Distributed PageRank Computation Based on Iterative Aggregation-Disaggregation Methods

Yangbo Zhu
Department of Electronic Engineering
Tsinghua University
Beijing 100084, China
zhuyangbo99@mails.tsinghua.edu.cn

Shaozhi Ye[*]
Department of Computer Science
University of California, Davis
CA 95616, USA
sye@ucdavis.edu

Xing Li
Department of Electronic Engineering
Tsinghua University
Beijing 100084, China
xing@cernet.edu.cn

ABSTRACT

PageRank has been widely used as a major factor in search engine ranking systems. However, global link graph information is required when computing PageRank, which causes prohibitive communication cost to achieve accurate results in distributed solution. In this paper, we propose a distributed PageRank computation algorithm based on *iterative aggregation-disaggregation* (IAD) method with *Block Jacobi* smoothing. The basic idea is divide-and-conquer. We treat each web site as a node to explore the block structure of hyperlinks. Local PageRank is computed by each node itself and then updated with a low communication cost with a coordinator. We prove the global convergence of the *Block Jacobi* method and then analyze the communication overhead and major advantages of our algorithm. Experiments on three real web graphs show that our method converges 5–7 times faster than the traditional Power method. We believe our work provides an efficient and practical distributed solution for PageRank on large scale Web graphs.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Algorithms, Performance

Keywords

PageRank, distributed search engines, iterative aggregation-disaggregation, Block Jacobi

[*]This work was conducted when this author was with Tsinghua University.

1. INTRODUCTION

The World Wide Web keeps growing. In April 2005, Google[1] announced to have indexed about 8 billion web pages. Only several giants can afford the prohibitive cost for maintaining and updating the index of billions of pages. Moreover, a considerable part of the high-quality Deep Web, which is estimated about 500 times larger than the static Web [20], is exclusive from crawlers.

Motivated by above reasons, distributed and collaborative search engines have been extensively studied. In a typical distributed search system, each node maintains the index of local-stored pages. Usually there are also some nodes serving as coordinators to provide global information for the other nodes.

A major challenge to distributed search engines is how to rank the query results on different nodes. The ranking factors in web search can be divided into two categories. The first is content-based relevance from traditional information retrieval, which can be easily handled by each node itself. The second is link-based authority rising in recent years. Among the most popular link analysis algorithms are PageRank [26] and HITS [16], both of which have been demonstrated to be successful in many web information retrieval applications, especially for large scale web search.

However, despite of their simple forms, both PageRank and HITS require the knowledge of the whole link graph for computation, which causes prohibitive communication overhead to achieve accurate results for distributed computation on large graphs. For example, even after compression, a web graph consisting of 118M vertices and 1G edges is 385M Bytes large [1]. Extensive studies have been conducted to shape PageRank suitable for distributed computation [7, 31].

In this paper, we propose a distributed PageRank computation (DPC) algorithm. The general idea is divide-and-conquer. We treat each web site as a node to make use of the underlying block structure of the Web [12]. Each node computes a PageRank vector for its local pages by links within sites and then updates its local PageRank through low volume communication with a given coordinator.

In a mathematical perspective, we prove that the DPC algorithm is equivalent to the classic *iterative aggregation-*

[1]http://www.google.com

disaggregation (IAD) method with *Block Jacobi* smoothing. We further present the proof for global convergence of the *Block Jacobi* method and make thorough analysis on the communication overhead and major advantages of our algorithm.

We use three real web graphs with several ten million vertices in our experiments. L_1 distance and Kendall's τ-distance are adopted for evaluation. The experimental results show that the DPC algorithm achieves better approximation than a recent work in [31]. And it converges 5–7 times faster than the traditional Power method. We believe that our work provides an efficient and practical distributed solution for PageRank computation on large scale Web graphs.

The remainder of this paper is organized as follows. Section 2 briefly reviews the PageRank algorithm. Section 3 proposes our DPC algorithm with theoretical analysis of its convergence properties and communication overhead. The experimental results on three real web graphs with several ten million pages are presented in Section 4. We discuss related work in Section 5 and conclude the whole paper with Section 6.

2. THE BASIC PAGERANK MODEL

PageRank has emerged as one of the dominant models for exploring link structures, partly due to its query-independence and immunity to spamming. We briefly review PageRank algorithm in this section. A reader familiar with PageRank may skip this section.

2.1 The Intuition

The basic idea of PageRank assumes that a link from page A to page B indicates that the author of A recommends page B. Thus a page linked by many other pages is important. Furthermore, a page linked by an important page is also important.

To formulate the original intuition with a mathematical model, the Web is viewed as a directed graph G with web pages as vertices, and hyperlinks as edges. Then PageRank is the stationary distribution of a random walk on the graph. In this random walk, a user visits web pages following hyperlinks or jumps to a random page with certain probability. Such a random walk is essentially a homogeneous first-order Markov chain.

