

Distributed Page Ranking in Structured P2P Networks

ShuMing Shi, Jin Yu, GuangWen Yang, DingXing Wang

Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China

E-mail: {ssm01, yujin}@mails.tsinghua.edu.cn; {ygw, dxwang}@mail.tsinghua.edu.cn

Abstract

This paper discusses the techniques of performing distributed page ranking on top of structured peer-to-peer networks. Distributed page ranking are needed because the size of the web grows at a remarkable speed and centralized page ranking is not scalable. Open System PageRank is presented in this paper based on the traditional PageRank used by Google. We then propose some distributed page ranking algorithms, partially prove their convergence, and discuss some interesting properties of them. Indirect transmission is introduced in this paper to reduce communication overhead between page rankers and to achieve scalable communication. The relationship between convergence time and bandwidth consumed is also discussed. Finally, we verify some of the discussions by experiments based on real datasets.

1. Introduction

Link structure based page ranking for determining the “importance” of web pages has become an important technique in search engines. In particular, the HITS [1] algorithm maintains a hub and authority score for each page, in which the authority and hub scores are computed by the linkage relationship of pages in the hyperlinked environment. The PageRank [2] algorithm used by Google [3] determines “scores” of web pages by compute the eigenvector of a matrix iteratively.

As size of the web grows, it becomes harder and harder for existing search engines to cover the entire web. We need distributed search engines which are scalable with respect to the number of pages and the number of users. In a distributed search engine, page ranking is not only needed as in its centralized counterpart for improving query results, but should be performed distributedly for scalability and availability.

A straightforward way to achieve distributed page ranking is simply scaling HITS or PageRank algorithms to distributed environment. But it is not a trivial thing to do that. Both HITS and PageRank are iterative algorithms. As each iteration step needs computation results of previous step, synchronize operation is needed. However, it is hard to achieve synchronous communication in wide spread distributed environment. In addition, page partitioning and

communication overhead must be considered carefully while performing distributed page ranking.

Structured peer-to-peer overlay networks have recently gained popularity as a platform for the construction of self-organized, resilient, large-scale distributed systems [6, 13, 14, 15]. In this paper, we try to perform effective page ranking on top of structured peer-to-peer networks. We first propose some distributed page ranking algorithms based on google’s PageRank [2] and present some interesting properties and results about them. As communication overhead is more important than CPU and memory usage in distributed page ranking, we then discuss strategies of page partitioning and ideas about alleviating communication overhead. By doing this, our paper makes the following contributions:

- We provide two distributed page ranking algorithms, partially prove their convergence, and verify their features by using a real dataset.
- We identify major issues and problems related to distributed page ranking on top of structured P2P networks.
- Indirect transmission is introduced in this paper to reduce communication overhead between page rankers and to achieve scalable communication.

The rest of the paper is as follows: After briefly reviewing the PageRank algorithm in section 2, a modification on PageRank for open systems is proposed in section 3. Issues in distributed page ranking are discussed one by one in section 4. Section 5 uses a real dataset to validate some of our discussions.

2. Brief Review of PageRank

The essential idea behind PageRank [2] is that if page u has a link to page v , then u is implicitly conferring some kind of importance to v . Intuitively, a page has high rank if it has many back links or it has a few highly ranked backlinks.

Let n be the number of pages, $R(u)$ be the rank of page u , and $d(u)$ be the out-degree of page u . For each page v , let B_v represent the set of pages pointing to v , then rank of v can be computed as follows:

$$R(v) = c \sum_{u \in B_v} \frac{R(u)}{d(u)} + (1-c)E(v) \quad (2.1)$$

The second term in the above expression is for avoiding rank sink [2].

Stated another way, let A be a square matrix with the rows and columns corresponding to web pages. Let $A_{u,v}=1/d(u)$ if there is an edge from u to v and $A_{u,v}=0$ if not. Then we can rewrite formula 2.1 as follows:

$$R = cAR + (1 - c)E \quad (2.2)$$

Then PageRank may be computed as algorithm 1.

```

R0 = S
loop
  Ri+1 = ARi
  D = ||Ri||1 - ||Ri+1||1
  Ri+1 = Ri+1 + dE
  δ = ||Ri+1 - Ri||1
while δ > ε

```

Algorithm 1: PageRank Algorithm

3. Open System PageRank

Algorithm 1 can't be simply scaled for distributed PageRank for two reasons: Firstly, as each machine only contains part of the whole link graph, so operations like $\|R_i\|$ is time-consuming. Secondly, each iteration step needs computation results of previous step, so synchronize operation is needed when the computation is distributed. In addition, formula 2.1 view pages crawled as a closed system; while in distributed systems, web pages in each machine must be views as open systems, for they must communication with pages in other machines to performing PageRank. All these demand PageRank for open systems.

