Scalable Data-driven PageRank: Algorithms, System Issues, and Lessons Learned

Xinxuan Li [1]

[1]University of Maryland Baltimore County

November 13, 2015

Outline

- **PageRank** is a numeric value that represents the importance of a page present on the web.
- when one page links to another page, it is effectively casting a vote for the other page.
- More votes implies more importance.
- Importance of each vote is taken into account when a page's PageRank is calculated.
- PageRank is Google's way of deciding a page's importance.
- It matters because it is one of the factors that determines a page's ranking in the search results.

Proposed by Sergey Brin and Larry Page

The PageRank Citation Ranking: Bringing Order to the Web

January 29, 1998

Abstract

The importance of a Web page is an inherently subjective matter, which depends on the readers interests, knowledge and attitudes. But there is still much that can be said objectively about the relative importance of Web pages. This paper describes PageRank, a method for rating Web pages objectively and mechanically, effectively measuring the human interest and attention devoted to them.

We compare PageRank to an idealized random Web surfer. We show how to efficiently compute PageRank for large numbers of pages. And, we show how to apply PageRank to search and to user navigation.

1 Introduction and Motivation

The World Wide Web creates many new challenges for information retrieval. It is very large and heterogeneous. Current estimates are that there are over 150 million web pages with a doubling life of less than one year. More importantly, the web pages are extremely diverse, ranging from "What is Joe having for lunch today?" to journals about information retrieval. In addition to these major challenges, search engines on the Web must also contend with inexperienced users and pages engineered to manipulate search engine ranking functions.

However, unlike "flat" document collections, the World Wide Web is hypertext and provides considerable auxiliary information on top of the text of the web pages, such as link structure and link text. In this paper, we take advantage of the link structure of the Web to produce a global "importance" ranking of every web page. This ranking, called PageRank, helps search engines and users quickly make sense of the vast heterogeneity of the World Wide Web.

1.1 Diversity of Web Pages

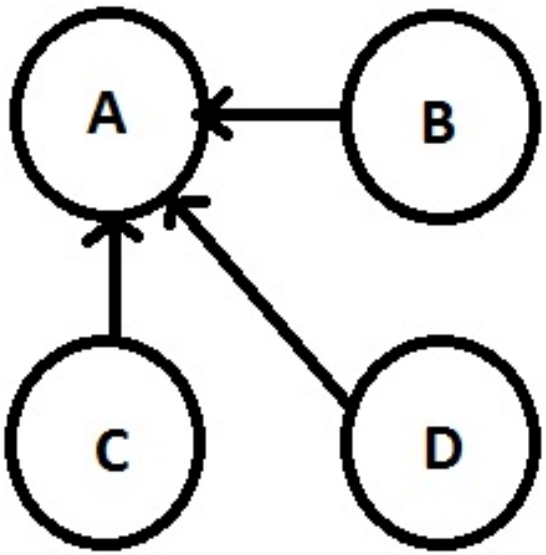
Although there is already a large literature on academic citation analysis, there are a number of significant differences between web pages and academic publications. Unlike academic papers which are scrupulously reviewed, web pages proliferate free of quality control or publishing costs. With a simple program, huge numbers of pages can be created easily, artificially inflating citation counts. Because the Web environment contains competing profit seeking ventures, attention getting strategies evolve in response to search engine algorithms. For this reason, any evaluation strategy which counts replicable features of web pages is prone to manipulation. Further, academic papers are well defined units of work, roughly similar in quality and number of citations, as well as in their purpose – to extend the body of knowledge. Web pages vary on a much wider scale than academic papers in quality, usage, citations, and length. A random archived message posting

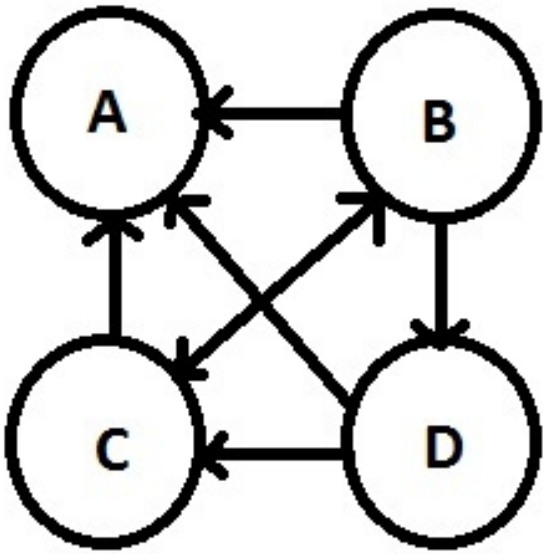
Example

Website	PageRank
http://www.google.com/	9 / 10
http://www.youtube.com/	9 / 10
http://www.harvard.edu/	8 / 10
http://www.math.umbc.edu/ kogan/	4 / 10