Before we formulate the PageRank algorithm, we introduce here some notations used in this paper. $\|v\|_1$ denotes the 1-norm of vector v. $\rho(M)$ denotes the *spectral radius*[2] of matrix M. We say $M > a$ if and only if $M_{ij} > a, \forall i, j$. Other binary relations (e.g. $\geq$, $\leq$, $<$, $=$) between a matrix and a scalar are defined likewise. $e = (1, \ldots, 1)^T$ is the uniform vector and I is the identity matrix. The size of e and I changes according to the context.

Assume the transition probability matrix of a random walk on directed graph G is P. Let N be the number of states in the Markov chain. π denotes the stationary probability vector of P. So π satisfies

$$\pi = P\pi \tag{2.1}$$

Thus, π is the principle eigenvector of P, corresponding to eigenvalue one.

[2]*Spectral radius* is the largest module of eigenvalues of M.

2.2 Technical Issues

A homogeneous finite Markov Chains has a unique positive stationary probability distribution if and only if it is irreducible and aperiodic. However, the existence of dangling nodes[3] makes the chain reducible. A simple remedy is to modify the model that when random walkers reach a dangling node, they pick a random page for the next state. Suppose that random walkers follow links with a probability d and jump to a random page with a probability $1 - d$. d is called the damping factor, which is 0.85 in this paper. Then the transition matrix satisfies

$$P_{ij} = \begin{cases} d/C_j + (1-d)/N & \text{if } j \to i \\ (1-d)/N & \text{if } j \nrightarrow i \text{ and } C_j \neq 0 \\ 1/N & \text{if } C_j = 0 \end{cases} \tag{2.2}$$

where C_j denotes the out-degree of page j. $j \to i$ denotes there is a link from j to i, and $j \nrightarrow i$ denotes there is no link from j to i. Finally we have $P \geq (1-d)/N$, and the existence of π is guaranteed.

A fairly straightforward way to obtain π is through Power methods [8], which employs iterative multiplication as follows:

$$\pi^{k+1} = P\pi^k \tag{2.3}$$

The eigenvector problem in (2.1) can also be formulated as a linear system:

$$(I - P)\pi = 0, e^T\pi = 1 \tag{2.4}$$

There are many alternative solutions for the linear system, such as *Jacobi* method, *Gauss-Seidel* method, *Successive Overrelaxation* method (SOR), *Symmetric Successive Overrelaxation* method (SSOR). There is a unified formulation for these algorithms. Split the coefficient matrix as $I - P = M - N$, where M is *nonsingular* and the splitting is *weak regular*[4]. Let $T = M^{-1}N$ be the iteration matrix. The general PageRank algorithm can be written as:

Algorithm 1 (PageRank(P, π^0, ϵ)).

Step 1. Let the initial approximation be π^0. Set $k = 0$.
Step 2. Compute

$$\tilde{\pi}^{k+1} = T\pi^k \tag{2.5}$$

Step 3. Normalize

$$\pi^{k+1} = \tilde{\pi}^{k+1}/\|\tilde{\pi}^{k+1}\|_1 \tag{2.6}$$

If $\|\pi^{k+1} - \pi^k\| < \epsilon$, quit with x^{k+1}. Otherwise, continue with Step 2 with k increased by 1.

Note that when $M = I$ and $N = P$, the iteration matrix is identical to that of Power method. Refer to [18] for a comprehensive review of PageRank.

3. DISTRIBUTED PAGERANK COMPUTATION

In this section, we propose our distributed PageRank computation (DPC) algorithm.

[3]A dangling node is a page with zero out-degree, i.e. an absorbing state.
[4]If $M^{-1} \geq 0$ and $M^{-1}N \geq 0$, the splitting is called *weak regular* [21].

3.1 The Basic Idea

The basic idea of our algorithm is divide-and-conquer. Each node in the distributed system computes PageRank vector for local pages. Unlike parallel computation algorithms performed by a cluster of machines connected with gigabit Ethernet [7], distributed algorithms require simple mechanism of interaction between nodes and low volume of communication traffic. Taking these constrains, we propose our Distributed PageRank Computation (DPC) algorithm here.

The web link graph has a natural block structure: the majority of hyperlinks are intra-host ones [12]. Therefore, the random walk on the web can be viewed as a nearly completely decomposable (NCD) Markov chain [24]. This property opens the door for the *iterative aggregation-disaggregation* (IAD) methods [29]. Before presenting the DPC algorithm, we introduce the classic IAD methods here. The notations and terminologies adopted in this section follow those used in [22].

3.2 IAD Methods

Let G be a set of integers $\{1, \ldots, N\}$. Let $G_1, \ldots, G_n, n \leq N$ be the aggregated groups of elements in G. The sets $G_i, i = 1, \ldots, n$, are mutually disjoint and $\cup_{i=1}^{n} G_i = G$. Let N_i be the order of set G_i, i.e. the number of elements in G_i.

Let R be the $n \times N$ *aggregation matrix*, which satisfies

$$R_{ij} = \begin{cases} 1 & j \in G_i \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

We partition the positive vector π as $(\pi_1^T, \pi_2^T, \ldots, \pi_n^T)^T$ according to $\{G_i\}$. π_i is a subvector with dimension N_i.