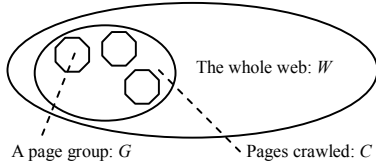


Fig.1. Different scopes of pages

In figure 1, the small ellipse contains pages grasped by a search engine. And the small octagon can be seen as a page group which comprises pages located on a single machine.

Figure 2 shows a web page group comprises four pages. Thick real lines denote link relationship between pages, for example, page P_1 points to page P_2 and P_4 . To avoid rank sink [2] and guarantee convergence of iteration, we can add a complete set of virtual edges between every pair of pages¹, as [8] has done. These virtual edges are denoted in Fig.2 by dashed lines with double arrows. Afferent links (edges pointed from pages in other groups to pages of this

group) are expressed by thin real lines. This kind of edges can also be viewed as some kind of rank source to this group. There are also edges pointed out from this group to pages in other groups, called efferent links which is denoted by dot-and-dashed lines.

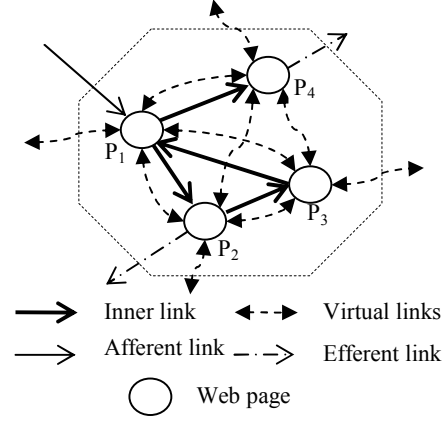


Fig.2. A web page group

Consider a page group G . For any page u in it, let $R(u)$, $d(u)$ be the rank and out-degree of u respectively. For each page v , let Bv represent the set of pages pointing to v in G . Assume that for each page u (with rank $R(u)$), $\alpha R(u)$ of its rank is used for real rank transmission (by inner or efferent links), while $\beta R(u)$ of its rank for virtual rank transmission ($\alpha + \beta = 1$).

For a page v , its rank can come from inner links, virtual links or afferent links, defined as $I(v)$, $V(v)$, and $X(v)$ respectively. We can easily know (use the same way as PageRank in section 2) that rank from inner links is:

$$I(v) = \alpha \sum_{u \in Bv} R(u) / d(u) \quad (3.1)$$

Now consider virtual links. Assume all virtual links have the same capacity, in other words, a page transmits the same amount of rank to other pages (include itself) by virtual links, then rank acquired from virtual links is:

$$V(v) = \sum_{u \in W} \beta R(u) / w = \frac{\beta}{w} \sum_{u \in W} R(u) = \beta E(v) \quad (3.2)$$

Here W is the entire web, and $w=|W|$. And $E(v)$ is the average page score over all pages in the whole web, the same meaning as in standard PageRank. For briefness, we can assume $E(v)=1$ for all pages in the group. The case when E is not uniform over pages can be used for personalized page ranking [5, 9].

Then ranks of all pages in the group can be expressed as follows:

$$\begin{aligned} R(v) &= I(v) + V(v) + X(v) \\ &= \alpha \sum_{u \in Bv} \frac{R(u)}{d(u)} + \beta E(v) + X(v) \end{aligned} \quad (3.3)$$

Or:

$$R = AR + (\beta E + X) \quad (3.4)$$

¹ Not limit to page pairs inside the group here. In fact, all pages (crawled and not crawled) in the whole web are included.

Here A is a square matrix with the rows and columns corresponding to web pages with $A_{u,v} = \alpha / d(u)$ if there is an edge from u to v and $A_{u,v}=0$ if not. Define $Y(v)$ as ranks ready for being sent to other page groups, we have:

$$Y = BR \quad (3.5)$$

Here B is a square matrix with $B_{u,v} = \beta / d(u)$ if $d(u) > 0$ and $A_{u,v}=0$ if not.

The main difference between standard PageRank and this variation is that: The former is for closed systems and the balance of rank carefully considered in each iteration step. While the later is for open systems and allow ranks to be flowed into and out of the system.

```
function R* = GroupPageRank(R0, X) {
  repeat
    R_{i+1} = AR_i + \beta E + X
    \delta = ||R_{i+1} - R_i||_1
  until \delta > \epsilon
  return R_i
}
```

Algorithm 2: PageRank algorithm for an open system

Using formula 3.4, rank of each page in the group can be solved iteratively (see Algorithm 2). The convergence of Algorithm 2 is guaranteed by the following theorems (refer to [7] for their proofs):

Theorem 3.1 Iteration $x = Ax + f$ converges for any initial value x_0 if and only if $\rho(A) < 1$. Here $\rho(A)$ is the spectral radius of matrix A .

