A Reordering for the PageRank problem

Amy N. Langville[†] and Carl D. Meyer[*]

March 2004

Abstract

We describe a reordering particularly suited to the PageRank problem, which reduces the computation of the PageRank vector to that of solving a much smaller system, then using forward substitution to get the full solution vector. We compare the theoretical rates of convergence of the original PageRank algorithm to that of the new reordered PageRank algorithm, showing that the new algorithm can do no worse than the original algorithm. We present results of an experimental comparison on five datasets, which demonstrate that the reordered PageRank algorithm can provide a speedup as much as a factor of 8. We also note potential additional benefits that result from the proposed reordering.

Key words: Markov chains, PageRank, reorderings, power method, convergence, stationary vector, dangling nodes

[†] Department of Mathematics,
N. Carolina State University,
Raleigh, NC 27695-8205, USA
anlangvi@unity.ncsu.edu
Phone: (919) 515-8146,
Fax: (919) 513-3798

[*] Department of Mathematics,
Center for Research in Scientific Computation,
N. Carolina State University, Raleigh, N.C. 27695-8205, USA
meyer@math.ncsu.edu
Phone: (919) 515-2384,
Fax: (919) 515-3798
Research supported in part by NSF CCR-ITR-0113121 and NSF DMS 9714811.

1 Introduction

It is well-known that many subsets of the web contain a large proportion of dangling nodes, webpages with no outlinks. Dangling nodes can result from many sources; a page containing an image, a postscript or pdf file, a page of data tables, or a page whose links have yet to be crawled by the search engine's spider. These dangling nodes can present philosophical, storage, and computational issues for a search engine such as Google that employs a ranking system for ordering retrieved webpages.

Let us introduce some notation. The hyperlink structure of the web defines a directed graph, which can be expressed as a sparse matrix. The founders of Google, Brin and Page, defined a hyperlink matrix as $\mathbf{P}_{i,j} = 1/|O_i|$, if there exists a hyperlink from page i to page j, and 0, otherwise. The scalar $|O_i|$ is the number of outlinks from page i. Thus, the nonzero rows of $\mathbf{P}$ sum to 1. The matrix $\mathbf{P}$ contains a $\mathbf{0}^T$ row for each dangling node. Brin and Page define the PageRank vector, a vector holding the global measure of importance for each page, to be the stationary vector for a Markov chain related to $\mathbf{P}$ [2, 3]. This definition is intuitive, as the stationary vector gives the long-run proportion of time the chain will spend in each state. For Brin and Page's application, the stationary elements represent the long-run proportion of time a random web surfer will spend on a page in the web, and thus, provides a measure of a page's importance or popularity. One problem with the hyperlink matrix $\mathbf{P}$ created strictly from the web's structure is that it is not stochastic, and a Markov chain is defined only for stochastic matrices. To remedy this and create the stochastic matrix called $\bar{\mathbf{P}}$, Brin and Page suggest replacing each $\mathbf{0}^T$ row (corresponding to a dangling node) of the sparse hyperlink matrix with a dense vector. The original fix for the dangling node problem used a uniform vector $\frac{1}{n}\mathbf{e}^T$, which was later replaced by the more general probability vector $\mathbf{v}^T$, known as the personalization or teleportation vector. Of course, if this suggestion were to be implemented explicitly, storage requirements would increase dramatically. Instead, the stochasticity fix can be modeled implicitly with the construction of one vector denoted $\mathbf{a}$. Element $a_i = 1$ if row i of $\mathbf{P}$ corresponds to a dangling node, and 0, otherwise. Then $\bar{\mathbf{P}}$ can be written as a rank-one update of $\mathbf{P}$; $\bar{\mathbf{P}} = \mathbf{P} + \mathbf{a}\mathbf{v}^T$.

