inlinks	2.09	2.096	2.094
outlinks	2.72	2.714	2.675
webpages	2.2	2.122	2.208
visitors	2.07	2.131	2.179

Table 1: Power law exponents of simulation results

Conclusion

- From the equations of extended stochastic process they derived an asymptotic formula for the exponent of the resulting power-law distribution.

- In order to explain the evolution of the Web graph both preferential and non-preferential processes are at work.

PageRank and BlockRank

Presented by:
Yu-Chung Chen
2006-4-10

Content

- “The PageRank Citation Ranking: Bring Order to the Web”, Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd, 1998
- “Exploiting the Block Structure of the Web for Computing PageRank”, Sepandar Kamvar, Taher Haveliwala, Christopher Manning, Gene Golub, 2003

Outline

- What is PageRank
- Exploit web block structure, BlockRank
- Applications

What is PageRank

- Web Link Structure
  - Forward/Back links
- Authority Matter!
- Random Surfer model

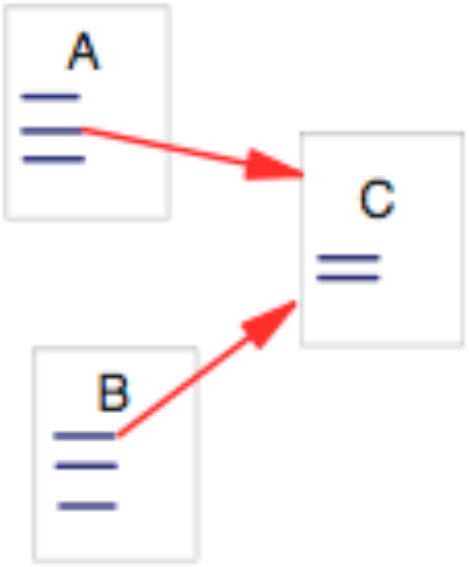
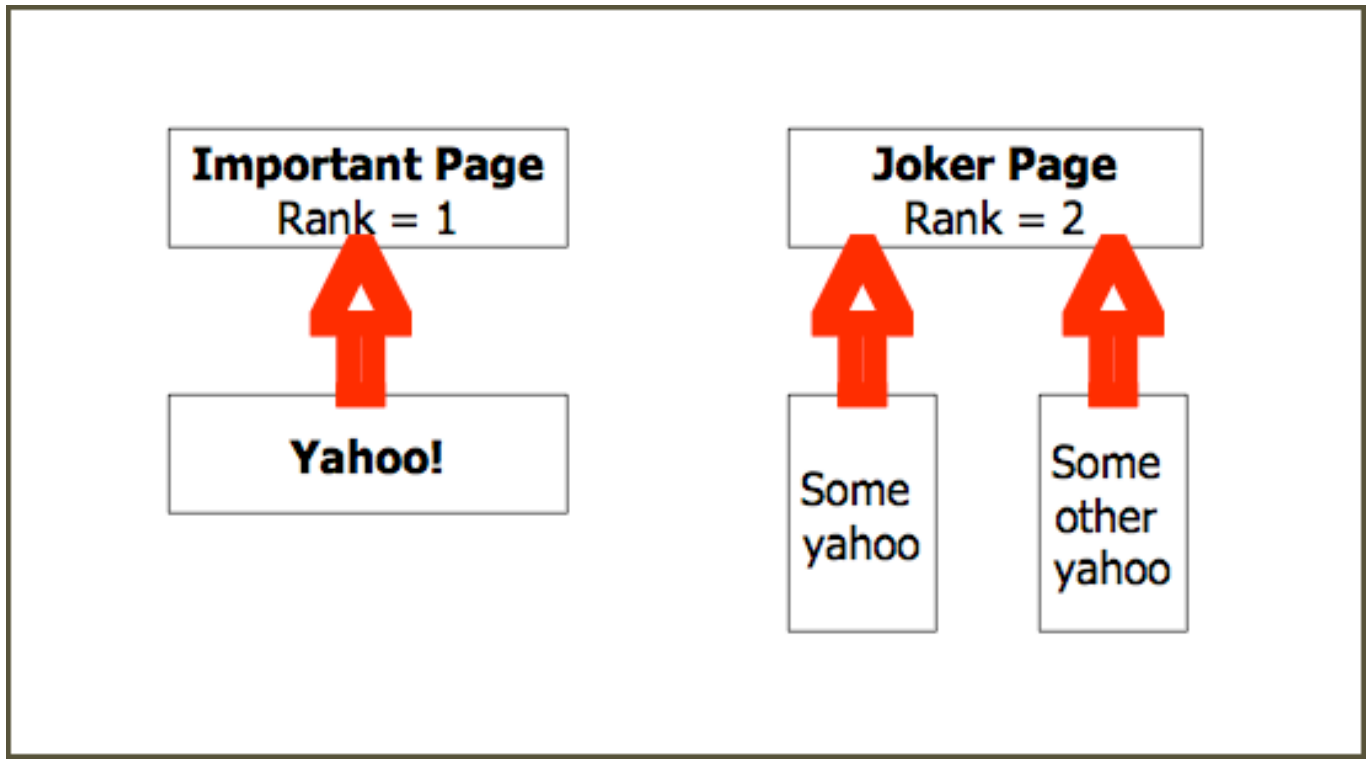


