Updating the Stationary Vector of an Irreducible Markov chain

Amy N. Langville* and Carl D. Meyer†

November 20, 2002

Abstract

Markov chain practitioners and researchers are familiar with the maxim: "If the elements of the Markov chain matrix $\mathbf{P}$ change and a new stationary vector is desired, simply begin an iterative method with the old stationary vector." This exploitation of the work of a nearby system seems like sound advice, so readers seldom question the maxim's value. However, in this paper, we show that this advice is actually seldom useful. This fact prompted us to generate better advice for the Markov chain updating problem. We present some traditional solutions to updating the stationary vector, followed by our new, quite promising, updating technique.

Key words: Markov chains, updating, stationary vector, PageRank updating, 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 order to address the problem of updating the stationary vector of an irreducible Markov chain, we begin by introducing our notation and the problem. Suppose a Markov chain practitioner has the transition probability matrix $\mathbf{P}$ and has solved, by one of the many solution techniques [22], for the stationary distribution vector $\boldsymbol{\pi}^T$. However, after some time, the matrix $\mathbf{P}$ has changed slightly, creating a new transition probability matrix $\tilde{\mathbf{P}}$ with a new stationary vector $\tilde{\boldsymbol{\pi}}^T$. The updating problem concerns finding $\tilde{\boldsymbol{\pi}}^T$. There are two types of updates to $\mathbf{P}$: element-updates and state-updates. Element-updates occur when elements of $\mathbf{P}$ change. State-updates occur when states are added to or deleted from the chain. In the next section, we discuss the naive approach to updating, which follows the old maxim referred to above.

2 Why the old Maxim often fails

For now, we restrict the discussion to element-updating and save state-updating for later sections. One hopes that the work of finding $\boldsymbol{\pi}^T$ would not be for naught; hopefully, $\mathbf{P}$ and $\boldsymbol{\pi}^T$ can be used to reduce the work needed to find $\tilde{\boldsymbol{\pi}}^T$. There are several updating methods for finding $\tilde{\boldsymbol{\pi}}^T$ when the updates affect only elements of $\mathbf{P}$. The naive approach begins an iterative method applied to $\tilde{\mathbf{P}}$ with $\boldsymbol{\pi}^T$ as the starting vector. Intuition counsels that if $\tilde{\mathbf{P}} \approx \mathbf{P}$, then $\tilde{\boldsymbol{\pi}}^T$ should be close to $\boldsymbol{\pi}$ and can thus be obtained, starting from $\boldsymbol{\pi}^T$, with only a few more iterations of the chosen iterative method. However, our experiments show that starting an iterative method from $\boldsymbol{\pi}^T$ takes almost as many iterations to converge to $\tilde{\boldsymbol{\pi}}^T$ as it does starting from the uniform vector, $\frac{1}{n}\mathbf{e}^T$. Why does this seemingly counterintuitive result happen? In short, because $\boldsymbol{\pi}^T$ is just not close enough to $\tilde{\boldsymbol{\pi}}^T$. This prompts the next question: "How close must $\boldsymbol{\pi}^T$ be to $\tilde{\boldsymbol{\pi}}^T$ to realize a savings in the number of iterations until convergence using $\mathbf{x}^{(0)T} = \boldsymbol{\pi}^T$?" "Very, very close," our experiments answer. More formally, we answer this closeness question by starting the power method with a vector that is ever so slightly perturbed from $\tilde{\boldsymbol{\pi}}^T$. Perhaps the simplest of all perturbations of a probability vector; add a small quantity ϵ to one element and subtract it from another element. That is,

$$\mathbf{x}^{(0)T} = \tilde{\boldsymbol{\pi}}^T + \epsilon \mathbf{e}_i^T - \epsilon \mathbf{e}_j^T,$$

where ϵ is a small positive constant and $\mathbf{e}_i$ is the i^{th} column of the identity matrix. Consider the power method applied to $\tilde{\mathbf{P}}$ to find $\tilde{\boldsymbol{\pi}}^T$, with $\mathbf{x}^{(0)T}$ as above.

$$\mathbf{x}^{(1)T} = \mathbf{x}^{(0)T}\tilde{\mathbf{P}} = \tilde{\boldsymbol{\pi}}^T\tilde{\mathbf{P}} + \epsilon \mathbf{e}_i^T\tilde{\mathbf{P}} - \epsilon \mathbf{e}_j^T\tilde{\mathbf{P}}.$$