Theorem 3.2 For any matrix A and matrix norm $\|\cdot\|$, $\rho(A) \leq \|A\|$

Theorem 3.3 Let $\|A\| < 1$, and $x_m = Ax_{m-1} + f$ converges to x^* , then

$$\|x^* - x_m\| \leq \frac{\|A\|}{1 - \|A\|} \|x_m - x_{m-1}\|$$

For Algorithm 2, we have $\rho(A) \leq \|A\|_\infty \leq \alpha$ by Theorem 3.2. Then, by Theorem 1, the iteration converges. Theorem 3.3 implies that we can use $\|x_m - x_{m-1}\|$ as termination condition of the iteration.

4. Distributed Page Ranking

In this section, we consider how to perform page ranking in a peer-to-peer environment. Assume there are K nodes (called *page rankers*) participating in page ranking, and each of them is in charge of a subset of the whole web pages to be ranked. Pages crawled by crawler(s) are partitioned into K groups and mapped onto K page rankers according to some strategy. Each page ranker runs a page ranking algorithm on it. Since there have links between pages of different page groups, page rankers need to communicate periodically to exchange updated ranking

values. Some key problems will be discussed in this section.

4.1. Web Page Partitioning

Different strategies can be adopted to divide web pages among page rankers: divide pages randomly, divide by the hash code of page URLs, or divide by the hash code of websites. As crawler(s) may revisit pages in order to detect changes and refresh the downloaded collection, one page may participate in dividing more than one time. The random dividing strategy doesn't fulfill this need for taking the risk of sending a page to different page rankers on different times. When performing page ranking, page scores may transmit between page rankers, causing communication overhead between nodes. Because number of inner-site links overcomes that of inter-site ones for a web site ([16] finds that 90% of the links in a page point to pages in the same site on average), divide at site-granularity instead of page-granularity can reduce communication overhead greatly. To sum up, dividing pages by hash code of websites is a something better strategy.

4.2. Distributed PageRank Algorithms

Two different algorithms, DPR1 and DPR2, are shown (see Algorithm 3 and 4) to performing distributed page ranking. Both of them contain a main loop, and in each loop, the algorithm first refreshes the value of X (for other groups may have sent new ranks by the afferent links of the group), and then compute vector R by one or more iteration steps, and lastly, compute new Y and send it to other nodes.

Note that each node runs the algorithm asynchronously, in other words, ranking programs in all the nodes can start at different time, execute at different 'speed', sleep for some time, suspend itself as its wish, or even shutdown. In fact, we can insert some delays before or after any instructions.

```
function DPR1() {
  R_0 = S
  X = 0
  loop
    X_{i+1} = Refresh X
    R_{i+1} = GroupPageRank(R_i, X_{i+1})
    Compute Y_{i+1} and send it to other nodes
    Wait for some time
  while true
}
```

Algorithm 3: Distributed PageRank Algorithm: DPR1

The difference between algorithm DPR1 and DPR2 lies in the style and frequency of refreshing input vector X and updating output vector Y . In each loop of algorithm DPR1,

new value of R is computed iteratively (by algorithm 2) until converge before updating and sending Y to other groups. While with DPR2, each node always uses the latest X it can be acquired to compute R and update the value of Y eagerly.

```

function DPR2() {
   $R_0 = S$ 
   $X = 0$ 
  loop
     $X_{i+1} = \text{Refresh } X$ 
     $R_{i+1} = AR_i + \beta E + X_{i+1}$ 
    Compute  $Y_{i+1}$  and send it to other nodes
    Wait for some time
  while true
}

```

Algorithm 4: Distributed PageRank Algorithm: DPR2

4.3. Convergence Analysis

Before analyzing convergence of the algorithms, we first give two interesting results for distributed PageRank (refer to Appendix for proof details):

Theorem 4.1 For a static link graph, sequence $\{R_1, R_2, \dots\}$ in algorithm DPR1 is monotonic for all nodes.

Theorem 4.2 For a static link graph, sequence $\{R_1, R_2, \dots\}$ in algorithm DPR1 has upper bound for all nodes.

As every bounded monotonic sequence converges, by theorem 4.1 and 4.2, algorithm DPR1 can converge.

