Updating PageRank using the Group Inverse and Stochastic Complementation

Amy N. Langville* and Carl D. Meyer†

November 19, 2002

Abstract

Since the seminal PageRank paper in 1998, [1], there has been an explosion in PageRank research by the information retrieval community. One open problem is the issue of updating PageRank. As the Web's pages are updated, so must the PageRanks of these pages be updated. In this paper, we propose three new updating techniques. The first two techniques are of theoretical value only, while the third technique is very promising and computationally practical. We use stochastic complementation along with aggregation/disaggregation methods to update PageRank exactly and in a very efficient manner. We demonstrate that our updating method severely outperforms the current updating technique of full recomputation. We conclude this research with suggestions for even greater improvements in updating.

Key words: Markov chains, information retrieval, PageRank, PageRank updating, group inverse, stochastic complementation, aggregation/disaggregation

*Department of Mathematics, Center for Research in Scientific Computation
N. Carolina State University, Raleigh, NC 27695-8205, USA
anlangvi@unity.ncsu.edu
Phone: (919) 513-4868, 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

In 1998, Brin and Page discovered their now famous PageRank information retrieval method, one of the first successful methods to exploit the hyperlink structure of the Web [1]. This method consists of two primary steps. First, a text scan is done over all webpages to select a subset of webpages containing query terms. Second, all webpages in this subset are sorted in decreasing order by their PageRank. Every

webpage is assigned, independent of all queries, a PageRank, a number between 0 and 1, capturing that page's importance. The PageRank concept posits that a page is important if several other important pages link back to it. This circular definition leads directly to a matrix eigenvector formulation. PageRank is computed for all webpages periodically by an eigenvector computation on a very large matrix. Soon after Brin and Page's success with *Google*, of which PageRank is the essential discriminating retrieval factor, other researchers discovered that the PageRank model was actually a Markov chain model [19, 23]. In fact, the PageRank vector containing the PageRank score of all webpages is actually the stationary solution vector of the underlying Markov chain. For example, for webpage 303, a PageRank score of .0925 can be interpretted in the Markov context as the proportion of time that a random Web surfer will spend viewing webpage 303. Thus, the higher the PageRank, the more important the document.

Once the Markov connnection to PageRank was discovered, several improvements to the PageRank idea were suggested. These were both computational and theoretical in nature [23, 19, 10]. In fact, Brin and Page's seminal paper started an explosion in PageRank research. Literally hundreds of papers exist on the topic of PageRank.

In 2001, Chien et al. wrote a paper of importance to the information retrieval community for two reasons [4]. First, the authors prove that PageRank is monotonic, thereby resolving an open question. The monotonicity of PageRank means that adding links into a page cannot decrease that page's PageRank. However, it is the second contribution of Chien et al. that is more pertinent to our work. The Web is an extremely volatile document collection with frequent updates. To address the Web's volatility, Chien et al. consider the problem of updating PageRank efficiently when changes are made to the link structure of the Web. They develop an algorithm that provides an approximate PageRank score without recomputation of stationary solution of the new, updated Markov chain. We improve their work, providing three new updating algorithms. The first two are of primarily theoretical value while the third is practical and very promising.

This paper is organized as follows. Section 2 introduces the Markov model of the Web and section 3 discusses the need for updating the model frequently. Section 4 briefly outlines Chien et al.'s method for updating the Markov model by approximating new PageRanks from old PageRanks. The remaining sections contain our contributions to updating PageRanks. Section 5 proposes an algorithm for exact updating of PageRanks using the group inverse [3, 12]. Section 6 compares the three major updating methods from a computational viewpoint. Section 7 discusses our second updating algorithm, an algorithm based on stochastic complementation. Section 8 returns to Chien et. al's updating method, highlighting its connection to stochastic complementation. And finally, by section 8.2, the foundation for our exact PageRank updating algorithm has been laid. We present our very promising iterative aggregation/disaggregation updating method and provide results from four test examples.

2 The Markov Model of the Web

We consider the hyperlink structure of the Web as a directed graph. The nodes represent webpages and the directed arcs represent hyperlinks. For example, consider the small document collection consisting of 6 webpages linked as in Figure 1.

The Markov model represents this graph with a square transition probability matrix $\mathbf{P}$ whose element p_{ij} is the probability of moving from state i (page i) to state j (page j) in one time step. For example, assume that, starting from any node (webpage), it is equally likely to follow any of the outgoing links to

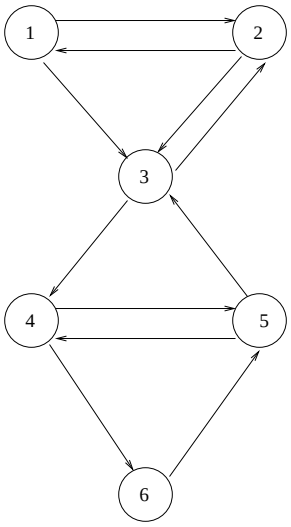

Figure 1: Directed graph representing web of 6 pages

arrive at another node. Thus,

$$\mathbf{P} = \begin{pmatrix} 0 & .5 & .5 & 0 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Any suitable probability distribution may be used across the rows. For example, if web usage logs show that a random surfer accessing page 2 is twice as likely to jump to page 1 as he or she is to jump to page 3, then the second row of $\mathbf{P}$, denoted $\mathbf{p}_2^T$, becomes

$$\mathbf{p}_2^T = (\,.6667 \quad 0 \quad .3333 \quad 0 \quad 0 \quad 0\,).$$

(Columns are similarly denoted. Column i of $\mathbf{P}$ is $\mathbf{p}_i$.) Brin and Page actually use the matrix $\bar{\mathbf{P}} = \alpha\mathbf{P} + \frac{(1-\alpha)}{n}\mathbf{E}$ in place of $\mathbf{P}$, where $0 \leq \alpha \leq 1$, n is the order of $\mathbf{P}$ and $\mathbf{E}$ is the matrix of all 1s. This linear combination of the original matrix $\mathbf{P}$ and a perturbation matrix $\mathbf{E}$ insures that $\bar{\mathbf{P}}$ is both stochastic and irreducible. From now on, when we use $\mathbf{P}$, the reader can assume that we are referring to Brin and Page's matrix $\bar{\mathbf{P}}$.

Regardless of the method for filling in the entries of $\mathbf{P}$, PageRank is determined by computing the stationary solution $\boldsymbol{\pi}^T$ of the Markov chain. The column vector $\boldsymbol{\pi}$ can be found by solving either the eigenvector problem

$$\boldsymbol{\pi}^T\mathbf{P} = \boldsymbol{\pi}^T,$$

or by solving the homogeneous linear system

$$\boldsymbol{\pi}^T(\mathbf{I} - \mathbf{P}) = \mathbf{0},$$

where $\mathbf{I}$ is the identity matrix. Both formulations are subject to an additional equation, the normalization equation $\boldsymbol{\pi}^T\mathbf{e} = 1$, where $\mathbf{e}$ is the column vector of all 1's. The normalization equation insures that $\boldsymbol{\pi}$ is a probability vector. The i^{th} element of $\boldsymbol{\pi}^T$, π_i, is the PageRank of page i.

The Markov model for the Web contains one state for each webpage. Thus, $\mathbf{P}$ is a $7,000,000,000 \times 7,000,000,000$ matrix since current estimates report 7 billion pages on the Web [6]. The sheer size of this matrix makes the webpage ranking problem challenging, fortunately, however, $\mathbf{P}$ is sparse. Brin and Page report reasonable times for finding the PageRank vector $\boldsymbol{\pi}$ despite using the notoriously slow power method [18]. Also, recall that the PageRank vector need only be computed once as PageRank is used as a post-processing information retrieval step to rank the importance of retrieved documents. One great advantage of PageRank is that it is query-independent; PageRanks will not change regardless of the content or number of queries entered by an information retrieval user. The only time a webpage's PageRank might change is when the actual structure of the Web is altered by updates.

3 Updates to the Web

One defining characteristic of the Web is its volatility. Literally thousands of modifications are made to the Web daily. Considering the Web as a graph, these modifications can be only one of two types: node modifications or link modifications. A node modification occurs when a webpage is added to or deleted from the Web. A link modification occurs when hyperlinks are added, deleted or changed. Each modification slightly changes the PageRank vector. Often, in practice, PageRank is computed once a month, thus accounting for Web changes on a monthly basis. However, as mentioned in the previous section, this PageRank calculation is computationally expensive due to the size of the $\mathbf{P}$ matrix. There is a tradeoff: PageRank should be recomputed as often as possible to give an accurate, up-to-date assessment of webpage rankings, yet PageRank takes a great deal of time and effort with each recomputation. One way to reduce the computational work of each recomputation is to exploit the previous PageRank computation, for example, from the previous month. The intuition here is that while several changes may be made to the Web over the period of a month, the vast majority of the Web's structure will remain unchanged.

Chien et al. suggest a method that exploits the unchanged part of the Web to approximate a new PageRank vector without the computation of the full PageRank vector of the updated Web [4]. Ng et. al. have also studied the PageRank updating problem [16, 17]. All these researchers have constrained themselves to the problem of link updates. Node updates create a much more difficult problem, since the size of the $\mathbf{P}$ matrix changes, whereas link updates only change the elements of $\mathbf{P}$. Addressing node updates is an interesting and necessary avenue for future work. For the remainder of the paper, the reader can interpret "updates" as link updates. We do not specifically address node updates until section 9. There we always use the term "node update" to distinguish it from link update.

4 Updating PageRank approximately

In this section, we detail the method for approximating the updated PageRank as described in [4]. The review of this method will serve the dual purpose of highlighting possible areas of improvement and introducing notation needed for the subsequent sections.

Consider again the small document collection of webpages depicted in Figure 1 with the PageRank stationary vector $\boldsymbol{\pi}^T$ given below.

$$\boldsymbol{\pi}^T = (\,.0741 \quad .1481 \quad .2222 \quad .2222 \quad .2222 \quad .1111\,)\,.$$

Now suppose one link update to this tiny Web is made, an edge from node 6 to node 4 is added. The new transition probability matrix $\tilde{\mathbf{P}}$ resembles the old transition matrix $\mathbf{P}$ exactly except the sixth row of $\tilde{\mathbf{P}}$ is [0 0 0 .5 .5 0]. The updated stationary vector $\tilde{\boldsymbol{\pi}}$ is

$$\tilde{\boldsymbol{\pi}}^T = (\,.0667 \quad .1333 \quad .2000 \quad .2667 \quad .2000 \quad .1333\,).$$

The goal in [4] is to approximate $\tilde{\boldsymbol{\pi}}^T$ without resorting to full recomputation by solving the updated system $\tilde{\boldsymbol{\pi}}^T\tilde{\mathbf{P}} = \tilde{\boldsymbol{\pi}}^T$. The method uses the known $\boldsymbol{\pi}^T$ and solves a much smaller Markov chain $\hat{\mathbf{P}}$ to derive the approximation to $\tilde{\boldsymbol{\pi}}^T$. Chien et al. propose that the six nodes be divided into two groups. One group Ω contains all the nodes not likely to have their PageRank affected by the link update, while G is the subset of nodes "near" the link update and likely to be affected. Determining which nodes belong to the supernode set Ω and which belong to the local graph set G is done by analyzing the transient behavior of the Markov chain [4]. A small Markov matrix $\hat{\mathbf{P}}$ is formed by lumping all nodes in Ω into one supernode with the same name. The elements $\hat{p}_{ij}$ are found by using the rules provided in [4]. The important point is that $\hat{\boldsymbol{\pi}}^T$ has been found by solving a generally much much smaller Markov chain and now the problem is to determine (or, if necessary, approximate) the correct, updated PageRank $\tilde{\boldsymbol{\pi}}^T$ given $\boldsymbol{\pi}^T$ and $\hat{\boldsymbol{\pi}}^T$. The rule given in [4] for creating an approximate PageRank vector $\bar{\boldsymbol{\pi}}^T$ is to let $\bar{\pi}_v = \pi_v$ for all $v \in \Omega$ and to let $\bar{\pi}_v = \hat{\pi}_v$ for all $v \in G$. For our example, using this rule and after normalizing the resulting vector so that a probability vector is obtained, one obtains the approximate PageRank vector $\bar{\boldsymbol{\pi}}$

$$\bar{\boldsymbol{\pi}}^T = (\,.0709 \quad .1418 \quad .2128 \quad .2554 \quad .1915 \quad .1276\,).$$

For this small example, we have previously computed the correct updated PageRank $\tilde{\boldsymbol{\pi}}^T$ so we can compare the approximate PageRank $\bar{\boldsymbol{\pi}}^T$ with $\tilde{\boldsymbol{\pi}}^T$. For this small example, the approximation $\bar{\boldsymbol{\pi}}^T$ does not even insure one digit of accuracy in the updated PageRanks. Chien et al. use $\|\tilde{\boldsymbol{\pi}}^T - \bar{\boldsymbol{\pi}}^T\|_1$ to judge the quality of their approximations. They note for their large, extensive examples that this distance is small, indicative of a good approximation. We feel this is the wrong measure of "goodness" or quality in this situation.