After one power iteration, the stationary vector $\tilde{\boldsymbol{\pi}}^T$ is present, since $\tilde{\boldsymbol{\pi}}^T = \tilde{\boldsymbol{\pi}}^T\tilde{\mathbf{P}}$, but it is severely polluted by the vectors $\epsilon \mathbf{e}_i^T\tilde{\mathbf{P}}$ and $\epsilon \mathbf{e}_j^T\tilde{\mathbf{P}}$. If ϵ is very small, then the effect of this pollution is small. Nevertheless, it takes several more power iterations to eradicate this pollution. Consider the next few power iterations,

$$\begin{aligned}
\mathbf{x}^{(2)T} &= \mathbf{x}^{(1)T}\tilde{\mathbf{P}} = \tilde{\boldsymbol{\pi}}^T\tilde{\mathbf{P}}^2 + \epsilon \mathbf{e}_i^T\tilde{\mathbf{P}}^2 - \epsilon \mathbf{e}_j^T\tilde{\mathbf{P}}^2, \\
&\vdots \\
\mathbf{x}^{(n)T} &= \mathbf{x}^{(n-1)T}\tilde{\mathbf{P}} = \tilde{\boldsymbol{\pi}}^T\tilde{\mathbf{P}}^n + \epsilon \mathbf{e}_i^T\tilde{\mathbf{P}}^n - \epsilon \mathbf{e}_j^T\tilde{\mathbf{P}}^n.
\end{aligned}$$

As $n \to \infty$, $\tilde{\mathbf{P}}^n \to \mathbf{e}\tilde{\boldsymbol{\pi}}^T$ and $\epsilon \mathbf{e}_i^T\tilde{\mathbf{P}}^n - \epsilon \mathbf{e}_j^T\tilde{\mathbf{P}}^n \to \epsilon\tilde{\boldsymbol{\pi}}^T - \epsilon\tilde{\boldsymbol{\pi}}^T = 0$ and, of course, the pollution is eradicated and the stationary solution $\tilde{\boldsymbol{\pi}}$ is reached. The effect of ϵ is to speed the eradication process. In fact, the rate of convergence of the power method on the perturbed system depends both on ϵ and the eigendistribution of $\tilde{\mathbf{P}}$.

The reasoning behind the previous statement can be demonstrated with a more complicated perturbation like $\tilde{\boldsymbol{\pi}}^T + \boldsymbol{\epsilon}^T$. We compare two cases: starting the power method applied to $\tilde{\mathbf{P}}_{k\times k}$ with an initial vector $\mathbf{x}^{(0)T} = \tilde{\boldsymbol{\pi}}^T + \boldsymbol{\epsilon}^T$ and starting the power method with an arbitrary starting vector, like $\mathbf{y}^{(0)T} = \frac{1}{k}\mathbf{e}^T$. We are interested in the number of iterations n required to meet a convergence criterion in each case. Suppose the convergence criterion is that the 1-norm of the residual be less than or equal to τ. Let $r(\mathbf{x}^{(n)T}) = \|\mathbf{x}^{(n)T}(\mathbf{I} - \tilde{\mathbf{P}})\|_1$ be the residual after the n^{th} iterate of the power method, starting from $\mathbf{x}^{(0)T}$ and $r(\mathbf{y}^{(n)T})$ be similarly defined. We begin this analysis with the following observations. First, for any starting probability vector $\mathbf{z}^{(0)T}$, requiring $r(\mathbf{z}^{(n)T}) \leq \tau$ can be achieved by requiring $\|\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1}\|_\infty \leq \tau$. This is because

$$\begin{aligned}
r(\mathbf{z}^{(n)T}) &= \|\mathbf{z}^{(n)T}(\mathbf{I} - \tilde{\mathbf{P}})\|_1 = \|\mathbf{z}^{(0)T}(\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1})\|_1 \\
&\leq \|\mathbf{z}^{(0)T}\|_1\|\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1}\|_\infty = \|\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1}\|_\infty = O(\lambda_2^n).
\end{aligned}$$

The magnitude of the subdominant eigenvalue, $|\lambda_2|$, provides a good practical estimate of the number of iterations n required for $\tilde{\mathbf{P}}^n \to \mathbf{e}\boldsymbol{\pi}^T$, which implies $\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1} \to 0$. In fact, the estimate of the number

of iterations required is the n such that $\lambda_2^n \leq \tau$, which implies that when $n \geq \frac{log\ \tau}{log\ \lambda_2}$, the convergence criterion is met. Choosing $n = \frac{log\ \tau}{log\ \lambda_2}$ is generally a very good approximation to the number of iterations required for the power method to converge to the stationary vector. Now we return to the two cases of interest. For the arbitrary starting vector, $\mathbf{y}^{(0)T} = \frac{1}{k}\mathbf{e}^T$, convergence will be achieved when

$$\|\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1}\|_\infty \leq \tau.$$