Figure 1: A and B are Backlinks of C

In the Old Days

- All backlinks created equal



Link Structure

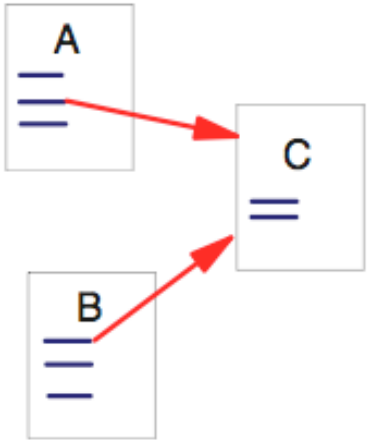


Figure 1: A and B are Backlinks of C

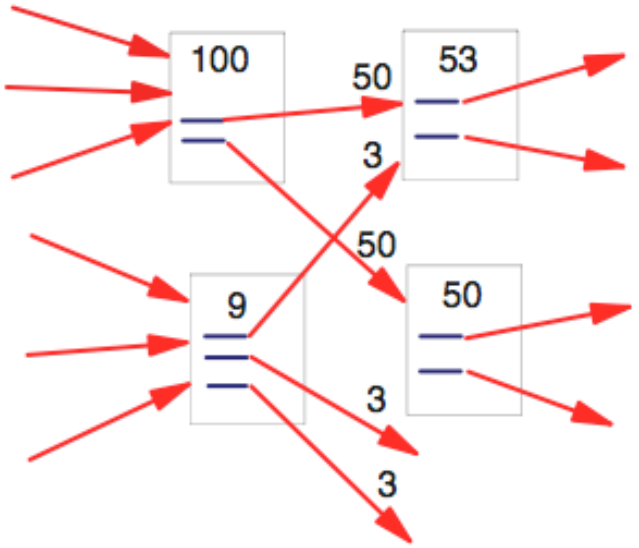


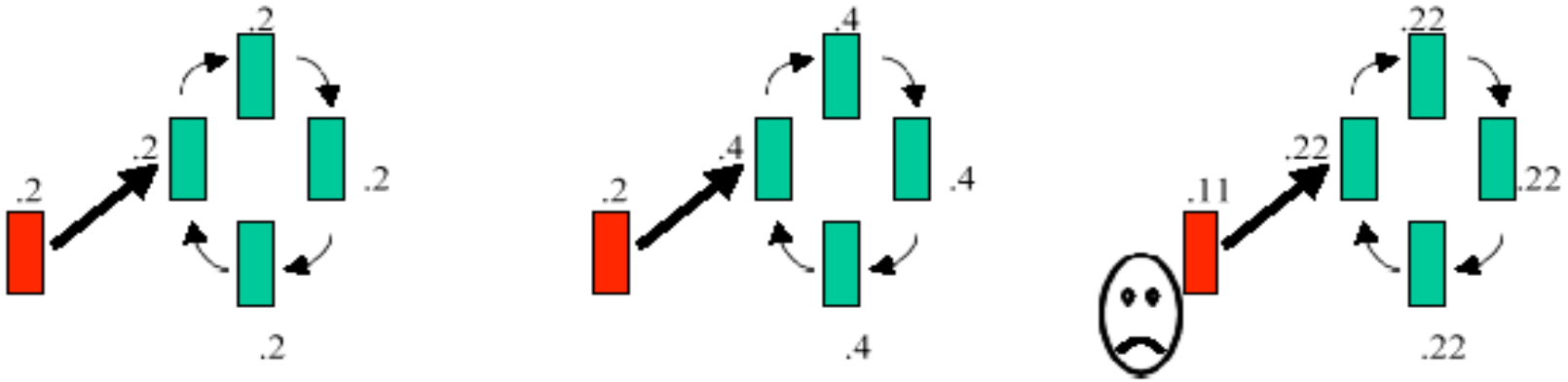
Figure 2: Simplified PageRank Calculation

Simple PageRank Definition

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

- Fu: Set of links from u
- Bu: Set of links to u
- Nu: |Fu|
- c: constant*
- R(u): Rank of u

Rank Sink



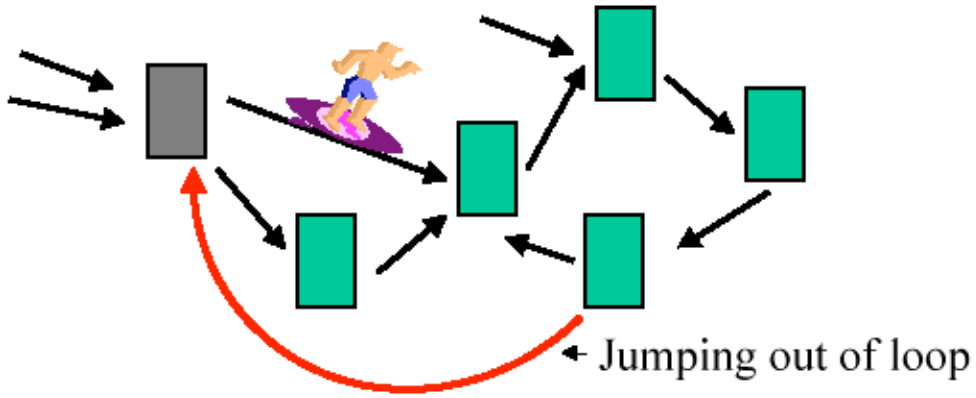
- The loop keeps accumulate rank, but never distribute any rank outside!

Escape Term

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} + cE(u)$$

- Solution: Rank source
- E(u) is a vector over web pages(for example, uniform or favorite page) that corresponds to a source of rank
- E(u) is a user designed parameter

Random Surfer Model



- Probability distribution of a random walk on the web graphs
- E(u) can be thought as the random surfer gets bored periodically and jumps to a different page and not kept in a loop forever

Markov Chain

- Discrete-time stochastic process
- Memory-less, based solely on present decision
- Random walks
  - Discrete-time stochastic process over a graph G=(V, E) with a transition probability matrix P
- Need to be aperiodic and irreducible*
  - Web graph is not strongly connected graph!
  - Add a new transition term to create a strongly connected transition graph

$$PageRank(p) = \frac{d}{n} + (1-d) \sum_{(q,p)\in E} PageRank(q)/outdegree(q)$$

Markov Chain(cont.)