Table: PageRank Table. Results are from http://www.seocentro.com/tools/search-engines/pagerank.html

Calculate PageRank





$$x_A = x_B + x_C + x_D = 0.75 \qquad x_A = \frac{x_B}{3} + \frac{x_C}{2} + \frac{x_D}{2} = 0.458$$

Hence, the general form for any node v is

$$x_v = \sum_{\omega \in S_v} \frac{{x_\omega}^k}{|\Gamma_\omega|}.$$

Teleportation Parameter

Teleportation Parameter α is the probability that a person will continue.

$$x_v = \alpha \Sigma_{\omega \in S_v} \frac{x_\omega}{|\Gamma_\omega|} + (1 - \alpha)$$

The sum of of all PageRank is N.

$$x_v = \alpha \Sigma_{\omega \in S_v} \frac{x_\omega}{|\Gamma_\omega|} + \frac{(1-\alpha)}{N}$$

The sum of of all PageRank is 1.
However, we are taking $x = \frac{x}{\|x\|_1}$. Both formats have same results.

Power Method

For every iteration,

$$x_v = \alpha \Sigma_{\omega \in S_v} \frac{x_\omega}{|\Gamma_\omega|} + \frac{(1-\alpha)}{N}$$

$$\implies \mathbf{x(t+1)} = \alpha \mathbf{P}^T \mathbf{x(t)} + (1-\alpha)\mathbf{e}.$$

P is a transition probability. when $t \to \infty$, $\mathbf{x(t+1)} \to \mathbf{x(t)}$.
$\|\mathbf{x}\|_1 = 1$ so $\mathbf{Ex} = \mathbf{1}$, where $\mathbf{E}$ is a matrix with all entries as 1.

$$\mathbf{x} = (\alpha \mathbf{P}^T + (1-\alpha)\mathbf{E})\mathbf{x}.$$

Example

$$v = \{\text{Dr.Kogan}, \text{Maria}, \text{Zois}, \text{Math students}, \text{Engineering students}\}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$P = D^{-1}A^T$$

$$\mathbf{r} = (1-\alpha)\mathbf{e} - (\mathbf{I} - \alpha\mathbf{P}^T)\mathbf{x}$$

Topology-driven PageRank

- Given a graph $G = (V, \xi)$
  where V is a vertex set, ξ is an edge set.
- The PageRank vector $\mathbf{x} = (1 - \alpha)\mathbf{e}$.
  Where α is a teleportation parameter between 0 and 1;
  $\mathbf{e}$ is the vector of all 1's.
- ${x_v}^{k+1} = \alpha \Sigma_{\omega \in S_v} \frac{{x_\omega}^k}{|\Gamma_\omega|} + (1 - \alpha)$
  where $x_v k$ is the node v's PageRank at kth-iteration;
  S_v is the set of incoming neighbors of node v;
  Γ_v is the set of outgoing neighbors of node v.
- The PageRank values are repeatedly computed until the difference between ${x_v}^k$ and ${x_v}^{k+1}$ is smaller than ϵ for all nodes.

Topology-driven PageRank

Define a row-stochastic matrix $\mathbf{P}$, such that $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ where $\mathbf{A}$ is an adjacency matrix and $\mathbf{D}$ is the degree diagonal matrix.
The **PageRank** problem requires solving the linear system:

$$(\mathbf{I} - \alpha\mathbf{P}^{\mathbf{T}})\mathbf{x} = (1 - \alpha)\mathbf{e}.$$

The residual is defined to be

$$\mathbf{r} = (1 - \alpha)\mathbf{e} - (\mathbf{I} - \alpha\mathbf{P}^{\mathbf{T}})\mathbf{x}.$$

Data-driven PageRank

- Given a graph $G = (V, \xi)$
  where V is a vertex set, ξ is an edge set.
- The PageRank vector $\mathbf{x} = (1 - \alpha)\mathbf{e}$.
- Pick a node in the working list V, computing the node's PageRank and adding its outgoing neighbors to the worklist.

