Internet Mathematics Vol. 4, No. 4: 299–327

A Two-Stage Algorithm for Computing PageRank and Multistage Generalizations

Chris P. Lee, Gene H. Golub, and Stefanos A. Zenios

Abstract. The PageRank model pioneered by Google is the most common approach for generating web search results. We present a two-stage algorithm for computing the PageRank vector where the algorithm exploits the lumpability of the underlying Markov chain. We make three contributions. First, the algorithm speeds up the PageRank calculation significantly. With web graphs having millions of webpages, the speed-up is typically in the two- to three-fold range. The algorithm can also embed other acceleration methods such as quadratic extrapolation, the Gauss-Seidel method, or the Biconjugate gradient stable method for an even greater speed-up; cumulative speed-up is as high as 7 to 14 times. The second contribution relates to the handling of dangling nodes. Conventionally, dangling nodes are included only towards the end of the computation. While this approach works reasonably well, it can fail in extreme cases involving aggressive personalization. We prove that our algorithm is the generally correct way of handling dangling nodes using probabilistic arguments. We also discuss variants of our algorithm, including a multistage extension for calculating a generalized version of the PageRank model where different personalization vectors are used for webpages of different classes. The ability to form class associations may be useful for building more refined models of web traffic.

1. Introduction

The commercial success of the PageRank approach [Page et al. 98] for ranking webpages has spawned much interest in the research community. The approach is based on a Markov chain model for web traffic and is intuitive and highly versatile. However, the calculation of the rankings, known as the PageRank vector, poses a daunting computational challenge [Moler 02]: As the number of webpages to be ranked is in the billions, the computation is time-consuming and can take several days. Computation is repeated frequently to maintain the currentness and relevance of the results as well as to model the preferences of different web surfers [Jeh and Widom 03]. The need for accelerated algorithms is clear.

The PageRank vector is the limiting distribution of a homogeneous discrete-time Markov chain. An accelerated algorithm is presented in this paper for its computation. The algorithm exploits the "lumpability"[Kemeny and Snell 60] of the underlying Markov chain and proceeds in two stages. In the first stage, the *dangling nodes* [Page et al. 98] are combined into a single "block node" and the limiting distribution of the resulting Markov chain is computed; in the second stage, the *nondangling nodes* are combined into a node and the limiting distribution is computed. (Throughout the paper the words *node*, *state*, and *page* are used interchangeably.) The two limiting distributions are then pasted together to give the PageRank vector. Note that this is not an approximation algorithm: the vector obtained is identical to the ordinary PageRank vector. Our approach can dramatically reduce the computing time and is conceptually elegant.

Several papers discuss acceleration methods based on numerical linear algebra techniques. A Gauss-Seidel style algorithm has been proposed [Arasu et al. 02]. Another acceleration method proceeds by periodically subtracting an approximation of the subdominant eigenvectors, a procedure known as quadratic extrapolation [Kamvar et al. 03c]. The Arnoldi method tries to separate the first few eigenvectors and has been found to be particularly suitable for the parallel processor environment [Golub and Greif 06]. Kamvar et al. found that when sorted by url, the Markov chain matrix has a block structure which can be exploited to speed up the calculation [Kamvar et al. 03b]. For other proposals, see [Kamvar et al. 03a, Corso et al. 05, Langville and Meyer 06b, Ipsen and Kirkland 06, Boldi et al. 05, Langville and Meyer 04, Zhu et al. 05, Yates et al. 05, Corso et al. 04]. A particularly valuable aspect of our algorithm is that it can be used in combination with many of these methods.

Our paper contributes to the PageRank literature in several ways. By focusing on the probabilistic interpretations of the underlying Markov chain, we gain insights into new methods of acceleration. In each of the two stages, a different state space reduction technique is used: in the first stage, states are combined by the technique of *lumping*; in the second stage, *state aggregation* is used. These techniques come from the applied probability literature (see, for example, [Cao and Stewart 85, Meyer 89, Simon and Ando 61]). In addition, we show that the conventional approach of including dangling nodes only towards the end of the computation [Page et al. 98] can fail. Our algorithm is a more general way of handling dangling nodes.

We also discuss generalizations of our algorithm with two examples. In the first example, the PageRank vector is computed with a multistage algorithm. In the second example, we discuss dividing webpages into different classes. The division can be based on host, subject, content, language, and so on. Each class of webpages is modeled by a different personalization vector. A multistage variant of our algorithm can then be used to compute the PageRank vector. The ability to incorporate class-dependent information is versatile and can potentially provide a more refined model for web traffic.

Here is the standard notation that we use throughout the paper. A boldface letter indicates a matrix if it's in uppercase or a vector if it's in lowercase; scalars are in italics. We let $\mathbf{v}(i)$ denote the ith element of $\mathbf{v}$. Similarly, $\mathbf{M}(i,j)$ denotes the element in the ith row of the jth column of $\mathbf{M}$; $\mathbf{M}(i:j,k:l)$ denotes the elements in rows i through j and in columns k through l; $\mathbf{M}(i,:)$ denotes the ith row. All untransposed vectors are column vectors. The 1-norm of a vector is the sum of the absolute values of its elements: $||\mathbf{v}||_1 = \sum_i |\mathbf{v}(i)|$. The notation $\mathbf{1}_n$ means an n-dimensional vector of 1s; when the dimensionality can be deduced from the context, the subscript is dropped. The cardinality of the set $\mathbb{A}$ is written as card($\mathbb{A}$).

2. Review of the PageRank Model

Extensive surveys of PageRank and its related methods have been conducted by Langville and Meyer and by Berkhin [Langville and Meyer 06a, Langville and Meyer 05a, Langville and Meyer 05b, Berkhin 05].

The main idea behind the PageRank model is to regard web surfing as a Markov chain. Consider a collection of webpages $\mathbb{S} = \{1, 2, ..., N\}$ and a *personalization vector* $\mathbf{u} \in \mathbb{R}^{N\times 1}$ representing a generic surfer's preferences for these webpages. (Specifically, the personalization vector is assumed to have positive components that sum to one.) Let this surfer be currently at $i \in \mathbb{S}$. We assume that, in the next time step, the surfer will move to $j \in \mathbb{S}$ with the following probability:

$$\mathbf{Q}(i,j) = \begin{cases} \frac{\mathbf{G}(i,j)}{\sum_{l=1}^{N} \mathbf{G}(i,l)} & \text{if } \mathbf{G}(i,m) = 1 \text{ for some } m \in \mathbb{S}, \\ \mathbf{u}(j) & \text{otherwise,} \end{cases}$$

where

$$\mathbf{G}(i,j) = \begin{cases} 1 & \text{if there is an outlink from } i \text{ to } j, \\ 0 & \text{otherwise.} \end{cases}$$

The interpretation is as follows: if the ith page has outlinks, then the surfer will click on one of the outlinks with a uniform probability; otherwise, the surfer will move to a random page based on the distribution provided by the personalization vector. A page that has no outlinks is called a *dangling page*. It is assumed that the same dynamics continue perpetually, giving rise to a homogeneous discrete-time Markov chain; $\mathbf{Q}$ is the transition probability matrix of this Markov chain. Let $\boldsymbol{\pi}_0 \in \mathbb{R}^{N\times 1}$ denote the probability distribution for the surfer's initial location; the location at time k is given by $\boldsymbol{\pi}_k^T = \boldsymbol{\pi}_0^T(\mathbf{Q})^k$.

The PageRank model assumes that the relative importance of a webpage is conferred by its limiting probability as $k \to \infty$, i.e., the relative frequency that the surfer will revisit the page in the long term. However, the specifications provided so far do not guarantee the existence of a unique limiting distribution. As such, for computational purposes, we use a "slightly shifted" Markov chain:

$$\mathbf{P} = c\mathbf{Q} + (1-c)\mathbf{1}_N\mathbf{u}^T,$$

where $\mathbf{1}_N\mathbf{u}^T$ is a rank-one matrix with rows $\mathbf{u}^T$ and c is some constant in $(0,1)$. The chain $\mathbf{P}$ closely approximates $\mathbf{Q}$ for c close to one. A typical value for c is between 0.85 and 0.95. It has been shown [Haveliwala and Kamvar 03] that c is related to the convergence rate of the algorithm for calculating the PageRank vector to be described in greater detail below. That $\mathbf{u}$ is positive ensures that $\mathbf{P}$ is an irreducible and aperiodic Markov chain (see [Berman and Plemmons 94, Kemeny and Snell 60]). By the Perron-Frobenius Theorem [Berman and Plemmons 94], a unique limiting distribution is guaranteed to exist, i.e., $\boldsymbol{\pi}^T = \lim_{k\to\infty} \boldsymbol{\pi}_0^T\mathbf{P}^k$, regardless of the initial distribution. The PageRank vector is defined to be this limiting distribution $\boldsymbol{\pi}$; $\mathbf{P}$ is sometimes referred to as the Google matrix. Let $\mathbb{S}_{\mathrm{D}}$ and $\mathbb{S}_{\mathrm{ND}}$ denote the dangling and nondangling subsets of $\mathbb{S}$, respectively; then,

$$\mathbf{P}(i,:) = \begin{cases} c\frac{\mathbf{G}(i,:)}{\sum_{l=1}^{N}\mathbf{G}(i,l)} + (1-c)\mathbf{u}^T & \text{if } i \in \mathbb{S}_{\mathrm{ND}}, \\ \mathbf{u}^T & \text{if } i \in \mathbb{S}_{\mathrm{D}}. \end{cases} \tag{2.1}$$

Each dangling row of $\mathbf{P}$ looks the same: $\mathbf{u}^T$. The nondangling rows are the sum of two components: the first component is given rise by the link structure ($\mathbf{G}$), while the second component comes from surfer preferences ($\mathbf{u}^T$).