However, for the closer starting vector, $\mathbf{x}^{(0)T} = \tilde{\boldsymbol{\pi}}^T + \boldsymbol{\epsilon}^T$, where $\boldsymbol{\epsilon}^T$ is a small perturbation vector with $\boldsymbol{\epsilon}^T\mathbf{e} = 0$ and $\|\boldsymbol{\epsilon}^T\|_1 < 1$, we expect to do less work. In fact,

$$r(\mathbf{x}^{(n)T}) \quad = \quad \|(\tilde{\boldsymbol{\pi}}^T + \boldsymbol{\epsilon}^T)(\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1})\|_1 = \|\boldsymbol{\epsilon}^T(\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1})\|_1 \leq \|\boldsymbol{\epsilon}^T\|_1\|\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1}\|_\infty.$$

If $\|\boldsymbol{\epsilon}^T\|_1 = 10^{-2}$, then to reach a tolerance $\tau = 10^{-8}$, the power method requires the number of iterations n such that $\|\tilde{\mathbf{P}}^n - \tilde{\mathbf{P}}^{n+1}\|_\infty \leq 10^{-6}$. Recall that an estimate of the n_x required to do this is given,

$$n_x \geq \frac{log\ \tau}{log\ \lambda_2} = \frac{log\ 10^{-6}}{log\ \lambda_2} = \frac{-6}{log\ \lambda_2},$$

whereas the n_y required by the arbitrary starting vector $\mathbf{y}^{(0)T}$ is

$$n_y \geq \frac{log\ \tau}{log\ \lambda_2} = \frac{log\ 10^{-8}}{log\ \lambda_2} = \frac{-8}{log\ \lambda_2},$$

An estimate of the difference in the bounds for the number of iterations in each case is

$$\frac{log\ \|\boldsymbol{\epsilon}^T\|_1}{log\ \lambda_2}.$$

As $\lambda_2 \rightarrow 1$, this difference grows. However, as $\lambda_2 \rightarrow 1$, the total number of iterations required with either starting vector grows dramatically too. Nevertheless, the point is that the savings achieved by a closer starting vector depend on both the closeness of the starting vector to the stationary vector (as measured by $\|\boldsymbol{\epsilon}^T\|_1$) and the size of the subdominant eigenvalue (λ_2). In practice, a starting vector is considered very close to the stationary vector if $\|\boldsymbol{\epsilon}^T\|_1 = 10^{-1}$ or 10^{-2}. Suppose $\|\boldsymbol{\epsilon}^T\|_1 = 10^{-1}$, then the power method with the close starting vector effectively stops when the residual is less than $\tau = 10^{-7}$, whereas the power method with the arbitrary vector continues until the residual is less than $\tau = 10^{-8}$. The number of additional iterations required to drive the residual norm from 10^{-7} to 10^{-8} depends primarily on λ_2. For example, if $\lambda_2 = \frac{1}{2}$, the number of additional iterations required by the "poorer" arbitrary starting vector is roughly $\frac{-1}{log\ 1/2} \approx 3$. If $\lambda_2 = \frac{2}{3}$, then the number of additional iterations is roughly 6. For $\lambda_2 = .8$, 10, and for $\lambda_2 = .9$, 22. Little savings is realized unless $\|\boldsymbol{\epsilon}^T\|_1$ is very small, a rather uncommon event in practice.

We present the following theorem as another way of arriving at the conclusion stated above: that the effectiveness of the power method with a close initial vector depends on the closeness of the initial vector to the final stationary vector and the magnitude of the subdominant eigenvalue.

Theorem 2.1 *Let* $\mathbf{P}$ *be a* $k \times k$ *aperiodic diagonalizable stochastic transition probability matrix with eigenvalues* $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_k$ *and stationary vector* $\boldsymbol{\pi}^T$*. The difference in the number of power iterations required starting with the uniform initial vector* ($\frac{1}{k}\mathbf{e}^T$) *and a perturbation of the stationary vector* ($\boldsymbol{\pi}^T + \boldsymbol{\epsilon}^T$)*, where* $\boldsymbol{\epsilon}^T\mathbf{e} = 0$ *and* $\|\boldsymbol{\epsilon}^T\|_1 < 1$*, is on the order of magnitude of*

$$\frac{log\ \|\boldsymbol{\epsilon}^T\|_1}{log\ \lambda_2}.$$

Proof Since $\mathbf{P}$ is diagonalizable, it has the spectral decomposition

$$\mathbf{P} = \lambda_1\mathbf{G}_1 + \lambda_2\mathbf{G}_2 + \cdots + \lambda_k\mathbf{G}_k, \quad \text{where } \lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_k \text{ and } \mathbf{G}_1 = \mathbf{e}\boldsymbol{\pi}^T.$$

We consider two cases:

1. The power method is applied to $\mathbf{P}$ using an initial vector $\mathbf{x}^{(0)T}$ that is close to $\boldsymbol{\pi}^T$. That is, $\mathbf{x}^{(0)T} = \boldsymbol{\pi}^T + \boldsymbol{\epsilon}^T$, where $\boldsymbol{\epsilon}^T\mathbf{e} = 0$ and $\|\boldsymbol{\epsilon}^T\|_1 < 1$ is small.
2. The power method is applied to $\mathbf{P}$ using $\mathbf{y}^{(0)T} = \frac{1}{k}\mathbf{e}^T$ as the initial vector.