Comparison

Algorithm 1. Topology-driven PageRank

Input: graph $G = (\mathcal{V}, \mathcal{E})$, α, ϵ
Output: PageRank $\mathbf{x}$

1: Initialize $\mathbf{x} = (1 - \alpha)\mathbf{e}$
2: **while** true **do**
3: **for** $v \in \mathcal{V}$ **do**
4: $x_v^{(k+1)} = \alpha \sum_{w \in \mathcal{S}_v} \frac{x_w^{(k)}}{|\mathcal{T}_w|} + (1 - \alpha)$
5: $\delta_v = |x_v^{(k+1)} - x_v^{(k)}|$
6: **end for**
7: **if** $\|\boldsymbol{\delta}\|_\infty < \epsilon$ **then**
8: break;
9: **end if**
10: **end while**
11: $\mathbf{x} = \frac{\mathbf{x}}{\|\mathbf{x}\|_1}$

Algorithm 2. Data-driven PageRank

Input: graph $G = (\mathcal{V}, \mathcal{E})$, α, ϵ
Output: PageRank $\mathbf{x}$

1: Initialize $\mathbf{x} = (1 - \alpha)\mathbf{e}$
2: **for** $v \in \mathcal{V}$ **do**
3: `worklist.push(`v`)`
4: **end for**
5: **while** `!worklist.isEmpty` **do**
6: v = `worklist.pop()`
7: $x_v^{new} = \alpha \sum_{w \in \mathcal{S}_v} \frac{x_w}{|\mathcal{T}_w|} + (1 - \alpha)$
8: **if** $|x_v^{new} - x_v| \geq \epsilon$ **then**
9: $x_v = x_v^{new}$
10: **for** $w \in \mathcal{T}_v$ **do**
11: **if** w is not in `worklist` **then**
12: `worklist.push(`w`)`
13: **end if**
14: **end for**
15: **end if**
16: **end while**
17: $\mathbf{x} = \frac{\mathbf{x}}{\|\mathbf{x}\|_1}$

Pull and push

Each active nodes v reads(pulls) its incoming neighbor's PageRank values and updates(pushes) residuals to its outgoing neighbors.

- **Pull**

$$x_v{}^{k+1} = \alpha \Sigma_{\omega \in S_v} \frac{x_\omega{}^k}{|\Gamma_\omega|} + (1 - \alpha)$$

- **Push**

$$\text{for } \omega \in \Gamma_\omega$$
$$r_\omega = r_\omega + \frac{r_v \alpha}{|\Gamma_\omega|}$$

Data-driven push-based PageRank

In the push-based algorithms, an active node updates its own value, and only pushes its neighbor's values.

1. Initialize $\mathbf{x} = (1-\alpha)\mathbf{e}$
2. $\mathbf{r}^0 = (1-\alpha)\alpha\mathbf{P}^T\mathbf{e}$
   $\mathbf{r} = (1-\alpha) - (\mathbf{I} - \alpha\mathbf{P}^T)(1-\alpha)\mathbf{e}$
3. Update all the active nodes v
$$x_v^{new} = x_v^{old} + r_v$$
4. For its outgoing neighbors ω
$$r_\omega^{new} = r_\omega^{old} + \frac{\alpha v}{|\Gamma_\omega|}$$
5. If $r_\omega^{new} > \epsilon$ or $r_\omega^{old} < \epsilon$ then send ω to push list(repeat 3).
6. Set $r_v = 0$.

Data-driven pull-push-based PageRank

1. Given a graph $G = (V, \xi)$
   where V is a vertex set, ξ is an edge set.
2. The PageRank vector $\mathbf{x} = (1 - \alpha)\mathbf{e}$.
3. Pick a active node in the working list V, computing the node's PageRank by reading its incoming neighbors.
4. Updating its out-coming neighbors follow the rule by push-based PageRank.