In Algorithm 1, we state the "standard" PageRank algorithm [Page et al. 98] for computing the PageRank vector. The algorithm takes an arbitrary initial probability vector and then multiplies it by $\mathbf{P}$ repeatedly until convergence. It is a modified version of the power method [Golub and Loan 96] that takes advantage of the sparseness of $\mathbf{G}$, which is extremely sparse because most webpages have only a handful of outlinks.

Algorithm 1. (Standard PageRank.)

```
Form $\tilde{\mathbf{P}}$, where $\tilde{\mathbf{P}}(i,j) = \begin{cases} \frac{\mathbf{G}(i,j)}{\sum_{l=1}^{N} \mathbf{G}(i,l)} & \text{if } i \in \mathbb{S}_{\mathrm{ND}}, \\ 0 & \text{if } i \in \mathbb{S}_{\mathrm{D}}. \end{cases}$
Select $\mathbf{y} \in \mathbb{R}^{N \times 1}, \mathbf{y} \geq 0, ||\mathbf{y}||_1 = 1$;
$\delta = ||\mathbf{y} - \mathbf{x}||_1$;
while $\delta \geq \epsilon$ do
    $\mathbf{x} = \mathbf{y}$;
    $\mathbf{y}^T = c\mathbf{x}^T\tilde{\mathbf{P}}$;
    $d = 1 - ||\mathbf{y}||_1$;
    $\mathbf{y} = \mathbf{y} + d\mathbf{u}$;
    $\delta = ||\mathbf{y} - \mathbf{x}||_1$;
end
```

Notice that $\mathbf{P}$, which is completely dense, is never explicitly formed or used. Instead we use $\tilde{\mathbf{P}}$, which is extremely sparse. The multiplication step $\mathbf{x}^T\tilde{\mathbf{P}}$ is very efficient, and a multiple of $\mathbf{u}$ is subsequently added. Under this approach, each iteration of the loop requires $O(N)$ operations; in contrast, each iteration would have required $O(N^2)$ operations if $\mathbf{P}$ was used—which would have been computationally prohibitive. The strategy of implementing a dense vector-matrix multiplication as a sparse vector-matrix multiplication plus a vector-vector addition will appear again in our algorithm.

3. Exploiting Lumpability

We review the lumpability of Markov chains (see [Kemeny and Snell 60, Dayar and Stewart 97]) and relate this property to $\mathbf{P}$.

The idea is that some Markov chains can have their state spaces partitioned ("lumped") into highly homogeneous blocks. Here, *homogeneity* refers to the property that all states within a block have identical transition probabilities to other blocks. With this type of Markov chains, one can easily derive a transition probability matrix describing dynamics at the block level. We show below how this property can be exploited to handle large Makov chains in a "divide-and-conquor" fashion roughly as follows: We first calculate the block-level transition probability matrix and compute its limiting distribution. Then, using this limiting distribution to describe dynamics between blocks, we figure out the dynamics within each block by computing the limiting distribution at the state level one block at a time. Working with only a portion of the state space at a time, the overall computing time is reduced.

The verification of lumpability is normally nontrivial. We'll show that the Markov chain associated with $\mathbf{P}$ is trivially lumpable. It should be emphasized that a lumpable chain is not necessarily *nearly completely decomposable* (NCD) and vice versa. An NCD chain is one that can be partitioned into blocks with little or no block-level traffic. For such a chain a different divide-and-conquor approach based on traditional aggregation/disaggregation techniques (see [Cao and Stewart 85, Meyer 89, Simon and Ando 61]) exists. Although sharing many similarities with our two-stage approach, aggregation/disaggregation techniques are iterative approximation algorithms where one switches between block-level and state-level calculations continually rather than sequentially. As it happens, $\mathbf{P}$ is both lumpable and NCD. We return to this point in Section 7.

Definition 3.1. Suppose that $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the transition probability matrix of a homogeneous discrete-time Markov chain with n states. Let $S_1, S_2, ..., S_r$ be disjoint blocks of states such that $\bigcup_{l=1}^{r} S_l = \{1, 2, ..., n\}$. The Markov chain is said to be *lumpable with respect to* $S_1, S_2, ...S_r$ if for every $l \in \{1, 2, ..., r\}$ and $m \in \{1, 2, ..., r\}$, every $i \in S_l$ satisfies

$$\sum_{j \in S_m} \mathbf{M}(i, j) = f(l, m). \tag{3.1}$$

That the right-hand side of (3.1) does not depend on i is what is meant by homogeneity.

Proposition 3.2. *Consider the Markov chain with states* $1, 2, ..., r$ *obtained by tracking the block-level transitions of* $\mathbf{M}$ *above. The resulting Markov chain (the "lumped chain") has an r-by-r transition probability matrix given by*

$$\mathbf{M}^{\mathrm{L}}(l, m) = f(l, m).$$

Furthermore, if $\mathbf{M}$ *is irreducible and aperiodic, so is* $\mathbf{M}^{\mathrm{L}}$.

Results on lumpable Markov chains can be found in older [Kemeny and Snell 60] and more recent [Dayar and Stewart 97] works. In particular, Dayar and Stewart provided the following small example:

$$\mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.2 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.1 & 0.4 & 0.2 \\ 0.5 & 0.1 & 0.1 & 0.3 \\ 0.5 & 0.3 & 0.2 & 0.0 \end{pmatrix} \end{array}$$

[Dayar and Stewart 97]. If we take $S_1 = \{1,3\}$ and $S_2 = \{2,4\}$, we see that 1 and 3 have the the same transition probability to S_2 $(0.3+0.1 = 0.1+0.3 = 0.4)$ while 2 and 4 have the same transition probability to S_1 $(0.3+0.4 = 0.5+0.2 = 0.7)$. Proposition 3.2 says that the block-level transition probability matrix, i.e., the "lumped chain," is given by

$$\mathbf{M}^{\mathrm{L}} = \begin{array}{c} \\ S_1 \\ S_2 \end{array} \!\!\begin{array}{c} \begin{array}{cc} S_1 & S_2 \end{array} \\ \begin{pmatrix} 0.6 & 0.4 \\ 0.7 & 0.3 \end{pmatrix} \end{array}.$$

We claim that the Markov chain associated with $\mathbf{P}$ is lumpable with respect to the partition where *all the dangling nodes are lumped into one block and each nondangling node is its own block.*

Proposition 3.3. *Define* $\mathbb{S}_k = \{k\}, \forall k \in \mathbb{S}_{\mathrm{ND}}$. *The homogeneous discrete-time Markov chain associated with* $\mathbf{P}$ *is lumpable with respect to* $\mathbb{S}_{\mathrm{D}}$ *and* $\mathbb{S}_k$, $\forall k \in \mathbb{S}_{\mathrm{ND}}$.

Proof. If $\mathbb{S}_l$ is a block consisted of a single nondangling node ($\mathbb{S}_l = \mathbb{S}_k$ for some $k \in \mathbb{S}_{\mathrm{ND}}$), then (3.1) is clearly true regardless of m.

Now let $\mathbb{S}_l$ be the block consisting of dangling nodes ($\mathbb{S}_l = \mathbb{S}_{\mathrm{D}}$). For all $i \in \mathbb{S}_l$, (2.1) yields

$$\sum_{j\in\mathbb{S}_m} \mathbf{P}(i,j) = \begin{cases} \mathbf{P}(i,m) = \mathbf{u}(m) & \text{if } \mathbb{S}_m = \mathbb{S}_k, k \in \mathbb{S}_{\mathrm{ND}}, \\ \sum_{j\in\mathbb{S}_{\mathrm{D}}} \mathbf{P}(i,j) = \sum_{j\in\mathbb{S}_{\mathrm{D}}} \mathbf{u}(j) & \text{if } \mathbb{S}_m = \mathbb{S}_{\mathrm{D}}. \end{cases} \quad \square$$

By lumping the dangling nodes into one block, we obtain a Markov chain with $\mathrm{card}(\mathbb{S}_{\mathrm{ND}}) + 1$ states compared to $\mathrm{card}(\mathbb{S}_{\mathrm{ND}}) + \mathrm{card}(\mathbb{S}_{\mathrm{D}})$ states in the original chain. The value $\mathrm{card}(\mathbb{S}_{\mathrm{D}})$ is typically several times larger than $\mathrm{card}(\mathbb{S}_{\mathrm{ND}})$. According to a previous block structure study [Kamvar et al. 03b], a 2001 crawl by Stanford's WebBase project [Hirai et al. 00] containing 290 million pages in total has only 70 million nondangling pages. The lumped chain is irreducible and aperiodic because its transition probability matrix is necessarily positive. The Perron-Frobenius Theorem guarantees the existence of a unique limiting distribution. The latter is a vector with $\mathrm{card}(\mathbb{S}_{\mathrm{ND}})+1$ elements: The $\mathrm{card}(\mathbb{S}_{\mathrm{ND}})$ elements associated with the singleton blocks equal the limiting probabilities of the dangling nodes that one would have gotten under the standard PageRank algorithm. (This follows from Propositions 3.2 and 3.3: since the lumped chain describes exactly the block-level dynamics of the original chain, the limiting probabilities associated with the singleton blocks in the lumped chain must equal the limiting probabilities associated with the nondangling nodes in the original chain that make up the singleton blocks.) This corresponds to the first stage of our algorithm.

At this point we need only the limiting probabilities associated with the dangling nodes to complete the PageRank vector. In the second stage of our algorithm, we use *state aggregation* to combine the nondangling nodes into one block while having each dangling node as its own block. The limiting distribution of the aggregated chain is then calculated. It is a $\text{card}(\mathbb{S}_{\text{D}}) + 1$ vector, with $\text{card}(\mathbb{S}_{\text{D}})$ of the elements identical to what one would have gotten for the dangling nodes under the standard PageRank algorithm. The complete PageRank vector is thus obtained. A critical step here is that in aggregating the nondangling nodes, the nondangling nodes need to be weighted using the (block-level) limiting distribution from the first stage. We now prove this.