The n^{th} power iterate for case 1 is $\mathbf{x}^{(n)T} = \mathbf{x}^{(0)T}\mathbf{P}^n$ and for case 2, $\mathbf{y}^{(n)T} = \mathbf{y}^{(0)T}\mathbf{P}^n$. The goal is to show that the difference in the number of iterations for case 1 and case 2 is roughly $\frac{log\ \|\boldsymbol{\epsilon}^T\|_1}{log\ \lambda_2}$. In order to do this, we estimate the number of iterations n required to insure that the respective residual norms, $r(\mathbf{x}^{(n)T}) = \|\mathbf{x}^{(n)T}(\mathbf{I}-\mathbf{P})\|_1$ and $r(\mathbf{y}^{(n)T}) = \|\mathbf{y}^{(n)T}(\mathbf{I}-\mathbf{P})\|_1$, are less than a given tolerance τ. Substituting the expressions for the n^{th} iterates, the residuals are

$$\begin{aligned} r(\mathbf{x}^{(n)T}) &= \|\mathbf{x}^{(n)T}(\mathbf{I}-\mathbf{P})\|_1 = \|\mathbf{x}^{(0)T}\mathbf{P}^n(\mathbf{I}-\mathbf{P})\|_1, \\ r(\mathbf{y}^{(n)T}) &= \|\mathbf{y}^{(n)T}(\mathbf{I}-\mathbf{P})\|_1 = \|\mathbf{y}^{(0)T}\mathbf{P}^n(\mathbf{I}-\mathbf{P})\|_1. \end{aligned}$$

Notice that $\mathbf{P}^n(\mathbf{I}-\mathbf{P})$ is a function of the diagonalizable $\mathbf{P}$ and thus, using the spectral decomposition, $\mathbf{P}^n(\mathbf{I}-\mathbf{P}) = \sum_{i=1}^k \lambda_i^n(1-\lambda_i)\mathbf{G}_i = \sum_{i=2}^k \lambda_i^n(1-\lambda_i)\mathbf{G}_i$. The residual for case 1 becomes

$$r(\mathbf{x}^{(n)T}) = \|(\boldsymbol{\pi}^T + \boldsymbol{\epsilon}^T)(\lambda_2^n(1-\lambda_2)\mathbf{G}_2 + \cdots + \lambda_k^n(1-\lambda_k)\mathbf{G}_k)\|_1 = \|\boldsymbol{\epsilon}^T\,(\lambda_2^n(1-\lambda_2)\mathbf{G}_2 + \cdots + \lambda_k^n(1-\lambda_k)\mathbf{G}_k)\|_1,$$

since $\boldsymbol{\pi}^T\mathbf{G}_i = 0$, for $i = 2, 3, \ldots, k$. And similarly,

$$r(\mathbf{y}^{(n)T}) = \|\frac{1}{k}\mathbf{e}^T\,(\lambda_2^n(1-\lambda_2)\mathbf{G}_2 + \cdots + \lambda_k^n(1-\lambda_k)\mathbf{G}_k)\|_1.$$

Using the triangle inequality to bound the residual norms,

$$r(\mathbf{x}^{(n)T}) \leq 2(k-1)\,\|\boldsymbol{\epsilon}^T\|_1\,\lambda_2^n\,\max_i \|\mathbf{G}_i\|_\infty,$$

since $\lambda_i^n \leq \lambda_2^n$ for $i = 3, \ldots, k$ and $|1-\lambda_i| \leq 2$. Similarly,

$$r(\mathbf{y}^{(n)T}) =\leq 2(k-1)\,\lambda_2^n\,\max_i \|\mathbf{G}_i\|_\infty.$$

To simplify the notation, let $\alpha = 2(k-1)\,\max_i \|\mathbf{G}_i\|_\infty$. Then

$$\begin{aligned} r(\mathbf{x}^{(n)T}) &\leq \tau \text{ whenever } \lambda_2^n \leq \frac{\tau}{\alpha\|\boldsymbol{\epsilon}^T\|_1} \text{ and} \\ r(\mathbf{y}^{(n)T}) &\leq \tau \text{ whenever } \lambda_2^n \leq \frac{\tau}{\alpha}. \end{aligned}$$

For case 1, the convergence criterion $r(\mathbf{x}^{(n)T}) \leq \tau$ is met when

$$n \geq \frac{log\ \tau - (log\ \alpha + log\ \|\boldsymbol{\epsilon}^T\|_1)}{log\ \lambda_2}.$$