   $r_\omega^{\text{new}} > \epsilon$ or $r_\omega^{\text{old}} < \epsilon$ then send ω to push list(repeat 3).

Comparison

Algorithm 3. Pull-Push-based PageRank

Input: graph $G = (\mathcal{V}, \mathcal{E})$, α, ϵ
Output: PageRank $\mathbf{x}$

1: Initialize $\mathbf{x} = (1-\alpha)\mathbf{e}$
2: Initialize $\mathbf{r} = \mathbf{0}$
3: **for** $v \in \mathcal{V}$ **do**
4: **for** $w \in \mathcal{S}_v$ **do**
5: $r_v = r_v + \frac{1}{|\mathcal{T}_w|}$
6: **end for**
7: $r_v = (1-\alpha)\alpha r_v$
8: **end for**
9: **for** $v \in \mathcal{V}$ **do**
10: `worklist.push(`v`)`
11: **end for**
12: **while** `!worklist.isEmpty` **do**
13: v = `worklist.pop()`
14: $x_v = \alpha \sum_{w \in \mathcal{S}_v} \frac{x_w}{|\mathcal{T}_w|} + (1-\alpha)$
15: **for** $w \in \mathcal{T}_v$ **do**
16: $r_w^{old} = r_w$
17: $r_w = r_w + \frac{r_v \alpha}{|\mathcal{T}_v|}$
18: **if** $r_w \geq \epsilon$ and $r_w^{old} < \epsilon$ **then**
19: `worklist.push(`w`)`
20: **end if**
21: **end for**
22: $r_v = 0$
23: **end while**
24: $\mathbf{x} = \frac{\mathbf{x}}{\|\mathbf{x}\|_1}$

Algorithm 4. Push-based PageRanl

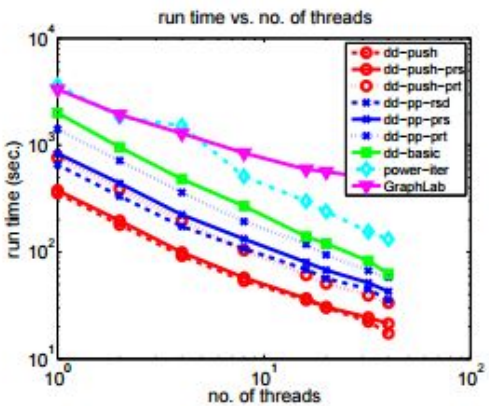
Input: graph $G = (\mathcal{V}, \mathcal{E})$, α, ϵ
Output: PageRank $\mathbf{x}$

1: Initialize $\mathbf{x} = (1-\alpha)\mathbf{e}$
2: Initialize $\mathbf{r} = \mathbf{0}$
3: **for** $v \in \mathcal{V}$ **do**
4: **for** $w \in \mathcal{S}_v$ **do**
5: $r_v = r_v + \frac{1}{|\mathcal{T}_w|}$
6: **end for**
7: $r_v = (1-\alpha)\alpha r_v$
8: **end for**
9: **for** $v \in \mathcal{V}$ **do**
10: `worklist.push(`v`)`
11: **end for**
12: **while** `!worklist.isEmpty` **do**
13: v = `worklist.pop()`
14: $x_v^{new} = x_v + r_v$
15: **for** $w \in \mathcal{T}_v$ **do**
16: $r_w^{old} = r_w$
17: $r_w = r_w + \frac{r_v \alpha}{|\mathcal{T}_v|}$
18: **if** $r_w \geq \epsilon$ and $r_w^{old} < \epsilon$ **then**
19: `worklist.push(`w`)`
20: **end if**
21: **end for**
22: $r_v = 0$
23: **end while**
24: $\mathbf{x} = \frac{\mathbf{x}}{\|\mathbf{x}\|_1}$