Proposition 3.4. *Suppose that $\mathbf{M} \in \mathbb{R}^{n\times n}$ is the transition probability matrix for an irreducible and aperiodic Markov chain and that we know $p < n$ elements of its limiting distribution $\boldsymbol{\varsigma}$. Without loss of generality, assume that these are the first p elements of $\boldsymbol{\varsigma}$ and let $\mathbf{M}$ and $\boldsymbol{\varsigma}$ be partitioned accordingly:*

$$\mathbf{M} = \begin{array}{c c} & \begin{array}{cc} p & n-p \end{array} \\ \begin{array}{c} p \\ n-p \end{array} & \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix} \end{array} \text{ and } \boldsymbol{\varsigma} = \begin{bmatrix} \boldsymbol{\varsigma}_p \\ \boldsymbol{\varsigma}_{n-p} \end{bmatrix}.$$

If

$$\mathbf{M}^{\text{A}} = \begin{array}{c c} & \begin{array}{cc} 1 & n-p \end{array} \\ \begin{array}{c} 1 \\ n-p \end{array} & \begin{pmatrix} \frac{\boldsymbol{\varsigma}_p^T}{\boldsymbol{\varsigma}_p^T\mathbf{1}}\mathbf{M}_{11}\mathbf{1} & \frac{\boldsymbol{\varsigma}_p^T}{\boldsymbol{\varsigma}_p^T\mathbf{1}}\mathbf{M}_{12} \\ \mathbf{M}_{21}\mathbf{1} & \mathbf{M}_{22} \end{pmatrix} \end{array},$$

then the last $(n-p)$ elements of the limiting distribution of $\mathbf{M}^{\text{A}}$ give $\boldsymbol{\varsigma}_{n-p}$.

Proof. From the stationarity equation for $\mathbf{M}$, i.e., $\boldsymbol{\varsigma}^T = \boldsymbol{\varsigma}^T\mathbf{M}$, we get

$$\begin{aligned} \boldsymbol{\varsigma}_p^T &= \boldsymbol{\varsigma}_p^T\mathbf{M}_{11} + \boldsymbol{\varsigma}_{n-p}^T\mathbf{M}_{21}, \\ \boldsymbol{\varsigma}_{n-p}^T &= \boldsymbol{\varsigma}_p^T\mathbf{M}_{12} + \boldsymbol{\varsigma}_{n-p}^T\mathbf{M}_{22}. \end{aligned}$$

Because $\mathbf{M}$ is irreducible and aperiodic, $\boldsymbol{\varsigma}$ is positive and $\mathbf{M}^{\text{A}}$ is irreducible and aperiodic. Take $\boldsymbol{\varrho}^T = \begin{pmatrix} \boldsymbol{\varsigma_p}^T\mathbf{1} & \boldsymbol{\varsigma}_{n-p}^T \end{pmatrix}$. Observe that

$$\begin{aligned} \boldsymbol{\varrho}^T\mathbf{M}^{\text{A}} &= \begin{pmatrix} (\boldsymbol{\varsigma}_p^T\mathbf{M}_{11} + \boldsymbol{\varsigma}_{n-p}^T\mathbf{M}_{21})\mathbf{1} & \boldsymbol{\varsigma}_p^T\mathbf{M}_{12} + \boldsymbol{\varsigma}_{n-p}^T\mathbf{M}_{22} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\varsigma}_p^T\mathbf{1} & \boldsymbol{\varsigma}_{n-p}^T \end{pmatrix} = \boldsymbol{\varrho}^T. \end{aligned}$$

That is, $\boldsymbol{\varrho}$ satisfies the stationarity equation for $\mathbf{M}^{\text{A}}$ and is thus the unique limiting distribution for $\mathbf{M}^{\text{A}}$. □

We combine Propositions 3.2 and 3.4 into the following theorem.

Theorem 3.5. *Suppose that* $\mathbf{M} \in \mathbb{R}^{n\times n}$ *is the transition probability matrix for an irreducible and aperiodic Markov chain that has a lumpable subset of* $n-p$ *states, i.e., the chain is lumpable with respect to the partition that these* $n-p$ *states are lumped into one block and the remaining* p *states are each its own block. Without loss of generality, we assume that states are reordered as follows:*

$$\mathbf{M} = \begin{array}{c c} & \begin{array}{cc} p & n-p \end{array} \\ \begin{array}{c} p \\ n-p \end{array} & \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix} \end{array}.$$

Suppose that the limiting distribution for $\mathbf{M}$ *is* $\boldsymbol{\varsigma} = \left[\begin{smallmatrix} \boldsymbol{\varsigma}_p \\ \boldsymbol{\varsigma}_{n-p} \end{smallmatrix}\right]$. *Then,*

(a) *The first* p *elements of the limiting distribution of the lumped chain* $\mathbf{M}^{\mathrm{L}}$ *give* $\boldsymbol{\varsigma}_p$*;*

(b) *supposing we form an aggregated chain* $\mathbf{M}^{\mathrm{A}}$ *using* $\boldsymbol{\varsigma}_p$ *from above, then the last* $n-p$ *elements of the limiting distribution of* $\mathbf{M}^{\mathrm{A}}$ *give* $\boldsymbol{\varsigma}_{n-p}$.

In other words, solving for the limiting distributions of $\mathbf{M}^{\mathrm{L}}$ *and* $\mathbf{M}^{\mathrm{A}}$ *yields the limiting distribution* $\boldsymbol{\varsigma} = \left[\begin{smallmatrix} \boldsymbol{\varsigma}_p \\ \boldsymbol{\varsigma}_{n-p} \end{smallmatrix}\right]$ *for* $\mathbf{M}$.

Proof. Because $\mathbf{M}^{\mathrm{L}}$ describes the dynamics of $\mathbf{M}$ at the block level (Proposition 3.2), the first p elements of the limiting distribution for $\mathbf{M}^{\mathrm{L}}$ must describe the limiting probabilities of visiting the first p blocks. Since these blocks are just the first p states of $\mathbf{M}$, their limiting probabilities are $\boldsymbol{\varsigma}_p$. This proves the first statement. The second statement follows from Proposition 3.4. $\square$

Note that Proposition 3.3 says that $\mathbf{P}$ satisfies the hypothesis of Theorem 3.5. Therefore, the "sequential lumping and aggregation" described in the theorem can be used to calculate the limiting distribution of $\mathbf{P}$. This is the basis for using the two-stage algorithm to compute the PageRank vector.

Remarks. (1) Previously, lumping and state aggregation were thought of as unrelated techniques for state space reduction. Theorem 3.5 shows that by using these two methods sequentially, one can calculate the limiting probabilities for complementary regions of the state space; this is generally applicable so long as the Markov chain is irreducible and aperiodic and has a lumpable subset. To our knowledge, this is the first paper to make that observation. (2) The concept is clearly generalizable. If the Markov chain has multiple lumpable subsets of states, one can lump and aggregate different subsets sequentially so as to calculate the limiting probabilities one subset at a time; Propositions 3.2 and 3.4 are sufficiently general for this purpose. Our multistage extensions in Section 8 are examples that make repeated applications of Propositions 3.2 and 3.4.

4. Two-Stage Algorithm

Having motivated the algorithm and established its probabilistic interpretation, we now give a step-by-step description of the algorithm. To recap, the algorithm is a two-stage process:

- **STAGE1**

  (a) Lump the dangling nodes into one block, and construct the transition probability matrix of the lumped chain.

  (b) Compute the limiting distribution of the lumped chain. This gives us $\boldsymbol{\pi}(k)$, $k \in \mathbb{S}_{\mathrm{ND}}$. As before, $\boldsymbol{\pi}$ is the PageRank vector.

- **STAGE2**

  (c) Compute the aggregation weights, i.e., $\boldsymbol{\pi}(k)/\sum_{m\in\mathbb{S}_{\mathrm{ND}}\boldsymbol{\pi}(m)}$, $k \in \mathbb{S}_{\mathrm{ND}}$.

  (d) Aggregate the nondangling nodes into one block by using the weights from (c).

  (e) Compute the limiting distribution of the aggregated chain. This yields $\boldsymbol{\pi}(k)$, $k \in \mathbb{S}_{\mathrm{D}}$.

In this section we describe an efficient numerical implementation of the two-stage algorithm. We emphasize three points: (1) The standard PageRank algorithm is an iterative algorithm based on the power method. We derive a similar algorithm for STAGE1. Because the latter can converge in as many or fewer iterations than the standard PageRank algorithm and because the lumped chain is smaller and has fewer nonzero elements than the original chain, our algorithm is significantly faster. (2) The aggregated chain in STAGE2 has a special structure allowing its limiting distribution to be quickly "extracted," which we describe in greater detail later. This makes the amount of work for STAGE2 quite negligible in the overall scheme of things. (3) Our implementation exploits sparsity whenever possible, and no dense matrices are ever formed or used. Here we focus on describing the steps involved; the proofs are deferred until the next section. For simplicity, we assume, without loss of generality, that $\mathbb{S}_{\mathrm{ND}} = \{1, 2, ..., K\}$ and

$\mathbb{S}_{\mathrm{D}} = \{K+1, K+2, ..., N\}$; $\mathbf{P}$ and $\mathbf{u}$ can be partitioned accordingly as

$$\mathbf{P} = \begin{array}{c c} & \begin{array}{cc} K & N-K \end{array} \\ \begin{array}{c} K \\ N-K \end{array} & \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{pmatrix} \end{array} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{1}_{N-K}\mathbf{u}_K^T & \mathbf{u}_K^T \end{pmatrix} \tag{4.1}$$

and

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_K \\ \mathbf{u}_{N-K} \end{bmatrix}.$$