For case 2, the convergence criterion is met when

$$n \geq \frac{log\ \tau - log\ \alpha}{log\ \lambda_2}.$$

The difference in the lowerbounds on n is $\frac{log\ \|\boldsymbol{\epsilon}^T\|_1}{log\ \lambda_2}$. Thus, the difference is on the order of magnitude of $\frac{log\ \|\boldsymbol{\epsilon}^T\|_1}{log\ \lambda_2}$. □

We close this section by presenting one final numerical example. Consider a 100-state Markov chain with stationary vector $\boldsymbol{\pi}^T$. Suppose the starting vector for the power method is $\mathbf{x}^{(0)T} = \boldsymbol{\pi}^T + \boldsymbol{\epsilon}^T$, which is a slight perturbation of $\boldsymbol{\pi}^T$ and further suppose $\lambda_2 = .7$. In fact, suppose that just 25 states of the $\boldsymbol{\pi}^T$ vector are perturbed by .001 in absolute magnitude. Then $\|\boldsymbol{\epsilon}^T\|_1 = 25(.001) = .025$. Despite the seemingly small change in $\boldsymbol{\pi}^T$ to create $\mathbf{x}^{(0)T}$, the power method still requires nearly the same number of iterations as the power method starting from the uniform vector. The estimate of the difference in the number of iterations required by the power method starting from the uniform vector and the power method starting from the "good" approximation is

$$\frac{log\ \|\boldsymbol{\epsilon}^T\|_1}{log\ \lambda_2} = \frac{log\ .025}{log\ .7} \approx 10.$$

In this case, using a "closer, better" starting vector does not provide a great advantage. Unless $\mathbf{x}^{(0)T}$ is extremely close to $\boldsymbol{\pi}^T$, restarting an iterative method with $\mathbf{x}^{(0)T}$ has little beneficial effect. In practice, this naive approach to updating the stationary vector often amounts to full recomputation of $\boldsymbol{\pi}^T$. After demonstrating the shortcomings of the naive approach, we move on to other, more sophisticated, element-updating methods.

3 Other Element-Updating Methods

Since, in practice, the Markov matrix $\mathbf{P}$ is often modified throughout an application's lifetime, MC researchers have been studying the updating problem for some time, hoping to find $\tilde{\boldsymbol{\pi}}^T$ inexpensively without resorting to full recomputation. There have been many papers on the topic of perturbation bounds for the stationary solution of a Markov chain [16, 11, 5, 9, 20, 7]. These papers aim to produce tight bounds on the difference between $\boldsymbol{\pi}^T$ and $\tilde{\boldsymbol{\pi}}^T$, showing that the magnitude of the changes in $\mathbf{P}$ gives information about the sensitivity of elements of $\boldsymbol{\pi}^T$. However, there are some papers whose aim is to produce more than just bounds; these papers show exactly how changes in $\mathbf{P}$ affect each element in $\boldsymbol{\pi}^T$. We present a few of these methods before developing our own solution to the element-updating problem. Most Markov chain updating papers deal with updates to elements of $\mathbf{P}$, i.e., the element-updating problem. In such papers, the size of $\mathbf{P}$ is assumed to be fixed. We keep this assumption until section 4, where we drop the assumption in order to address the state-updating problem.

3.1 Updating with the group inverse

The first method for element-updating the stationary vector uses the group inverse [17, 3]. We begin this section with some definitions. 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 [17, 3, 18, 7, 5, 9] for numerous theorems and examples. Reference [7] contains an algorithm for computing $\mathbf{Q}^{\#}$ in $4n^3/3$ flops.

Meyer and Shoaf solved the element-updating problem in 1980, however, due to the expense of their solution it has remained a theoretical contribution only. We restate the theorem from [18] for element-updating the stationary vector of an ergodic Markov chain.

Theorem 3.1 *(Theorem 4.3 from [18]): 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* $\boldsymbol{\pi}^T$ *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* $\mathbf{Q}^{\#}$.

The theorem above requires that updates to elements of $\mathbf{P}$ be in the same row. To handle multiple row updates to elements of $\mathbf{P}$, the above theorem must be applied sequentially, one row at a time. Summarizing the group inverse updating method, 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 this updating method? Computationally, "no" as the group inverse updating algorithm is an $O(n^3)$ algorithm. Nevertheless, the element-updating problem has been solved *in theory* since 1980.

Similar analyses use mean first passage times, the fundamental matrix, or an **LU** factorization to update $\boldsymbol{\pi}^T$ exactly [5, 20, 12, 8]. Yet these are also expensive means of obtaining $\tilde{\boldsymbol{\pi}}^T$ and remain computationally impractical.