Then we define the $N \times n$ *disaggregation matrix* $S(\pi)$ as follows:

$$\mathbf{S}(\pi) = \begin{pmatrix} S(\pi)_1 & 0 & \ldots & 0 \\ 0 & S(\pi)_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & S(\pi)_n \end{pmatrix} \tag{3.2}$$

where $S(\pi)_i = (\pi_i / \|(\pi_i\|_1)$ is a column vector denoting the *censored stationary distribution* of pages in node G_i. Note that $RS(\pi) = I$.

Let $T = M^{-1}N$ be a matrix arising from some splitting of $I - P = M - N$. To solve the linear system $(I - P)\pi = 0$, we can adopt the following algorithm:

Algorithm 2 (IAD method).
Step 1. Select a positive initial approximation π^0, $\|\pi^0\| = 1$. Set $k = 0$.
Step 2. Construct the aggregated matrix $RPS(\pi^k)$ and solve the linear system

$$RPS(\pi^k) z^k = z^k \tag{3.3}$$

where $\|z\| = 1$.
Step 3. Compute

$$\tilde{\pi}^{k+1} = TS(\pi^k) z^k \tag{3.4}$$

Step 4. Normalize

$$\pi^{k+1} = \tilde{\pi}^{k+1} / \|\tilde{\pi}^{k+1}\|_1 \tag{3.5}$$

If $\|\pi^{k+1} - \pi^k\| < \epsilon$, quit with π^{k+1}. Otherwise, the algorithm continues with Step 2 with k increased by 1.

3.3 DPC Algorithm

First, we define some notations for following discussion. The transition matrix P is partitioned into blocks according to $\{G_i\}$:

$$\mathbf{P} = \begin{pmatrix} P_{11} & P_{12} & \ldots & P_{1n} \\ P_{21} & P_{22} & \ldots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \ldots & P_{nn} \end{pmatrix} \tag{3.6}$$

We denote the ith block row with

$$\mathbf{P_{i*}} \triangleq (P_{i1}, \ldots, P_{in}) \tag{3.7}$$

and denote the ith block column with

$$\mathbf{P_{*i}} \triangleq \begin{pmatrix} P_{1i} \\ \vdots \\ P_{ni} \end{pmatrix} \tag{3.8}$$

Each diagonal block P_{ii} is square and stands for the intra-node link matrix of node G_i, while the off-diagonal blocks stand for the inter-node link structure. Moreover, the $n \times n$ aggregated matrix $A = RPS(\pi)$ is the transition matrix between nodes. It is straightforward that A satisfies irreducibility and aperiodicity when P does.

We now present our new algorithm:

Algorithm 3 (DPC algorithm).
Step 1. Each node G_i constructs its local transition matrix Q_i, which contains only the pages in G_i. Let the initial approximation be

$$\pi_i^0 = PageRank\ (Q_i, e/N_i, \epsilon) \tag{3.9}$$

Set $k = 0$.
Step 2. Construct the aggregated matrix $A^k = RPS(\pi^k)$. Solve the associated linear system.

$$z^k = PageRank\ (A^k, e/n, \epsilon) \tag{3.10}$$

This step can be called the solution on the coarse level.
Step 3. Each node G_i constructs an $(N_i + 1) \times (N_i + 1)$ extended local transition matrix

$$B_i^k = \begin{pmatrix} P_{ii} & (P_{i*}S(\pi^k)z^k - P_{ii}\pi_i^k z_i)/(1 - z_i^k) \\ e^T P_{*i} & \alpha^k \end{pmatrix} \tag{3.11}$$

where the scalar α^k ensures the column sum of B_i^k is one.
Compute the extended local PageRank vector

$$\begin{pmatrix} \omega_i^{k+1} \\ \beta_i^{k+1} \end{pmatrix} = PageRank\ (B_i^k, e/(N_i + 1), \epsilon) \tag{3.12}$$

where β_i^{k+1} is a scalar. This step can be called the smoothing on the fine granularity.
Before being sent to the center, the local vector is multiplied by a factor.

$$\tilde{\pi}_i^{k+1} = \frac{1 - z_i^k}{\beta_i^{k+1}} \omega_i^{k+1} \tag{3.13}$$

Step 4. Normalize

$$\pi^{k+1} = \tilde{\pi}^{k+1} / \|\tilde{\pi}^{k+1}\|_1 \tag{3.14}$$

If $\|\pi^{k+1} - \pi^k\| < \epsilon$, the algorithm quit with π^{k+1}. Otherwise, continue with Step 2 with k increased by 1.

REMARK 1. *The DPC algorithm is essentially equivalent to an IAD method with T being a Block Jacobi iteration matrix. Proof in exact arithmetic is given in* Appendix A.

3.4 Convergence Analysis

The IAD method was first proposed by Takahashi in 1975 [29]. It has been widely used to accelerate convergence of iterative methods for solving linear systems and minimization problems. After thirty years, the global convergence of IAD method is still an open problem. The difficulty partly comes from that the disaggregation step $S(\pi)z$ is *nonlinear* [4]. Some convergence properties of IAD method are analyzed in [22, 23, 27]. In order to justify the DPC algorithm to some extent, we prove the convergence of the *Block Jacobi* method in PageRank scenario.