The 1-norm of the error is $\|\tilde{\boldsymbol{\pi}}^T - \bar{\boldsymbol{\pi}}^T\|_1 = \sum_{i=1}^{n} |\tilde{\pi}_i - \bar{\pi}_i|$, where n is the dimension of the vectors, or the number of webpages. We present the following example to demonstrate the inadequacies of the 1-norm as a measure of the quality of the approximation. Suppose

$$\begin{aligned}
\tilde{\boldsymbol{\pi}}^T &= (\,.199 \quad .199 \quad .199 \quad .199 \quad .199 \quad .005\,),\\
\bar{\boldsymbol{\pi}}^{(1)^T} &= (\,.199 \quad .199 \quad .199 \quad .199 \quad .194 \quad .010\,),\\
\bar{\boldsymbol{\pi}}^{(2)^T} &= (\,.199 \quad .204 \quad .194 \quad .199 \quad .200 \quad .006\,),
\end{aligned}$$

where $\bar{\boldsymbol{\pi}}^{(1)T}$ and $\bar{\boldsymbol{\pi}}^{(2)T}$ are different approximations to the correct PageRank $\tilde{\boldsymbol{\pi}}^T$ obtained from different $\Omega - G$ groupings of the nodes. Intuition predicts that $\bar{\boldsymbol{\pi}}^{(2)T}$ is the better approximation to $\tilde{\boldsymbol{\pi}}^T$. Notice that the sixth element of $\bar{\boldsymbol{\pi}}^{(1)T}$ is in error of the true PageRank for node 6 by 100%, while the sixth element of $\bar{\boldsymbol{\pi}}^{(2)T}$ is in error of the true PageRank for node 6 by 20%. While $\bar{\boldsymbol{\pi}}^{(1)T}$ may have more elements matching elements of $\tilde{\boldsymbol{\pi}}^T$, most readers will report $\bar{\boldsymbol{\pi}}^{(2)T}$ as closer to $\tilde{\boldsymbol{\pi}}^T$. However, using the 1-norm to gauge error, one might conclude that $\bar{\boldsymbol{\pi}}^{(1)T}$ is the better approximation since

$$\|\tilde{\boldsymbol{\pi}}^T - \bar{\boldsymbol{\pi}}^{(1)T}\|_1 = .010 \;<\; .012 = \|\tilde{\boldsymbol{\pi}}^T - \bar{\boldsymbol{\pi}}^{(2)T}\|_1.$$

The problem with the 1-norm in this context is that it uses absolute errors, whereas relative errors are more appropriate. A cumulative relative 1-norm is defined as

$$\text{relative 1-norm} = \sum_{i=1}^{n} \frac{|\tilde{\pi}_i - \bar{\pi}_i|}{\tilde{\pi}_i}.$$

The relative 1-norm of the error of approximation $\bar{\boldsymbol{\pi}}^{(1)T}$ is 1.0251, while the relative 1-norm of the error of approximation $\bar{\boldsymbol{\pi}}^{(2)T}$ is .2553. The number .2553 means that sum of entry-wise error percentages is

.2553. Clearly, the relative 1-norm is the better measure to use. A small absolute 1-norm does not give a good indication of the quality of the approximation. Therefore, the absolute 1-norm errors of $O(10^{-7})$ reported by Chien et al. do not tell the complete story. One should be cautious about inferring too much from their low reported errors.

Before introducing our improvements to the above method, we recap its advantages and disadvantages. The most advantageous aspect of the method is that $\tilde{\boldsymbol{\pi}}^T$ is approximated from only $\boldsymbol{\pi}^T$ and $\hat{\boldsymbol{\pi}}^T$. The new large system with $\tilde{\mathbf{P}}$ is never solved, only the much smaller system $\hat{\mathbf{P}}$ is used. However, two groups of nodes, Ω and G, must be formed and after this computation and the computation of $\hat{\boldsymbol{\pi}}^T$, merely an approximation $\bar{\boldsymbol{\pi}}^T$ to $\tilde{\boldsymbol{\pi}}^T$ is obtained. An approximation whose quality was questioned above. Our first suggestion to improving this method uses the group inverse to update $\boldsymbol{\pi}^T$, creating $\tilde{\boldsymbol{\pi}}^T$, exactly. However, this improvement is of theoretical value only, it provides no practical implementation benefits. Our second improvement uses stochastic complementation to produce exact updates, yet it, too, is very costly, bearing no practical value. Unlike our first two improvements, our final improvement does bear practical value. We use stochastic complementation and aggregation/disaggregation to update PageRank, and with very little additional computation update $\boldsymbol{\pi}^T$ to create the exact $\tilde{\boldsymbol{\pi}}^T$ vector. In the remaining sections, we present each method in turn, saving the best for last.

5 Improvement 1: Computing the updated PageRank $\tilde{\pi}^T$ exactly using the group inverse

In this section, we discuss our method for finding the new, updated PageRank $\tilde{\boldsymbol{\pi}}^T$ exactly.

We, like Chien et al., avoid solving the new large system with $\tilde{\mathbf{P}}$. However, the resulting vector obtained by our method gives $\tilde{\boldsymbol{\pi}}^T$ exactly, by using only $\boldsymbol{\pi}^T$ and $(\mathbf{I}-\mathbf{P})^{\#}$, where $(\mathbf{I}-\mathbf{P})^{\#}$ represents the group inverse of $(\mathbf{I}-\mathbf{P})$ [3, 12]. Unfortunately, the computation of $(\mathbf{I}-\mathbf{P})^{\#}$ is expensive. We hope that even though our group inverse exact updating method is impractical, it may generate new research possibilities with its theoretical contributions. We pause now to define the group inverse and give an algorithm for its computation.

5.1 Computation of the Group Inverse

Let $\mathbf{Q} = \mathbf{I} - \mathbf{P}$. The group inverse of $\mathbf{Q}$, $\mathbf{Q}^{\#}$, is defined as the unique matrix satisfying the three equations: $\mathbf{Q}\mathbf{Q}^{\#}\mathbf{Q} = \mathbf{Q}$, $\mathbf{Q}^{\#}\mathbf{Q}\mathbf{Q}^{\#} = \mathbf{Q}^{\#}$ and $\mathbf{Q}\mathbf{Q}^{\#} = \mathbf{Q}^{\#}\mathbf{Q}$. Meyer and his various coworkers have made it very clear that $\mathbf{Q}^{\#}$ is a fundamental quantity governing the sensitivity of the stationary distribution of an ergodic Markov chain. See [12, 3, 14, 7, 5, 8] for numerous theorems and examples. Reference [7] contains an algorithm for computing $\mathbf{Q}^{\#}$. Since our PageRank update method requires $\mathbf{Q}^{\#}$, we include the algorithm here.

Algorithm 1: Computing $\mathbf{Q}^{\#}$.

1. Obtain an $\mathbf{LU}$ decomposition of $\mathbf{Q}$.

$$\mathbf{Q} = \mathbf{LU}.$$

2. Obtain the stationary vector $\boldsymbol{\pi}^T$ from step 1 by backsubstitution.

3. Solve successively

$$\mathbf{LU}\mathbf{z}_i = \mathbf{e}_i - \pi_i \mathbf{e},$$

where $\mathbf{e}_i$ is a vector of all 0's except the i^{th} position contains a 1, $\mathbf{e}$ is the vector of all 1's and $\mathbf{z}_i$ is a column vector.

4. Normalize $\mathbf{z}_i$.

$$\mathbf{q}_i^{\#} = z_i - (\boldsymbol{\pi}^T \mathbf{z}_i)\mathbf{e}.$$

5. Adjoin the column vectors $\mathbf{q}_i^{\#}$ to form the $n \times n$ matrix $\mathbf{Q}^{\#}$.

The calculation of $\mathbf{Q}^{\#}$ requires $4n^3/3$ flops.

5.2 Updating PageRank exactly using the group inverse

We begin by stating a theorem from [14] that we need for our PageRank updating method.

Theorem 5.1 *(Theorem 4.3 from [14]): Let $\mathbf{P}$ be the transition matrix of an ergodic Markov chain and suppose that the i^{th} row of $\mathbf{P}$, $\mathbf{p}_i^T$, is modified to produce the matrix $\tilde{\mathbf{P}}$, which is also the transition matrix of an ergodic chain. If $\boldsymbol{\pi}^T$ and $\tilde{\boldsymbol{\pi}}^T$ denote the stationary probability vector of $\mathbf{P}$ and $\tilde{\mathbf{P}}$ respectively, then*

$$\tilde{\boldsymbol{\pi}}^T = \boldsymbol{\pi}^T - \pi_i \mathbf{b}_i^T,$$

where π_i is the i^{th} component of $\boldsymbol{\pi}$ and

$$\mathbf{b}_i^T = \frac{[\mathbf{p}_i^T - \tilde{\mathbf{p}}_i^T]\mathbf{Q}^{\#}}{1 + [\mathbf{p}_i^T - \tilde{\mathbf{p}}_i^T]\mathbf{q}_i^{\#}},$$

where $\mathbf{q}_i^{\#}$ is the i^{th} column of $\mathbf{Q}^{\#}$.

We now use theorem 5.1 to update the PageRank for our running example. Recall

$$\mathbf{P} = \begin{pmatrix} 0 & .5 & .5 & 0 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \pi^T = (\,.0741 \quad .1481 \quad .2222 \quad .2222 \quad .2222 \quad .1111\,).$$

The sixth row of $\mathbf{P}$ is altered so that a new transition matrix $\tilde{\mathbf{P}}$ is formed. Adding a link from node 6 to node 4 gives $\tilde{\mathbf{p}}_6^T = [\,0 \quad 0 \quad 0 \quad .5 \quad .5 \quad 0]$. We desire $\tilde{\boldsymbol{\pi}}^T$, the stationary vector for the updated $\tilde{\mathbf{P}}$ system. Using theorem 5.1,

$$\begin{aligned} \tilde{\boldsymbol{\pi}}^T &= \boldsymbol{\pi}^T - \pi_6 \frac{[\mathbf{p}_6^T - \tilde{\mathbf{p}}_6^T]\mathbf{Q}^{\#}}{1 + [\mathbf{p}_6^T - \tilde{\mathbf{p}}_6^T]\mathbf{q}_6^{\#}} \\ &= (\,0.0667 \quad 0.1333 \quad 0.2000 \quad 0.2666 \quad 0.2000 \quad 0.1333\,). \end{aligned}$$

We pause to summarize our findings. Given $\mathbf{P}$ and $\boldsymbol{\pi}^T$, we compute $\tilde{\boldsymbol{\pi}}^T$ exactly without solving the updated Markov system with $\tilde{\mathbf{P}}$. However, $\mathbf{Q}^{\#}$ must be computed instead. Have we gained anything with our method? Computationally, probably not, but theoretically, possibly. The next section outlines a method for updating $\mathbf{Q}^{\#}$, so that this expensive quantity need not be recomputed with each link update to the Web. To introduce the $\mathbf{Q}^{\#}$ updating method, we consider the case of multiple link updates.

5.3 Updating PageRank exactly for Multiple Link Updates

We begin this section by stating another theorem from [14], one which enables us to handle multiple link updates.

Theorem 5.2 *(Theorem 4.2 of [14]): Suppose that* $\mathbf{P}$ *is the transition matrix of an ergodic Markov chain and suppose that the* i^{th} *row of* $\mathbf{P}$, $\mathbf{p}_i^T$, *is modified producing* $\tilde{\mathbf{P}}$, *which is also the transition matrix of an ergodic Markov chain. Let* $\mathbf{Q} = \mathbf{I} - \mathbf{P}$ *and* $\tilde{\mathbf{Q}} = \mathbf{I} - \tilde{\mathbf{P}}$. *If* π_i *represents the* i^{th} *component of the stationary probability vector and* $\mathbf{e}$ *is the column vector of all 1's, then*

$$\tilde{\mathbf{Q}}^{\#} = \mathbf{Q}^{\#} + \pi_i \mathbf{e}\mathbf{b}_i^T[\mathbf{Q}^{\#} - \mathbf{b}_i^T \mathbf{q}_i^{\#} \mathbf{I}] - \mathbf{q}_i^{\#} \mathbf{b}_i^T,$$

where $\mathbf{b}_i^T$ *is defined as in Theorem 5.1.*

Given $\boldsymbol{\pi}^T$ and $\mathbf{Q}^{\#}$, both $\tilde{\boldsymbol{\pi}}^T$ and $\tilde{\mathbf{Q}}^{\#}$ can be obtained exactly with only $3n^2 + 2n$ multiplications for an n-state Markov chain. This means that with each link update, both the stationary vector and the group inverse should be updated.

Suppose four link updates are made to a given Web. In order to use Theorem 5.1 and obtain $\tilde{\boldsymbol{\pi}}^T$ without solving the $\tilde{\mathbf{P}}$ system, we must make sequential updates. Let $\tilde{\boldsymbol{\pi}}^{(j)T}$ and $[\mathbf{Q}^{(j)}]^{\#}$ represent the stationary vector and the group inverse, respectively, after the j^{th} update is made. After the first link update, $\tilde{\boldsymbol{\pi}}^{(1)T}$ is obtained. However, in proceeding to make the next update, we encounter a problem. Theorem 5.1 requires $[\mathbf{Q}^{(1)}]^{\#}$, the group inverse corresponding to the Markov system after the first update. This is worrisome. It implies that after the first update in which $\tilde{\boldsymbol{\pi}}^{(1)T}$ is obtained, $[\mathbf{Q}^{(1)}]^{\#}$ must be computed in order to obtain $\tilde{\boldsymbol{\pi}}^{(2)T}$. Fortunately, Meyer and Shoaf provide the saving theorem, theorem 5.2 above, for updating the group inverse [14]. After a link update, when $\boldsymbol{\pi}^T$ is updated to obtain $\tilde{\boldsymbol{\pi}}^T$, $\mathbf{Q}^{\#}$ must also be updated to obtain $\tilde{\mathbf{Q}}^{\#}$. When multiple link updates are made, with our new notation, this means with link update j, $\boldsymbol{\pi}^{(j)T}$ and $[\mathbf{Q}^{(j)}]^{\#}$ are updated to obtain $\boldsymbol{\pi}^{(j+1)T}$ and $[\mathbf{Q}^{(j+1)}]^{\#}$.

We outline our procedure for updating PageRank for the case of multiple link updates.

Algorithm 2: Updating PageRank with multiple link updates.

Begin with an initial Web graph with Markov matrix $\mathbf{P}^{(0)}$, stationary vector $\boldsymbol{\pi}^{(0)T}$ and group inverse $[\mathbf{Q}^{(0)}]^{\#}$. Assume link update j, $j = 1 : J$, where J is the total number of link updates, affects row i. For each link update j, do

1. Compute the change vector $\mathbf{c}_i^T$, a row vector.

$$\mathbf{c}_i^T = [\mathbf{p}_i^{T(j-1)} - \mathbf{p}_i^{T(j)}].$$

2. Compute $\mathbf{b}_i^T$.

$$\mathbf{b}_i^T = \frac{\mathbf{c}_i^T [\mathbf{Q}^{(j-1)}]^{\#}}{1 + \mathbf{c}_i^T [\mathbf{q}_i^{(j-1)}]^{\#}}.$$

3. Compute the updated PageRank vector $\boldsymbol{\pi}^{(j)^T}$.

$$\boldsymbol{\pi}^{(j)^T} = \boldsymbol{\pi}^{(j-1)^T} - \pi_i^{(j-1)} \mathbf{b}_i^T.$$

4. Compute the updated group inverse, $[\mathbf{Q}^{(j)}]^{\#}$, which is needed for the PageRank update corresponding to the next link update.

$$[\mathbf{Q}^{(j)}]^{\#} = [\mathbf{Q}^{(j-1)}]^{\#} + \pi_i^{(j-1)}\mathbf{e}\mathbf{b}_i^T \left\{[\mathbf{Q}^{(j-1)}]^{\#} - \mathbf{b}_i^T[\mathbf{q}_i^{(j-1)}]^{\#}\mathbf{I}\right\} - [\mathbf{q}_i^{(j-1)}]^{\#}\mathbf{b}_i^T.$$

There is no doubt that our algorithm 2 provides exact updates, as opposed to the approximate updates to PageRank proposed in [4]. However, we concede that in practice our algorithm 2 can not compete. The approximation method operates on the smaller matrix $\hat{\mathbf{P}}$ and is an $O(\hat{n}^2)$ algorithm where $\hat{n}$ is the size of $\hat{\mathbf{P}}$ and $\hat{n} << n$. In fact, full recomputation of the updated PageRanks using the updated $n \times n$ matrix $\tilde{\mathbf{P}}$ is superior computationally to our algorithm 2. The power method for computing $\tilde{\boldsymbol{\pi}}^T$ requires an unknown number of iterations, yet each iteration requires, in general, $O(n^2)$ multiplications, where n is the size of the original $\mathbf{P}$ and updated $\tilde{\mathbf{P}}$ matrices. However, due to the extreme sparsity of these matrices this is reduced to much much less than $O(n^2)$ multiplications. Also, since Brin and Page report good results in obtaining PageRanks with only 50 power iterations, the full recomputation method severely outperforms our algorithm 2. However, we hope that while our updating method based on the group inverse may be practically of little value, it may be theoretically valuable, sparking other opportunities for advancement in information retrieval.