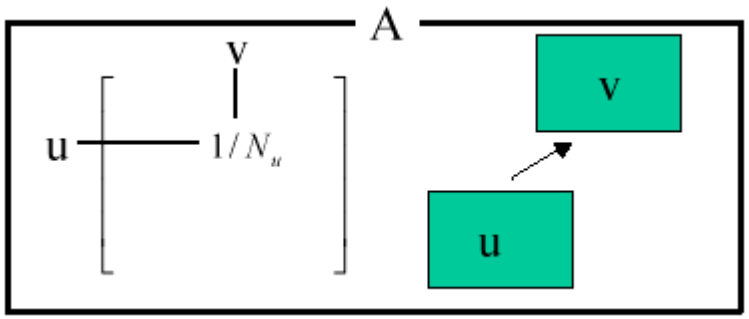
- According Markov theory, the PageRank(u) becomes the probability of being at ‘u’ page after a lot of clicks
- R is the solution to:

$$\mathbf{R} = \begin{bmatrix} (1-q)/N \\ (1-q)/N \\ \vdots \\ (1-q)/N \end{bmatrix} + q \begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & \cdots & \ell(p_1,p_N) \\ \ell(p_2,p_1) & \ddots & & \\ \vdots & & \ell(p_i,p_j) & \\ \ell(p_N,p_1) & & & \ell(p_N,p_N) \end{bmatrix} \mathbf{R}$$

- Solution to eigensystem
- Empirical results implies q = 0.85

Matrix Notation

$$R = c(A^T + E \times e^T)R$$



- Write to matrix form: $R=cA^TR+cE$
- R is the dominant eigenvector and c is the dominant eigenvalue of $(A + E \times e^T)$ because c is maximized
- Broken down to Eigenvalue problem, can be solved efficiently
  - Characteristic polynomial: not scalable
  - Power iterative method

Compute PageRank

$$
\begin{aligned}
R_0 &\leftarrow S \\
\text{loop}: & \\
R_{i+1} &\leftarrow AR_i \\
d &\leftarrow \|R_i\|_1 - \|R_{i+1}\|_1 \\
R_{i+1} &\leftarrow R_{i+1} + dE \\
\delta &\leftarrow \|R_{i+1} - R_i\|_1 \\
\text{while } \delta &> \epsilon
\end{aligned}
$$

Implementation

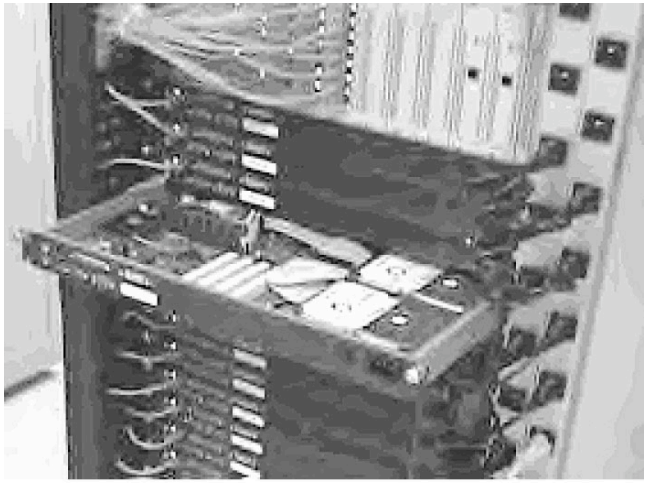
- 24 million pages
- 75 million URLs
- Memory and disk storage
  - Mem: weight vector: 4 bytes float
  - Disk: Matrix A: linear disk access
- 75000000*4/1000000 = 300MB/75million URLS
  - Fit into memory or multiple passes
- 6 minutes/iteration per machine