3.2 Updating the stationary vector approximately with Aggregation

The second method for element-updating a Markov chain comes from [4]. This method was initially created for a specific Markov chain application, updating the search engine, Google's, PageRank measure. However, the method is not limited to information retrieval and can be applied to any Markov chain. In contrast to the group inverse updating method, this method is short on computation, however, it only produces an approximation $\bar{\boldsymbol{\pi}}^T$ to $\tilde{\boldsymbol{\pi}}^T$. [1]

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 states of the Markov chain be divided into two groups. One group Ω contains all the states not likely to have their stationary probabilities affected (or affected in a negligible way) by the element updates, while G is the subset of states "near" the updates and likely to be affected. Determining which states belong to the superstate set Ω and which belong to the local graph set G is done by analyzing the transient behavior of the Markov chain [4]. Specifically, the transient flow of the chain is simulated as follows. Begin the flow simulation with equal flow in each state involved in a link update. For a small number of time steps, let this flow dissipate according to the probabilities found in the Markov probability matrix. Stop and determine the states with the greatest amount of flow. These states are "near" the link updates and belong to G. A small Markov matrix $\hat{\mathbf{P}}$ is formed by lumping all states in Ω into one superstate with the same name. The elements $\hat{p}_{ij}$ are found by using the rules provided in [4]. The important point is that $\hat{\boldsymbol{\pi}}^T$ is found by solving a generally much, much smaller Markov chain and now the problem is to determine (or, if necessary, approximate) the correct, updated $\tilde{\boldsymbol{\pi}}^T$ given $\boldsymbol{\pi}^T$ and $\hat{\boldsymbol{\pi}}^T$. The rule given in [4] for creating an approximate stationary 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 $v \in G$.

Chien et al. use the absolute 1-norm, $\|\tilde{\boldsymbol{\pi}}^T - \bar{\boldsymbol{\pi}}^T\|_1$, to judge the quality of their approximations. They note for their large, extensive examples that this distance is small ($O(10^{-7})$), indicative of a good

[1] There are some rare cases in which a minor modification to the method of [4] produces $\tilde{\boldsymbol{\pi}}^T$ exactly. This is explained in detail in [13].

approximation. We feel this is the wrong measure of "goodness" or quality in this situation. The more appropriate measure of the quality of the approximation is the cumulative relative 1-norm defined as

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

A small absolute 1-norm does not give a good indication of the quality of the approximation of [4]. Since, for their experiments, $\boldsymbol{\pi}^T$ is a probability vector with several million entries, the value of an entry is very small. Many entries may be on the order of 10^{-6}. Thus, estimates of these probabilities will also be very small. Using absolute errors to compare probabilities $\boldsymbol{\pi}_i$ and $\bar{\boldsymbol{\pi}}_i$, like .000000713 and .000000913, is deceiving. The absolute error is very low, $.0000002 = 2 \cdot 10^{-7}$. However, the relative error is .0000002/.000000713 = .28. The estimated probability is in error of the true probability by 28%; relative error provides a much more meaningful measure. 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 comparing the approximate updating method of [4] to aggregation methods in the next section, 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 states, 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.

3.2.1 Comparing the method of [4] to Aggregation Methods

Since the method of [4] aggregates the states in Ω into one superstate, we naturally explored its relationship to aggregation methods. With a little work, the method of [4] can be couched as an aggregation method, following the familiar aggregation framework outlined in [22]. The algorithm below is a restatement of the method of [4] in the language of aggregation. Assume that $\tilde{\mathbf{P}}$ has been partitioned as

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

Algorithm 1: Approximate element-updating method of [4] as an aggregation method

1. Let $\tilde{\boldsymbol{\pi}}^T = (\boldsymbol{\pi}_\Omega^T \quad \pi_{g_1} \quad \pi_{g_2} \quad \cdots \quad \pi_{g_{n_G}})$ be the given initial approximation to the solution $\tilde{\boldsymbol{\pi}}^T$. That is, let the old stationary vector $\boldsymbol{\pi}^T$ be the initial approximation to the desired updated stationary vector $\tilde{\boldsymbol{\pi}}^T$. The states in G are labeled $\{g_1, g_2, \ldots, g_{n_G}\}$.
2. Compute $\mathbf{s}_1^T$, the approximation[2] to $\tilde{\mathbf{s}}_1^T$.

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

[2]The notation and choice of the letter $\mathbf{s}$ was a deliberate effort to emphasize the connection with the exact aggregation method of stochastic complementation, described briefly in the subsequent paragraphs. The vector $\mathbf{s}_1^T$ is an approximation to the censored stationary vector $\tilde{\mathbf{s}}_1^T$ of stochastic complementation.