Theorem 4.1 and 4.2 also holds for DPR2 if $R_0=0$. This can be proved similarly by viewing each page as a group.

For convenience of proof, we presume that the link-graph is **static** (no link/node insertion and deletion), and also assume $S=0$ for DPR2. However, we believe the two algorithms *DO* converge without these constrains (although Theorem 4.1 and 4.2 don't hold anymore with *dynamic* link graph).

Can the two algorithms converge to the same vector as centralized page ranking algorithm? The answer is "Yes", according to our experiments.

4.4. Reducing Communication Overhead

When web pages are partitioned into groups and mapped onto page rankers, each group *potentially* has links pointing to nearly *all* other groups, which causes one-to-one communication. Figure 3 shows some fictitious nodes (represented by small circles) and part of the communications (arrowed-lines) between them. We call this kind of communication as *direct transmission*. Given N as the total number of page rankers, although the allowing of asynchronous operations in each machine can reduce communication overhead in some degree, $O(N^2)$ messages are still needed to be transmitted between nodes per iteration. That is essentially not scalable.

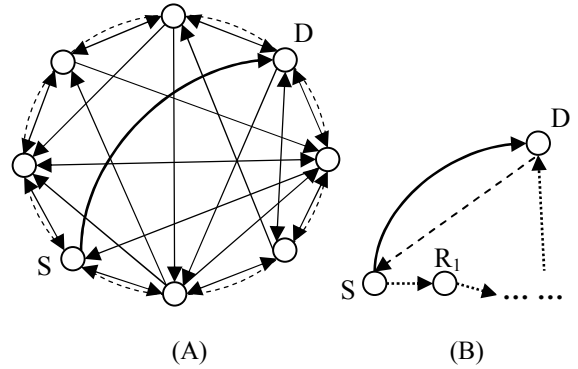


Fig.3. Direct transmission in performing distributed page ranking. (A) The communication is nearly one-to-one with direct transmission. (B) Finding the IP address and port of the destination by using lookup operations in structured P2P networks.

Moreover, if the number of page rankers is large (i.e. more than 1000), it is impossible to have one node knowing all the other nodes. Actually, in P2P networks [6, 13, 14, 15], one node commonly has roughly some dozens of neighbors. When one source node S wants to send a message to a destination D , it must know the IP address and port of D first. This is implemented by a lookup message in structured P2P networks, see (B) of Figure 3. Assuming averagely h hops are needed for a lookup, these lookup messages increase the communication overhead up to $O(hN^2)$.

For each message, it has to go through the network stack at the sender and the receiver. Thus it is copied to and from kernel space twice, incurring two context switches between the kernel and the user mode.

To reduce the number of messages, we provide an alternative way to achieve scalable communication: *indirect transmission*. With indirect transmission, updated page scores are *not* sent to their destinations directly, instead, they are transferred several times before get to their destinations. In other words, indirect transmission uses the routing path of structured P2P networks to transfer data --- something opposite to the spirit of P2P. Figure 4 shows the key idea of indirect transmission. In the figure, node B need to transmit updated page scores to other machines, instead of sending data to all destinations directly (after finding the IP addresses of the destinations), it packs the data into packages and send them to its neighbors respectively. When a machine A receives some packages (from its neighbors B, C, D , and E), it unpacks them, recombines the data in them according to their destinations, and forms new packages. Then these new packages are sent to each neighbor of A . As a result, data containing page scores reach the destination after a series of packing and unpacking. Figure 5 shows the communication pattern between nodes using indirect transmission. We can see that, use indirect transmission

scheme, data are transferred only between neighbors. Hence, only $O(N)$ messages are needed per iteration. However, as messages are sent indirectly, much bandwidth may be consumed. Assume it takes averagely h hops to route a message to its destination, total bandwidth consumed can be $O(hN)$.

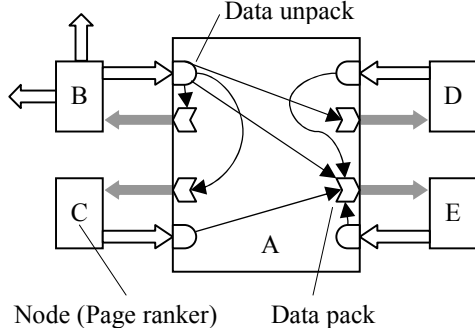


Fig.4. Explanation for indirect transmission. Data are unpacked and recombined on each node.