Before the existence of the stationary PageRank vector could be guaranteed, Brin and Page had to make one final adjustment to the hyperlink matrix. Because the web is reducible, their original PageRank algorithm would often get caught in a rank sink, causing convergence problems. To remedy this, and thereby guaranteeing the existence and uniqueness of the PageRank vector, Brin and Page added another rank-one update, this time an irreducibility adjustment in the form of a dense perturbation matrix $\mathbf{e}\mathbf{v}^T$ that creates direct connections between each page. The stochastic, irreducible matrix called $\bar{\bar{\mathbf{P}}}$ is given by

$$\begin{aligned}\bar{\bar{\mathbf{P}}} &= \alpha\,\bar{\mathbf{P}} + (1-\alpha)\,\mathbf{e}\mathbf{v}^T \\ &= \alpha\,\mathbf{P} + \alpha\,\mathbf{a}\mathbf{v}^T + (1-\alpha)\,\mathbf{e}\mathbf{v}^T \\ &= \alpha\,\mathbf{P} + (\alpha\,\mathbf{a} + (1-\alpha)\,\mathbf{e})\mathbf{v}^T,\end{aligned}$$

where $0 < \alpha < 1$ and $\mathbf{e}$ is the vector of all ones. Brin and Page, and consequently, Google then use the power method to compute the unique stationary vector of this enormous Markov chain, thereby producing their famous PageRank vector.

We now turn to the philosophical issue of the presence of dangling nodes. In one of their early papers [1], Brin and Page report that they "often remove dangling nodes during the computation of PageRank, then add them back in after the PageRanks have converged." From this vague statement it is hard to say exactly how Brin and Page compute PageRank for the dangling nodes. However, the removal of dangling nodes at any time during the power method does not make intuitive sense. Some dangling nodes should receive high PageRank. For example, a very authoritative pdf file could have many inlinks from respected sources, and thus, should receive a high PageRank. Simply removing the dangling nodes biases the PageRank vector unjustly. (See [8] for additional arguments against removal of dangling nodes.) Further, incorporating dangling nodes into the PageRank power method is very simple and inexpensive. The power method follows the iterative formula below.

$$\begin{aligned}\mathbf{x}^{(k)T} &= \mathbf{x}^{(k-1)T}\bar{\bar{\mathbf{P}}} = \alpha\mathbf{x}^{(k-1)T}\bar{\mathbf{P}} + (1-\alpha)\mathbf{x}^{(k-1)T}\mathbf{e}\mathbf{v}^T \\ &= \alpha\mathbf{x}^{(k-1)T}\bar{\mathbf{P}} + (1-\alpha)\mathbf{v}^T\end{aligned}$$

$$= \quad \alpha\mathbf{x}^{(k-1)T}\mathbf{P} + \alpha\mathbf{x}^{(k-1)T}\mathbf{a}\mathbf{v}^T + (1-\alpha)\mathbf{v}^T, \tag{1}$$

since $x^{(k-1)T}$ is a probability vector, and thus, $\mathbf{x}^{(k-1)T}\mathbf{e} = 1$. This shows that the power method applied to $\bar{\bar{\mathbf{P}}}$ can be implemented with vector-matrix multiplications on the extremely sparse $\mathbf{P}$, and $\bar{\bar{\mathbf{P}}}$ and $\bar{\mathbf{P}}$ are never formed or stored. Since the vector-matrix multiplications involve enormous entities, an important question to ask is how many iterations can we expect until the power method converges to the PageRank vector. It has been proven that the rate of convergence of the power method applied to the Google matrix (after the stochasticity and irreducibility adjustments) is the rate at which $\alpha^k \to 0$. Since Google uses $\alpha = .85$, one can expect roughly 114 power iterations to give a convergence tolerance (as measured by the norm of the difference of successive iterates) less than 10^{-8}. Further, this means that on the order of $O(114 \cdot nnz(\mathbf{P}))$ operations must be performed to compute the PageRank vector with that prescribed tolerance, where $nnz(\mathbf{P})$ is the number of nonzeros in $\mathbf{P}$. Since $\mathbf{P}$ is so sparse, this is not a prohibitive cost, but it does nevertheless, take days of computation.

At this point, we have arrived at the computational issue associated with the dangling nodes. Since the dangling nodes form identical rows in the $\mathbf{P}$ matrix, they can be conveniently lumped into one class and the theory of lumpable and aggregated Markov chains can be used to compute the PageRank vector quickly. Lee et al. were the first to notice and exploit the structure provided by the dangling nodes [8]. Their iterative power method implementation on aggregated Markov chains led to drastic improvements in the computation of PageRank. On some sample datasets dangling nodes make up over 80% of the webpages, meaning the Lee et al. algorithm can compute PageRank with a factor of 5 speedup, since the aggregated chain is roughly 1/5 the size of the full chain.