In practice, the 7 billion-node Web has a very small proportion of "active" nodes with frequent link updates. For each active node, our exact PageRank update algorithm (algorithm 2) may be applied. When the number of active nodes is small, our algorithm may be a valuable alternative to the other updating methods. In the next section, we compare the computational effort and advantages of the three updating methods discussed so far (full recomputation, the method in [4] and our method).

6 Comparison of three PageRank updating methods

A detailed comparison of the various PageRank updating methods provides an indication of which method applies in a given situation. We refer to full recomputation of $\tilde{\boldsymbol{\pi}}^T$ using the power method as method 1. Method 2 is from [4] and method 3 is our group inverse algorithm 2. We discuss each method in turn.

6.1 Method 1: full recomputation of $\tilde{\pi}$ using the power method

Assumptions:

1. The power method converges to the accepted level of tolerance in 100 iterations. This is a reasonable assumption as Brin and Page [18] report acceptable convergence with 50 power iterations on a 322 million-node Web.
2. Since $\mathbf{P}$ is very sparse, vector-matrix multiplications can be reduced to much less than the standard n^2 multiplications. We define the average number of incoming links to a node as I. It is reasonable to assume that $I \leq 10$ [**?**]. Notice $I << n$.

Thus, each vector-matrix multiplication ($\mathbf{x}^T\mathbf{P}$) in an iteration of the power method requires an average of I^2 multiplications. The full recomputation of $\tilde{\boldsymbol{\pi}}^T$ then requires at most about $100In \leq 100(10)n = 1000n = O(n)$ multiplications, and, of course, gives the exact updated PageRanks.

6.2 Method 2: approximating $\tilde{\pi}^T$ using $\hat{\pi}^T$ and π^T

Assumptions:

1. The size $\hat{n}$ of $\hat{\mathbf{P}}$, after the nodes least likely to be affected by the link updates are lumped into one supernode, is much less than n, the size of the original $\mathbf{P}$ matrix. That is, $\hat{n} << n$.
2. The division of nodes into the two groups, Ω and G, is less than $O(n)$. This division is determined by analyzing the short-run behavior of the system near all link updates. In fact, [4] uses a breadth first search with a predetermined number of time steps near each link update to determine the subset of nodes belonging to G.

Then the computation of $\bar{\boldsymbol{\pi}}^T$, the approximation to $\tilde{\boldsymbol{\pi}}^T$, requires at most $O(n) + O(\hat{n}^2)$ time, assuming no sparsity in $\hat{\mathbf{P}}$ is exploited in the computation of $\hat{\boldsymbol{\pi}}^T$. In practice, using multiple small breadth first searches near each link update and exploiting the sparsity in $\hat{\mathbf{P}}$, this time can be reduced to much less than $O(n)$ time. However, this method is not an exact updating method. Method 2 must be used in conjunction with an exact updating method, like Method 1, to periodically return the approximations to exact PageRanks.

6.3 Method 3: group inverse updating method

Assumptions:

1. $\mathbf{Q}^{\#}$ is dense, as it almost always is, regardless of the sparsity of $\mathbf{P}$.
2. Updating $\boldsymbol{\pi}^T$ to form $\tilde{\boldsymbol{\pi}}^T$ and $\mathbf{Q}^{\#}$ to form $\tilde{\mathbf{Q}}^{\#}$ using Theorems 5.1 and 5.2 requires $3n^2 + 2n$ multiplications, where n is the size of $\mathbf{P}$. This can not, in general, be reduced, due to the density of $\mathbf{Q}^{\#}$.

Thus, the exact updating of PageRanks using the group inverse requires $J(3n^2 + 2n)$ multiplications for J link updates.

It becomes clear that the lack of sparsity of $\mathbf{Q}^{\#}$ prohibits our method 3 from becoming practically useful. Similarly, it is the extreme sparsity of $\mathbf{P}$ that makes the generally slow power method so prevalent in PageRank computation schemes. If, for some application or subset of the Web, $\mathbf{P}$ loses its sparsity and only a few link updates are needed, then perhaps our exact updating method may be preferred to the approximate updating method of [4].

7 Improvement 2: Updating PageRank exactly using stochastic complementation

Our second improvement for updating PageRanks, like our first improvement, is computationally impractical. Yet it does produces exact updates. This method, method 4, is based on stochastic complementation. We begin by defining this concept.

Definition 7.1 *(Definition 2.1 from [?]): Let* $\mathbf{P}$ *be an* $n \times n$ *irreducible stochastic matrix with a* k*-level partition*

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1k} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \cdots & \mathbf{P}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{k1} & \mathbf{P}_{k2} & \cdots & \mathbf{P}_{kk} \end{pmatrix}$$

in which all diagonal blocks are square. For a given index i*, let* $\mathbf{P}_i$ *denote the principal block submatrix of* $\mathbf{P}$ *obtained by deleting the* i^{th} *row and* i^{th} *column of blocks from* $\mathbf{P}$*, and let* $\mathbf{P}_{i*}$ *and* $\mathbf{P}_{*i}$ *designate*

$$\mathbf{P}_{i*} = \begin{pmatrix} \mathbf{P}_{i1} & \mathbf{P}_{i2} & \cdots & \mathbf{P}_{i,i-1} & \mathbf{P}_{i,i+1} \cdots \mathbf{P}_{ik} \end{pmatrix} \quad \textit{and} \quad \mathbf{P}_{*i} = \begin{pmatrix} \mathbf{P}_{1i} \\ \vdots \\ \mathbf{P}_{i-1,i} \\ \mathbf{P}_{i+1,i} \\ \vdots \\ \mathbf{P}_{ki} \end{pmatrix}.$$

That is, $\mathbf{P}_{i*}$ *is the* i^{th} *row of blocks with* $\mathbf{P}_{ii}$ *removed, and* $\mathbf{P}_{*i}$ *is the* i^{th} *column of blocks with* $\mathbf{P}_{ii}$ *removed. The* ***stochastic complement*** *of* $\mathbf{P}_{ii}$ *in* $\mathbf{P}$ *is defined to be the matrix*

$$\mathbf{S}_{ii} = \mathbf{P}_{ii} + \mathbf{P}_{i*}(\mathbf{I} - \mathbf{P}_i)^{-1}\mathbf{P}_{*i}.$$

For example, the stochastic complement of the $\mathbf{P}_{22}$ in the three-level partitioned matrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{pmatrix}$$

is

$$\mathbf{S}_{22} = \mathbf{P}_{22} + \begin{pmatrix} \mathbf{P}_{21} & \mathbf{P}_{23} \end{pmatrix} \begin{pmatrix} \mathbf{I} - \mathbf{P}_{11} & -\mathbf{P}_{13} \\ -\mathbf{P}_{31} & \mathbf{I} - \mathbf{P}_{33} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{P}_{12} \\ \mathbf{P}_{32} \end{pmatrix}.$$

Next we state the coupling theorem and show how this enables exact updating of PageRanks.

Theorem 7.1 *(The coupling theorem, theorem 4.1 from [?]): If* $\mathbf{P}$ *is an* $n \times n$ *irreducible stochastic matrix partitioned as*

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1k} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \cdots & \mathbf{P}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{k1} & \mathbf{P}_{k2} & \cdots & \mathbf{P}_{kk} \end{pmatrix}$$

with square diagonal blocks, then the stationary distribution vector for $\mathbf{P}$ *is given by*

$$\boldsymbol{\pi}^T = \begin{pmatrix} \xi_1 \mathbf{s}_1^T & \xi_2 \mathbf{s}_2^T & \cdots & \xi_k \mathbf{s}_k^T \end{pmatrix},$$

where $\mathbf{s}_i^T$ *is the unique stationary distribution vector for the stochastic complement*

$$\mathbf{S}_{ii} = \mathbf{P}_{ii} + \mathbf{P}_{i*}(\mathbf{I} - \mathbf{P}_i)^{-1}\mathbf{P}_{*i},$$

and where

$$\boldsymbol{\xi}^T = (\,\xi_1 \quad \xi_2 \quad \cdots \quad \xi_k\,),$$

is the unique stationary distribution vector for the $k \times k$ irreducible stochastic matrix $\mathbf{C}$ whose entries are defined by

$$c_{ij} = \mathbf{s}_i^T \mathbf{P}_{ij} \mathbf{e}.$$

*The matrix $\mathbf{C}$ is called the **coupling matrix** and the scalars ξ_i are called the coupling factors.*

For readers familiar with aggregation/disaggregation methods for nearly completely decomposable (NCD) Markov chains, the results of the above theorem can be viewed as an "exact aggregation" method. We now apply the coupling theorem to better understand the approximate updating method (method 2), this time from the viewpoint of stochastic complementation. Consider the tiny 6-node Web with transition probability matrix

$$\mathbf{P} = \begin{pmatrix} 0 & .5 & .5 & 0 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Assume one link update is made, a link from 6 to 4 is added, creating

$$\tilde{\mathbf{P}} = \begin{pmatrix} 0 & .5 & .5 & 0 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & .5 & .5 & 0 \end{pmatrix}.$$

As mentioned in section 4, we use the rules in [4] to group the nodes into two groups, Ω and G. Suppose $\Omega = \{1, 2, 3\}$ and $G = \{4, 5, 6\}$. We use this grouping to partition $\tilde{\mathbf{P}}$ into two levels, one for Ω and one for G.

$$\tilde{\mathbf{P}} = \left(\begin{array}{ccc|ccc} 0 & .5 & .5 & 0 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right).$$

Using definition 7.1, we form the two stochastic complements for $\tilde{\mathbf{P}}$

$$\tilde{\mathbf{S}}_{11} = \begin{pmatrix} 0 & 0.5000 & 0.5000 \\ 0.5000 & 0 & 0.5000 \\ 0 & 0.5000 & 0.5000 \end{pmatrix} \quad \text{and} \quad \tilde{\mathbf{S}}_{22} = \begin{pmatrix} 0 & 0.5000 & 0.5000 \\ 1 & 0 & 0 \\ .5000 & 0.5000 & 0 \end{pmatrix}.$$

The stationary distributions of the stochastic complements are

$$\tilde{\mathbf{s}}_1^T = (\,0.1667 \quad 0.3333 \quad 0.5000\,) \quad \text{and} \quad \tilde{\mathbf{s}}_2^T = (\,0.4445 \quad 0.3333 \quad 0.2222\,).$$

Thus, the coupling matrix $\tilde{\mathbf{C}}$ is formed according to the rule in Theorem 7.1 as

$$\tilde{\mathbf{C}} = \begin{pmatrix} .75 & .25 \\ .1667 & .8333 \end{pmatrix}.$$

The stationary distribution of this smaller Markov chain, called $\tilde{\boldsymbol{\xi}}^T$ in our coupling theorem language, is

$$\tilde{\boldsymbol{\xi}}^T = \begin{pmatrix} .4 & .6 \end{pmatrix}.$$

Thus, by the coupling theorem, the stationary distribution for $\tilde{\mathbf{P}}$ is

$$\begin{aligned} \tilde{\boldsymbol{\pi}}^T &= \begin{pmatrix} \tilde{\xi}_1\tilde{\mathbf{s}}_1^T & \tilde{\xi}_2\tilde{\mathbf{s}}_2^T \end{pmatrix} = \begin{pmatrix} .4 \cdot (0.1667 & 0.3333 & 0.5) & .6 \cdot (.4445 & .3333 & .2222) \end{pmatrix} \\ &= \begin{pmatrix} 0.0667 & 0.1333 & 0.2000 & .2667 & .2000 & .1333 \end{pmatrix}. \end{aligned}$$

Using stochastic complements, we have updated the PageRanks exactly. However, stochastic complements, much like the group inverse, are computationally expensive. Updating with stochastic complements requires obtaining the 2×1 stationary vector $\tilde{\boldsymbol{\xi}}^T$, the $|\Omega| \times 1$ stationary vector $\tilde{\mathbf{s}}_1^T$ and the $|G| \times 1$ stationary vector $\tilde{\mathbf{s}}_2^T$. The cardinality of G, $|G|$, is generally much much smaller than the cardinality of Ω, $|\Omega|$. Thus, we have traded finding one large stationary vector $\tilde{\boldsymbol{\pi}}^T$ for one slightly smaller stationary vector $\tilde{\mathbf{s}}_1^T$ and two very small stationary vectors $\tilde{\boldsymbol{\xi}}^T$ and $\tilde{\mathbf{s}}_2^T$. In addition, forming the stochastic complements, $\mathbf{S}_{11}$ and $\mathbf{S}_{22}$ requires matrix inversion of matrices of order $|G|$ and $|\Omega$. Thus, it appears that, like the group inverse updating method, updating exactly using stochastic complements may be of theoretical value only. Fortunately, there is a way to circumvent the costly computation of stochastic complementation yet still use the theory of stochastic complementation and the coupling theorem to update PageRanks. In the next section, we show the connection between the approximate updating method of [4] and stochastic complementation.