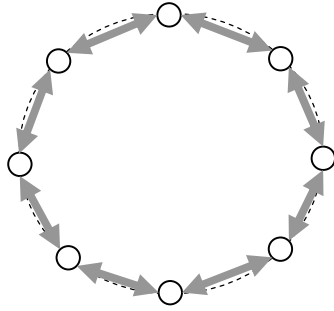


Fig.5. Communication between nodes using indirect transmission. Assume each node has two neighbors here.

To compare these two kinds of communication pattern, assume there are N nodes responsible for the ranking computation of W web pages. Using the “hash-by-site” strategy in section 4.1, one page has only about 1 URL pointing to other sites [16]. Define l as the average size of one link, and r as the average size of a lookup message for a destination node. Assume it takes averagely h hops to route a message to its destination. Considering an iteration of the DPR1 or DPR2 algorithm, with indirect transmission, the size of data should be transferred between nodes is roughly:

$$D_{it} = hlW \quad (4.1)$$

Whereas with direct transmission, the size of data transferred is about:

$$D_{dt} = lW + hrN^2 \quad (4.2)$$

Formula 4.2 is because a node must know the IP addresses and ports of destinations before sending updated page scores to them. So some lookup messages must be sent first, as shown in Figure 3 (B).

Now consider the number of messages. With indirect transmission, the average number of messages per iteration is:

$$S_{it} = gN \quad (4.3)$$

Here g is the average number of neighbors per node. While with direct transmission, the average number of messages per iterations is roughly:

$$S_{dt} = (h+1)N^2 \quad (4.4)$$

From the above four formulas, we can see that indirect transmission is more scalable than direct transmission, in terms of the size of data and the number of messages transferred. Direct transmission seems better only for small N .

4.5 Convergence Time vs. Bandwidth

We analyze the relationship between convergence time and bandwidth consumed in this section. Only indirect transmission is considered here.

Consider an example of computing the page ranking of 3 billion (Google indexes more than 3 billion web documents [18]) web pages over 1000 page rankers. That is, we have $W=3GB$ and $N=1000$ in formula 4.1 and 4.2. Define T as the minimal time interval between two iterations.

Link information exchange between page rankers has format of $\langle url_from, url_to, score \rangle$, which means that an URL url_from with ranking $score$ has an outlink to URL url_to . Given an average URL size of 40 bytes [16], the average size of one link is roughly 100 bytes. So we have:

$$L = 100 \text{ bytes} \quad (4.5)$$

The communication overhead should not exceed the capacity of the internet and upstream/downstream bandwidth of page rankers themselves. So we consider the following two constrains:

Bisection Bandwidth: One way to estimate the internet’s capacity is to look at the backbone cross-section bandwidth. The sum of bisection bandwidth of Internet backbones in the U.S. was about 100 gigabits in 1999 [17]. That is used by [17] to estimate the feasibility of peer-to-peer web indexing and searching. We also use it as our internet bisection bandwidth constrain. Assume one percent of the internet bisection bandwidth is allowed to be used by page ranking, that is, 1 gigabit, or 100MB per second.

Upstream/Downstream Bandwidth: Each node has an upstream and downstream bottleneck bandwidth when it connects to the internet. Data transfer should not exceed bottleneck bandwidth of nodes.

According to the bisection bandwidth constrain, we have:

$$D_{it} = hlW < T * 100MB / s \quad (4.6)$$

For Pastry [6] with 1000 nodes, the average number of hops is about 2.5. Thus we have $T > 7500s$ from formula 4.6. That means, with distributed page ranking, the time interval between two iterations is at least 2 hours.

Now consider the second constrain. Define B as the bottleneck bandwidth of each node, we have:

$$\frac{D_{it}}{N} < TB \quad (4.7)$$

With $T=7500s$, we have $B \geq 100KB$.

Table 1 shows some minimal time intervals between iterations for different number of page rankers. Notice that for Pastry with 10,000 and 100,000 nodes, the average number of hops h is about 3.5 and 4.0 respectively [6]. The minimal node bottleneck bandwidth needed for different number of nodes is also showed in Table 1.

Some techniques can be adopted to reduce convergence time, i.e. compression. This problem is left as future work.

Table.1. The minimal time interval between iterations and the minimal node bottleneck bandwidth needed for distributed page ranking

# of Page Rankers	1,000	10,000	100,000
Time per Iteration	7500s	10500s	12000s
Bottleneck Bandwidth Needed	100KB/s	10KB/s	1KB/s

5. Experiments

We run a simulator to verify the discussion in previous sections.

Datasets The link graph adopted for experiments is generated from Google programming contest data [3] which includes a selection of HTML web pages from 100 different sites in the “edu” domain. This link graph contains nearly 1M pages with overall 15M links. Although slightly small, this is the largest *real* dataset can be obtained by us now.