First we present a lemma from [5]:

LEMMA 1. *The iteration scheme*

$$\pi^{k+1} = T\pi^k / \|T\pi^k\|_1 \tag{3.15}$$

converges when the following conditions are satisfied:
(C_1) $\rho(T) = 1$
(C_2) T is irreducible
(C_3) T is acyclic

Neumann and Plemmons [25] proves that the iteration matrix derived from any *weak regular* splitting of the matrix $I - P$ satisfies condition (C_1) and (C_2) if the matrix P is stochastic and irreducible.

Let D be the block diagonal of $I - P$. Let L be the block strictly lower triangular part of P, and U be the block strictly upper triangular part of P. Arising from the splitting $I - P = D - (L + U)$, the iteration matrix of *Block Jacobi* methods is

$$T = D^{-1}(L + U) \tag{3.16}$$

Since $(I - P_{ii})^{-1} \geq 0$ [24] and $(L + U) \geq 0$, the splitting above is *weak regular*. Because P is stochastic and irreducible, T satisfies (C_1) and (C_2).

However, the acyclicity of P is not sufficient to guarantee the acyclicity of T [14]. Fortunately, in the PageRank scenario we have the following lemma:

LEMMA 2. *If $P > 0$ is the transition matrix of a Markov chain and is partitioned according to* (3.6)*. Let T be the iteration matrix defined in* (3.16)*, i.e. the Block Jacobi matrix. T is acyclic if and only if $n > 2$.*

PROOF. See *Appendix B.* $\square$

Now (C_1), (C_2) and (C_3) in Lemma 1 are all satisfied when $n > 2$. Consequently, we have:

THEOREM 1. *If $P > 0$ is the transition matrix of a Markov chain and is partitioned according to* (3.6)*. Let T be the iteration matrix defined in* (3.16)*, i.e. the Block Jacobi matrix. If $n > 2$, The iterative scheme* (3.15) *always converges to the fix point $\hat{x}$ of $P\hat{x} = \hat{x}$*

Courtois [5] proves that when T is cyclic, the iteration scheme (3.15) ultimately converges to a vector composed of subvectors which are parallel to the corresponding subvectors of $\hat{x}$. Therefore $\hat{x}$ can be achieved by an additional single iteration of IAD methods.

3.5 Communication Overhead

In this section, we analyze the communication overhead of our algorithm. All messages are in the form of a vector, no matrix is transferred. Because the vector v is usually sparse, it is transferred as a stream of (index i, value $v_i > 0$) pairs. In practice, the index is a combination of the node ID and a hash value of the URL string. Let $\mathrm{Pos}(\cdot)$ denote the number of positive elements in a vector or a matrix. So the size of message is proportional to $\mathrm{Pos}(v)$. In practice, we use sparse matrix $\bar{P}$ instead of P. Let $\bar{L}$ and $\bar{U}$ be the block strictly lower and upper triangular part of $\bar{P}$ separately.

$$\bar{P}_{ij} = \begin{cases} d/C_j & \text{if } j \to i \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

Step 1 of DPC algorithm needs trivial communication.

In Step 2, node G_i sends the coordinator a vector $\bar{P}_{*i}\pi_i$, which is equal to the ith column of $\bar{P}S(\pi)$. Note that the ith subvector of $\bar{P}_{*i}\pi_i$ is sent as a scalar $e^T\bar{P}_{ii}\pi_i$. The amount of communication traffic is proportional to $\mathrm{Pos}((\bar{L} + \bar{U})S(\pi))$, which is much smaller than $\mathrm{Pos}(\bar{L} + \bar{U})$. Table 1 shows the comparison in real web graphs.

In Step 3, the coordinator sends the ith subvector of $\bar{P}S(\pi)z$ to node G_i. So the communication cost is $\mathrm{Pos}((\bar{L} + \bar{U})S(\pi)z)$, which is much smaller than N.

In Step 4, local nodes send the vector $\tilde{\pi}_i^{k+1}$ to the coordinator, who performs the normalization. The communication cost is $O(N)$.

To sum up, the entire communication overhead is of the magnitude $O\big(\mathrm{Pos}((\bar{L} + \bar{U})S(\pi))\big) + O(N)$, which is roughly equivalent to that of the LPR-Ref-2 algorithm in [31].

Table 1: Comparison of number of positive elements

	$\mathrm{Pos}(\bar{L} + \bar{U})$	$\mathrm{Pos}(\mathrm{Pos}((\bar{L} + \bar{U})S(\pi)))$
ST01	40M	8M(20.0%)
ST03	484M	165M(34.1%)
CN04	150M	35M(23.3%)

3.6 Advantages of DPC

The DPC algorithm has three major advantages over standard PageRank algorithm.

1. As most of the computation in DPC is solving PageRank vectors, many acceleration methods are ready to be used, such as the extrapolation method in [13].