Back to 1998…

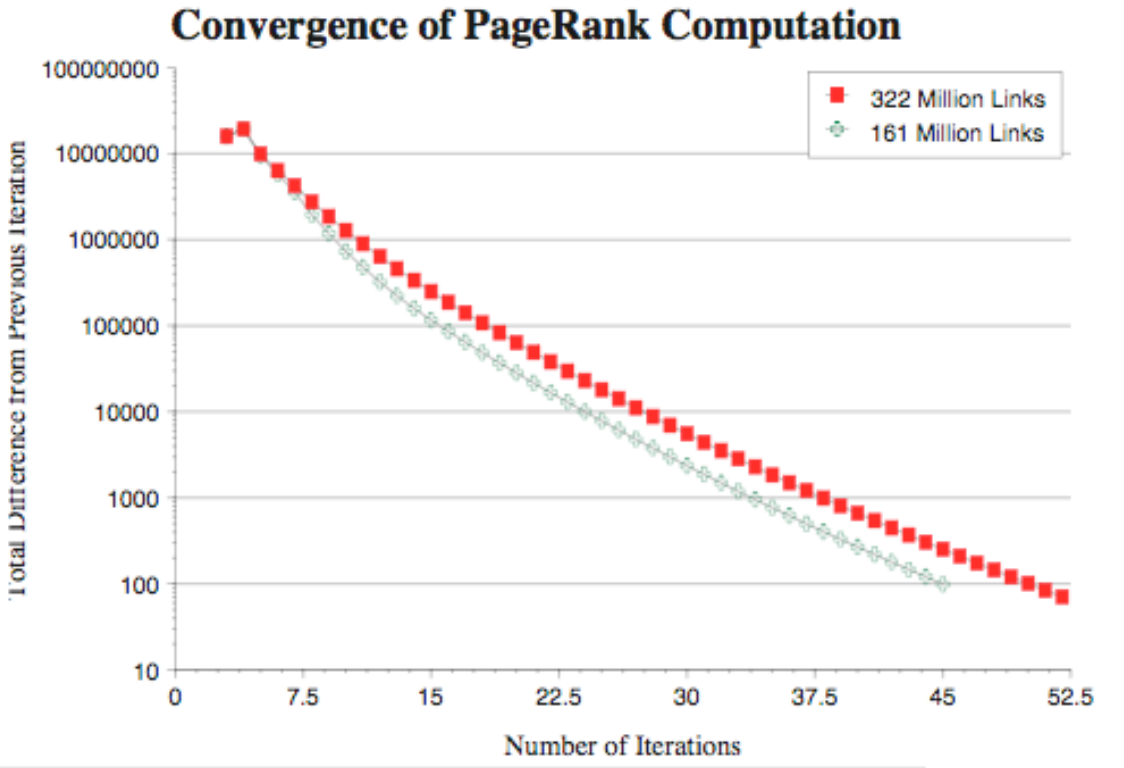
- In 1998, it took 5 days to index on 24 million page database

Now…

- Today: Google cluster and Google File system
  - 719 racks, 63,272 machines, 126,544 CPUs
  - 126,544 Gb RAM, 5,062Tb of disk space
  - http://www.tnl.net/blog/entry/How_many_Google_machines



Convergence



- O(log|V|) due to rapidly mixing web graph G
- Good initial ranking -> quick convergence*

Personalized PageRank

- Rank source E can be initialized:
  - Uniformly
    - All pages are treated the same, not good
      - Copyright, mailing list archives
  - Total weigh on a single page
    - Bad too
  - Everything in-between
    - About News, sports, etc

Issues - Quality

- Users are no random walkers
- Reinforcing effects/bias towards main pages/sites
- Linkage spam
- Manipulation by commercial interests
  - Cost to buy 1 link from an important page or a link from many non-important pages
  - Hilltop, only trust experts