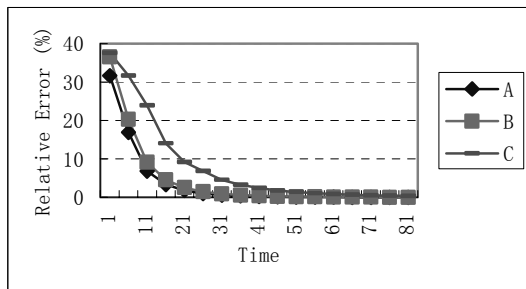


Fig.6. Distributed PageRank converges to the ranks of centralized PageRank. ($K=1000$. A: $p=1, T_1=0, T_2=6$; B: $p=0.7, T_1=0, T_2=6$; C: $p=0.7, T_1=0, T_2=15$).

Experiment Setup To simulate the asynchronism of computation on different nodes, each group u waits for $T_w(u, m)$ time units before starting a new loop step m . In our experiment, $T_w(u, m)$ follows exponential distribution for a fixed u , and the mean waiting time of each page group are randomly selected from $[T_1, T_2]$ (T_1 and T_2 are parameters that can be adjusted). To simulate potential network failures, we assume vector Y may fail to be sent to other groups with a probability p . We run the simulation many times with different values of T_1, T_2, p and K (here K is the number of page groups or page rankers).

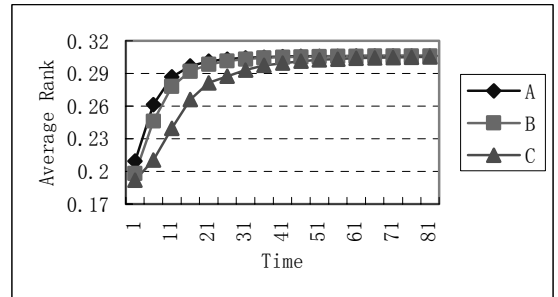


Fig.7. Rank sequence generated by DPR1 is monotonic. ($K=100$. A: $p=1, T_1=0, T_2=6$; B: $p=0.7, T_1=0, T_2=6$; C: $p=0.7, T_1=0, T_2=15$).

Let R, R^* be ranks obtained by distributed PageRank and its centralized counterpart, define the relative error as $\frac{\|R-R^*\|}{\|R^*\|}$. We use relative error as a metric for the difference between them. Figure.6 shows that the relative error decreases over time.

Figure.7 shows the monotonic property of rank sequence generated by DPR1. Notice that the average rank is only 0.3 when converges. That is because a large proportion of links point to outside of the dataset (only 7M of the whole 15M links point to pages in the dataset).



Fig.8. Comparison between different page ranking algorithms. CPR means centralized page ranking. The threshold relative error is 0.01%. ($p=1, T_1=15, T_2=15$).

Figure 8 shows the convergence of different page ranking algorithms. We can see that DPR1 converges

more quickly than DPR2. DPR1 even need fewer iteration steps than the centralized page ranking algorithm to converge. Another conclusion seen from the figure is that the number of page rankers has little effect on the converge speed.

6. Related Works

In addition to the two seminal algorithms [1, 2] using link analysis for web search, much work has been done on the efficient computation of PageRank [4, 8], using PageRank for personalized or topic-sensitive web search [5, 9], utilizing or extending them for other tasks [10, 11], etc. To our knowledge, there has no discussion till now about distributed page ranking in public published materials.

Another kind of related work may be parallel methods of solution of linear equation systems for computers with multiprocessors. There are two ways of solving the linear system which can both be parallelized: direct methods and iterative methods. Most of the methods are not suitable to solve our problem because they require matrix inversions that are prohibitively expensive for a matrix of the size and sparsity of the web-link matrix. Please see [12] for details of them.

7. Conclusions and Future Work

Distributed page ranking are needed because size of the web grows at a remarkable speed and centralized page ranking is not scalable. PageRank can be modified slightly for open systems. To do page ranking distributedly, pages can be partitioned by hash code of their websites. Distributed PageRank converges to the ranks of centralized PageRank. Indirect transmission can be adopted to achieve scalable communication. The convergence time is judged by network bisection bandwidth and the bottleneck bandwidth of nodes.

Future works include: Doing more experiments (and using larger datasets) to discover more interesting phenomena in distributed page ranking. And explore more methods for reducing communication overhead and convergence time.

References

- [1] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In Proceedings of the Ninth Annual ACM/SIAM Symposium on Discrete Algorithms. San Francisco, California, January 1998.
- [2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford University Database Group, 1998.