4.1. STAGE1

The $(K+1)$-by-$(K+1)$ transition probability matrix for the lumped chain is given by

$$\mathbf{P}^{(1)} = \begin{array}{c c} & \begin{array}{cc} K & 1 \end{array} \\ \begin{array}{c} K \\ 1 \end{array} & \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12}\mathbf{1} \\ \mathbf{u}_K^T & \mathbf{u}_{N-K}^T\mathbf{1} \end{pmatrix} \end{array}.$$

Let $\mathbf{x} \in \mathbb{R}^{(K+1)\times 1}$ be an arbitrary probability distribution. We implement $\mathbf{x}^T\mathbf{P}^{(1)}$ (which would have been a dense vector-matrix multiplication) as a sparse vector-matrix multiplication plus a vector-vector addition:

$$\mathbf{x}^T\mathbf{P}^{(1)} = c\mathbf{x}(1:K)^T\tilde{\mathbf{P}}^{(1)} + (1 - c + c\mathbf{x}(K+1))\,\tilde{\mathbf{u}}^T, \tag{4.2}$$

where

$$\tilde{\mathbf{P}}^{(1)}(i,:) = \left(\frac{\mathbf{G}(i,1:K)}{\sum_{l=1}^{N}\mathbf{G}(i,l)} \quad 1 - \frac{\sum_{l=1}^{K}\mathbf{G}(i,l)}{\sum_{l=1}^{N}\mathbf{G}(i,l)} \right) \text{ for } 1 \le i \le K, \tag{4.3}$$

$$\tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{u}_K \\ 1-\alpha \end{pmatrix}, \tag{4.4}$$

and $\alpha = \mathbf{u}_K^T\mathbf{1}$. Note that $\tilde{\mathbf{P}}^{(1)}$ is K-by-$(K+1)$ and has different dimensions from $\mathbf{P}^{(1)}$. Equation (4.2) requires only $O(K)$ operations. STAGE1 is a version of the power method modified to incorporate (4.2), as seen in Algorithm 2.

STAGE1 converges to the limiting distribution of $\mathbf{P}^{(1)}$ and yields K components of the PageRank vector. Between the two stages, STAGE1 takes up most of the computing time. Note that $\mathbf{P}^{(1)}$ is just the upper left part of $\mathbf{P}$ plus a column and a row. Working with $\mathbf{P}^{(1)}$, therefore, requires much less work than with $\mathbf{P}$. The speed-up factor is related to the ratio $\frac{K}{N}$ as well as the distribution of nonzero elements within $\mathbf{P}$.

Algorithm 2. (STAGE1.)

```
Form $\tilde{\mathbf{P}}^{(1)}$ and $\tilde{\mathbf{u}}$ via (4.3) and (4.4).
Select $\mathbf{y} \in \mathbb{R}^{(K+1)}, \mathbf{y} \geq 0, ||\mathbf{y}||_1 = 1$;
$\delta = ||\mathbf{y} - \mathbf{x}||_1$;
while $\delta \geq \epsilon$ do
    $\mathbf{x} = \mathbf{y}$ ;
    $\mathbf{y}^T = c\mathbf{x}(1:K)^T \tilde{\mathbf{P}}^{(1)} + (1 - c + c\mathbf{x}(K+1))\, \tilde{\mathbf{u}}^T$ ;
    $\delta = ||\mathbf{y} - \mathbf{x}||_1$;
end
```

4.2. STAGE2

The transition probability matrix for the aggregated chain is

$$\mathbf{P}^{(2)} = \begin{array}{c} \\ 1 \\ N-K \end{array} \begin{array}{c} \begin{array}{cc} 1 & N-K \end{array} \\ \begin{pmatrix} \boldsymbol{\eta}^T \mathbf{P}_{11} \mathbf{1} & \boldsymbol{\eta}^T \mathbf{P}_{12} \\ \alpha \mathbf{1}_{N-K} & \mathbf{1}_{N-K} \mathbf{u}_K^T \end{pmatrix} \end{array},$$

where

$$\boldsymbol{\eta}(k) = \frac{\boldsymbol{\pi}(k)}{\sum_{m=1}^{K} \boldsymbol{\pi}(m)}, k = 1, 2, \ldots, K. \tag{4.5}$$

Note that $\mathbf{P}^{(2)}$ is a rank-two matrix—all rows starting from the second row are identical. For an arbitrary probability distribution $\mathbf{x} \in \mathbb{R}^{(N-K+1)\times 1}$, we have

$$\mathbf{x}^T \mathbf{P}^{(2)} = c\mathbf{x}(1) \begin{pmatrix} \beta & \mathbf{w}^T \end{pmatrix} + (1 - c\mathbf{x}(1)) \begin{pmatrix} \alpha & \mathbf{u}_{N-K}^T \end{pmatrix},$$

where

$$\mathbf{w}^T = \sum_{i=1}^{K} \boldsymbol{\eta}(i) \frac{\mathbf{G}(i, K+1:N)}{\sum_{l=1}^{N} \mathbf{G}(i,l)}, \tag{4.6}$$

$$\beta = 1 - \mathbf{w}^T \mathbf{1}. \tag{4.7}$$

Here, the vector-matrix multiplication is replaced by the summation of two vectors.

Whereas many iterations are needed for STAGE1 to converge, STAGE2 is guaranteed to terminate quickly. This is because $\mathbf{P}^{(2)}$ is a rank-two matrix and the simplicity of its eigenvalue system allows the limiting distribution to be quickly extracted if it is not yet available after three power steps. The extraction is by means of the Aitken extrapolation [Kamvar et al. 03c]. In Algorithm 3, we state the algorithm while a formal proof is deferred to Section 5.

Algorithm 3. (STAGE2.)

```
Compute $\boldsymbol{\eta}$ via (4.5).
Form $\mathbf{w}$ and $\beta$ via (4.6) and (4.7).
Select $\mathbf{x}^{(0)} \in \mathbb{R}^{(N-K+1)}, \mathbf{x}^{(0)} \geq 0, ||\mathbf{x}^{(0)}||_1 = 1$;
for $i = 1 : 3$ do
    $\left(\mathbf{x}^{(i)}\right)^T = c\mathbf{x}^{(i-1)}(1)\left(\beta \quad \mathbf{w}^T\right) + \left(1 - c\mathbf{x}^{(i-1)}(1)\right)\left(\alpha \quad \mathbf{u}_{N-K}^T\right)$;
end
if $||\mathbf{x}^{(3)} - \mathbf{x}^{(2)}||_1 < \epsilon$ then
    $\mathbf{z} = \mathbf{x}^{(3)}$;
else
    /* Aitken extrapolation */
    for $i = 1 : (N-K+1)$ do
        $\mathbf{v}(i) = \frac{\left(\mathbf{x}^{(2)}(i) - \mathbf{x}^{(1)}(i)\right)^2}{\mathbf{x}^{(3)}(i) - 2\mathbf{x}^{(2)}(i) + \mathbf{x}^{(1)}(i)}$;
    end
    $\mathbf{z} = \mathbf{x}^{(1)} - \mathbf{v}$;
end
```

Typically, STAGE1 can take about 100 iterations to converge. The precise number of iterations needed depends on c (see [Haveliwala and Kamvar 03]). Because STAGE2 converges in 3 iterations and each iteration is just the summation of two vectors, it requires much less work. Also, because $\mathbf{P}^{(2)}$ is rank-two, only two vectors need to be stored.

5. Convergence Analysis

We address two issues. First, we prove that STAGE1 converges in as many or fewer iterations than the standard PageRank algorithm. Then, we show that $\mathbf{P}^{(2)}$, being rank-two, allows us to extract the limiting distribution with Aitken extrapolation.

5.1. Convergence of STAGE1

Our strategy is to show that regardless of what initial distribution is used with the standard PageRank algorithm, we can always construct a related initial distribution for STAGE1 so that STAGE1 converges in as many or fewer iterations. We begin with a lemma. The lemma says that if we initialize standard PageRank and STAGE1 with state-level and block-level versions of the same underlying limiting distribution, after each iteration the distributions produced by the two algorithms will be state-level and block-level versions of each other.

Lemma 5.1. *Let* $\mathbf{x}^{(0)} \in \mathbb{R}^{N\times 1}$ *be given. Define*

$$(\mathbf{y}^{(0)})^T = (\mathbf{x}^{(0)})^T \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{0} & \mathbf{1}_{N-K} \end{pmatrix},$$

where $\mathbf{I}$ *is the* K*-by-*K *identity matrix and* $\mathbf{0}$ *is an* $(N-K)$*-by-*K *matrix of zeros. Consider two sequences of iterates:*

$$(\mathbf{x}^{(l+1)})^T = (\mathbf{x}^{(l)})^T \mathbf{P},$$
$$(\mathbf{y}^{(l+1)})^T = (\mathbf{y}^{(l)})^T \mathbf{P}^{(1)},$$

for $l = 0, 1, 2, \ldots.$ *Then,*

$$(\mathbf{y}^{(l)})^T = (\mathbf{x}^{(l)})^T \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{0} & e_{N-K} \end{pmatrix}$$

for $l = 0, 1, 2, \ldots.$

Proof. The claim is true for $l = 0$ by assumption. Now, suppose that the claim is true for $l = l_0$, then

$$\begin{aligned} (\mathbf{y}^{(l_0)})^T \mathbf{P}^{(1)} &= (\mathbf{x}^{(l_0)})^T \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{0} & \mathbf{1}_{N-K} \end{pmatrix} \mathbf{P}^{(1)} \\ &= (\mathbf{x}^{(l_0)})^T \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{0} & \mathbf{1}_{N-K} \end{pmatrix} \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{u}_K^T & \mathbf{u}_K^T \end{pmatrix} \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{0} & \mathbf{1}_{N-K} \end{pmatrix} \\ &= (\mathbf{x}^{(l_0)})^T \mathbf{P} \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{0} & \mathbf{1}_{N-K} \end{pmatrix} \\ &= (\mathbf{x}^{(l_0+1)})^T \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{0} & \mathbf{1}_{N-K} \end{pmatrix}. \end{aligned}$$

Because $(\mathbf{y}^{(l_0)})^T \mathbf{P}^{(1)} = (\mathbf{y}^{(l_0+1)})^T$, the claim must also hold for $l = l_0 + 1$. Induction completes the proof. □

This result is not surprising given Proposition 3.2: $\mathbf{P}^{(1)}$ captures exactly the block-level dynamics of $\mathbf{P}$. The following proposition says successive block-level distributions cannot differ by more than the difference between successive state-level distributions.