In this paper, we present an analogous formulation of the Lee et al. algorithm (section 2). Our formulation uses a linear system formulation rather than a Markov chain formulation. Then we extend the idea of exploiting the dangling nodes to reduce computation, thereby producing a convenient reordering of the Google matrix, which has several advantageous properties (section 3). In section 4, we present the results of our reordering algorithm on several sample datasets.

2 A Linear System Formulation for exploiting Dangling Nodes

Although the size of the Google problem makes the power method one of the few practical solution methods for obtaining the PageRank vector, there are other formulations of the problem that are theoretically possible. The linear system formulation of the Google problem is

$$\boldsymbol{\pi}^T(\mathbf{I} - \alpha\mathbf{P}) = \mathbf{v}^T \quad \text{with} \quad \boldsymbol{\pi}^T\mathbf{e} = 1, \tag{2}$$

where $\boldsymbol{\pi}^T$ is the unknown PageRank vector.

The coefficient matrix $(\mathbf{I} - \alpha\mathbf{P})$ has many nice properties, which were proven in [7].

Properties of $(\mathbf{I} - \alpha\mathbf{P})$:

1. $(\mathbf{I} - \alpha\mathbf{P})$ is nonsingular.
2. $(\mathbf{I} - \alpha\mathbf{P})$ is an $\mathbf{M}$-matrix.
3. The row sums of $(\mathbf{I} - \alpha\mathbf{P})$ are either $1-\alpha$ for nondangling nodes or 1 for dangling nodes.
4. $\|\mathbf{I} - \alpha\mathbf{P}\|_\infty = 1 + \alpha$.

5. Since $(\mathbf{I}-\alpha\mathbf{P})$ is an $\mathbf{M}$-matrix, $(\mathbf{I}-\alpha\mathbf{P})^{-1}\geq 0$.
6. The row sums of $(\mathbf{I}-\alpha\mathbf{P})^{-1}$ are equal to 1 for the dangling nodes and less than or equal to $\frac{1}{1-\alpha}$ for the nondangling nodes.
7. The condition number $\kappa_\infty(\mathbf{I}-\alpha\mathbf{P})\leq\frac{1+\alpha}{1-\alpha}$.
8. The row of $(\mathbf{I}-\alpha\mathbf{P})^{-1}$ corresponding to dangling node i is $\mathbf{e}_i^T$, where $\mathbf{e}_i$ is the i^{th} column of the identity matrix.

The last property makes the computation of the PageRank vector especially efficient. Suppose the rows and columns of $\mathbf{P}$ are permuted (i.e., the indices are reordered) so that the rows corresponding to dangling nodes are at the bottom of the matrix.

$$\mathbf{P}=\begin{array}{cc} & \begin{array}{cc} ND & D \end{array} \\ \begin{array}{c} ND \\ D \end{array} & \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \end{array},$$

where ND is the set of nondangling nodes and D is the set of dangling nodes. Then the coefficient matrix in the sparse linear system formulation becomes

$$(\mathbf{I}-\alpha\mathbf{P})=\begin{pmatrix} \mathbf{I}-\alpha\mathbf{P}_{11} & -\alpha\mathbf{P}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

and the inverse of this matrix is

$$(\mathbf{I}-\alpha\mathbf{P})^{-1}=\begin{pmatrix} (\mathbf{I}-\alpha\mathbf{P}_{11})^{-1} & \alpha(\mathbf{I}-\alpha\mathbf{P}_{11})^{-1}\mathbf{P}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

Therefore, the PageRank vector $\boldsymbol{\pi}^T=\mathbf{v}^T(\mathbf{I}-\alpha\mathbf{P})^{-1}$ can be written as

$$\boldsymbol{\pi}^T=(\,\mathbf{v}_1^T(\mathbf{I}-\alpha\mathbf{P}_{11})^{-1}\quad|\quad\alpha\mathbf{v}_1^T(\mathbf{I}-\alpha\mathbf{P}_{11})^{-1}\mathbf{P}_{12}+\mathbf{v}_2^T\,),$$

where the personalization vector $\mathbf{v}^T$ has been partitioned into nondangling $(\mathbf{v}_1^T)$ and dangling $(\mathbf{v}_2^T)$ sections. In summary, we now have an algorithm that computes the PageRank vector using only the nondangling portion of the web, exploiting the rank-one structure of the dangling node fix.