2. The aggregated matrix A and the local transition matrices B_i are small enough to fit into main memory. Thus, the iterations require less disk I/O, which greatly accelerates the computation.

3. In DPC algorithm, the local PageRank vectors for many nodes converge quickly. In standard PageRank algorithm, the convergence rate is mainly determined by slow-converge nodes and much computation is wasted in recomputing the PageRank on the already-converged nodes [11]. Figure 1 shows the distribution of number of iterations for local PageRank computation using Power method in our experiments.

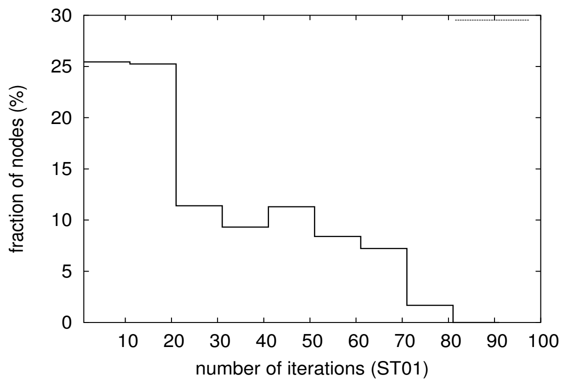

Figure 1: Histogram of distribution over number of iterations for local PageRank computation. The x-axis gives the number of iterations, and the y-axis shows the fraction of nodes completing their local PageRank computation within x iterations ($\epsilon = 1e^{-5}$).

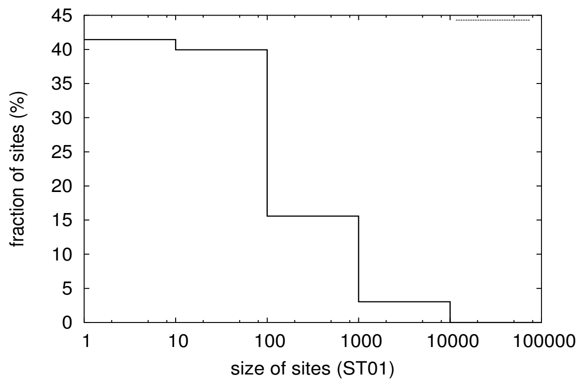

Figure 2: Histogram of distribution over size of sites. The x-axis gives the magnitude of the number of pages hosted by a site, and the y-axis shows the fraction of sites of the size.

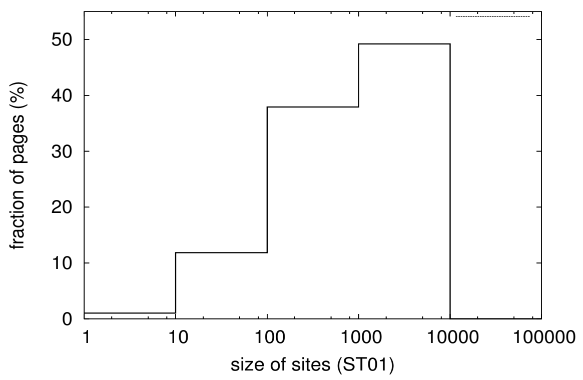

Figure 3: Histogram of distribution over number of pages hosted by sites of different size. The x-axis gives the magnitude of the number of pages hosted by a site, and the y-axis shows the fraction of pages hosted by all sites of that size.

4. EXPERIMENTAL RESULTS

We use three real web graphs to evaluate the proposed algorithm. We also implement the classic Power method and the LPR-Ref-2 algorithm in [31] for comparison.

4.1 Experimental Setup

The CN04 graph is from our two-week crawling in Aug. 2004, starting from thousands of well-known web sites in China. In order to obtain pages of high-quality, the crawl was performed in breath-first fashion.

The other two graphs, ST01[5] and ST03[6], come from Stanford WebBase project. Table 2 summarizes these three data sets. The three graphs vary in density of links and degree of inter-sites coupling. Figure 2 shows the distribution of the size of web sites, and Figure 3 shows the distribution of pages hosted by nodes of different size. Both figures are based on the data set ST01, the other two data sets have similar distributions.

Table 2: Characteristics of data sets

	ST01	ST03	CN04
number of URLs	65M	49M	88M
number of links	607M	1185M	485M
number of sites	542K	25K	697K
average out-degree	9.28	24.05	5.53
inter-node link	6.53%	40.85%	30.99%

The simulation is carried out on one machine. First, all URLs are sorted lexicographically. Anchors in the tail of URLs are ignored. For example, "http://a.edu/b.htm#c" and "http://a.edu/b.htm#d" are considered to be identical. Then we partition web pages into groups according to sites.

4.2 Evaluation Metrics

L_1 distance and Kendall's τ-distance [15] are adopted to evaluate our algorithm. Both measure certain kind of similarity between π and $\hat{\pi}$.

The first metric is the L_1 distance $\|\pi - \hat{\pi}\|_1$. We have $0 < \|\pi - \hat{\pi}\|_1 < 2$.