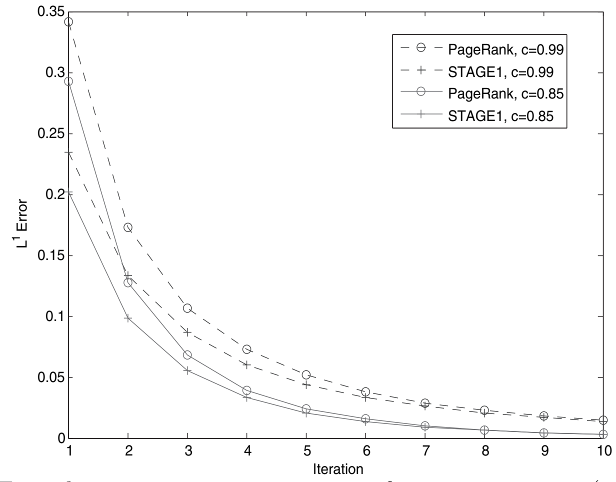


Figure 1. Error between successive iterates after ten iterations (web graph = US2004).

Proposition 5.2. *Let* $\mathbf{x}^{(l)}$ *and* $\mathbf{y}^{(l)}$ *be defined as above. Then,*

$$||\mathbf{y}^{(l+1)} - \mathbf{y}^{(l)}||_1 \leq ||\mathbf{x}^{(l+1)} - \mathbf{x}^{(l)}||_1$$

for $l = 0, 1, 2, \ldots.$

Proof. Following Lemma 5.1,

$$\begin{aligned}
&||\mathbf{y}^{(l+1)} - \mathbf{y}^{(l)}||_1 - ||\mathbf{x}^{(l+1)} - \mathbf{x}^{(l)}||_1 \\
&= |\mathbf{y}^{(l+1)}(K+1) - \mathbf{y}^{(l)}(K+1)| - \sum_{i=K+1}^{N} |\mathbf{x}^{(l+1)}(i) - \mathbf{x}^{(l)}(i)| \\
&= |\sum_{i=K+1}^{N} \mathbf{x}^{(l+1)}(i) - \mathbf{x}^{(l)}(i)| - \sum_{i=K+1}^{N} |\mathbf{x}^{(l+1)}(i) - \mathbf{x}^{(l)}(i)| \\
&\leq 0.
\end{aligned}$$

□

Figure 1 gives a graphical example. The difference between successive iterates from STAGE1 is dominated by that of successive iterates from the standard PageRank algorithm. As a result, when the two algorithms use the same ϵ as the termination criterion, STAGE1 will always converge in the same or fewer iterations.

5.2. Convergence of STAGE2

To show that STAGE2 computes the unique limiting distribution of $\mathbf{P}^{(2)}$, i.e., the left eigenvector associated with the eigenvalue of one, we begin with a lemma that describes the eigenvalue system of $\mathbf{P}^{(2)}$.

Lemma 5.3. *For brevity, we denote $M = (N - K + 1)$. Thus, $\mathbf{P}^{(2)}$ is M-by-M.*

(a) One is the dominant eigenvalue of $\mathbf{P}^{(2)}$; its algebraic and geometric multiplicities are both one. Zero is also an eigenvalue and has a geometric multiplicity of $(M - 2)$.

(b) If $\mathbf{P}^{(2)}$ does not have a third distinct eigenvalue, then

$$\mathbf{x}^T\mathbf{P}^{(2)}\mathbf{P}^{(2)} = \mathbf{x}^T\mathbf{P}^{(2)}\mathbf{P}^{(2)}\mathbf{P}^{(2)}$$

for any $\mathbf{x} \in \mathbb{R}^{M\times 1}$ (i.e., in three iterations the sequence converges to either zero or the left dominant eigenvector).

Proof. The matrix $\mathbf{P}^{(2)}$ is positive and rank-two, and its rows sum to one. The Perron-Frobenius Theorem (see [Berman and Plemmons 94, pp. 27–32] or [Horn and Johnson 85, pp. 508–511]) establishes the first claim.

Next, suppose that $\mathbf{P}^{(2)}$ does not have a third distinct eigenvalue. The algebraic multiplicity of the eigenvalue zero is necessarily $(M - 1)$. The Jordan canonical form of $\mathbf{P}^{(2)}$ establishes the second claim. (See [Golub and Loan 96, p. 317] or [Horn and Johnson 85, pp. 121–131].) $\square$

The next proposition can be summarized as follows: Consider an arbitrary vector repeatedly multiplied by $\mathbf{P}^{(2)}$. If convergence does not occur after three iterations, $\mathbf{P}^{(2)}$ must have a third eigenvalue between one and zero in modulus, and all subsequent iterates are contained in the span of the first and second eigenvectors.

Proposition 5.4. *Let $\mathbf{x} \in \mathbb{R}^{M\times 1}$ be arbitrary. If*

$$\mathbf{x}^T\mathbf{P}^{(2)}\mathbf{P}^{(2)} \neq \mathbf{x}^T\mathbf{P}^{(2)}\mathbf{P}^{(2)}\mathbf{P}^{(2)},$$

then

(a) there exists another eigenvalue λ of $\mathbf{P}^{(2)}$ such that $0 < |\lambda| < 1$;

(b) for $l = 1, 2, ...,$

$$\mathbf{x}^T \left(\mathbf{P}^{(2)}\right)^l = c_1 \mathbf{v}_1^T + c_2 \lambda^l \mathbf{v}_2^T,$$

where

$$\mathbf{v}_1^T \mathbf{P}^{(2)} = \mathbf{v}_1^T,$$

$$\mathbf{v}_2^T \mathbf{P}^{(2)} = \lambda \mathbf{v}_2^T,$$

and c_1 and c_2 are constants.

Proof. The first statement follows directly from Lemma 5.3. An examination of the geometric multiplicities of $\mathbf{P}^{(2)}$ reveals the existence of a full set of eigenvectors that span $\mathbb{R}^{M \times 1}$. Writing $\mathbf{x}$ as a linear combination of these eigenvectors establishes the second statement. □

When the assumption of Proposition 5.4 holds, part (b) of the proposition says that we can obtain the dominant eigenvector by subtracting away the second eigenvector. This is exactly what Aitken extrapolation does [Kamvar et al. 03c].

6. Numerical Experiments

We conducted numerical experiments on Stanford's "Hedge" cluster consisting of Sun Fire X4100 systems each with two 2.8 GHz AMD Opteron processors and 8 GB of RAM running 64-bit Linux. The algorithms were implemented in C and compiled as MATLAB-callable mex functions. This approach gave us the speed, efficiency, and low-level control of C within the graphical and interactive environment of MATLAB. Some of our code was modified from the implementation of David Gleich (Institute for Computational and Mathematical Engineering, Stanford University), which is publicly available at http://www.stanford.edu/~dgleich/. Summary statistics of the web graphs used are given in Table 1. The data US2004 came from a July 2004 crawl of the websites of 14 American universities, and AU2006 came from an April 2006 crawl of all 38 Australian universities; both web graphs are available from the Statistical Cybermetrics Research Group at the University of Wolverhampton (http://cybermetrics.wlv.ac.uk/). Both graphs contained four million or more webpages and 24 million links and were representative of the typical crawl in terms of the ratio of dangling to nondangling nodes (three to five is typical) and the distribution of links. The data WIKI2005 came from a November 2005 crawl of Wikipedia with all non-article pages removed by Gleich. This

	N	K	nnz($\tilde{\mathbf{P}}$)	nnz($\tilde{\mathbf{P}}^{(1)}$)	nnz($\mathbf{u}$)	nnz($\tilde{\mathbf{u}}$)
US2004	6,411,252	1,585,057	23,883,438	14,932,701	6,411,252	1,585,058
AU2006	3,907,649	1,225,553	23,782,896	18,272,067	3,907,649	1,225,554
WIKI2005	1,634,989	1,562,432	19,753,078	19,998,918	1,634,989	1,562,433

Table 1. Web graphs used in the experiments: nnz indicates the number of nonzero elements.

latter web graph had an unusually low fraction of dangling nodes, thus allowing us to examine the algorithm's performance under extreme cases. (See Table 1.)

We conducted three numerical experiments, the results of which are provided below. The first experiment verified some of the convergence properties we proved earlier. The second experiment tested the speed of the two-stage algorithm relative to the standard PageRank algorithm. Lastly, we experimented with the embedding of other acceleration methods within the two-stage algorithm.

6.1. Convergence Behavior

We tested the convergence of our algorithm against the standard PageRank algorithm. Both algorithms were set to a convergence tolerance (ϵ) of 10^{-8}.

First, we tested if our algorithm converged at all. On all three webgraphs the algorithm converged readily to the desired tolerance. We found the L^1 error between successive iterates of standard PageRank to dominate that of STAGE1 (Figure 1), exactly as Proposition 5.2 predicted; STAGE1 always converged in as many and occasionally fewer iterations than did standard PageRank. The constant c affected the convergence of both algorithms in the same way—both algorithms converged slower when c was closer to one. This is because the second eigenvalue of $\mathbf{P}$ is a function of c [Haveliwala and Kamvar 03].

In addition, we tested whether our algorithm converged to the right vector. Theorem 3.5 assumed the use of the exact solution from STAGE1 in constructing the STAGE2 matrix. In practice, STAGE1 is terminated finitely, and this could impact the accuracy of the solution from STAGE2.