8 Relationship between Stochastic Complementation and the method of [4]

Instead of partitioning the nodes of $\tilde{\mathbf{P}}$ into two levels, partition them into n_G+1 levels, where $n_G = |G|$. The supernode comprises one level while each node in $G = \{g_1, g_2, \ldots, g_{n_G}\}$ creates its own level. This partitions the original $\mathbf{P}$ matrix into

$$\mathbf{P} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{ccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \cdots & \mathbf{P}_{1,n_G+1} \\ \mathbf{P}_{21}\mathbf{e} & \mathbf{P}_{22} & \mathbf{P}_{23} & \cdots & \mathbf{P}_{2,n_G+1} \\ \mathbf{P}_{31}\mathbf{e} & \mathbf{P}_{32} & \mathbf{P}_{33} & \cdots & \mathbf{P}_{3,n_G+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{n_G+1,1} & \mathbf{P}_{n_G+1,2} & \mathbf{P}_{n_G+1,3} & \cdots & \mathbf{P}_{n_G+1,n_G+1} \end{pmatrix} \end{array}$$

and the updated $\tilde{\mathbf{P}}$ matrix into

$$\tilde{\mathbf{P}} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{ccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \cdots & \mathbf{P}_{1,n_G+1} \\ \tilde{\mathbf{P}}_{\mathbf{21}}\mathbf{e} & \tilde{\mathbf{P}}_{\mathbf{22}} & \tilde{\mathbf{P}}_{\mathbf{23}} & \cdots & \tilde{\mathbf{P}}_{\mathbf{2,n_G+1}} \\ \tilde{\mathbf{P}}_{\mathbf{31}}\mathbf{e} & \tilde{\mathbf{P}}_{\mathbf{32}} & \tilde{\mathbf{P}}_{\mathbf{33}} & \cdots & \tilde{\mathbf{P}}_{\mathbf{3,n_G+1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{P}}_{\mathbf{n_G+1,1}} & \tilde{\mathbf{P}}_{\mathbf{n_G+1,2}} & \tilde{\mathbf{P}}_{\mathbf{n_G+1,3}} & \cdots & \tilde{\mathbf{P}}_{\mathbf{n_G+1,n_G+1}} \end{pmatrix} \end{array}.$$

Note that only links of nodes in G can be removed or added, thus $\tilde{\mathbf{P}}_{1,j} = \mathbf{P}_{1,j}$ for all $j = 1, 2, \ldots, n_G+1$.

Reviewing the previous section's rules for forming the stochastic complements reveals that $\mathbf{S}_{11}$ is an $|\Omega| \times |\Omega|$ matrix while $\mathbf{S}_{22}, \mathbf{S}_{33}, \ldots, \mathbf{S}_{n_G+1,n_G+1}$ are 1×1 matrices. Reference [?] proves that $\mathbf{S}_{ii}$ are always stochastic, thus $\mathbf{S}_{22} = \mathbf{S}_{33} = \ldots = \mathbf{S}_{n_G+1,n_G+1} = 1$. The stationary vectors $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{n_G+1}$ for these matrices are also 1. The same holds for $\tilde{\mathbf{S}}_{22}, \tilde{\mathbf{S}}_{33}, \ldots, \tilde{\mathbf{S}}_{n_G+1,n_G+1}$ and $\tilde{\mathbf{s}}_2, \tilde{\mathbf{s}}_3, \ldots, \tilde{\mathbf{s}}_{n_G+1}$.

We now compare the coupling matrix $\mathbf{C}$ of stochastic complementation with the $\hat{\mathbf{P}}$ matrix of [4]. Using the rules of Theorem 7.1, the coupling matrix $\mathbf{C}$ corresponding to the original $\mathbf{P}$ system is

$$\mathbf{C} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{ccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \mathbf{s}_1^T\mathbf{P}_{11}\mathbf{e} & \mathbf{s}_1^T\mathbf{P}_{12} & \mathbf{s}_1^T\mathbf{P}_{13} & \cdots & \mathbf{s}_1^T\mathbf{P}_{1,n_G+1} \\ \mathbf{P}_{21}\mathbf{e} & \mathbf{P}_{22} & \mathbf{P}_{23} & \cdots & \mathbf{P}_{2,n_G+1} \\ \mathbf{P}_{31}\mathbf{e} & \mathbf{P}_{32} & \mathbf{P}_{33} & \cdots & \mathbf{P}_{3,n_G+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{n_G+1,1}\mathbf{e} & \mathbf{P}_{n_G+1,2} & \mathbf{P}_{n_G+1,3} & \cdots & \mathbf{P}_{n_G+1,n_G+1} \end{pmatrix} \end{array}$$

and the coupling matrix $\tilde{\mathbf{C}}$ for the updated system is

$$\tilde{\mathbf{C}} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{ccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \tilde{\mathbf{s}}_1^T\mathbf{P}_{11}\mathbf{e} & \tilde{\mathbf{s}}_1^T\mathbf{P}_{12} & \tilde{\mathbf{s}}_1^T\mathbf{P}_{13} & \cdots & \tilde{\mathbf{s}}_1^T\mathbf{P}_{1,n_G+1} \\ \tilde{\mathbf{P}}_{21}\mathbf{e} & \tilde{\mathbf{P}}_{22} & \tilde{\mathbf{P}}_{23} & \cdots & \tilde{\mathbf{P}}_{2,n_G+1} \\ \tilde{\mathbf{P}}_{31}\mathbf{e} & \tilde{\mathbf{P}}_{32} & \tilde{\mathbf{P}}_{33} & \cdots & \tilde{\mathbf{P}}_{3,n_G+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{P}}_{n_G+1,1}\mathbf{e} & \tilde{\mathbf{P}}_{n_G+1,2} & \tilde{\mathbf{P}}_{n_G+1,3} & \cdots & \tilde{\mathbf{P}}_{n_G+1,n_G+1} \end{pmatrix} \end{array}.$$

Chien et. al's version of the coupling matrix is $\hat{\mathbf{P}}$. Using the rules in [4],

$$\hat{\mathbf{P}} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{ccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \mathbf{s}_1^T\mathbf{P}_{11}\mathbf{e} & \mathbf{s}_1^T\mathbf{P}_{12} & \mathbf{s}_1^T\mathbf{P}_{13} & \cdots & \mathbf{s}_1^T\mathbf{P}_{1,n_G+1} \\ \tilde{\mathbf{P}}_{21}\mathbf{e} & \tilde{\mathbf{P}}_{22} & \tilde{\mathbf{P}}_{23} & \cdots & \tilde{\mathbf{P}}_{2,n_G+1} \\ \tilde{\mathbf{P}}_{31}\mathbf{e} & \tilde{\mathbf{P}}_{32} & \tilde{\mathbf{P}}_{33} & \cdots & \tilde{\mathbf{P}}_{3,n_G+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{P}}_{n_G+1,1}\mathbf{e} & \tilde{\mathbf{P}}_{n_G+1,2} & \tilde{\mathbf{P}}_{n_G+1,3} & \cdots & \tilde{\mathbf{P}}_{n_G+1,n_G+1} \end{pmatrix} \end{array}.$$

Notice that $\hat{\mathbf{P}}$ is neither the coupling matrix for the original system $\mathbf{P}$ nor the coupling matrix for the updated system $\tilde{\mathbf{P}}$. However, $\hat{\mathbf{P}}$ differs from $\tilde{\mathbf{C}}$ only by its first row. $\hat{\mathbf{P}}$ can be considered an approximate coupling matrix for the updated system. In fact, there are two cases that arise. In some rare examples, $\mathbf{s}_1^T = \tilde{\mathbf{s}}_1^T$ and thus $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$, implying that the approximation method of [4] actually produces all information needed for exact updating. The more common case is that $\hat{\mathbf{P}} \approx \tilde{\mathbf{C}}$. We discuss these two cases.

8.1 Case 1: $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$

Case 1 occurs when $\mathbf{s}_1^T = \tilde{\mathbf{s}}_1^T$, which happens when $\mathbf{S}_{11} = \tilde{\mathbf{S}}_{11}$. In general, one can not expect the first stochastic complements for the original system and the updated system to be equal. However, Theorem 8.1, presented later, provides conditions under which this event occurs. If $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$, then the approximate updating method of [4] can be modified to give exact updates. By the theory of stochastic complementation,

$$\begin{aligned} \tilde{\boldsymbol{\pi}}^T &= (\tilde{\boldsymbol{\pi}}_\Omega^T \mid \tilde{\boldsymbol{\pi}}_G^T) \\ &= (\tilde{\boldsymbol{\xi}}_1 \tilde{\mathbf{s}}_1^T \mid \tilde{\boldsymbol{\xi}}_2 \tilde{\mathbf{s}}_2^T \;\; \tilde{\boldsymbol{\xi}}_3 \tilde{\mathbf{s}}_3^T \;\; \cdots \;\; \tilde{\boldsymbol{\xi}}_{n_G+1} \tilde{\mathbf{s}}_{n_G+1}^T) \\ &= (\tilde{\boldsymbol{\xi}}_1 \mathbf{s_1}^T \mid \tilde{\boldsymbol{\xi}}_2 \tilde{\boldsymbol{\xi}}_3 \;\; \cdots \;\; \tilde{\boldsymbol{\xi}}_{n_G+1}), \end{aligned}$$

where $\tilde{\boldsymbol{\xi}}^T$ is the stationary vector of $\hat{\mathbf{P}}$. Therefore, elements 2 through n_G+1 in $\tilde{\boldsymbol{\xi}}^T$ are the exact updated PageRanks. To obtain the exact updated PageRanks corresponding to nodes in Ω, one simply scales the old $\mathbf{s}_1^T = \frac{\boldsymbol{\pi}_\Omega^T}{\boldsymbol{\pi}_\Omega^T \mathbf{e}}$ by $\tilde{\boldsymbol{\xi}}_1$. The method of [4] does not recognize this relationship. Despite producing all the values needed to construct the exact updated PageRanks, it produces approximate PageRanks instead. The method of [4] uses the stationary vector from $\hat{\mathbf{P}}$, called $\hat{\boldsymbol{\pi}}^T$, to form $\bar{\boldsymbol{\pi}}^T$, its approximation to $\tilde{\boldsymbol{\pi}}^T$. In fact, $\bar{\pi}_v = \pi_v$ for all $v \in \Omega$ and $\bar{\pi}_v = \hat{\pi}_v$ for all $v \in G$. Then a normalization step, $\bar{\boldsymbol{\pi}}^T = \frac{\bar{\boldsymbol{\pi}}^T}{\bar{\boldsymbol{\pi}}^T \mathbf{e}}$, is applied to insure the probabilities sum to 1. It is this normalization step that steers the approximation method of [4] off-course. The theory of stochastic complementation proves that the PageRanks obtained from solving the condensed system $\hat{\boldsymbol{\pi}}^T \mathbf{P} = \hat{\boldsymbol{\pi}}^T$ (or equivalently, the coupled system $\tilde{\boldsymbol{\xi}}^T \mathbf{C} = \tilde{\boldsymbol{\xi}}^T$) are the exact updated PageRanks for each node in G. Normalizing only moves these PageRanks further away from their true values. Our modification to the method of [4] is simple: when $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$, let $\bar{\boldsymbol{\pi}}_G^T = \hat{\boldsymbol{\pi}}_G^T$ and let $\bar{\boldsymbol{\pi}}_\Omega^T = \frac{\boldsymbol{\pi}_\Omega^T}{\boldsymbol{\pi}_\Omega^T \mathbf{e}} \cdot \hat{\pi}_1$. By the theory of stochastic complementation, $\bar{\boldsymbol{\pi}}^T = \tilde{\boldsymbol{\pi}}^T$.

8.1.1 Updating Example for Case 1

We now return to the 6-node example from section 7. This example satisfies case 1 ($\hat{\mathbf{P}} = \tilde{\mathbf{C}}$) and shows that the most costly computation in the stochastic complementation method, computing the large stationary vector $\mathbf{s}_1$, can be avoided by our simple modification to the approximate updating method of [4].

Recall that the original $\mathbf{P}$ matrix is

$$\mathbf{P} = \begin{pmatrix} 0 & .5 & .5 & 0 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

One link update is made, creating the updated matrix $\tilde{\mathbf{P}}$.

$$\tilde{\mathbf{P}} = \begin{pmatrix} 0 & .5 & .5 & 0 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & .5 & .5 & 0 \end{pmatrix}.$$

The six nodes are grouped into two classes: those nodes likely to be affected by the link update are put into G, while those likely to be unaffected are in Ω. Suppose $\Omega = \{1, 2, 3\}$ and $G = \{4, 5, 6\}$. Then $\tilde{\mathbf{P}}$ can be partitioned into 4 levels; one level for each node in G and one additional level for all the nodes lumped into Ω.

$$\tilde{\mathbf{P}} = \left(\begin{array}{ccc|c|cc} 0 & .5 & .5 & 0 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & .5 & .5 \\ \hline 0 & 0 & .5 & .5 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right).$$

Using definition 7.1, we form the four stochastic complements for $\tilde{\mathbf{P}}$

$$\tilde{\mathbf{S}}_{11} = \begin{pmatrix} 0 & 0.5000 & 0.5000 \\ 0.5000 & 0 & 0.5000 \\ 0 & 0.5000 & 0.5000 \end{pmatrix}, \quad \tilde{\mathbf{S}}_{22} = 1, \quad \tilde{\mathbf{S}}_{33} = 1, \quad \text{and} \quad \tilde{\mathbf{S}}_{44} = 1.$$

The stationary distributions of the stochastic complements are

$$\tilde{\mathbf{s}}_1^T = (\,0.1667 \quad 0.3333 \quad 0.5000\,), \quad \tilde{\mathbf{s}}_2^T = 1, \quad \tilde{\mathbf{s}}_3^T = 1, \quad \text{and} \quad \tilde{\mathbf{s}}_4^T = 1.$$

Thus, the coupling matrix $\tilde{\mathbf{C}}$ is formed according to the rule in Theorem 7.1 as

$$\tilde{\mathbf{C}} = \begin{pmatrix} .75 & .25 & 0 & 0 \\ 0 & 0 & .5 & .5 \\ .5 & .5 & 0 & 0 \\ 0 & .5 & .5 & 0 \end{pmatrix}.$$

Using definition 7.1 to form the stochastic complements and stationary distributions for $\mathbf{P}$, one can verify that $\tilde{\mathbf{S}}_{11} = \mathbf{S}_{11}$, and thus $\tilde{\mathbf{s}}_1^T = \mathbf{s}_1^T$, which implies $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$. The stationary vector of the coupling matrix, called $\tilde{\boldsymbol{\pi}}^T$ in the language of stochastic complementation and $\hat{\boldsymbol{\pi}}^T$ in the language of [4], is

$$\tilde{\boldsymbol{\xi}}^T = \hat{\boldsymbol{\pi}}^T = (\,.4 \quad .2667 \quad .2 \quad .1333\,).$$

Using our modification to the method of [4], the exact updated PageRanks can be obtained without forming any of the costly matrices or stationary vectors of stochastic complementation. In fact, $\tilde{\mathbf{s}}_1^T$ is known, since $\tilde{\mathbf{s}}_1^T = \mathbf{s}_1^T = \frac{\boldsymbol{\pi}_\Omega^T}{\boldsymbol{\pi}_\Omega^T \mathbf{e}}$, the normalized old PageRanks for the set Ω.

$$\begin{aligned} \tilde{\boldsymbol{\pi}}^T &= (\,\tilde{\boldsymbol{\pi}}_\Omega^T \;\mid\; \tilde{\boldsymbol{\pi}}_G^T\,) \\ &= (\,\tilde{\boldsymbol{\xi}}_1 \mathbf{s_1}^T \;\mid\; \tilde{\xi}_2 \quad \tilde{\xi}_3 \quad \tilde{\xi}_4\,) \\ &= (\,.4 \cdot (0.1667 \;\; 0.3333 \;\; 0.5) \quad .2667 \cdot 1 \quad .2 \cdot 1 \quad .1333 \cdot 1\,) \\ &= (\,0.0667 \quad 0.1333 \quad 0.2000 \quad .2667 \quad .2000 \quad .1333\,). \end{aligned}$$

In summary, realizing the connection between the method of [4] and stochastic complementation allows us to update PageRanks exactly with no additional work beyond the work of [4]. However, this method is severely limited, as it only applies in the event that $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$. In the next subsection, we discuss conditions under which this event occurs.