- [3] <http://www.google.com>
- [4] T. H. Haveliwala. Efficient computation of PageRank. Stanford University Technical Report, 1999.
- [5] G. Jeh and J. Widom. Scaling personalized web search. Stanford University Technical Report, 2002.
- [6] Rowstron, A. and P. Druschel. Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems. in IFIP/ACM Middleware. 2001. Heidelberg, Germany.
- [7] Owe Axelsson. Iterative Solution Methods. Cambridge University Press. 1994
- [8] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, etc. Extrapolation Methods for Accelerating PageRank Computations. Stanford University Technical Report, 2002.
- [9] T. H. Haveliwala. Topic-sensitive PageRank. In Proceedings of the Eleventh International World Wide Web Conference, 2002.
- [10] D. Rafiei and A.O. Mendelzon. What is this page known for? Computing web page reputations. In Proceedings of the Ninth International World Wide Web Conference, 2000.
- [11] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. In Proceedings of the Eighth International World Wide Web Conference, 1999.
- [12] Vipin Kumar, Ananth Grama, etc. Introduction to Parallel Computing, Design and Analysis of Algorithms. The Benjamin/Cummings Publishing Company.
- [13] Ratnasamy, S., et al. A Scalable Content-Addressable Network. in ACM SIGCOMM. 2001. San Diego, CA, USA.
- [14] Stoica, I., et al. Chord: A scalable peer-to-peer lookup service for Internet applications. in ACM SIGCOMM. 2001. San Diego, CA, USA.
- [15] Zhao, B. Y., Kubiawicz, J.D., and Josep, A.D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, UC Berkeley, EECS, 2001.
- [16] Junghoo Cho and Hector Garcia-Molina. Parallel crawlers. In Proc. of the 11th International World-Wide Web Conference, 2002.
- [17] Jinyang Li, Boon Thau Loo, Joseph M. Hellerstein, M. Frans Kaashoek, David R. Karger and Robert Morris. On the Feasibility of Peer-to-Peer Web Indexing and Search. In Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), 2003
- [18] Google Press Center: Technical Highlights. <http://www.google.com/press/highlights.html>.

Appendix

Notations: For a vector r , define $r \geq 0$ if and only all elements of it are larger than or equal to zero. For a matrix A , define $A \geq 0$ if and only if all elements of it are larger

than or equal to zero. For two vectors r_1 and r_2 , define $r_1 \geq r_2$ if and only if each element of r_1 is larger than or equal to the corresponding element of r_2 .

Lemma 1 For a square matrix $A \geq 0$, and a vector $f \geq 0$.

If $\|A\|_\infty < 1$ and $r = Ar + f$, then $r \geq 0$

Proof: Let k be dimension of A, f , and r . Assume r_0 is the smallest element of r with no loss of generality. If the lemma doesn't hold, then $r_0 < 0$, so

$$r_0 = \left(\sum_{i=1}^k A_{0i}r_i\right) + f_0 \geq r_0 \left(\sum_{i=1}^k A_{0i}\right) + f_0 > r_0 + f_0$$

A contradiction! So the lemma holds.

Lemma 2. Given a square matrix $A \geq 0$ and two vectors $f_1 \geq 0, f_2 \geq 0$, if $\|A\|_\infty < 1, r_1 = Ar_1 + f_1$ and $r_2 = Ar_2 + f_2$, then: $f_1 \geq f_2 \Rightarrow r_1 \geq r_2$

Proof: From $r_1 = Ar_1 + f_1$ and $r_2 = Ar_2 + f_2$ can get:

$$(r_1 - r_2) = A(r_1 - r_2) + (f_1 - f_2)$$

We get $r_1 - r_2 \geq 0$ by theorem 1, so $r_1 \geq r_2$.

Proof of Theorem 4.1

Theorem 4.1 For a static link graph, sequence $\{R_1, R_2, \dots\}$ in algorithm DPR1 is monotonic for each node.

Proof: We define $R_{u,i}$ as rank vector R_i on node (or page group) u , and define $R_{u,i}(j)$ as the j 'th element of $R_{u,i}$. $X_{u,i}$ and $Y_{u,i}$ are defined similarly. Define $t_r(u,i)$ as the time when value of $R_{u,i}$ is computed. Similarly define $t_x(u,i)$ and $t_y(u,i)$. Then we need only to prove that for any page group u and integer m , if $m > 0$, then

$$R_{u,m} \leq R_{u,m+1} \quad (*)1 \quad \text{and} \quad X_{u,m} \leq X_{u,m+1} \quad (*)2$$

If (*)2 is proved, by the following statement (#1), formula (*)1 will be proved either. Now we focus on the proof of statement (*)2.