In theory, we could reduce the inaccuracy in STAGE1's solution to an arbitrarily low level by setting the tolerance parameter ϵ to machine precision. Then, the STAGE2 matrix would be as "exact" as floating point storage would permit—that is, there would not be any difference between using this finitely-terminated solution from STAGE1 or the "exact" solution, because in either case the rounding error occurring from the storage of the STAGE2 matrix would mask whatever additional accuracy the "exact" solution would provide. Fortunately, we found

		First K	Last $(N-K)$
	0.99	1.6828×10^{-10}	5.4302×10^{-9}
	0.95	3.3834×10^{-11}	4.6170×10^{-9}
US2004	$c = 0.85$	1.0224×10^{-11}	3.4593×10^{-9}
	0.99	7.2604×10^{-11}	3.1710×10^{-9}
	0.95	1.5937×10^{-11}	3.2452×10^{-9}
AU2006	$c = 0.85$	5.2734×10^{-12}	2.3439×10^{-9}
	0.99	4.7914×10^{-11}	3.7866×10^{-11}
	0.95	2.6340×10^{-11}	3.4605×10^{-11}
WIKI2005	$c = 0.85$	1.9982×10^{-11}	2.6271×10^{-11}

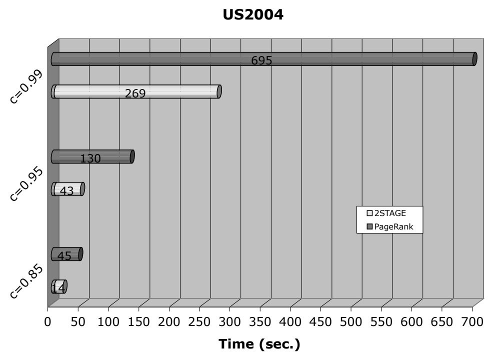
Table 2. The L^1 error between the first K and last $(N-K)$ components of solutions computed by the two-stage algorithm and the standard PageRank algorithm.

that we did not need to do this. Using an ϵ of 10^{-8} (for both STAGE1 and standard PageRank), we measured the L^1 error between solutions from the two algorithms. The results are provided in Table 2. For all three webgraphs and all values of c, the two solutions were less than $\epsilon = 10^{-8}$ apart. It's interesting to note that, in general, the error arising from STAGE1 tended to be smaller than the error arising from STAGE2, which in turn tended to be less than ϵ. The first part can be related to Lemma 5.1: when STAGE1 and the standard PageRank algorithm converged after the same number of iterations, the final solutions from the two algorithms are, in fact, block-level and state-level versions of the same distribution. As such, the two solutions tended to be very close to each other (save for rounding errors). The second part was due to this proximity carrying over to STAGE2, yielding an aggregated matrix very close to the exact aggregated matrix. Consequently, the error arising from STAGE2 also remained small.

In short, the two-stage algorithm properly converged to the same solution as in standard PageRank.

6.2. Computing Time

On all three web graphs our algorithm converged faster than PageRank. Generally, the speed-up can be related to two factors. The ratio of nondangling to dangling nodes, i.e., $\frac{K}{N}$, determines the relative dimensionality of the vector operations: when $\frac{K}{N}$ is small, the vectors involved in the addition, multiplication, and norm-taking steps are all much smaller in STAGE1 than in the standard algorithm. On the other hand, the distribution of nonzero elements in $\tilde{\mathbf{P}}$ affects the number of nonzero elements that will ultimately be in $\tilde{\mathbf{P}}^{(1)}$ and thus the matrix operations. The overall speed-up is thus related to $\frac{K}{N}$ and $\frac{\mathrm{nnz}(\tilde{\mathbf{P}}^{(1)})}{\mathrm{nnz}(\tilde{\mathbf{P}})}$.



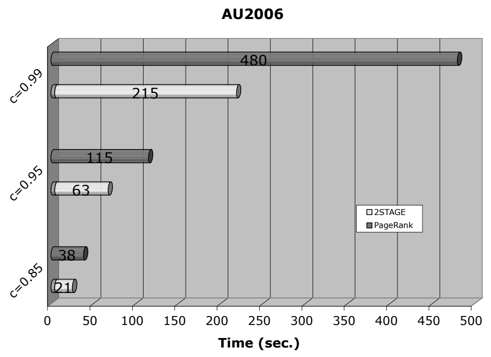


Figure 2. Computing time for two typical web graphs. The base section of each 2STAGE bar represents the total time spent on Steps (a), (c), (d), and (e) (see Section 4); the remaining length indicates the computing time for the limiting distribution of $\mathbf{P}^{(1)}$ (Step (b)).

US2004 had $\frac{K}{N} = 0.25$ and $\frac{\mathrm{nnz}(\tilde{\mathbf{P}}^{(1)})}{\mathrm{nnz}(\tilde{\mathbf{P}})} = 0.63$; AU2006 had $\frac{K}{N} = 0.31$ and $\frac{\mathrm{nnz}(\tilde{\mathbf{P}}^{(1)})}{\mathrm{nnz}(\tilde{\mathbf{P}})} = 0.75$. As shown in Figure 2, the speed-up for these two matrices were in the two- to threefold range, which was consistent with our expectations. The speed-up also depended on c: when c was far from one, the fixed overhead associated with the algorithm (Steps (a), (c), (d), and (e) in Section 4) became relatively significant and limited the overall speed-up.

In short, the amount of speed-up depended on the web graph used as well as c. With some web graphs, we saw a speed-up of as much as five to seven times. In the rare case when the graph had very few dangling nodes, the algorithm produced little advantage. Our experimentation with WIKI2006 ($\frac{K}{N} = 0.96$ and $\frac{\mathrm{nnz}(\tilde{\mathbf{P}}^{(1)})}{\mathrm{nnz}(\tilde{\mathbf{P}})} = 0.98$) showed a speed-up that was either negligible or very moderate (10%), depending on the value of c.

		PR	QE	GS	BiCGSTAB
	0.99	2.5	**3.0**	1.7	2.3
	0.95	2.7	3.0	1.3	**3.8**
US2004	$c = 0.85$	2.4	2.2	1.1	**3.6**
	0.99	**2.2**	1.8	1.1	1.9
	0.95	1.7	1.2	(0.8)	**2.3**
AU2006	$c = 0.85$	1.5	1.1	(0.9)	**2.6**
	0.99	1.1	1.3	**1.4**	1.1
	0.95	**1.1**	1.1	1.0	(0.9)
WIKI2005	$c = 0.85$	1.0	1.0	1.0	**1.0**

Table 3. Additional factor of speed-up when the two-stage speed-up was applied. Very occasionally the speed-up did not offset the overhead of the algorithm (Steps (a), (c), (d), and (e)), leading to an overall slowdown.

6.3. Embedding Other Acceleration Methods

One may apply to STAGE1 any acceleration method devised for the standard PageRank algorithm. We call the procedure of accelerating STAGE1 embedding. We considered the embedding of three acceleration methods: quadratic extrapolation [Kamvar et al. 03c], Gauss-Seidel [Arasu et al. 02], and the biconjugate gradient stable algorithm [Gleich and Zhukov 05, Gleich et al. 04]. Results are summarized in Tables 3 and 4. We abbreviate these acceleration methods QE, GS, and BiCGSTAB, respectively; standard PageRank and the two-stage algorithm were labeled PR and 2S, respectively.

Table 3 shows the additional speed-up the embedding of an acceleration method yielded compared to a direct application of that acceleration method alone. Roughly speaking, GS and BiCGSTAB benefited from embedding the most, although the gain depended on the web graph used and c. Very occasionally, the additional speed-up did not offset the overhead of embedding (Steps (a), (c), (d),

		Algorithm	Speed-up	Iterations	PR Iterations
	0.99	2S+GS	13.5	210	1,103
	0.95	2S+GS	7.1	60	230
US2004	$c = 0.85$	2S+GS	4.0	28	75
	0.99	2S+GS	5.5	302	1,100
	0.95	2S+GS	4.3	85	234
AU2006	$c = 0.85$	2S+GS	2.9	34	78
	0.99	2S+BiCGSTAB	2.0	145	847
	0.95	2S+QE	1.2	150	167
WIKI2005	$c = 0.85$	2S+QE	1.1	49	55

Table 4. The fastest among the eight possible algorithms: PR, QE, GS, BiCGSTAB, 2S, 2S+QE, 2S+GS, and 2S+BiCGSTAB. The speed-up factor is with respect to standard PageRank.

and (e) in Section 4), leading to a slight overall slowdown. Table 4 shows the fastest possible algorithm to solve each PageRank problem, i.e., the algorithm with the greatest cumulative speed-up. Overall, some type of embedding was always best, although the optimal embedding scheme could vary. A safe choice seemed to be the embedding of GS, which yielded a cumulative speed-up as high as 7.1 to 13.5 times. Please note that our implementations for QE, GS, and BiCGSTAB were not parallelized, which could change things.

We also verified (results not shown) that all embedding schemes computed a solution that was within ϵ of the solution computed by standard PageRank.

7. Dangling Nodes

A webpage is dangling if it has no outlinks. Dangling nodes are important for several reasons:

- Most webpages are dangling. Eiron et al. reported that dangling nodes can outnumber the nondanglings nodes by almost four to one [Eiron et al. 04].
- Dangling nodes contain much information about the structure of the web, i.e., all the information represented in $\mathbf{P}_{12}$ arises from dangling nodes. In our datasets, $\mathbf{P}_{12}$ often contained half of all the links.
- Dangling nodes tend to occur in the "frontier" of the web where things are changing most rapidly [Eiron et al. 04].
- Dangling nodes can have a high ranking. Eiron et al. reported many of the highest ranked webpages are dangling [Eiron et al. 04].
- Certain types of URLs are naturally dangling. These include downloadable files such as PDFs, images, MP3s, movies, and so on.
- Outgoing traffic from dangling nodes is modeled by a personalization vector: the PageRank model says that when there are no outlinks, the user will move to a random page based on her preferences, i.e., by directly entering a URL in the browser. This is quite intuitive and recognizes the fact that internet traffic is a function of both structure (links) and user behavior (preferences).