8.1.2 Conditions under which $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$

We know that $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$ if $\tilde{\mathbf{s}}_1^T = \mathbf{s}_1^T$, which occurs when $\tilde{\mathbf{S}}_{11} = \mathbf{S}_{11}$. The following theorem formally states the conditions on $\tilde{\mathbf{P}}$ that lead to the event $\tilde{\mathbf{S}}_{11} = \mathbf{S}_{11}$.

Theorem 8.1 *Assume the n nodes are partitioned into two sets, Ω and G. This partitions the matrix $\mathbf{P}$ into*

$$\mathbf{P} = \left(\begin{array}{c|c} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \hline \mathbf{P}_{21} & \mathbf{P}_{22} \end{array}\right)$$

with a conformably partitioned PageRank vector

$$\boldsymbol{\pi}^T = (\, \boldsymbol{\pi}_\Omega^T \quad | \quad \boldsymbol{\pi}_G^T \,).$$

Some link updates are made to the Web, resulting in a new matrix $\tilde{\mathbf{P}}$,

$$\tilde{\mathbf{P}} = \left(\begin{array}{c|c} \tilde{\mathbf{P}}_{11} & \tilde{\mathbf{P}}_{12} \\ \hline \mathbf{P}_{21} & \mathbf{P}_{22} \end{array}\right)$$

with a conformably partitioned PageRank vector

$$\tilde{\boldsymbol{\pi}}^T = (\, \tilde{\boldsymbol{\pi}}_\Omega^T \quad | \quad \tilde{\boldsymbol{\pi}}_G^T \,).$$

Suppose the nodes are partitioned into the sets Ω and G so that the following assumptions hold.

1. $\mathbf{P}_{11} = \tilde{\mathbf{P}}_{11}$.
2. $\mathbf{P}_{12} = \tilde{\mathbf{P}}_{12}$.
3. $\mathbf{P}_{21} = \tilde{\mathbf{P}}_{21}$.
4. $\mathbf{P}_{21}$ *contains exactly one nonzero row.*

(NOTE: The first three assumptions are trivial. The sets Ω and G can always be formed so that these hold. However, it is the fourth assumption that is especially restrictive, as is discussed later.)

Our modification to the method of [4] involves fixing $\tilde{\boldsymbol{\pi}}_G^T$ at the resulting stationary vector from the smaller Markov chain (the $\tilde{\mathbf{C}}$ matrix in the coupling theorem language and $\hat{\mathbf{P}}$ matrix from [4]) and scaling $\tilde{\boldsymbol{\pi}}_\Omega^T$ by $\tilde{\xi}_1 \cdot \frac{\boldsymbol{\pi}_\Omega^T}{\boldsymbol{\pi}_\Omega^T \mathbf{e}}$. If the four assumptions hold, then our modification produces exact *updated PageRanks.*

Proof Our goal is to show that the four assumptions create a situation in which $\tilde{\mathbf{s}}_1^T = \mathbf{s}_1^T$. Then clearly, $\tilde{\boldsymbol{\pi}}_\Omega^T = \tilde{\xi}_1 \tilde{\mathbf{s}}_1^T = \tilde{\xi}_1 \mathbf{s}_1^T = \tilde{\xi}_1 \cdot \frac{\boldsymbol{\pi}_\Omega^T}{\boldsymbol{\pi}_\Omega^T \mathbf{e}}$ and the conclusion holds. In order to show $\tilde{\mathbf{s}}_1^T = \mathbf{s}_1^T$, we show $\tilde{\mathbf{S}}_{11} = \mathbf{S}_{11}$.

Let the j^{th} row of $\tilde{\mathbf{P}}_{21}$ be the lone nonzero row. Since $\tilde{\mathbf{P}}$ is a stochastic matrix, the j^{th} row of $\tilde{\mathbf{P}}_{22}$ is the only row not summing to 1. Further, the j^{th} row of $(\mathbf{I} - \tilde{\mathbf{P}}_{22})$ is the only row with a nonzero

row sum. Writing this statement mathematically, there exists a scalar α such that $(\mathbf{I}-\tilde{\mathbf{P}}_{22})\mathbf{e}=\alpha\mathbf{e}_j$. Since $(\mathbf{I}-\mathbf{P}_{22})$ is nonsingular (see [**?**]), the previous identity can be rewritten as $(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}\mathbf{e}_j=\frac{1}{\alpha}\mathbf{e}$. Without loss of generality, we assume only one link update is made. That is, only the i^{th} row of $\mathbf{P}_{22}$ is changed, resulting in $\tilde{\mathbf{P}}_{22}$. Note that $i\neq j$, since i represents the updated row of $\mathbf{P}_{22}$ and j represents the nonzero row of $\mathbf{P}_{21}$. If $i=j$, then $\mathbf{P}_{21}\neq\tilde{\mathbf{P}}_{\mathbf{21}}$, which violates assumption 3. Let $\mathbf{b}^T$ be the new updated row i in $(\mathbf{I}-\tilde{\mathbf{P}}_{22})$. That is, $\mathbf{b}^T=\mathbf{e}_i^T(\mathbf{I}-\tilde{\mathbf{P}}_{22})$. Assumptions 1 through 3 give $\mathbf{b}^T\mathbf{e}=\mathbf{e}_i^T(\mathbf{I}-\tilde{\mathbf{P}}_{22})\mathbf{e}=\mathbf{e}_i^T\mathbf{e}-\mathbf{e}_i^T\tilde{\mathbf{P}}_{22}\mathbf{e}=1-1=0$. Using the previous equality,

$$\mathbf{b}^T(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}\mathbf{e}_j=\mathbf{b}^T(\frac{1}{\alpha}\mathbf{e})=\frac{1}{\alpha}\mathbf{b}^T\mathbf{e}=\mathbf{0}.$$

Using the above statement along with the Sherman-Morrison formula for rank-1 updating of an inverse [12], we see that the j^{th} columns of $(\mathbf{I}-\mathbf{P}_{22})^{-1}$ and $(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}$ are equal. Recall that the i^{th} row of $(\mathbf{I}-\mathbf{P}_{22})$ is removed and $\mathbf{b}^T$ is inserted in its place, forming $(\mathbf{I}-\tilde{\mathbf{P}}_{22})$. That is, $\mathbf{I}-\tilde{\mathbf{P}}_{22}=\mathbf{I}-\mathbf{P}_{22}+\mathbf{e}_i(\mathbf{b}^T-\mathbf{e}_i(\mathbf{I}-\mathbf{P}_{22}))$. By the Sherman-Morrison formula,

$$(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}=(\mathbf{I}-\mathbf{P}_{22})^{-1}-\frac{(\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{e}_i(\mathbf{b}^T(\mathbf{I}-\mathbf{P}_{22})^{-1}-\mathbf{e}_i^T)}{\mathbf{b}^T(\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{e}_i}.$$

Since $j\neq i$, the j^{th} columns of $(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}$ and $(\mathbf{I}-\mathbf{P}_{22})^{-1}$ are related by the following formula.

$$\begin{aligned}(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}\mathbf{e}_j &= (\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{e}_j-\frac{(\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{e}_i(\mathbf{b}^T(\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{e}_j-0)}{\mathbf{b}^T(\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{e}_i}\\ &= (\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{e}_j,\end{aligned}$$

as $\mathbf{b}^T(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}\mathbf{e}_j=0$. The equality of the j^{th} columns of $(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}$ and $(\mathbf{I}-\mathbf{P}_{22})^{-1}$ implies that the first stochastic complements $\mathbf{S}_{11}$ and $\tilde{\mathbf{S}}_{11}$ are equal. Recall that

$$\begin{aligned}\mathbf{S}_{11} &= \mathbf{P}_{11}+\mathbf{P}_{12}(\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{P}_{21} \quad \text{and}\\ \tilde{\mathbf{S}}_{11} &= \tilde{\mathbf{P}}_{11}+\tilde{\mathbf{P}}_{12}(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}\tilde{\mathbf{P}}_{21}\\ &= \mathbf{P}_{11}+\mathbf{P}_{12}(\mathbf{I}-\tilde{\mathbf{P}}_{22})^{-1}\mathbf{P}_{21},\end{aligned}$$

by assumptions 1, 2 and 3. In the matrix multiplication of $(\mathbf{I}-\mathbf{P}_{22})^{-1}\mathbf{P}_{21}$, only the j^{th} column of $(\mathbf{I}-\mathbf{P}_{22})^{-1}$ is involved since, by assumption 4, only the j^{th} row of $\mathbf{P}_{21}$ is nonzero. Since $\mathbf{S}_{11}=\tilde{\mathbf{S}}_{11}$, the stationary distributions associated with these stochastic complements, $\mathbf{s}_1^T$ and $\tilde{\mathbf{s}}_1^T$ are equal. By the theory of stochastic complementation developed earlier in this section, we know that

$$\tilde{\boldsymbol{\pi}}^T=(\,\tilde{\boldsymbol{\pi}}_\Omega^T \quad | \quad \tilde{\boldsymbol{\pi}}_G^T\,)=(\,\tilde{\xi}_1\tilde{\mathbf{s}}_1^T \quad | \quad \tilde{\xi}_2\tilde{\mathbf{s}}_2^T\,).$$

The $\tilde{\boldsymbol{\pi}}_G^T$ obtained from the coupling matrix as suggested in [4] was shown to be exact earlier (since when $\tilde{\mathbf{s}}_1^T=\mathbf{s}_1^T$, $\tilde{\mathbf{C}}=\hat{\mathbf{P}}$). Now $\tilde{\boldsymbol{\pi}}_\Omega^T$ can be obtained by appropriately scaling, as we suggest, the old PageRank values, $\boldsymbol{\pi}_\Omega^T$.

$$\tilde{\boldsymbol{\pi}}_\Omega^T=\tilde{\xi}_1\tilde{\mathbf{s}}_1^T=\tilde{\xi}_1\mathbf{s}_1^T=\tilde{\xi}_1\frac{\boldsymbol{\pi}_\Omega^T}{\boldsymbol{\pi}_\Omega^T\mathbf{e}}.$$

□

We assess the satisfiability and meaning of the assumptions of Theorem 8.1. The sets Ω and G can easily be formed so that the first three assumptions are satisfied. For instance, for each link update from node i to node j, put both i and j in G, as well as all nodes k such that $p_{i,k}>0$. With this simple rule,

the first three assumptions are satisfied. However, the fourth assumption is very restrictive and in some cases may be impossible to satisfy. Nevertheless, checking that the fourth assumption holds is trivial, yet this check immediately provides the researcher with information about the accuracy of the updated PageRanks arrived at using the method of [4]. This fourth assumption is equivalent to requiring that G be formed so that it contains exactly one node with outlinks to the set Ω. All other nodes in G should only outlink to other nodes in G. This suggests a fruitful line of research: determining a heuristic to partition the n nodes into the two sets Ω and G so that this fourth restriction is met, or further still, determining conditions on $\mathbf{P}$, Ω and G that enable the researcher to determine whether or not such an optimal partition exists.

8.2 Case 2: $\hat{\mathbf{P}} \approx \tilde{\mathbf{C}}$

The above theorem shows that it is very rare that $\hat{\mathbf{P}} = \tilde{\mathbf{C}}$. It is much more likely that $\hat{\mathbf{P}} \approx \tilde{\mathbf{C}}$. In this case, the condensed matrix $\hat{\mathbf{P}}$ of [4] is an approximation to the coupling or aggregation matrix $\tilde{\mathbf{C}}$. In this section, we describe the basic aggregation/disaggregation (A/D) method, which is similar to stochastic complementation except it uses an approximate aggregation matrix rather than an exact aggregation matrix. We then use the basic aggregation/disaggregation method of [21], modifying it for our particular updating problem, to calculate updated *exact* PageRanks with little additional work.

Notice again that $\hat{\mathbf{P}}$ differs from $\tilde{\mathbf{C}}$ by only its first row. $\hat{\mathbf{P}}$ uses $\mathbf{s}_1^T$ as an estimate of the unknown $\tilde{\mathbf{s}}_1^T$. Our aggregation/disaggregation algorithm tries to improve these estimates of $\tilde{\mathbf{s}}_1^T$ with repeated A/D steps.

First, we present the basic A/D algorithm and discuss its uses. Then we present our modification to this A/D algorithm, tailoring it to the PageRank updating problem.