3. Construct the aggregation matrix $\hat{\mathbf{P}}$ according to the rules in [4].

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

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

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

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

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

We now compare the approximate aggregation matrix of step 4 above to the exact aggregation matrix. Stochastic complementation [15] enables this comparison; stochastic complementation is an exact aggregation method, which produces an exact aggregation (or coupling) matrix in step 4. Notice in the exact coupling matrix below that only the first row of $\hat{\mathbf{P}}$ and $\tilde{\mathbf{C}}$ differ. The exact coupling matrix $\tilde{\mathbf{C}}$ is

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

where $\tilde{\mathbf{s}}_1^T$ is the stationary distribution vector for the stochastic complement $\tilde{\mathbf{S}}_{11}$. $\tilde{\mathbf{S}}_{11}$ is the transition matrix for the *censored* Markov chain corresponding to the first grouping of states, Ω. This censored matrix is obtained from

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

where $\tilde{\mathbf{P}}$ has been partitioned as

$$\tilde{\mathbf{P}} = \begin{array}{c} \\ \Omega \\ G \end{array} \begin{array}{c} \begin{array}{cc} \Omega & G \end{array} \\ \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \tilde{\mathbf{P}}_{21} & \tilde{\mathbf{P}}_{22} \end{pmatrix} \end{array}.$$

Replacing the $\hat{\mathbf{P}}$ matrix with $\tilde{\mathbf{C}}$ in the aggregation algorithm above yields the exact updated stationary vector $\tilde{\boldsymbol{\pi}}^T$ after one aggregation step. However, $\tilde{\mathbf{s}}_1^T$ is needed to build the exact coupling matrix. To compute $\tilde{\mathbf{s}}_1^T$, $\tilde{\mathbf{S}}_{11}$, a matrix the size of the set Ω must be formed and solved. Thus, the exact aggregation method of stochastic complementation, like the group inverse updating method, is prohibitively expensive. See [15, 22] for further discussion of this point. Nevertheless, the impetus for our new IAD updating algorithm, presented in the next section, is the connection between the approximate updating method of [4] and stochastic complementation.

3.3 The IAD updating method

Because the $\hat{\mathbf{P}}$ matrix of [4] differed from the exact coupling matrix $\tilde{\mathbf{C}}$ in the first row only, we explored an iterative aggregation/disaggregation method (IAD). Rather than applying just one aggregation step, why not do this repeatedly, interspersing the aggregation steps with disaggregation steps needed to move repeated aggregation steps off the well-documented fixed point [22]. IAD is most commonly applied to nearly completely decomposable (NCD) Markov chains, where it has had proven success. However, IAD is usually unsuccessful in reducing work when applied to a general Markov chain. Nevertheless, we have found that, for the updating problem, IAD works well, even when applied to non-NCD Markov chains. Below is our IAD algorithm.

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

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

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: Stationary Vector 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 stationary vector $\boldsymbol{\pi}^T$ be the initial approximation to the desired updated stationary 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].

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

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

$$\bar{\boldsymbol{\pi}}^{(m)T} = \left(\hat{\pi}_1^{(m-1)}\bar{\mathbf{s}}_1^{(m-1)T} \quad \hat{\pi}_{g_1}^{(m-1)} \quad \hat{\pi}_{g_2}^{(m-1)} \quad \cdots \quad \hat{\pi}_{g_{n_G}}^{(m-1)} \right).$$

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.

At each iteration of the IAD algorithm, we improve $\hat{\mathbf{P}}$ so that $\hat{\mathbf{p}}_1^T \rightarrow \tilde{\mathbf{c}}_1^T$. This convergence is quick as long as $\mathbf{s}_1^T \approx \tilde{\mathbf{s}}_1^T$ (or equivalently, $\boldsymbol{\pi}_\Omega \approx \tilde{\boldsymbol{\pi}}_\Omega$). Consequently, it is important that Ω contain the nodes least likely to have their stationary probability changed by the updates to elements in $\mathbf{P}$. This is a key assumption of our method and shows the effect on and importance of the Ω/G partitioning for our algorithm. We feel more work should be done regarding the partition. Nevertheless, we are enthused by our preliminary results, which follow.

3.3.1 Testing the IAD element-updating algorithm

In this section we compare our IAD updating algorithm with the current, most popular alternative, full recomputation using $\mathbf{x}^{(0)T} = \boldsymbol{\pi}^T$ (the naive approach). We compare the IAD-power algorithm to full recomputation with the power method, but note that the IAD disaggregation step may be executed with any iterative method. For example, one could compare IAD-GMRES with full recomputation using GMRES. We only include the power method experiments as the results are similar using various solution techniques.