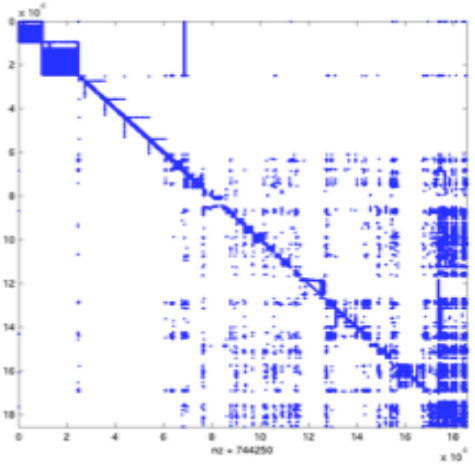
Issues - Speed

- Argue: Time is insignificant compared to building full text index, but…
- Re-compute ranks every few months
  - Web changes faster!
- WWW conference 2003: Google becoming up to 5 times faster
  - BlockRank: 3X the current calculation speed!
  - Extrapolation
  - Adaptive PageRank

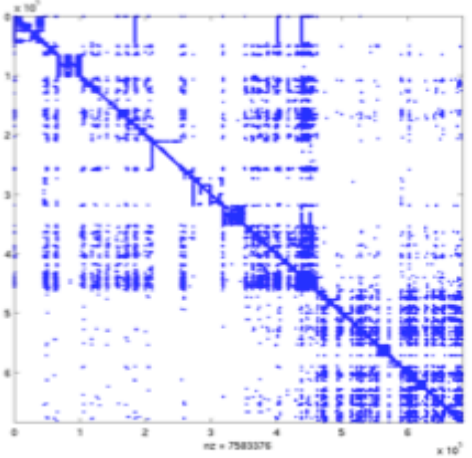
BlockRank

- Observations: web link graph is nested block structure
  - Pages under the same domain/host link to pages under the same domain/host
  - Internal links: 80% of all links per page
- Exploit this structure to speedup PageRank computation
- 3-stage algorithm

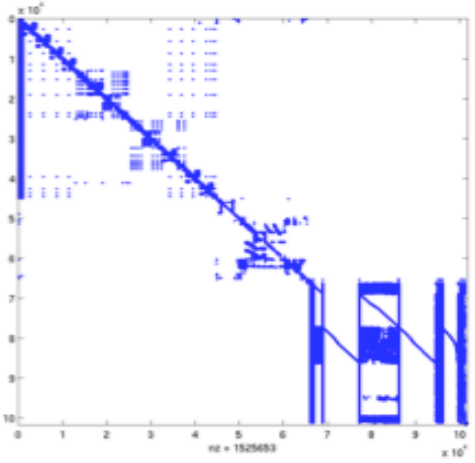
Block Structure



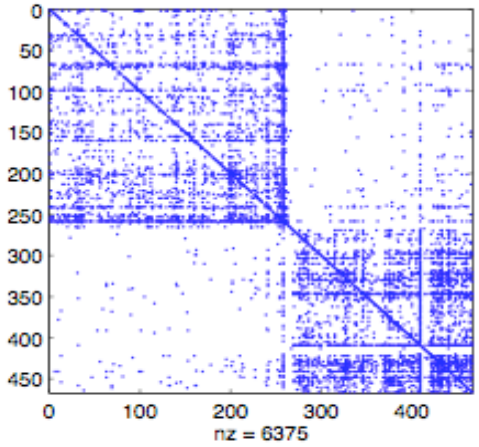
(a) IBM



(b) Stanford/Berkeley



(c) Stanford-50



(d) Stanford/Berkeley Host Graph

Experiment Setup & Observations

		Domain		Host	
Full	*Intra*	953M links	83.9%	899M links	79.1%
	Inter	183M links	16.1%	237M links	20.9%
DNR	*Intra*	578M links	95.2%	568M links	93.6%
	Inter	29M links	4.8%	39M links	6.4%

Table 2: Hyperlink statistics on LARGEWEB for the full graph (*Full*: 291M nodes, 1.137B links) and for the graph with dangling nodes removed (*DNR*: 64.7M nodes, 607M links).