Algorithm 1:

1. Solve for $\boldsymbol{\pi}_1^T$ in $\boldsymbol{\pi}_1^T(\mathbf{I}-\alpha\mathbf{P}_{11})=\mathbf{v}_1^T$.
2. Compute $\boldsymbol{\pi}_2^T=\alpha\boldsymbol{\pi}_1^T\mathbf{P}_{12}+\mathbf{v}_2^T$.
3. Normalize $\boldsymbol{\pi}^T=[\boldsymbol{\pi}_1^T\ \ \boldsymbol{\pi}_2^T]/\|[\boldsymbol{\pi}_1^T\ \ \boldsymbol{\pi}_2^T]\|_1$.

This algorithm is much simpler and cleaner, but equivalent, to the specialized iterative method proposed by Lee et al. [8], which exploits the dangling nodes to reduce computation of the PageRank vector, sometimes by a factor of 5.

3 A PageRank Algorithm based on a Reordering of the Google Matrix

3.1 The Reordered PageRank Algorithm

The linear system formulation of section 2 leads to a deeper examination of the structure of the Google matrix $\mathbf{P}$. Since the presence of zero rows in the matrix is so advantageous, might more zero rows be discovered in the submatrix $\mathbf{P}_{11}$, which is needed to solve the system in step 1 of Algorithm 1. This process of locating zero rows can be repeated recursively on smaller and smaller submatrices of $\mathbf{P}$, continuing until a submatrix is created that has no zero rows. The result of this process is a decomposition of the $\mathbf{P}$ matrix that looks like Figure 1. This process amounts to a simple reordering of the indices of the Markov chain. The left pane shows the original $\mathbf{P}$ matrix and the right pane is

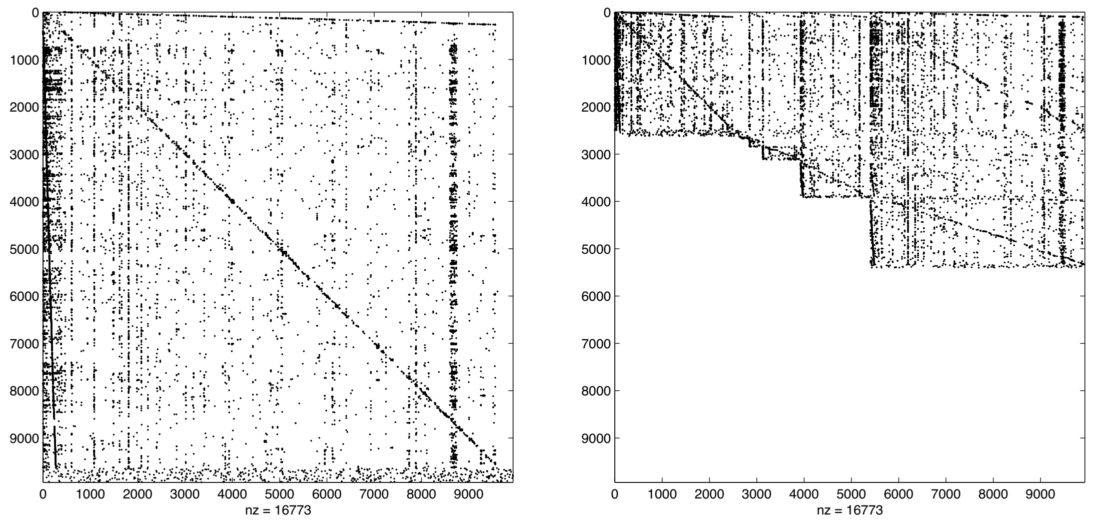

Figure 1: Original and reordered $\mathbf{P}$ matrix for `CA.dat`

the reordered matrix according to the recursive dangling node idea. The dataset `CA.dat` (available from `http://www.cs.cornell.edu/Courses/cs685/2002fa/`) is a typical subset of the web. It contains 9,664 nodes and 16,773 links, pertaining to the query topic of "california".