Then we introduce the Kendall's τ metric. Let $K(\pi, \hat{\pi})$ be an $N \times N$ matrix, whose elements are

$$K_{ij}(\pi,\hat{\pi}) = \begin{cases} 1 & \pi_i \geq \pi_j \text{ and } \hat{\pi}_i < \hat{\pi}_j \\ 1 & \pi_i < \pi_j \text{ and } \hat{\pi}_i \geq \hat{\pi}_j \\ 0 & otherwise \end{cases} \quad (4.1)$$

That is to say, $K_{ij}(\pi, \hat{\pi}) = 1$ if page i and j are in different order in π and $\hat{\pi}$.

Kendall's τ-distance is defined as follows:

$$\text{KDist}(\pi,\hat{\pi}) = \frac{\sum_{1\leq i<j\leq N} K_{ij}(\pi,\hat{\pi})}{N(N-1)/2} \quad (4.2)$$

where $0 \leq \text{KDist}(\pi, \hat{\pi}) \leq 1$.

4.3 Accuracy of a Single Iteration

The distributed computation heavily relies on the communication between nodes. When the network is under heavy

[5]ftp://db.stanford.edu/pub/webbase/Links2001.tar.gz
[6]ftp://db.stanford.edu/pub/webbase/Crawl-2003-04.tar.gz

load, it is possible for the iteration to be executed only once and wait for a long time to reconnect the coordinator. Then the accuracy of π^1 is virtually important. In fact, the LPR-Ref-2 algorithm proposed in [31] can be performed only once. Table 3 shows the L_1 distance between π^1 and $\hat{\pi}$.

Table 3: Accuracy after One Iteration ($\|\pi^1 - \hat{\pi}\|_1$)

	Power Method	LPR-Ref-2	DPC
ST01	0.539	0.207	0.023
ST03	0.523	0.477	0.124
CN04	0.140	0.146	0.014

It is costly to compute Kendall's τ-distance when N is large, because the computational complexity is $O(N^2)$. On a box with Intel Pentium 4 2.4GHz processor, it takes about 4 minutes to compute the Kendall's τ-distance between two vectors when $N = 10^5$. Since the N in our experiments are of the order of magnitude of 10^8, we have to adopt *Monte Carlo* method to estimate the true Kendall's τ-distance. KDist$(\pi, \hat{\pi})$ can be viewed as the probability of two randomly picked pages with different order in π and $\hat{\pi}$. So we can estimate the probability by random sampling. In practice, we pick 10^{10} random pairs as a sample. Repeated running of the *Monte Carlo* method empirically demonstrates that the variance of approximations is small. Thus, the sampling method is reliable. Table 4 presents the KDist$(\pi_1, \hat{\pi})$ of different methods.

Table 4: Accuracy of One Iteration (KDist$(\pi^1, \hat{\pi})$)

	Power Method	LPR-Ref-2	DPC
ST01	0.190	0.142	0.071
ST03	0.155	0.070	0.010
CN04	0.093	0.087	0.013

4.4 Convergence Rate

When the network condition is acceptable for communication, iterations of the DPC Algorithm are carried out until convergence is reached. Because the LPR-Ref-2 algorithm in [31] can be run only once, its convergence rate cannot be evaluated. Table 5 shows the number of iterations needed to achieve convergence. The DPC algorithm converges 5–7 times faster than Power Method. Figure 4 compares the convergence rate of Power method and that of DPC algorithm on graph ST01. The results on the other two graphs are similar.

Table 5: Number of Iterations for Convergence ($\epsilon = 10^{-5}$)

	Power Method	DPC
ST01	54	11
ST03	42	7
CN04	34	5

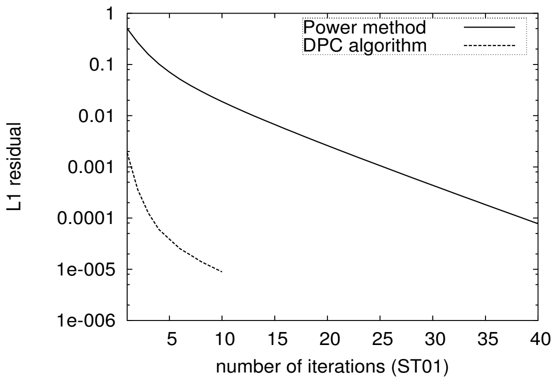

Figure 4: Convergence rate for Power method vs. DPC algorithm. The x-axis is the number of iterations, and the y-axis is the magnitude of L_1 residual ($\|\pi^{k+1} - \pi^k\|_1$).

5. RELATED WORK

Much work has been done on PageRank acceleration. Kamvar et al. [12] uses Step 1 and 2 of IAD to obtain an initial vector for subsequent iterations. Lee et al. [19] presents a fast PageRank algorithm which lumps dangling nodes into a single state. Broder et al. [2] proposes graph aggregation as an efficient PageRank approximation method. Eiron et al. [6] exploits the hierarchical structure on different levels of granularity to rank the web pages seen while not crawled, which they named as web frontier. Gleich et al. [7] implements a class of iterative algorithms for PageRank computation on a parallel computer. Langville and Meyer [17] employs a modified two-block IAD to accelerate the updating of PageRank vector. Ipsen and Kirkland [9] analyzes the asymptotic convergence rate of the method proposed in [17]. Wang and DeWitt [31] proposes a framework for distributed search system and a distributed algorithm for PageRank approximation.