3 Stage Algorithm

- 1. Local PageRanks of pages for each host are computed independently
- 2. Calculate BlockRanks of hosts in Block Graph
- 3. Local PageRanks are weighted by the ‘importance’ of the corresponding host
- 4. Standard PageRank algorithm using 2. Weighted aggregates as starting vector

Formulations

0. Sort the web graph lexicographically as described in Section 3, exposing the nested block structure of the web.
1. Compute the local PageRank vector $\vec{l}_J$ for each block J.

foreach *block* J **do**
$\vec{l}_J = pageRank(G_{JJ}, \vec{s}_J, \vec{v}_J)$;
end

2. Compute block transition matrix B and BlockRanks $\vec{b}$.

$$B = L^T A S$$
$$\vec{b} = pageRank(B, \vec{v}_k, \vec{v}_k)$$

3. Find an approximation $\vec{x}^{(0)}$ to the global PageRank vector $\vec{x}$ by weighting the local PageRanks of pages in block J by the BlockRank of J.

$$\vec{x}^{(0)} = L\vec{b}$$

4. Use this approximation as a start vector for a standard PageRank iteration.

$$\vec{x}^{(0)} = pageRank(G, \vec{x}^{(0)}, \vec{v})$$

Algorithm 3: BlockRank Algorithm

BlockRank Advantages

- Speedup due to caching effects*
  - Now CPU cache and Memory
- Converge quickly
- 1st step can be done completely parallel or distributed fashion
- Results of 1st step can be reused

Experiment Results

Algorithm	Wallclock time
Standard	180m 36s
Standard (using url-sorted links)	87m 44s
BlockRank (no pipelining)	81m 19s
BlockRank (w/ pipelining)	57m 06s

Table 6: Wallclock running times for 4 algorithms for computing PageRank with $c = 0.85$ to a residual of less than 10^{-3}.

	PageRank	BlockRank
STANFORD/BERKELEY	50	27
LARGEWEB	28	18

Table 7: Number of iterations needed to converge for standard PageRank and for BlockRank (to a tolerance of 10^{-4} for STANFORD/BERKELEY, and 10^{-3} for LARGEWEB).

Experiment Results(cont.)