In general, after this symmetric reordering, the hyperlink matrix $\mathbf{P}$ has the following structure

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \cdots & \mathbf{P}_{1b} \\ & \mathbf{0} & \mathbf{P}_{23} & \cdots & \mathbf{P}_{2b} \\ & & \mathbf{0} & \cdots & \mathbf{P}_{3b} \\ & & & \ddots & \\ & & & & \mathbf{0} \end{pmatrix}, \tag{3}$$

where $b \geq 2$ is the number of square diagonal blocks in the reordered matrix. Therefore, the coefficient matrix of the linear system formulation of the PageRank problem (2) has the following structure.

$$(\mathbf{I}-\alpha\mathbf{P}) = \begin{pmatrix} \mathbf{I}-\alpha\mathbf{P}_{11} & -\alpha\mathbf{P}_{12} & -\alpha\mathbf{P}_{13} & \cdots & -\alpha\mathbf{P}_{1b} \\ & \mathbf{I} & -\alpha\mathbf{P}_{23} & \cdots & -\alpha\mathbf{P}_{2b} \\ & & \mathbf{I} & \cdots & -\alpha\mathbf{P}_{3b} \\ & & & \ddots & \\ & & & & \mathbf{I} \end{pmatrix}. \tag{4}$$

As a result, the PageRank system in equation (2) after reordering can be solved by forward substitution. The only system that must be solved directly is the first subsystem, $\boldsymbol{\pi}_1^T(\mathbf{I}-\alpha\mathbf{P}_{11}) = \mathbf{v}_1^T$, where $\boldsymbol{\pi}^T$ and $\mathbf{v}^T$ have also been partitioned according to the number and size of the blocks. The remaining subvectors of $\boldsymbol{\pi}^T$ are computed quickly and efficiently by forward substitution. In the `CA.dat` example, a $2,622\times 2,622$ system can be solved instead of the full $9,664 \times 9,664$ system, or even the once-reduced $5,132 \times 5,132$ system of Algorithm 1. The reordered PageRank algorithm is an extension of the dangling node method of section 2. This reordered PageRank idea can also be written in a Markov chain formulation, thereby extending the ideas in [8]. The steps of the reordered PageRank (in its linear system formulation) are enumerated in Algorithm 2.

Algorithm 2:

1. Reorder the states of the original Markov chain, so that the reordered matrix has the structure given in equation (3).
2. Solve for $\boldsymbol{\pi}_1^T$ in $\boldsymbol{\pi}_1^T(\mathbf{I}-\alpha\mathbf{P}_{11}) = \mathbf{v}_1^T$.
3. Compute $\boldsymbol{\pi}_2^T = \alpha\boldsymbol{\pi}_1^T\mathbf{P}_{12} + \mathbf{v}_2^T$.
4. Compute $\boldsymbol{\pi}_3^T = \alpha\boldsymbol{\pi}_1^T\mathbf{P}_{13} + \alpha\boldsymbol{\pi}_2^T\mathbf{P}_{23} + \mathbf{v}_3^T$.

   $\vdots$
5. Compute $\boldsymbol{\pi}_b^T = \alpha\boldsymbol{\pi}_1^T\mathbf{P}_{1b} + \alpha\boldsymbol{\pi}_2^T\mathbf{P}_{2b} + \cdots + \alpha\boldsymbol{\pi}_{b-1}^T\mathbf{P}_{b-1,b} + \mathbf{v}_b^T$.
6. Normalize $\boldsymbol{\pi}^T = [\boldsymbol{\pi}_1^T \;\; \boldsymbol{\pi}_2^T \;\; \cdots \;\; \boldsymbol{\pi}_b^T]/\|[\boldsymbol{\pi}_1^T \;\; \boldsymbol{\pi}_2^T \;\; \cdots \;\; \boldsymbol{\pi}_b^T]\|_1$.