For any page group u and integer m , by lemma 2, we have

$$X_{u,m} \leq X_{u,m+1} \Rightarrow R_{u,m} \leq R_{u,m+1}, Y_{u,m} \leq Y_{u,m+1} \quad (\#1)$$

And its equivalent statement:

$$(\exists j, s.t. Y_{u,m}(j) > Y_{u,m+1}(j)) \Rightarrow \exists i, s.t. X_{u,m}(i) > X_{u,m+1}(i) \quad (\#2)$$

$$(\exists j, s.t. R_{u,m}(j) > R_{u,m+1}(j)) \Rightarrow \exists i, s.t. X_{u,m}(i) > X_{u,m+1}(i)$$

Formula (#1) implies that, for any group, high rank value of afferent links means high page ranks and high scores of efferent links. Now we prove by contradiction. Assume that formula (*)2 doesn't hold for a page group u_1 , that is, there exist a page with index j and an integer $m_1 > 0$, such that $X_{u_1, m_1}(j) > X_{u_1, m_1+1}(j)$. As the value of $X(j)$ comes from efferent links of other groups, there must have a group u_2 with page i and iteration step m_2 , such that

$Y_{u_2, m_2}(i) > Y_{u_2, m_2+1}(i)$. Note that $m_2 > 0$ and $t_y(u_2, m_2 + 1) < t_x(u_1, m_1 + 1)$. Therefore, by formula (#2), we see that formula (*)2 doesn't hold for page group u_2 and integer m_2 . Moreover, we have:

$$t_r(u_2, m_2 + 1) < t_y(u_2, m_2 + 1) < t_x(u_1, m_1 + 1) < t_r(u_1, m_1 + 1)$$

Repeat the above process, we get two infinite sequences: $\{u_1, u_2, \dots\}, \{m_1, m_2, \dots\}$ satisfying the following formula:

$$t_r(u_1, m_1 + 1) > t_r(u_2, m_2 + 1) > \dots$$

The above statement implies (u_i, m_i) and (u_j, m_j) are different states for any $i \neq j$, that is, there are infinite states before (u_i, m_i) . But there can't be infinite times of iterations up to a certain time, a contradiction! Therefore, formula (*)2, and so formula (*)1, holds for any page group u .

Proof of Theorem 4.2

Theorem 4.2 For a static link graph, sequence $\{R_1, R_2, \dots\}$ in algorithm DPR1 has upper bound for each node.

Proof: Define $R_{u,i}, X_{u,i}, Y_{u,i}, t_r(u,i), t_x(u,i), t_y(u,i)$ etc as in the proof of theorem 4.1. In addition, define R_u^* as the ultimate rank vector of group u if centralized PageRank is performed on all the page groups (instead of on each page group respectively). And define X_u^* and Y_u^* similarly. Then we need only to prove that for any page group u and integer m , if $m > 0$, then

$$R_{u,m} \leq R_u^* \quad (*)1 \quad \text{and} \quad X_{u,m} \leq X_u^* \quad (*)2$$

As in the proof of Theorem 4.1, we just need to prove (*)2.

For any page group u and integer $m > 0$, because $R_{u,m} = AR_{u,m} + \beta E + X_{u,m}$ and $R_u^* = AR_u^* + \beta E + X_u^*$, by lemma 2 we have

$$X_{u,m} \leq X_u^* \Rightarrow R_{u,m} \leq R_u^*, Y_{u,m} \leq Y_u^* \quad (\#1)$$

And its equivalent statement:

$$(\exists j, s.t. Y_{u,m}(j) > Y_u^*(j)) \Rightarrow \exists i, s.t. X_{u,m}(i) > X_u^*(i) \quad (\#2)$$

$$(\exists j, s.t. R_{u,m}(j) > R_u^*(j)) \Rightarrow \exists i, s.t. X_{u,m}(i) > X_u^*(i)$$

Prove by contradiction. Assume formula (*)2 doesn't hold for page u_1 and integer $m_1 > 0$, then, use the same process as in the proof of theorem 4.1, we get two infinite sequences $\{u_1, u_2, \dots\}$ and $\{m_1, m_2, \dots\}$ satisfying the following formula:

$$t_r(u_1, m_1 + 1) > t_r(u_2, m_2 + 1) > \dots$$

We can get a contradiction by the same reasoning as in the proof of theorem 4.1. Thus, theorem 4.2 is proved.

² Because, by algorithm, $Y_{u,0}$ is never sent to other groups.