8.2.1 Aggregation/Disaggregation Methods for block Markov Chains

The A/D algorithm to follow has proven success when applied to nearly completely decomposable (NCD) Markov chains. An NCD Markov chain has states that can be partitioned into subsets, where interactions among states in the same subset are strong, while interactions among states in different subsets are weak. An NCD Markov chain with N subsets has the following block structure

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1N} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \cdots & \mathbf{P}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{N1} & \mathbf{P}_{N2} & \cdots & \mathbf{P}_{NN} \end{pmatrix},$$

where the elements of off-diagonal blocks are small compared to the elements of diagonal blocks. The stationary solution of the NCD system can still be found by solving $\boldsymbol{\pi}^T\mathbf{P} = \boldsymbol{\pi}^T$ and $\boldsymbol{\pi}^T\mathbf{e} = 1$. However, since the elements of $\mathbf{P}_{ij}$ for $i \neq j$ are small compared to those of $\mathbf{P}_{ii}$, it is tempting to consider each P_{ii} as an independent system with its own stationary vector. In the completely decomposable case, all off-diagonal blocks are $\mathbf{0}$ and

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P}_{NN} \end{pmatrix}.$$

The pioneers of NCD work, Simon and Ando [20], decided to approximate $\boldsymbol{\pi}^T$ by first treating the nearly completely decomposable system as a completely decomposable system. The full stationary vector $\boldsymbol{\pi}^T$ is partitioned into

$$\boldsymbol{\pi}^T = (\boldsymbol{\pi}_1^T \quad \boldsymbol{\pi}_2^T \quad \cdots \quad \boldsymbol{\pi}_N^T),$$

where each $\boldsymbol{\pi}_i^T$ is an $n_i \times 1$ vector, with n_i being the order of $\mathbf{P}_{ii}$. For the completely decomposable system, $\boldsymbol{\pi}_i^T$ is found from $\boldsymbol{\pi}_i^T \mathbf{P}_{ii} = \boldsymbol{\pi}_i^T$. However, in applying this logic to the NCD system, $\mathbf{P}_{ii}$ are not stochastic; they are substochastic, since the off-diagonal blocks do hold some probability mass. One way to circumvent this problem is to compute the Perron vector $\mathbf{u}_i^T$ for the substochastic system, $\mathbf{u}_i^T \mathbf{P}_{ii} = \lambda_{i_1} \mathbf{u}_i^T$, $\mathbf{u}_i^T \mathbf{e} = 1$, where λ_{i_1} is the Perron root of $\mathbf{P}_{ii}$ and $\mathbf{u}_i^T$ is its corresponding lefthand eigenvector. If the off-diagonal blocks are truly small in comparison to the diagonal blocks, then $\mathbf{u}_i^T$ is a good approximation to the stationary vector for block i. However, simply concatenating these $\mathbf{u}_i^T$'s does not give an approximation to the stationary vector $\bar{\boldsymbol{\pi}}^T$ for the full system $\mathbf{P}$, as the concatenation does not produce a probability vector. However, if each subvector is weighted by the probability of the chain being in that subsets of states, then an approximate probability vector for $\boldsymbol{\pi}^T$ is obtained. The probability of the chain being in block i is simply $\boldsymbol{\pi}_i^T \mathbf{e}$. But $\boldsymbol{\pi}_i^T$ is unknown. To remedy this second problem, an aggregation matrix $\mathbf{A}$ is formed to approximate to the probability of being in block i. The aggregation matrix is an $N \times N$ stochastic matrix whose (i,j)-element holds the probability of moving from block i to block j. Thus, the stationary vector $\boldsymbol{\xi}^T$ of $\mathbf{A}$ contains the probabilities of being in each block. To form Simon and Ando's approximation $\bar{\boldsymbol{\pi}}^T$ to $\boldsymbol{\pi}^T$, the stationary vector of the NCD system $\mathbf{P}$, let

$$\bar{\boldsymbol{\pi}}^T = (\xi_1 \mathbf{u}_1^T \quad \xi_2 \mathbf{u}_2^T \quad \ldots \quad \xi_N \mathbf{u}_N^T).$$

This is the basic aggregation step for approximating the stationary solution of an NCD system. Researchers then suggested repeating this aggregation step to iteratively form better and better approximations to $\boldsymbol{\pi}^T$. However, they discovered that inputting the approximate probability vector $\bar{\boldsymbol{\pi}}^T$ back into the aggregation step gave no improvement. The aggregation step is stationary. Yet, applying a disaggregation step, that is, one iteration of the power method on the approximation $\bar{\boldsymbol{\pi}}^T$, then another aggregation step, does lead to improvement. The basic iterative aggregation/disaggregation (IAD) method of [21] is below.

Algorithm 3: IAD Algorithm

1. Let $\boldsymbol{\pi}^{(0)T} = (\boldsymbol{\pi}_1^{(0)T} \quad \boldsymbol{\pi}_2^{(0)T} \quad \cdots \quad \boldsymbol{\pi}_N^{(0)T})$ be the given initial approximation to the solution $\boldsymbol{\pi}^T$ and set $m = 1$.

2. Compute $\boldsymbol{\phi}^{(m-1)T}$ by computing

$$\boldsymbol{\phi}_i^{(m-1)T} = \frac{\boldsymbol{\pi}_i^{(m-1)T}}{\boldsymbol{\pi}_i^{(m-1)T} \mathbf{e}},$$

   for $i = 1, 2, \ldots, N$.

3. Form the aggregation matrix $\mathbf{A}^{(m-1)}$ with elements

$$[\mathbf{A}^{(m-1)}]_{ij} = \boldsymbol{\phi}_i^{(m-1)T} \mathbf{P}_{ij} \mathbf{e}.$$

4. Compute the stationary vector $\boldsymbol{\xi}^{(m-1)T}$ for $\mathbf{A}^{(m-1)}$ by solving

$$\boldsymbol{\xi}^{(m-1)T} \mathbf{A}^{(m-1)} = \boldsymbol{\xi}^{(m-1)T}, \quad \boldsymbol{\xi}^{(m-1)T} \mathbf{e} = 1.$$

5. Compute the approximate stationary vector $\bar{\boldsymbol{\pi}}^{(m)T}$

$$\bar{\boldsymbol{\pi}}^{(m)T} = \begin{pmatrix} \xi_1^{(m-1)}\boldsymbol{\phi}_1^{(m-1)T} & \xi_2^{(m-1)}\boldsymbol{\phi}_2^{(m-1)T} & \cdots & \xi_N^{(m-1)}\boldsymbol{\phi}_N^{(m-1)T} \end{pmatrix}.$$

6. Compute $\boldsymbol{\pi}^{(m)T}$ by applying one iteration of the power method to $\bar{\boldsymbol{\pi}}^{(m)T}$.

$$\boldsymbol{\pi}^{(m)T} = \bar{\boldsymbol{\pi}}^{(m)T}\mathbf{P}.$$

7. Test for convergence. If not satisfied, set $m = m + 1$ and go to step 2.

If step 6, the disaggregation step, is ignored and $\bar{\boldsymbol{\pi}}^{(m)T}$ is input back into step 2, the same aggregation matrix is formed and the method stagnates at a fixed point. The disaggregation step moves the method off the fixed point, improves the aggregation matrix and allows the IAD to converge to the stationary vector $\boldsymbol{\pi}^T$ quickly.

8.2.2 PageRank Updating IAD Algorithm

IAD methods are known to work well on NCD systems. However, they are not recommended for general systems [21]. There is no reason to believe that the partitioning of the PageRank $\tilde{\mathbf{P}}$ matrix according to the sets Ω and G creates an NCD matrix. Nevertheless, due $\hat{\mathbf{P}}$'s closeness to the exact aggregation matrix $\tilde{\mathbf{C}}$, we experimented with a modification to the above IAD method. Our PageRank updating IAD algorithm is below.

Recall that for PageRank updating, we have partitioned $\tilde{\mathbf{P}}$ into

$$\tilde{\mathbf{P}} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{ccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \cdots & \mathbf{P}_{1,n_G+1} \\ \tilde{\mathbf{P}}_{\mathbf{21}} & \tilde{\mathbf{P}}_{\mathbf{22}} & \tilde{\mathbf{P}}_{\mathbf{23}} & \cdots & \tilde{\mathbf{P}}_{\mathbf{2,n_G+1}} \\ \tilde{\mathbf{P}}_{\mathbf{31}} & \tilde{\mathbf{P}}_{\mathbf{32}} & \tilde{\mathbf{P}}_{\mathbf{33}} & \cdots & \tilde{\mathbf{P}}_{\mathbf{3,n_G+1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{P}}_{\mathbf{n_G+1,1}} & \tilde{\mathbf{P}}_{\mathbf{n_G+1,2}} & \tilde{\mathbf{P}}_{\mathbf{n_G+1,3}} & \cdots & \tilde{\mathbf{P}}_{\mathbf{n_G+1,n_G+1}} \end{pmatrix} \end{array}.$$

with a conformably partitioned stationary vector

$$\tilde{\boldsymbol{\pi}}^T = \begin{pmatrix} \tilde{\boldsymbol{\pi}}_\Omega^T & \tilde{\pi}_{g_1} & \tilde{\pi}_{g_2} & \cdots & \tilde{\pi}_{g_{n_G}} \end{pmatrix}.$$

Algorithm 4: PageRank Updating IAD Algorithm

1. Let $\tilde{\boldsymbol{\pi}}^{(0)T} = \begin{pmatrix} \boldsymbol{\pi}_\Omega^{(0)T} & \pi_{g_1}^{(0)} & \pi_{g_2} & \cdots & \pi_{g_{n_G}}^{(0)} \end{pmatrix}$ be the given initial approximation to the solution $\tilde{\boldsymbol{\pi}}^T$ and set $m = 1$. That is, let the old PageRank vector $\boldsymbol{\pi}^T$ be the initial approximation to the desired updated PageRank vector $\tilde{\boldsymbol{\pi}}^T$.

2. Compute $\bar{\mathbf{s}}_1^{(m-1)T}$, the approximation to $\tilde{\mathbf{s}}_1^T$.

$$\bar{\mathbf{s}}_1^{(m-1)T} = \frac{\tilde{\boldsymbol{\pi}}_\Omega^{(m-1)T}}{\tilde{\boldsymbol{\pi}}_\Omega^{(m-1)T}\mathbf{e}}.$$

3. Construct the aggregation matrix $\hat{\mathbf{P}}^{(m-1)}$ according to the rules in [4].

$$\hat{\mathbf{P}}^{(m-1)} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{cccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \bar{\mathbf{s}}_1^{(m-1)T}\mathbf{P}_{11}\mathbf{e} & \bar{\mathbf{s}}_1^{(m-1)T}\mathbf{P}_{12} & \bar{\mathbf{s}}_1^{(m-1)T}\mathbf{P}_{13} & \cdots & \bar{\mathbf{s}}_1^{(m-1)T}\mathbf{P}_{1,n_G+1} \\ \tilde{\mathbf{P}}_{21}\mathbf{e} & \tilde{\mathbf{P}}_{22} & \tilde{\mathbf{P}}_{23} & \cdots & \tilde{\mathbf{P}}_{2,n_G+1} \\ \tilde{\mathbf{P}}_{31}\mathbf{e} & \tilde{\mathbf{P}}_{32} & \tilde{\mathbf{P}}_{33} & \cdots & \tilde{\mathbf{P}}_{3,n_G+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{P}}_{n_G+1,1}\mathbf{e} & \tilde{\mathbf{P}}_{n_G+1,2} & \tilde{\mathbf{P}}_{n_G+1,3} & \cdots & \tilde{\mathbf{P}}_{n_G+1,n_G+1} \end{pmatrix} \end{array}.$$

Note that only the first row of the aggregation matrix $\hat{\mathbf{P}}$ changes at each iteration.

4. Find the stationary vector $\hat{\boldsymbol{\pi}}^{(m-1)T}$ of $\hat{\mathbf{P}}^{(m-1)}$ by solving

$$\hat{\boldsymbol{\pi}}^{(m-1)T}\hat{\mathbf{P}}^{(m-1)} = \hat{\boldsymbol{\pi}}^{(m-1)T}, \quad \hat{\boldsymbol{\pi}}^{(m-1)T}\mathbf{e} = 1.$$

5. Compute the approximate stationary vector $\bar{\boldsymbol{\pi}}^{(m)T}$.

$$\bar{\boldsymbol{\pi}}^{(m)T} = \begin{pmatrix} \hat{\pi}_1^{(m-1)}\bar{\mathbf{s}}_1^{(m-1)T} & \hat{\pi}_{g_1}^{(m-1)} & \hat{\pi}_{g_2}^{(m-1)} & \cdots & \hat{\pi}_{g_{n_G}}^{(m-1)} \end{pmatrix}.$$

6. Do one power iteration with $\bar{\boldsymbol{\pi}}^{(m)T}$ to create $\tilde{\boldsymbol{\pi}}^{(m)T}$.

$$\tilde{\boldsymbol{\pi}}^{(m)T} = \bar{\boldsymbol{\pi}}^{(m)T}\tilde{\mathbf{P}}.$$

7. Test for convergence. If not satisfied, set $m = m + 1$ and go to step 2.

8.2.3 Testing the IAD PageRank Updating method

In order for our IAD PageRank updating method to be practical, it should require a small number of outer iterations. That is, m would ideally be much smaller than the number of power iterations required for full recomputation of $\tilde{\boldsymbol{\pi}}^T$ using $\tilde{\boldsymbol{\pi}}^T\tilde{\mathbf{P}} = \tilde{\boldsymbol{\pi}}^T$. In this section, we compare our IAD updating algorithm with full recomputation in terms of the number of iterations until convergence to the updated PageRanks. We do not report clock times, since they may be misleading for our small examples. The main expense of PageRank computation is the power iteration step. Our IAD method significantly reduces the number of power iterations and only adds the minor additional expense of a stationary vector solve on a very, very small Markov chain. Tables 1-5 report our findings. For the experiments, we use 4 small subnetworks of the Web and one tiny artificial network.

- `tiny10.dat`: artificial network with 10 nodes and 21 links
- `movies.dat`: network created from crawl of Web. All 451 nodes pertain to the topic of "movies". There are 713 links.
- `mathworks.dat`: network created from crawl of MathWorks website. There are 517 nodes and 13,531 links. Network supplied by Cleve Moler [15].
- `impeachment.dat`: "impeachment" network with 1336 nodes and 2604 links.
- `abortion.dat`: "abortion" network with 1693 nodes and 4325 links. The three Web subnetworks (movies, impeachment and abortion) were supplied by Ronny Lempel and used in [11].

We describe the notation used in Tables 1-5.

- For all experiments, the PageRank linear combination factor α is .90. The stopping criterion is that the 1-norm of the residual be less than 10^{-10}. The IAD algorithm was coded in Matlab and run on a SUN Ultra 10.
- The scalars r and a refer to the number of links removed and added, respectively, to the original $\mathbf{P}$ matrix to create the updated matrix $\tilde{\mathbf{P}}$. These links are chosen randomly for each program run. Sometimes the link updates have little effect on the subdominant eigenvalue of $\tilde{\mathbf{P}}$ and sometimes the link updates significantly change the subdominant eigenvalue of $\tilde{\mathbf{P}}$, making the power method require many fewer or many more iterations.
- The cardinality of the set G of the nodes most likely to be affected by the link updates, denoted $|G|$, is varied for each dataset and each choice of r and a. As $|G|$ increases, we hypothesize that our IAD method should require fewer iterations.
- The number of power iterations required by the full recomputation method for updating is reported. This is compared with the number of IAD iterations required by our exact updating method.