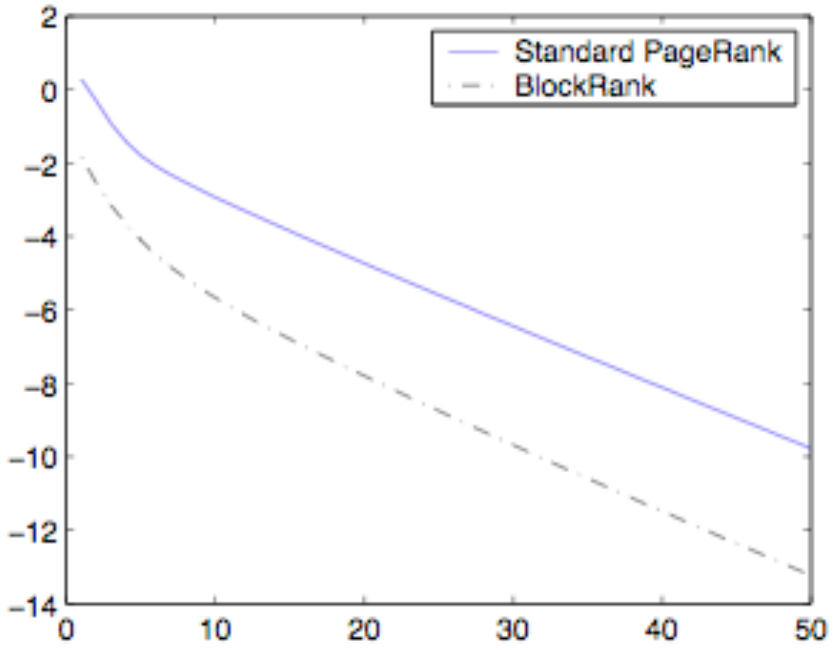
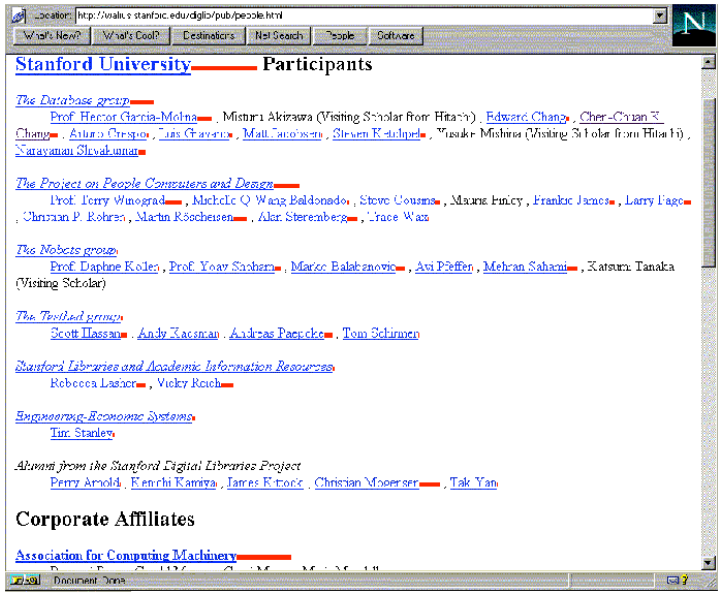


Figure 5: Convergence rates for standard PageRank (solid line) vs. BlockRank (dotted line). The x-axis is the number of iterations, and the y-axis is the log of the L_1-residual. STANFORD/BERKELEY data set; $c = 0.85$.

Applications - PageRank

- Estimate web traffic
- Backlink predictor
- Better search engine quality
- Check out Google.com!



Applications - BlockRank

PageRank/BlockRank Highlights

- PageRank is a global ranking based on the web's graph structure
- PageRank uses backlink information
- PageRank can be thought as random surfer model
- BlockRank: exploit block structure to speedup and advantages
- Various applications

Thank you for your attention!

Questions?

Backup Notes

More Implementations

- Unique integer ID for each URL
- Sort and Remove dangling Links
- Iterating until converge
- Add back dangling links and re-compute

Convergence

- G(V,E) is an expander with factor alpha if for all subsets S:|As| >= alpha|s|
- Eigenvalue separation: largest eigenvalue is sufficiently larger than the second-largest eigenvalue
- Random walk converges fast to a limiting probability distribution on a set of nodes in the graph

Google File System

- Performance, scalability, reliability and availability
- It’s normal to have hardware component failures
- Huge number of huge files
- Mutations
- Constraint specific file system

Google File System(cont.)

- Master: Handle meta-data
- Chunk server: Hold chunked data
  - 64MB per chunk
- Clients: Access to tera-bytes of data

Google File System(cont.)

- Reduce master workload
  - Reduce interaction with master
- Keep metadata in memory
- Availability!
- Replication!
  - Multiple replicated data chunks
  - Master state replication, and shadow master
- Fast recovery

References

- http://www-db.stanford.edu/~backrub/google.html
- http://www.tnl.net/blog/entry/How_many_Google_machines
- http://www.interesting-people.org/archives/interesting-people/200405/msg00013.html
- http://www.stanford.edu/~sdkamvar/research.html
- http://en.wikipedia.org/wiki/PageRank
- http://en.wikipedia.org/wiki/Markov_chain
- http://en.wikipedia.org/wiki/Eigenvector
- http://en.wikipedia.org/wiki/Power_method
- http://www.webmasterworld.com/forum34/523.htm