3.2 Analysis of the Reordered PageRank Algorithm

Step 1 of Algorithm 2 is a modification to Tarjan's algorithm, which requires $O(nnz(\mathbf{P}))$ work. Recall that one power iteration takes $O(nnz(\mathbf{P}))$ operations, so the reordering amounts to one power iteration on the full matrix. The forward substitution steps of Algorithm 2, steps 3–5, amount to slightly less than the work of one power iteration. Thus, the small system solve of step 2 is the most time-consuming step. Some hyperlink matrices $\mathbf{P}$ can be particularly suited to the reordered PageRank algorithm. In this case, the large sets of zero rows are consecutively extracted, and the remaining $\mathbf{P}_{11}$ block is very small. In the worst case, $b = 2$ and $nnz(\mathbf{P}_{11})/nnz(\mathbf{P})$ is close to 1. Regardless, the smaller system $\boldsymbol{\pi}_1^T(\mathbf{I}-\alpha\mathbf{P}_{11}) = \mathbf{v}_1^T$ can be solved by any appropriate direct or iterative method. However, we have found that the sparsity of the $\mathbf{P}_{11}$ matrix makes a simple iterative method very attractive. The Jacobi method is especially simple due to the properties of the $\mathbf{P}_{11}$ matrix. The diagonal elements of $\mathbf{P}_{11}$ are 0, since by convention, no webpage links to itself. This means that the diagonal matrix $\mathbf{D}$ involved in the splitting of the coefficient matrix $(\mathbf{I}-\alpha\mathbf{P}_{11})$ is the identity matrix $\mathbf{I}$. Therefore, the Jacobi iterates are

$$\mathbf{x}^{(k)T} \quad = \quad \alpha\mathbf{x}^{(k-1)T}\mathbf{P}_{11} + \mathbf{v}_1^T.$$

This stationary iterative process converges to a unique solution for all starting iterates because $\rho(\alpha\mathbf{P}_{11}) \leq 1$. ($\mathbf{P}_{11}$ is substochastic.) The rate of convergence of the iterative reordered PageRank process is the same as or better than the rate of convergence of the standard PageRank power method. In fact, the reordered PageRank method will require about $O(\frac{log(\tau)}{log(\alpha)} \cdot nnz(\mathbf{P}_{11}))$ operations to reach a convergence tolerance of τ. In summary, Algorithm 2, the reordered PageRank method, requires roughly $\frac{nnz(\mathbf{P}_{11})}{nnz(\mathbf{P})}\%$ of the time that the original PageRank method requires. Further, acceleration methods, such as extrapolation and preconditioners, can be applied to the small $\mathbf{P}_{11}$ system to achieve even greater speedups.

4 Experiments with the Reordering Method

We compare the reordered PageRank algorithm to the original PageRank algorithm, experimenting on five datasets. The first three datasets are small, containing less than 10,000 pages. The remaining two dataset are much larger with one containing over 325,000 pages and the other over 450,000 pages. Each dataset is a different type of subset of the web. The `EPA.dat` dataset contains 5,042 pages linking to the EPA webpage. The `CA.dat` dataset contains 9,664 pages pertaining to the query topic of "california." `NCSU.dat` is a crawl that started at the N.C. State homepage and stopped once 10,000 pages were reached. Most of the pages are internal to the NCSU domain, but some are external. The dataset `ND.dat` contains 325,729 pages within the Notre Dame domain. Finally, `SU450k.dat` contains 451,237 pages from a 2001 crawl of the Stanford domain by the Stanford WebBase project.

The experiments in this section were conducted on a 867 MHz Mac G4 with 1.5 GB of RAM. We used $\alpha = .9$ as the scaling factor and $\tau = 10^{-10}$ as the convergence tolerance. The original PageRank algorithm and the reordered PageRank algorithm (Algorithm 2) were run on each dataset. The results are summarized in Table 1.

Table 1: Comparison of Original PageRank and Reordered PageRank algorithms

		`EPA.dat`	`CA.dat`	`NCSU.dat`	`ND.dat`	`SU450k.dat`
PageRank	Time (sec.)	3.80	9.63	13.17	177.16	237.37
	Iterations	159	176	162	166	164
	$n(\mathbf{P})$	5,042	9,664	10,000	325,729	451,237
	$nnz(\mathbf{P})$	9,563	16,873	101,118	1,497,134	1,082,604
Reordered PR	Time	.59	1.22	7.65	130.54	52.84
	Iterations	155	169	160	170	145
	b	10	9	5	18	12
	$n(\mathbf{P}_{11})$	704	2,622	7,136	127,472	84,861
	$nnz(\mathbf{P}_{11})$	1,330	5,238	79,230	1,191,761	267,566
Speedup	Estimate: $\frac{nnz(\mathbf{P})}{nnz(\mathbf{P}_{11})}$	7.2	6.4	1.3	1.3	4.0
	Actual (time)	**6.4**	**7.9**	**1.7**	**1.4**	**4.5**

The reordered PageRank algorithm beats the original PageRank algorithm by a factor of almost 8 on some datasets. The speedup depends on the ratio $\frac{nnz(\mathbf{P})}{nnz(\mathbf{P}_{11})}$, which depends on the properties of a particular dataset and the number of zero rows that can be successively squeezed out of the submatrices. The reordered PageRank algorithm is guaranteed to outperform the original PageRank method as long as some dangling nodes are present, i.e., $b \geq 2$.