Our algorithm provides an efficient way for handling dangling nodes. Previously, it was suggested [Page et al. 98] that dangling nodes could be handled as follows: begin by applying the standard PageRank algorithm to $\mathbf{P}_{11}$ alone; upon convergence, take the limiting distribution, pad it with additional elements, and use that as the initial distribution for $\mathbf{P}$. The belief was that $\mathbf{P}$ would now converge in relatively few iterations. While this trick works well in many practical situations, it can fail in more extreme cases when aggressive personalization is used. (A small example is provided in the appendix.) Indeed, this is not the probabilistically correct way of handling dangling nodes. To see why, we appeal to the theory of stochastic complementation [Meyer 89]. Using the same partition as in (4.1), the theory says the first K components of $\mathbf{P}$'s limiting distribution, when normalized, equal the limiting distribution of

$$\mathbf{S}_{11} = \mathbf{P}_{11} + \mathbf{P}_{12}(\mathbf{I} - \mathbf{P}_{22})^{-1}\mathbf{P}_{21}.$$

Sometimes $\mathbf{S}_{11}$ is also known as the *stochastic complement* of $\mathbf{P}_{11}$. The trick described above amounts to assuming that $\mathbf{S}_{11}$ and $\mathbf{P}_{11}$ have the same (or similar) limiting distributions, which is generally incorrect.

So what would be a sufficient condition for the trick to work? Note that if $\mathbf{P}$ was a nearly completely decomposable (NCD) matrix with respect to this partition (which it isn't), $\mathbf{P}_{21}$ and $\mathbf{P}_{12}$ would be close to zero, and $\mathbf{S}_{11}$ and $\mathbf{P}_{11}$ would be nearly identical; in that case, the trick would work very well. In any event, we believe our algorithm has rendered the trick unnecessary: Our algorithm handles dangling nodes in a probabilistically correct manner (see Propositions 3.2, 3.3, 3.4, and 5.4) while achieving the desired speed-up.

Note that the property of a matrix being NCD, like lumpability, refers to a specific partition. While $\mathbf{P}$ is not NCD with respect to the partition in question, i.e., dangling versus nondangling nodes, it *is* NCD with respect to the partition that the rows and columns of $\mathbf{P}$ are symmetrically permuted by the order of the webpage's host [Kamvar et al. 03b].

8. Multistage Generalizations

Our algorithm takes on a Markov chain (i.e., probabilistic) point of view. Nevertheless, there is an alternative, nonprobabilistic approach that emphasizes the algebraic structure. For example, the reduction through lumping can be equivalently derived by means of the Neumann expansion. (We're indebted to Amy Langville for pointing this out; see also [Langville and Meyer 06b].)

Nevertheless, the Markov chain interpretation is intuitive and easily generalized. The two-stage algorithm can be viewed as sequentially alternating between complementary parts of the state space. This "divide-and-conquer" approach can also make use of a multistage algorithm. Here we mention two such variants of our algorithm based on repeated applications of Propositions 3.2 and 3.4. We leave it to future works to explore the numerical aspects.

8.1. Computing PageRank in Three Stages

We give a three-stage example for computing the PageRank vector.

As before, we have K nondangling nodes and $N-K$ dangling nodes, for a total of N nodes. Assume that among the nondangling nodes, $K2$ nodes point only to the dangling nodes. We will refer to these nodes as being *weakly dangling*. Thus, we have $N-K$ dangling nodes, $K2$ weakly dangling nodes, and $K1 = K - K2$ nodes that are neither. We denote these subsets by $\mathbb{S}_{\mathrm{D}}$, $\mathbb{S}_2$, and $\mathbb{S}_1$, respectively. (Note that $\mathbb{S}_1 \cup \mathbb{S}_2 = \mathbb{S}_{\mathrm{ND}}$.) Matrix $\mathbf{P}_{11}$ in (4.1) can be symmetrically permuted so that $\mathbb{S}_1$ occupies the leading part of $\mathbf{P}_{11}$ and $\mathbb{S}_2$ occupies the bottom:

$$\mathbf{P}_{11} = \begin{array}{c} \\ K1 \\ K2 \end{array} \begin{array}{c} \begin{array}{cc} \quad K1 \quad & \quad\quad K2 \quad \end{array} \\ \begin{pmatrix} \mathbf{P}_{11,11} & \mathbf{P}_{11,12} \\ (1-c)\mathbf{1}_{K2}\mathbf{u}_{K1}^T & (1-c)\mathbf{1}_{K2}\mathbf{u}_{K2}^T \end{pmatrix} \end{array}.$$

A three-stage algorithm that successively collapses pairs of subsets can be used to compute the PageRank vector.

First, by lumping we can combine $\mathbb{S}_{\mathrm{D}}$ and $\mathbb{S}_2$ each into a block. (Check that once $\mathbb{S}_{\mathrm{D}}$ is lumped, $\mathbb{S}_2$ becomes lumpable.) This gives rise to a transition probability matrix that is $(K1+2)$-by-$(K1+2)$. The limiting distribution can be computed, as before, by the power method. Upon convergence, $\boldsymbol{\pi}(i)$ for $i \in \mathbb{S}_1$, $\sum_{i\in\mathbb{S}_2} \boldsymbol{\pi}(i)$, and $\sum_{i\in\mathbb{S}_{\mathrm{D}}} \boldsymbol{\pi}(i)$ are obtained (Proposition 3.2). This is Stage 1.

In Stage 2, we consider a second Markov chain where we first lump $\mathbb{S}_{\mathrm{D}}$ into one block (which we know from Proposition 3.3 is lumpable) before aggregating this block with $\mathbb{S}_1$ to yield a single block. In doing so, we use $\boldsymbol{\pi}(i)$, $i \in \mathbb{S}_1$, and $\sum_{i\in\mathbb{S}_{\mathrm{D}}} \boldsymbol{\pi}(i)$ that we obtained in Stage 1 to calculate the aggregation weights. The resulting matrix is $(K2+1)$-by-$(K2+1)$ and rank-two. We know how to calculate its limiting distribution already. We now have $\boldsymbol{\pi}(i)$ for $i \in \mathbb{S}_{\mathrm{ND}} = \mathbb{S}_1 \cup \mathbb{S}_2$ (Proposition 3.4).

In Stage 3, our problem looks identical to the beginning of the second stage in the original two-stage algorithm. Once this stage is completed, we have the PageRank vector.

8.2. Differentiating Webpages by Class

The personalization vector has traditionally been used to model the preferences of the surfer. We may use multiple such vectors in a multistage extension of our algorithm to enable customization with respect to *classes of webpages*. The idea is as follows. In the PageRank model, each row of $\mathbf{P}$ amounts to transition probabilities out of the corresponding page. Each row, therefore, is really a conditional probability distribution parameterized by a page index. In the same vein, we can replace the personalization vector with a collection of vectors, each one representing the transition probabilities out of a corresponding *webpage class*. (Here, parameterization is with respect to a class index $m \in M$.) Instead of inserting the same personalization vector everywhere, we insert different vectors for different classes; ordinary PageRank is the special case when $M = 1$. We will call these vectors *personalized class vectors*. ("Personalized" because we can still use a different set of vectors for different individuals.) PageRank as a model for web traffic now has three components: a structural component (the link structure), a component for content (webpage class), and a component for surfer behavior (surfer preferences).

There are natural ways to form class associations between webpages. Possible associations are by domain, host, topic, language, or file type. A good example is association by language. Webpages belonging to the same language class are likely to have more traffic among themselves. In that case, one may model within-language traffic with high probabilities and between-language traffic with low probabilities.

We now describe a multistage algorithm for computing the PageRank vector under this model. Suppose that webpages are divided into M classes, which we denote as $\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_M$. In the permuted form of (4.1), what used to be the rank-one portion in the bottom of the matrix is now rank-M. We denote dangling nodes belonging to class m by $(\mathbb{C}_m \cap \mathbb{S}_{\mathrm{D}})$.

In Stage 1, we lump each $(\mathbb{C}_m \cap \mathbb{S}_{\mathrm{D}})$ into a single block. The lumpability here is trivially verifiable and completely analogous to Proposition 3.3. The resulting matrix is $(K + M)$-by-$(K + M)$, and we can compute its limiting distribution by the power method. Upon convergence, we obtain $\boldsymbol{\pi}(i)$ for $i \in \mathbb{S}_{\mathrm{ND}}$ and $\sum_{i \in (\mathbb{C}_m \cap \mathbb{S}_{\mathrm{D}})} \boldsymbol{\pi}(i)$ for $m = 1, 2, \ldots, M$. This is assured by Proposition 3.2.

For Stage $n + 1$, we work with a Markov chain where $(\mathbb{C}_m \cap \mathbb{S}_{\mathrm{D}})$, $m \neq n$, are each lumped into one block. The resulting $M - 1$ blocks are then aggregated with $\mathbb{S}_{\mathrm{ND}}$ to form a single block. The weights here are calculated from $\boldsymbol{\pi}(i)$ for $i \in \mathbb{S}_{\mathrm{ND}}$ and $\sum_{i \in (\mathbb{C}_m \cap \mathbb{S}_{\mathrm{D}})} \boldsymbol{\pi}(i)$ for $m \neq n$, which we have from Stage 1. We thus obtain a $(\mathrm{card}(\mathbb{C}_n \cap \mathbb{S}_{\mathrm{D}}) + 1)$-by-$(\mathrm{card}(\mathbb{C}_n \cap \mathbb{S}_{\mathrm{D}}) + 1)$ matrix that is rank-two. Its limiting distribution can be very efficiently computed by a procedure similar to Algorithm 3. This yields $\boldsymbol{\pi}(i)$, $i \in (\mathbb{C}_n \cap \mathbb{S}_{\mathrm{D}})$, as assured by Proposition 3.4.