Table 1: Comparison of IAD and full recomputation updating methods on `tiny10.dat` ($n = 10$)

r	a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	2	2	28	76
		5	24	34
		8	3	198

Table 2: Comparison of IAD and full recomputation updating methods on `movies.dat` ($n = 451$)

r	a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	2	10	10	20
		20	9	20
		50	7	22
		100	6	20
		250	3	20
10	10	10	10	21
		20	11	20
		50	8	20
		100	7	20
		250	4	22
50	50	10	15	22
		20	11	20
		50	11	21
		100	8	22
		250	5	24
100	100	10	22	190
		20	13	186
		50	12	24
		100	9	20
		250	5	22

Table 3: Comparison of IAD and full recomputation updating methods on `mathworks.dat` ($n = 517$)

r	a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	2	10	37	63
		20	36	63
		50	16	63
		100	12	63
		250	10	63
10	10	10	46	63
		20	44	63
		50	17	63
		100	13	63
		250	9	63
50	50	10	44	61
		20	47	62
		50	19	60
		100	15	64
		250	15	62
100	100	10	53	63
		20	48	62
		50	19	65
		100	17	57
		250	15	60
		500	4	61
250	250	10	47	47
		20	46	43
		50	26	54
		100	25	56
		250	18	52
		500	3	60

8.2.4 Discussion of IAD Results

Work involved in IAD method

It is clear that the IAD updating method requires many fewer iterations than the full recomputation method. However, we need to examine the work required with each IAD iteration to give a fair comparison.

First, there is a preprocessing step: the n nodes of the network must be partitioned into the two sets, Ω and G. This is a one-time cost and should be done carefully as a good partition can speed convergence of the IAD method. As suggested in [4], we implemented a transient analysis to partition the nodes into Ω and G. Specifically, we find all k nodes i and j between whom a link has been removed or added. We then form a starting vector $\mathbf{x}^{(0)T}$ for the power method. This vector has zeros everywhere, except at the positions corresponding to the k nodes, there is a $\frac{1}{k}$. We then run three power iterations with this starting vector to obtain $\mathbf{x}^{(3)T}$, where $\mathbf{x}^{(k+1)T} = \mathbf{x}^{(k)T}\tilde{\mathbf{P}}$. We take the indices of the top $|G|$ values in $\mathbf{x}^{(3)T}$ to be the nodes in G. This assumes that $|G|$ has been preassigned. The size of G affects the convergence of the IAD method and should be chosen judiciously. Determining the most appropriate partitioning of the nodes into the two sets Ω and G is an area of future study. In summary, the one-time partitioning cost for our examples is 3 power iterations.

Table 4: Comparison of IAD and full recomputation updating methods on `impeachment.dat` ($n = 1336$)

r	a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	2	10	101	164
		20	15	164
		50	17	164
		100	8	164
		250	2	164
		500	5	164
10	10	10	67	164
		20	22	164
		50	19	163
		100	10	44
		250	7	164
		500	5	163
50	50	10	69	164
		20	25	164
		50	21	164
		100	21	163
		250	7	164
		500	4	163
100	100	10	135	164
		20	24	164
		50	23	164
		100	18	36
		250	9	165
		500	6	165
250	250	10	143	165
		20	26	163
		50	23	164
		100	23	166
		250	12	165
		500	6	163

Next, each IAD iteration requires the formation and storage of the small Markov chain $\hat{\mathbf{P}}$. The order of $\hat{\mathbf{P}}$ is $|G|+1$. Note the tradeoff: as $|G|$ increases, the number of IAD iterations decreases, yet the size of $\hat{\mathbf{P}}$ increases, requiring more inner iteration work. Regardless, at each subsequent iteration, only the first row of $\hat{\mathbf{P}}$ needs to be recomputed and stored. The remainder of $\hat{\mathbf{P}}$ is fixed. Step 5 of the IAD method requires the stationary solution of this small Markov chain. We used the power method to find $\hat{\boldsymbol{\pi}}^T$, yet the small size of $\hat{\mathbf{P}}$ frees us to implement any linear system technique. One observation of applying the power method to $\hat{\mathbf{P}}$ is worth mentioning. A very high tolerance level on the residual of the power iterates (10^{-12}) is needed in step 5. We found that a low tolerance level (10^{-6}) could cause the IAD method to stagnate and never reach the desired tolerance of the outer iteration (step 7). In our future work, since $\hat{\mathbf{P}}$ is small, we plan to use a direct method, such as GTH [9]. This will improve the stability and accuracy of $\hat{\boldsymbol{\pi}}^T$ computed in step 5. We emphasize that, in applying the IAD method to a network the size of Google's database, the computation of step 5 is negligible compared to one power iteration in step 6 with the Web-sized $\tilde{\mathbf{P}}$.

Finally, each iteration of IAD requires one power iteration in step 6. This is the main expense for the solution of the global Markov chain. The strength of the IAD method is its ability to significantly

Table 5: Comparison of IAD and full recomputation updating methods on `abortion.dat` ($n = 1693$)

r	a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	2	10	105	167
		20	93	167
		50	60	167
		100	58	167
		250	9	167
10	10	10	103	169
		20	109	167
		50	114	167
		100	95	167
		250	9	167
50	50	10	125	167
		20	126	168
		50	123	167
		100	113	167
		250	11	167
100	100	10	134	168
		20	133	167
		50	130	168
		100	115	167
		250	12	167
		500	5	167
200	200	10	128	167
		20	140	167
		50	141	168
		100	124	167
		250	12	167
		500	10	167

reduce the number of these power iterations and still produce exact updates. In information retrieval, the implications for a practical search engine, like Google, are great. The method of full recomputation may never be needed again. Instead, the IAD method can be used to update on a more frequent basis. Rather than expending a great deal of computational power each month for recomputation, expend less energy on a more frequent weekly or even daily basis. The IAD updating method also allows PageRank implementors, like Google, increased flexibility. The IAD algorithm contains a convergence test in step 7. Google engineers may set this tolerance level low, meaning IAD converges to estimates of PageRank that are exact to within a certain, lower tolerance level. This lower tolerance means that less IAD iterations are required. Perhaps Google may decide to update daily, with less accuracy, yet still update exactly on a montly basis, by either full recomputation or IAD with a very high tolerance level. We envision that our IAD method for updating PageRank will make the full recomputation method obsolete one day. PageRank can now be updated on a much more frequent basis with much less work. We have implemented an IAD-Power method, but of course, any suitable iterative method, like SOR, GMRES, etc. may be used [22].

9 Node Updates

In the section 3, we mentioned the types of updates that are made to the Web, distinguishing between link updates and node updates. In this section, we demonstrate that the IAD algorithm also accomodates node updates. To our knowledge, this is the first algorithm to address the more challenging problem of node updating. The size of the Markov matrix changes with node updates so that the new Markov matrix becomes

$$\tilde{\mathbf{P}} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{ccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \mathbf{P} & \tilde{\mathbf{P}}_{12} & \tilde{\mathbf{P}}_{13} & \cdots & \tilde{\mathbf{P}}_{1,n_G+1} \\ \tilde{\mathbf{P}}_{21} & \tilde{\mathbf{P}}_{22} & \tilde{\mathbf{P}}_{23} & \cdots & \tilde{\mathbf{P}}_{2,n_G+1} \\ \tilde{\mathbf{P}}_{31} & \tilde{\mathbf{P}}_{32} & \tilde{\mathbf{P}}_{33} & \cdots & \tilde{\mathbf{P}}_{3,n_G+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{P}}_{n_G+1,1} & \tilde{\mathbf{P}}_{n_G+1,2} & \tilde{\mathbf{P}}_{n_G+1,3} & \cdots & \tilde{\mathbf{P}}_{n_G+1,n_G+1} \end{pmatrix} \end{array}.$$

Notice that $\tilde{\mathbf{P}}$ is just $\mathbf{P}$ augmented by rows and columns corresponding to the additional nodes. Since the Markov matrix grows in size, so does the stationary vector. Step 1 of the IAD algorithm must be modified. Using $\boldsymbol{\pi}^T$ as the starting vector for the node-updating IAD algorithm will not work. We modify step 1 for the node-updating problem,

1. Let $\tilde{\boldsymbol{\pi}}^{(0)T} = \begin{pmatrix} \boldsymbol{\pi}_{\Omega}^{(0)T} & 0^{(0)} & 0 & \cdots & 0^{(0)} \end{pmatrix}$ be the given initial approximation to the solution $\tilde{\boldsymbol{\pi}}^T$ and set $m = 1$. That is, let the old stationary vector $\boldsymbol{\pi}^T$ be $\boldsymbol{\pi}_{\Omega}^{(0)T}$ and initially set the stationary probabilities for the new nodes to 0.

The other IAD steps remain the same. Thus, one algorithm addresses both updating problems, link-updating and node-updating.

When the Markov chain downsizes with the removal of nodes, node-downdates, then step 1 of the IAD method is similarly modified.

1. Let $\tilde{\boldsymbol{\pi}}^{(0)T} = \frac{\boldsymbol{\pi}_{O}^{(0)T}}{\boldsymbol{\pi}_{O}^{(0)T}\mathbf{e}}$ be the given initial approximation to the solution $\tilde{\boldsymbol{\pi}}^T$, where O is the set of the original nodes remaining after the removal of the downdated nodes. Set $m = 1$.

Since the results of the node-downdating experiments mimic the results of the node-updating experiments, to avoid redundancy we only present the results of the node-updating experiments.

We tested the IAD algorithm on $\mathbf{P}$ matrices that were node-updated. We use the same datasets as the link-updating experiments and the same notation for the tables. Here a represents the number of nodes added. For each node added, we randomly selected outlinking states. The number of outlinks for each new node is a random integer between 1 and 15, as is fitting for PageRank information retrieval applications [2].

Table 6: Comparison of IAD and full recomputation node-updating methods on `tiny10.dat` ($n = 10$, $l = 21$)

a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	2	31	34
	5	15	34
	8	10	34

Table 7: Comparison of IAD and full recomputation node-updating methods on `movies.dat` ($n = 451$, $l = 713$)

a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	10	9	20
	20	8	20
	50	7	20
	100	6	20
	250	5	20
10	10	9	19
	20	9	20
	50	9	20
	100	8	19
	250	6	20
50	10	11	19
	20	10	20
	50	9	20
	100	10	19
	250	8	19
100	10	15	20
	20	12	20
	50	14	20
	100	12	20
	250	11	19

9.1 Further Interesting Observations

Following an intuitive hunch about the importance of the Ω/G partitioning on IAD convergence, we explored this topic in depth. Tables 1-10 reveal an interesting trend. Regardless of the number of link updates (size of r and a) or node updates (size of a), the plot of the number of IAD iterations for varying sizes of G exhibits a constant shape for each dataset. Consider the `impeachment.dat` results in table 3. As $|G|$ increases from 10 to 20, there is a huge dropoff in the number of IAD iterations required, regardless of the number of link updates. This *dropoff point* occurred in almost all datasets for both link-updating and node-updating. We increased the granularity of $|G|$ to pinpoint the dropoff point for each dataset and number of link updates. Graph 2 shows the dropoff point for `mathworks.dat`. Notice that regardless of the number of link updates, the plots all have the same shape, with a dropoff point near 40. After the dropoff point, little is gained despite the increasing amount of work. The best choice for $|G|$ is just after this dropoff point. This dropoff point occurred at the same point in the node-updating experiments for this same dataset. See table 3.

Table 8: Comparison of IAD and full recomputation node-updating methods on `mathworks.dat` ($n = 517$, $l = 13,531$)

a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	10	38	63
	20	40	63
	50	32	63
	100	13	63
	250	12	63
10	10	46	63
	20	43	63
	50	36	63
	100	14	63
	250	11	63
50	10	50	64
	20	45	64
	50	16	63
	100	15	64
	250	13	63
100	10	50	64
	20	48	63
	50	17	64
	100	15	64
	250	15	64
250	10	55	64
	20	52	64
	50	21	64
	100	17	64
	250	16	64

The existence of the dropoff point suggests that perhaps the most appropriate size for G is problem-dependent. In fact, that is exactly what we discovered. We pinpointed the dropoff point for all datasets. For example, for `mathworks.dat`, there was a great deal of progress in reducing the number of IAD iterations as $|G|$ moved from 35 to 36 to 37 and so on up to the dropoff point near 40. Regardless of the number of updates (size of r and a), this always occurred. We also noted that as r and a increase, the change in the dominant eigenvalues of $\tilde{\mathbf{P}}$ is barely perceptible. We next examined the eigendistribution of the stochastic complements to see if marked changes there might predict the dropoff point and, in turn, the most appropriate size of G. Why examine the stochastic complements? Meyer [13] has shown that the eigenvalues of the stochastic complements govern the convergence of an exact aggregation method. Since the IAD method is so close to an exact aggregation method, studying stochastic complements seems a good place to start.