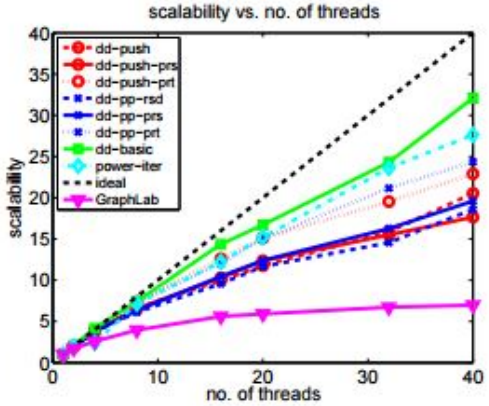
Scheduling

- Schedule process the node has largest residual first. Define: a node v's priority p_v to be the residual per unit work. Let $p_v = \frac{p_v}{d_v}$ For pull-push-based PageRank:$d_v = |\Gamma_v| + |S_v|$. For push-based PageRank:$d_v = |\Gamma_v|$
- NUMA-aware OBIM priority scheduler(stealing)
  This scheduler uses an approximate priority consensus protocol to inform a per-thread choice to search for stealable high-priority work or to operate on local near-high-priority work.
- Bulk-synchronous priority scheduler.
  This scheduler operates in rounds. Each rounds, all items with priority above a threshold are executed. Generated tasks and unexecuted items are placed in the next round. Threshold are updated each round based on the distribution of priorities observed recomputed every round.

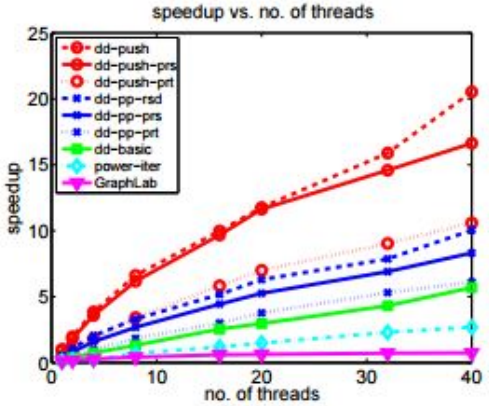
Results



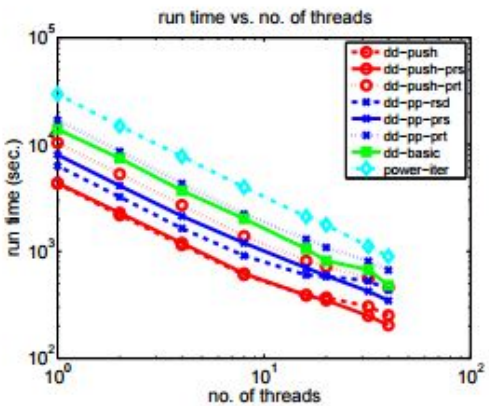
(a) pld run time



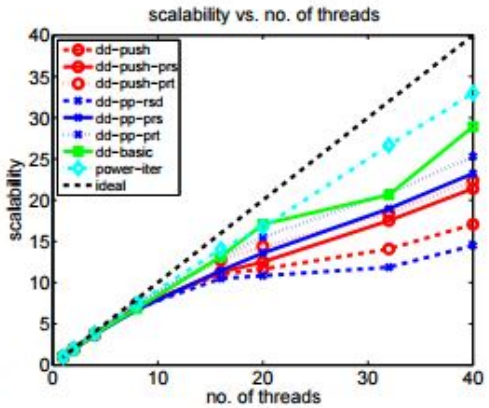
(b) pld scalability



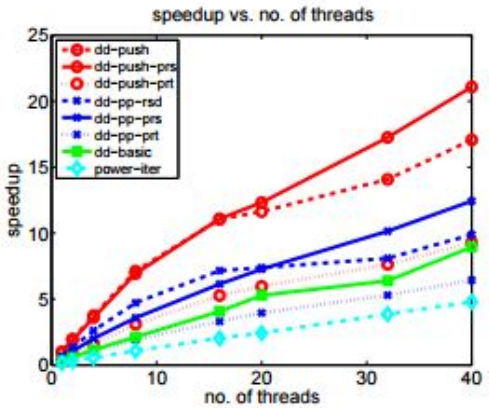
(c) pld speedup



(d) twitter run time



(e) twitter scalability



(f) twitter speedup

References

- Whang, J., Lenharth, A., Dhillon, I., Pingali, K., Larsson Traff, J., Hunold, S., and Versaci, F. (2015). Scalable Data-Driven PageRank: Algorithms, System Issues, and Lessons Learned. doi:10.1007/978-3-662-48096-034
- Frank McSherry, A Uniform Approach to Accelerated PageRank Computation, 2005