The entire PageRank vector is available after $(M + 1)$ stages.

9. Concluding Remarks

We presented a two-stage algorithm for computing the PageRank vector. In the first stage, we focus on a Markov chain where dangling nodes are lumped into one block; in the second stage, we aggregate the nondangling nodes. Limiting distributions of these two chains give the PageRank vector. The bulk of the work is in computing the lumped chain. Numerical experimentation showed that the algorithm can finish in a fraction of the time than what's normally required. Furthermore, only a part of the transition probability matrix is explicitly formed at any given time. On machines where memory is limited, the performance gap will likely widen. We also examined the practice of excluding dangling nodes until the last stages of computation and found that it could fail in extreme cases. Our algorithm achieves the same desired speed-up while being probabilistically correct.

The algorithm has a divide-and-conquer theme that can be generalized. We gave two examples. In the first example, we showed that the PageRank vector could be computed using a three-stage algorithm. In the second example, we applied a multistage algorithm to a generalized version of the PageRank model where the personalization vector was replaced with a collection of personalized class vectors. In practice, the class vectors can be used to model the contribution of a webpage's transition probabilities associated with some underlying class of webpages. The resulting model allows greater versatility in the modeling of web traffic.

10. Appendix

As this example demonstrates, including the dangling nodes only during the last stages of computation can fail in extreme cases. Take as the link matrix

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Here, $K = 2$ and $N = 4$. Take $c = 0.85$ and $\mathbf{u}^T = ((1 - 3a) \quad a \quad a \quad a)$, where $a = \frac{43}{138}$. Thus,

$$\mathbf{P} = \begin{pmatrix} 0.0098 & 0.3301 & 0.3301 & 0.3301 \\ 0.8598 & 0.0467 & 0.0467 & 0.0467 \\ 0.0652 & 0.3116 & 0.3116 & 0.3116 \\ 0.0652 & 0.3116 & 0.3116 & 0.3116 \end{pmatrix}.$$

It can be shown that the limiting distribution for the leading 2-by-2 submatrix of $\mathbf{P}$ is $(0.61 \quad 0.39)$ while for the entire matrix it is $(0.25 \quad 0.25 \quad 0.25 \quad 0.25)$ (i.e., the leading 2-by-2 submatrix yields a worse starting iterate than the uniform distribution).

Acknowledgments. We thank Stephanie Leung (McKinsey & Co.), who led our software development, and Hector Garcia-Molina, Andreas Paepcke, Sriram Raghavan, and Gary Wesley of the Stanford WebBase project, who provided the data used in early versions of this manuscript. Data used in the current version were prepared by David Gleich (Stanford). We also benefitted from discussions with Sepandar Kamvar (Google), Wang Lam (Kosmix), Amy Langville (College of Charleston), Sebastiano Vigna (University of Milano), and Chen Grief (University of British Columbia). Our coauthor Gene Golub passed away recently and will be deeply missed.

Inquiries, including requests for source code, should be sent to Chris P. Lee, The Wharton School, University of Pennsylvania, 3730 Walnut Street, Suite 500, Philadelphia, PA 19104 (cpclee@wharton.upenn.edu).

References

[Arasu et al. 02] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. "PageRank Computation and the Structure of the Web: Experiments and Algorithms." In *Posters of the Eleventh International World Wide Web Conference*, no. 173. Available at http://www2002.org/CDROM/poster/173.pdf, 2002.

[Berkhin 05] P. Berkhin. "A Survey on PageRank Computing." *Internet Math.* 2:1 (2005), 73–120.

[Berman and Plemmons 94] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Philadelphia, PA: SIAM Press, 1994.

[Boldi et al. 05] P. Boldi, M. Santini, and S. Vigna. "PageRank as a Function of the Damping Factor." In *Proceedings of the 14th International Conference on World Wide Web*, pp. 557–566. New York: ACM Press, 2005.

[Cao and Stewart 85] W. L. Cao and W. J. Stewart. "Iterative Aggregation/Disaggregation Techniques for Nearly Uncoupled Markov Chains." *J. ACM* 32 (1985), 702–719.

[Corso et al. 04] G. M. Del Corso, A. Gulli, and F. Romani. "Exploiting Web Matrix Permutations to Speedup PageRank Computation." Technical report, Dipartimento di Informatica, University of Pisa, Italy, 2004.

[Corso et al. 05] G. M. Del Corso, A. Gulli, and F. Romani. "Fast PageRank Computation via a Sparse Linear System." *Internet Math.* 2:3 (2005), 251–273.

[Dayar and Stewart 97] T. Dayar and W. J. Stewart. "Quasi-Lumpability, Lower Bounding Coupling Matrices, and Nearly Completely Decomposable Markov Chains." *SIAM J. Matrix Anal. Appl.* 18:2 (1997), 482–498.

[Eiron et al. 04] N. Eiron, K. S. McCurley, and J. A. Tomlin. "Ranking the Web Frontier." In *Proceedings of the 13th International Conference on World Wide Web*, pp. 309–318. New York: ACM Press, 2004.

[Gleich and Zhukov 05] D. Gleich and L. Zhukov. "Scalable computing for Power Law Graphs: Experience with Parallel PageRank." Technical report, Yahoo!, 2005.

[Gleich et al. 04] D. Gleich, L. Zhukov, and P. Berkhin. "Fast Parallel PageRank: A Linear System Approach." Technical report, Yahoo!, 2004.

[Golub and Greif 06] G. H. Golub and C. Greif. “An Arnoldi-type Algorithm for Computing PageRank.” *BIT Numerical Mathematics* 46:4 (2006), 759–771.

[Golub and Loan 96] G. H. Golub and C. F. Van Loan. *Matrix Computation*, third edition. Baltimore, MD: Johns Hopkins University Press, 1996.

[Haveliwala and Kamvar 03] T. H. Haveliwala and S. D. Kamvar. “The Second Eigenvalue of the Google Matrix.” Technical report, Stanford University, 2003.

[Hirai et al. 00] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. “WebBase: A Repository of Web Pages.” In *Proceedings of the Ninth International World Wide Web Conference*, no. 296. Available at http://www9.org/w9cdrom/296/296.html, 2000.

[Horn and Johnson 85] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1985.

[Ipsen and Kirkland 06] I. C. F. Ipsen and S. Kirkland. “Convergence Analysis of a PageRank Updating Algorithm by Langville and Meyer.” *SIAM J. Matrix Anal. Appl.* 27:4 (2006), 952–967.

[Jeh and Widom 03] G. Jeh and J. Widom. “Scaling Personalized Web Search.” In *Proceedings of the 12th International Conference on World Wide Web*, pp. 271–279. New York: ACM Press, 2003.

[Kamvar et al. 03a] S. D. Kamvar, T. H. Haveliwala, and G. H. Golub. “Adaptive Methods for the Computation of PageRank.” Technical report, Stanford University, 2003.

[Kamvar et al. 03b] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. “Exploiting the Block Structure of the Web for Computing PageRank.” Technical report, Stanford University, 2003.

[Kamvar et al. 03c] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. “Extrapolation Methods for Accelerating PageRank Computations.” In *Proceedings of the 12th International Conference on World Wide Web*, pp. 261–270. New York: ACM Press, 2003.

[Kemeny and Snell 60] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. New York: D. Van Norstrand, 1960.

[Langville and Meyer 04] A. N. Langville and C. D. Meyer. “Updating PageRank with Iterative Aggregation.” In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pp. 392–393. New York: ACM Press 2004.

[Langville and Meyer 05a] A. N. Langville and C. D. Meyer. “Deeper inside PageRank.” *Internet Math.* 1:3 (2005), 335–400.

[Langville and Meyer 05b] A. N. Langville and C. D. Meyer. “A Survey of Eigenvector Methods for Web Information Retrieval.” *SIAM Rev* 47:1 (2005), 135–161.

[Langville and Meyer 06a] A. N. Langville and C. D. Meyer. *Google’s PageRank and Beyond: The Science of Search Engine Rankings*. Princeton, NJ: Princeton University Press, 2006.

[Langville and Meyer 06b] A. N. Langville and C. D. Meyer. "A Reordering for the PageRank Problem." *SIAM J. Sci. Comput.* 27:6 (2006), 2112–2120.

[Meyer 89] C. D. Meyer. "Stochastic Complementation, Uncoupling Markov Chains, and the Theory of Nearly Reducible Systems." *SIAM Rev.* 31:2 (1989), 240–272.

[Moler 02] C. Moler. "The World's Largest Matrix Computation." *MATLAB News & Notes* (October 2002), 12–13.

[Page et al. 98] L. Page, S. Brin, R. Motwani, and T. Winograd. "The PageRank Citation Ranking: Bringing Order to the Web." Technical report, Stanford Digital Library Technologies Project, 1998.

[Simon and Ando 61] H. A. Simon and A. Ando. "Aggregation of Variables in Dynamic Systems." *Econometrica* 29 (1961), 111–138.

[Yates et al. 05] R. B. Yates, P. Boldi, and C. Castillo. "The Choice of a Damping Function for Propagating Importance in Link-Based Ranking." Technical report, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, 2005.

[Zhu et al. 05] Y. Zhu, S. Ye, and X. Li. "Distributed PageRank Computation Based on Aggregation-Disaggregation Methods." In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pp. 578–585. New York: ACM Press, 2005.

Chris P. Lee, The Wharton School, University of Pennsylvania, 3730 Walnut Street, Suite 500, Philadelphia, PA 19104 (cpclee@wharton.upenn.edu)

Gene H. Golub[†] (November 16, 2007)

Stefanos A. Zenios, Graduate School of Business, Stanford University, 518 Memorial Way, Stanford, CA 94305-5015 (stefzen@leland.stanford.edu)

Received April 23, 2007; accepted February 24, 2008.