The partitioning scheme for $\tilde{\mathbf{P}}$ (one large class the size of Ω and $|G|$ small one-state classes) provides

Table 9: Comparison of IAD and full recomputation node-updating methods on `impeachment.dat` ($n = 1336$, $l = 2604$)

a	$\lvert G\rvert$	IAD Iterations	Full Recomputation Iterations
2	10	93	164
	20	91	164
	50	17	164
	100	16	164
	250	7	164
10	10	21	164
	20	47	175
	50	18	163
	100	18	164
	250	7	164
50	10	118	164
	20	120	164
	50	20	164
	100	20	164
	250	8	164
100	10	123	164
	20	63	164
	50	20	164
	100	11	164
	250	9	164
250	10	132	164
	20	66	164
	50	23	164
	100	13	164
	250	12	164

very specific information about the stochastic complements corresponding to states in G. Recall that

$$\tilde{\mathbf{P}} = \begin{array}{c} \\ \Omega \\ g_1 \\ g_2 \\ \vdots \\ g_{n_G} \end{array} \begin{array}{c} \begin{array}{ccccc} \Omega & g_1 & g_2 & \cdots & g_{n_G} \end{array} \\ \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \cdots & \mathbf{P}_{1,n_G+1} \\ \tilde{\mathbf{P}}_{21} & \tilde{\mathbf{P}}_{22} & \tilde{\mathbf{P}}_{23} & \cdots & \tilde{\mathbf{P}}_{2,n_G+1} \\ \tilde{\mathbf{P}}_{31} & \tilde{\mathbf{P}}_{32} & \tilde{\mathbf{P}}_{33} & \cdots & \tilde{\mathbf{P}}_{3,n_G+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{P}}_{n_G+1,1} & \tilde{\mathbf{P}}_{n_G+1,2} & \tilde{\mathbf{P}}_{n_G+1,3} & \cdots & \tilde{\mathbf{P}}_{n_G+1,n_G+1} \end{pmatrix} \end{array}.$$

Note that each block $\tilde{\mathbf{P}}_{g_i,g_i}$ is a 1×1 block for all $i = 1, \ldots, n_G$. From [13], we know the stochastic complements $\tilde{\mathbf{S}}_{ii}$ corresponding to these $\tilde{\mathbf{P}}_{g_i,g_i}$ are also 1×1 matrices, and more importantly, are stochastic. Therefore, the censored stationary probability vector $\tilde{\mathbf{s}}_i^T$ of $\tilde{\mathbf{S}}_{ii}$ is 1 and the eigenvalue for the 1×1 stochastic complement for nodes in G is always 1. Thus, to understand the convergence of the IAD method, which is an approximation to exact aggregation by stochastic complementation, only $\tilde{\mathbf{S}}_{11}$ (or $\tilde{\mathbf{S}}_{\Omega,\Omega}$) need be examined. The eigendistribution of $\tilde{\mathbf{S}}_{11}$ does indeed unlock the mystery of the dropoff point. Table 11 shows the evolving dominant eigenvalues of $\tilde{\mathbf{S}}_{11}$ for `mathworks.dat` as $|G|$ increases to the observed dropoff point.

The dropoff point occurs when the subdominant eigenvalue of $\tilde{\mathbf{S}}_{11}$ drops dramatically. For `mathworks.dat`, this occurs when $|G|$ moves from 35 to 40, as $\lambda_2(\tilde{\mathbf{S}}_{11})$ drops from .6054 to .4018. After the dropoff point,

Table 10: Comparison of IAD and full recomputation node-updating methods on `abortion.dat` ($n = 1693$, $l = 4325$)

a	$\|G\|$	IAD Iterations	Full Recomputation Iterations
2	10	89	167
	20	92	167
	50	86	167
	100	87	168
	250	10	167
10	10	103	167
	20	96	167
	50	106	167
	100	103	167
	250	10	167
50	10	120	167
	20	118	167
	50	117	167
	100	111	167
	250	12	166
100	10	127	167
	20	124	167
	50	113	166
	100	112	167
	250	12	167
250	10	132	167
	20	132	167
	50	128	166
	100	120	167
	250	13	167

Table 11: Dominant eigenvalues of the first stochastic complement, $\tilde{\mathbf{S}_{11}}$ for `mathworks.dat` as $|G|$ increases ($r = 50$, $a = 50$)

$\|G\| = 10$	$\|G\| = 20$	$\|G\| = 30$	$\|G\| = 35$	$\|G\| = 38$	$\|G\| = 40$	$\|G\| = 45$	$\|G\| = 50$	$\|G\| = 100$
$\sigma(\tilde{\mathbf{S}}_{11})$	$\sigma(\tilde{\mathbf{S}}_{11})$	$\sigma(\tilde{\mathbf{S}}_{11})$	$\sigma(\tilde{\mathbf{S}}_{11})$	$\sigma(\tilde{\mathbf{S}}_{11})$	$\sigma(\tilde{\mathbf{S}}_{11})$	$\sigma(\tilde{\mathbf{S}}_{11})$	$\sigma(\tilde{\mathbf{S}}_{11})$	$\sigma(\tilde{\mathbf{S}}_{11})$
1	1	1	1	1	1	1	1	1
.7206	.6891	.6610	**.6054**	.4431	**.4018**	.4012	.4005	.3857
.6551	.6167	.4021	.4016	.4105	.3524	.3512	.3099	.2895
.4118	.4065	.3517	.3133	.3134	.3107	.3100	.2809	.2597

little change in $\lambda_2(\tilde{\mathbf{S}}_{11})$ is observed and little is gained in terms of convergence. Reference [13] explains how the stochastic complements each individually absorb a large subdominant eigenvalue of $\tilde{\mathbf{P}}$, speeding convergence. For example, suppose $\tilde{\mathbf{P}}$ has two large subdominant eigenvalues with the remaining eigenvalues far removed from 1. Suppose a three-level partition is used, creating three stochastic complements. Then one stochastic complement uses the unit eigenvalue of $\tilde{\mathbf{P}}$ as its unit eigenvalue, while the other two stochastic complements each use one of the two large subdominant eigenvalues of $\tilde{\mathbf{P}}$ as their unit eigenvalues. The non-unit eigenvalues of each stochastic complement are thus far removed from 1, speeding convergence to the censored stationary vectors. This numerically advantageous division of labor occurs

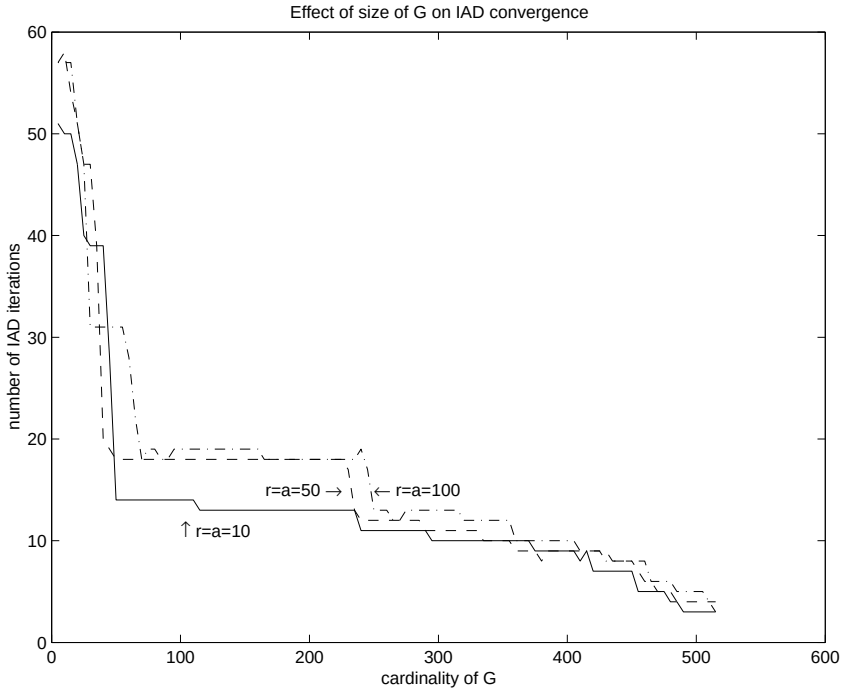

Figure 2: Dropoff point ≈ 40 for `mathworks.dat` for link-updating experiments

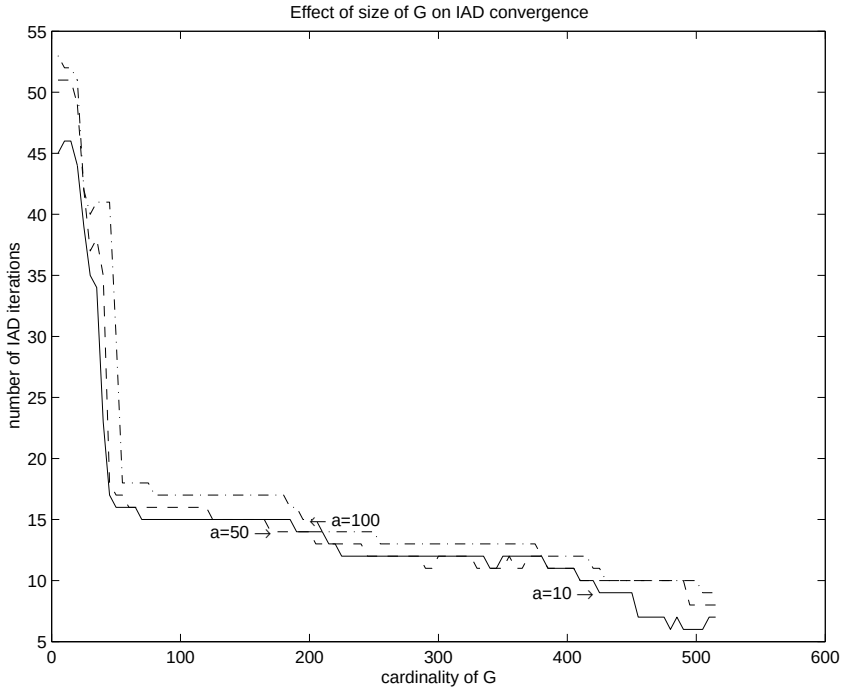

Figure 3: Dropoff point ≈ 40 for `mathworks.dat` for node-updating experiments

naturally. In the context of the updating problem, with our unique partitioning method, this means the IAD algorithm converges quickly when all the large subdominant eigenvalues of $\tilde{\mathbf{P}}$ have been absorbed by the 1×1 stochastic complements corresponding to nodes in G, which always have quick convergence due to their 1×1 size. One means of pushing these large subdominant eigenvalues of $\tilde{\mathbf{P}}$ into the stochastic complements is to increase the size of G until this set contains a cluster of nodes responsible for the large subdominant eigenvalues of $\tilde{\mathbf{P}}$. Our first-pass suggestion for determining the dropoff point is to employ a transient analysis, then increase $|G|$ until $\lambda_2(\tilde{\mathbf{S}}_{11})$, or the estimate of this coefficient of ergodicity, is acceptably small. However, we feel that better ways of determining the optimal G set must exist. In the future, we plan to examine the connectivity structure of $\tilde{\mathbf{P}}$, searching for densely linked clusters of nodes. Nevertheless, for now, we feel Markov chain researchers, especially those in information retrieval, like Google engineers, can make great use of our suggestions for determining G and $|G|$. In fact, if our observations on the constancy of the eigendistribution of $\mathbf{P}$ despite link updates hold for Web-sized graphs, then Google engineers can determine the set G and the dropoff point for their dataset once and these calculations should hold for many, many subsequent months.

10 Conclusions and Future Work

We discussed several new algorithms for updating PageRank, only one of which is practical. The IAD updating method is most promising. It is an exact updating method that exploits the old PageRanks to create the new PageRanks. It severely beats the current updating method of full recomputation in terms of the number of iterations required for convergence. We see three exciting possibilities for even greater improvements to our IAD method. First, in step 4, we plan to apply the GTH algorithm to find $\hat{\boldsymbol{\pi}}$. Second, we have several ideas for accelerating the power method of step 6. Third, another area of continued study is the Ω/G partitioning. We would like to formalize this step, providing heuristics to cheaply determine the optimal size of G or members of G. Finally, we emphasize that this IAD algorithm handles both link updates and node updates. It is, to our knowledge, the only algorithm to address the challenging problem of node updating. And, of course, we also note the applicability of the IAD algorithm to general Markov chain updating problems and perhaps to eigenvector updating problems as well.

Acknowledgements We thank Ronny Lempel for generously supplying the three datasets, `movies.dat`, `impeachment.dat` and `abortion.dat`. We also thank Cleve Moler for sharing his Mathworks dataset, `mathworks.dat`, and other web crawling m-files.

References

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30 (1998) 107-117.

[2] A. Broder, R. Kumar, F. Maghoul, et al. Graph structure in the Web. *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, May 2000, Elsevier Science.

[3] S. L. Campbell and C. D. Meyer. *Generalized Inverses of Linear Transformations*. Pitman: San Francisco, 1979.

[4] S. Chien, C. Dwork, R. Kumar, and D. Sivakumar. Towards exploiting link evolution.

[5] G. Cho and C. D. Meyer. Comparison of perturbation bounds for the stationary distribution of a Markov chain. *Linear Algebra and its Applications*, (2001) to appear??.

[6] Cyveillance. Sizing the Internet. White paper. July 2000. http://www.cyveillance.com/web/us/corporate/white-paper.htm.

[7] R. E. Funderlic and C. D. Meyer. Sensitivity of the stationary distribution vector for an ergodic Markov chain. *Linear Algebra and its Applications*, 76(1986) 1-17.

[8] G. H. Golub and C. D. Meyer. Using the QR factorization and group inverse to compute, differentiate and estimate the sensitivity of stationary probabilities for Markov chains. *SIAM Journal on Algebraic and Discrete Methods*, 17(1986) 273-281.

[9] W. K. Grassmann, M. I. Taskar and D. P. Heyman. Regenerative analysis and steady state distributions for Markov chains. *Operations Research*, 33(5):1107-1116, 1985.

[10] T. H. Haveliwala. Efficient computation of PageRank. *Technical Report*, Computer Science Department, Stanford University, 1999.

[11] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proceedings of the 9th International World Wide Web Conference*, May 2000.

[12] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM: Philadelphia, 2000.

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

[14] C. D. Meyer and J. M. Shoaf. Updating finite Markov chains by using techniques of group matrix inversion. *Journal of Statistical Computation and Simulation*, 11(1980) 163-181.

[15] C. Moler. The world's largest matrix computation. *Matlab News and Notes*, October 2002, 12-13.

[16] A. Y. Ng, A. X. Zheng and M. I. Jordan. Stable algorithms for link analysis. *Proceedings of the 24th Annual International ACM SIGIR Conference*. ACM, 2001.

[17] A. Y. Ng, A. X. Zheng and M. I. Jordan. Link analysis, eigenvectors and stability. *Seventh International Joint Conference on Artificial Intelligence*, 2001.

[18] L. Page, S. Brin, R. Motwami, and T. Winograd. The PageRank citation ranking: bringing order to the Web. *Technical Report*, Computer Science Department, Stanford University, 1998.

[19] D. Rafiei and A. O. Mendelzon. What is this page known for? Computing webpage reputations. In *Proceedings of the 9th International World Wide Web Conference*, 823-835, Amsterdam, May 2000, Elsevier Science.

[20] H. A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111-138, 1961.

[21] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press: Princeton, 1994.

[22] W. J. Stewart and W. Wu. Numerical experiments with iteration and aggregation for Markov chains. ORSA Journal on Computing, 4(3):336-350, 1992.

[23] D. Zhang and Y. Dong. An efficient algorithm to rank Web resources. *Computer Networks*, 33(2000), 449-455.