The most relevant work to our algorithm is probably the one proposed by Vantilborgh [30]. Cao and Stewart [3] establishes conditions for local convergence of IAD with *Block Jacobi* smoothing. Stewart et al. [27] proposes an IAD method with *Block Gauss-Seidel* smoothing and establishes some regularity conditions to guarantee the convergence. Kafeety et al. [10] outlines a general framework for IAD. Recent work by Marek et al. [23] analyzes some local and global convergence properties of IAD.

6. CONCLUSION AND FUTURE WORK

This paper proposes a distributed PageRank computation algorithm based on *iterative aggregation-disaggregation* (IAD) methods with *Block Jacobi* smoothing. The basic idea is divide-and-conquer. We group web pages by sites to make use of the block structure of link graphs. To reduce the communication cost, nodes are organized in star topology. Each node computes its PageRank vector of local-stored pages and communicates with a coordinator for global updating information. We prove the global convergence of the *Block Jacobi* method and then analyze communication overhead of our algorithm. Three primary advantages of our algorithm are presented. Experiments on three real web graphs demonstrate that our method achieves better approximation than

LPR-Ref-2 in [31] and accelerates convergence by a factor of 5–7. We believe our work provides an efficient and practical distributed solution for PageRank on large scale Web graphs.

Several questions remain to be investigated in our future work:

1. How to update PageRank vectors efficiently within our framework?

2. Since $\rho(T) = 1$, is it possible to compute PageRank with asynchronous iterations [28]?

7. ACKNOWLEDGEMENTS

The authors are grateful to Stanford Database Group for sharing ST01 and ST03 data sets. We also thank Dr. Yuan Wang and Dr. David J. DeWitt for offering their data sets used in [31]. We would like to thank the anonymous reviewers for their insightful comments.

8. REFERENCES

[1] P. Boldi and S. Vigna. The webgraph framework i: compression techniques. In *Proc. of the WWW'04 Conf.*, pages 595–602, 2004.

[2] A. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient pagerank approximation via graph aggregation. In *Proc. of the WWW'04 Conf.*, pages 484–485, 2004.

[3] W. Cao and W. Stewart. Iterative aggregation/ disaggregation techniques for nearly uncoupled markov chains. *J. ACM*, 32(3):702–719, 1985.

[4] F. Chatelin. Iterative aggregation/disaggregation methods. In *Proc. of Int. Workshop on Mathematical Computer Performance and Reliability*, 1983.

[5] P. Courtois and P. Semal. Block iterative algorithm for stochastic matrices. *Linear Algebra and its Application, Vol 76*, pages 59–70, 1986.

[6] N. Eiron, K. McCurley, and J. Tomlin. Ranking the web frontier. In *Proc. of the WWW'04 Conf.*, pages 309–318, 2004.

[7] D. Gleich, L. Zhukov, and P. Berkhin. Fast parallel pagerank: A linear system approach. Technical report, Yahoo Corp., 2004.

[8] G. Golub and C. V. Loan. *Matrix computations (3rd ed.)*. Johns Hopkins Univ. Press, 1996.

[9] I. Ipsen and S. Kirkland. Convergence analysis of a pagerank updating algorithm by langville and meyer. *SIAM J. Matrix Anal. Appl., to appear*, 2004.

[10] H. Kafeety, C. Meyer, and W. Stewart. A general framework for iterative aggregation/ disaggregation methods. In *Proc. of the 4th Copper Mountain Conf. on Iterative Methods*, 1992.

[11] S. Kamvar, T. Haveliwala, and G. Golub. Adaptive methods for the computation of pagerank. Technical report, Stanford Univ., 2003.

[12] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford Univ., 2003.

[13] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Extrapolation methods for accelerating pagerank computations. *Proc. of the WWW'03 Conf.*, pages 261–270, 2003.

[14] L. Kaufman. Matrix methods for queueing problems. *SIAM J. Sci. Statist. Comput.*, 4(3):525–552, 1983.

[15] M. Kendall and J. Gibbons. *Rank Correlation Methods*. Edward Arnold, London, 5 edition, 1990.

[16] J. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[17] A. Langville and C. Meyer. Updating pagerank with iterative aggregation. In *Proc. of the WWW'04 Conf.*, 2004.

[18] A. Langville and C. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2005.

[19] C. Lee, G. Golub, and S. Zenios. A fast two-stage algorithm for computing pagerank and its extensions. Technical report, Stanford Univ., 2003.

[20] P. Lyman, H. Varian, J. Dunn, A. Strygin, and K. Swearingen. How much information project. Technical report, Univ. of California, Berkeley, 2003.

[21] I. Marek and P. Mayer. Iterative aggregation/ disaggregation methods for computing some characteristics of markov chains. In *Proc. of the Third Int. Conf. on Large-Scale Scientific Computing*, pages 68–80, 2001.