5 Future Work

While the relationship between the spectrums of $\bar{\bar{\mathbf{P}}}$ and $\bar{\mathbf{P}}$ is known [4, 7], the relationship of these spectrums to that of $\mathbf{P}$ is not known. The reordering presented here may illuminate this unknown relationship. The symmetric reordering shows that the eigenvalues of $\mathbf{P}$ are the eigenvalues of $\mathbf{P}_{11}$ plus many 0 eigenvalues for the $\mathbf{0}$ diagonal blocks. The size of $\mathbf{P}_{11}$ may be small enough that its subdominant eigenvalues and eigenvectors can be computed. These subdominant eigenvectors can provide beneficial information about the community structure of a subset of the web [5]. Further, since $\bar{\mathbf{P}}$ is created from a rank-one update to $\mathbf{P}$, perturbation theory for eigenvalues and eigenvectors can provide additional information about the PageRank vector with respect to its original hyperlink structure ($\mathbf{P}$, not $\bar{\mathbf{P}}$ or $\bar{\bar{\mathbf{P}}}$).

Another area with potential benefit is the application of aggregation ideas to the reordered $\mathbf{P}$ matrix. Perhaps Perron complementation [10] or stochastic complementation [9] can be used for the updating problem [6]. One difficult hurdle is the reducibility of $\mathbf{P}$. However, it is possible that stochastic or Perron complements may be written ingeniously using rank-one update formulas.

6 Conclusions

Reorderings for linear systems have been well-studied. Some popular reorderings, such as the minimum degree reordering, the reverse Cuthill-McKee reordering, and the Dulmage–Mendelson reordering, do not give the nice structure of equation (4), which requires only one small system solve plus a quick forward substitution that amounts to the work of one power iteration. This reordering naturally produces an efficient alternative to the original PageRank algorithm called the reordered PageRank algorithm. The reordered algorithm has a rate of convergence that is as good as or better than the rate of convergence of the original method. The reordered algorithm produces a dataset-dependent speedup over the original algorithm. We presented experimental results comparing the reordered PageRank algorithm to the original PageRank algorithm on five datasets, showing a factor of 8 speedup on one dataset. This reordering and its linear system formulation also open the door for new PageRank acceleration techniques beyond the usual power method acceleration techniques.

Acknowledgements The paper [8] by Chris Lee of Stanford University and discussions with Michele Benzi of Emory University sparked the reordering idea presented in this paper. We thank Cleve Moler (Mathworks), Ronny Lempel (IBM Research Labs), and Jon Kleinberg (Cornell) for providing us with several small datasets that we used for testing. Our large dataset for the Stanford web crawl came from Chris Lee.

References

[1] Sergey Brin, Rajeev Motwani, Lawrence Page, and Terry Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21:37–47, 1998.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 33:107–117, 1998.

[3] Sergey Brin, Lawrence Page, R. Motwami, and Terry Winograd. The PageRank citation ranking: bringing order to the web. Technical report, Computer Science Department, Stanford University, 1998.

[4] Taher H. Haveliwala and Sepandar D. Kamvar. The second eigenvalue of the google matrix. Technical report, Stanford University, 2003.

[5] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46, 1999.

[6] Amy N. Langville and Carl D. Meyer. Updating the stationary vector of an irreducible Markov chain. Technical Report crsc02-tr33, N. C. State, Mathematics Dept., CRSC, 2002.

[7] Amy N. Langville and Carl D. Meyer. Deeper inside pagerank. *Internet Mathematics Journal*, 2003. Submitted in September 2003.

[8] Chris Pan-Chi Lee, Gene H. Golub, and Stefanos A. Zenios. Partial state space aggregation based on lumpability and its application to pagerank. Technical report, Stanford University, 2003.

[9] Carl D. Meyer. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM Review*, 31(2):240–272, 1989.

[10] Carl D. Meyer. Uncoupling the Perron eigenvector problem. *Linear Algebra and its Applications*, 114/115:69–74, 1989.