[22] I. Marek and P. Mayer. Convergence theory of some classes of iterative aggregation-disaggregation methods for computing stationary probability vectors of stochastic matrices. *Linear Algebra and Its Applications*, 363:177–200, 2003.

[23] I. Marek and I. Pultarova. A note on local and global convergence analysis of iterative aggregation -disaggregation methods. *Submitted to Linear Algebra and Applications*, 2005.

[24] C. Meyer. Stochastic complementation, uncoupling markov chains, and the theory of nearly reducible systems. *SIAM Review, Vol. 31, :2*, pages 240–272, 1989.

[25] M. Neumann and R. Plemmons. Convergent nonnegative matrices and iterative methods for consistent linear systems. *Numer. Math.*, 31:265–279, 1978.

[26] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Univ., 1998.

[27] G. Stewart, W. Stewart, and D. McAllister. A two stage iteration for solving nearly completely decomposable markov chains. *Recent Advances in Iterative Methods*, IMA Vol. Math. Appl. 60:201–216, 1993.

[28] D. Szyld. The mystery of asynchronous iterations convergence when the spectral radius is one. Research Report 98-102, Temple Univ., 1998.

[29] Y. Takahashi. A lumping method for numerical calculations of stationary distributions of markov chains. Technical Report B-18, Dept. of Information Sciences, Tokyo Institute of Technology, 1975.

[30] H. Vantilborgh. The error of aggregation in decomposable systems. Technical Report R453, Philipps Research Laboratory, Brussels, Belgium, 1981.

[31] Y. Wang and D. DeWitt. Computing pagerank in a distributed internet search engine system. In *Proc. of VLDB'04 Conf.*, pages 420–431, 2004.

APPENDIX

A. PROOF OF REMARK 1

Here we show that DPC Algorithm is essentially identical to an IAD method with Step 3 being a *Block Jacobi* smoothing.

Comparing Algorithm 3 with Algorithm 2, Step 1,2 and 4 are identical. Now we prove that the Step 3 of Algorithm 3 is equivalent to that of Algorithm 2 with $T = D^{-1}(L + U)$.

In Step 3 of Algorithm 2, from $\tilde{\pi}^{k+1} = D^{-1}(L+U)S(\pi^k)z^k$, we have

$$\begin{aligned}\tilde{\pi}_i^{k+1} &= (I - P_{ii})^{-1} \sum_{1\le j\le n, j\ne i} P_{ij}S_j z_j \\ &= (I - P_{ii})^{-1}(\sum_{1\le j\le n} P_{ij}S_j z_j - P_{ii}S_i z_i) \\ &= (I - P_{ii})^{-1}(P_{i*}S(\pi^k)z^k - P_{ii}\pi_i^k z_i) \quad \text{(A.1)}\end{aligned}$$

where $\tilde{\pi}_i$ is the ith subvector of $\tilde{\pi}$.

In Step 3 of Algorithm 3, from (3.12), we have:

$$\omega_i^{k+1} = \frac{1 - z_i^k}{\beta_i^{k+1}}(I - P_{ii})^{-1}(P_{i*}S(\pi^k)z^k - P_{ii}\pi_i^k z_i) \quad \text{(A.2)}$$

From (3.13), we have:

$$\begin{aligned}\tilde{\pi}_i^{k+1} &= \frac{\beta_i^{k+1}}{1 - z_i^k}\omega_i^{k+1} \\ &= (I - P_{ii})^{-1}(P_{i*}S(\pi^k)z^k - P_{ii}\pi_i^k z_i) \quad \text{(A.3)}\end{aligned}$$

Comparing (A.3) with (A.1), we conclude that Algorithm 3 is theoretically an IAD method with *Block Jacobi* smoothing.

B. PROOF OF LEMMA 2

Partition T into blocks according to G_i. The diagonal blocks $T_{ii} = 0$ and the off-diagonal blocks $T_{ij} > 0, i \ne j$. Consider T^2 as the square of T, which is also partitioned accordingly. When $n > 2$, $\forall i, j, \exists \hat{k}$ that satisfies $\hat{k} \ne i$ and $\hat{k} \ne j$. Consequently, we have

$$T_{ij}^2 = \sum_{1\le k\le n} T_{ik}T_{kj} \ge T_{i\hat{k}}T_{\hat{k}j} > 0 \quad \text{(B.1)}$$

It can be easily verified that $T^m > 0$ when $m \ge 2$. Thus T is acyclic.

When $n = 2$,

$$T^m = \begin{pmatrix} 0 & ((I - P_{11})^{-1}P_{12})^m \\ ((I - P_{22})^{-1}P_{21})^m & 0 \end{pmatrix} \quad \text{(B.2)}$$

when m is odd. And

$$T^m = \begin{pmatrix} ((I - P_{11})^{-1}P_{12})^m & 0 \\ 0 & ((I - P_{22})^{-1}P_{21})^m \end{pmatrix} \quad \text{(B.3)}$$

when m is even. Thus, T is cyclic.