In order for our IAD 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 stationary vector. We do not report clock times, since they may be misleading for our small examples. The main expense of stationary vector 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 one tiny chain and 4 small Markov chains. The 4 small chains arise from a particular Markov application of increasing interest in information retrieval. The popular search engine, Google, uses the stationary vector (known as PageRank to IR researchers) of a Markov chain modeling web surfer's behavior to rank the importance of retrieved documents [1]. The 4 small chains are subnetworks of the Web while the tiny 10-node network was artificially created. The networks are labeled and described below. We adopt the information retrieval language and refer to states as nodes and elements of $\mathbf{P}$ as links in the graph corresponding to the transition matrix.

- `tiny10.dat`: artificial network with $n = 10$ nodes and $l = 21$ 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 [19].
- `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 [14].

We describe the notation used in Tables 1-5.

- The stopping criterion is that the 1-norm of the residual be less than 10^{-10}. The updating algorithms were coded in Matlab and run on a SUN Ultra 10 workstation.
- The scalars r and a refer to the number of links removed and added, respectively, to the original matrix $\mathbf{P}$ 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 element-updating methods on `tiny10.dat` ($n = 10$, $l = 21$)

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

3.3.2 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

Table 2: Comparison of IAD and full recomputation element-updating methods on `movies.dat` ($n = 451$, $l = 713$)

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

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.

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 [10]. 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 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

Table 3: Comparison of IAD and full recomputation element-updating methods on `mathworks.dat` ($n = 517$, $l = 13,531$)

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

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 [21].

4 What happens when the size of P changes?

Recall in the first section that we distinguished between two types of updates to the transition probability matrix **P**: element-updates and state-updates. Having satisfactorily resolved the issue of element-updates, we now turn to state-updates and consider the question: "How does one update the stationary

Table 4: Comparison of IAD and full recomputation element-updating methods on `impeachment.dat` ($n = 1336$, $l = 2604$)

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

vector when the size of the transition matrix changes?" Theoretically, this is a much more challenging problem. To our knowledge, no answer to this question, theoretical or numerical, exists. We propose a numerical solution. In fact, the same IAD algorithm detailed in section 3.3 can be used to accomodate state-updates. This is convenient from an implementation perspective; the same algorithm handles both types of updates.

After state additions, the new Markov matrix becomes

Table 5: Comparison of IAD and full recomputation element-updating methods on `abortion.dat` ($n = 1693$, $l = 4325$)

r	a	$\lvert G\rvert$	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

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

Notice that $\tilde{\mathbf{P}}$ is just $\mathbf{P}$ augmented by rows and columns corresponding to the additional states. 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 state-updating IAD algorithm will not work. We modify step 1 for the state-updating problem,

1. Let $\tilde{\boldsymbol{\pi}}^{(0)T} = \begin{pmatrix} \boldsymbol{\pi}_\Omega^{(0)T} & 0 & 0 & \cdots & 0 \end{pmatrix}$ 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 states to 0.

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

When the Markov chain downsizes with the removal of states, state-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 states remaining after the removal of the downdated states. Set $m = 1$.

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

We tested the IAD algorithm on $\mathbf{P}$ matrices that were state-updated. We use the same datasets as the element-updating experiments and the same notation for the tables. Here a represents the number of states 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 [6, 2].

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

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

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 element-updating and node-updating. We increased the granularity of $|G|$ to pinpoint the dropoff point for each dataset and number of link updates. Figure 1 shows the dropoff point for the `mathworks.dat` link-updating experiments. 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 location in the node-updating experiments for this same dataset. See Figure 2.

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 [15] has shown that the eigenvalues of the stochastic complements govern the convergence of an exact aggregation method.

Table 7: Comparison of IAD and full recomputation state-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

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 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 [15], 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, little change in $\lambda_2(\tilde{\mathbf{S}}_{11})$ is observed and little is gained in terms of convergence. Reference [15] explains

Table 8: Comparison of IAD and full recomputation state-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

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 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

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

a	$\|G\|$	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

these calculations should hold for many, many subsequent months.

5 Conclusions and Future Work

We discussed several algorithms for updating the stationary vector of a Markov chain in the presence of element-updates and state-updates. Our IAD updating algorithm is an exact updating method that exploits the old stationary vector to create the new stationary vector. This method 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 5 of the algorithm, we plan to apply the stable GTH direct method to find $\hat{\boldsymbol{\pi}}^T$. Second, we have several ideas for accelerating the power method of step 6. The third area of continued study is the Ω/G partitioning. We would like to formalize this step, providing heuristics to cheaply determine the optimal size and members of G, as this partitioning plays such a crucial role in the convergence of the IAD method. Finally, we emphasize that this IAD algorithm handles both element-updates and state-updates. It is, to our knowledge, the only algorithm to address the challenging problem of state-updating. We also note the potential applicability of the IAD algorithm to eigenvector updating problems and linear system updating problems as well.

Table 10: Comparison of IAD and full recomputation state-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

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.

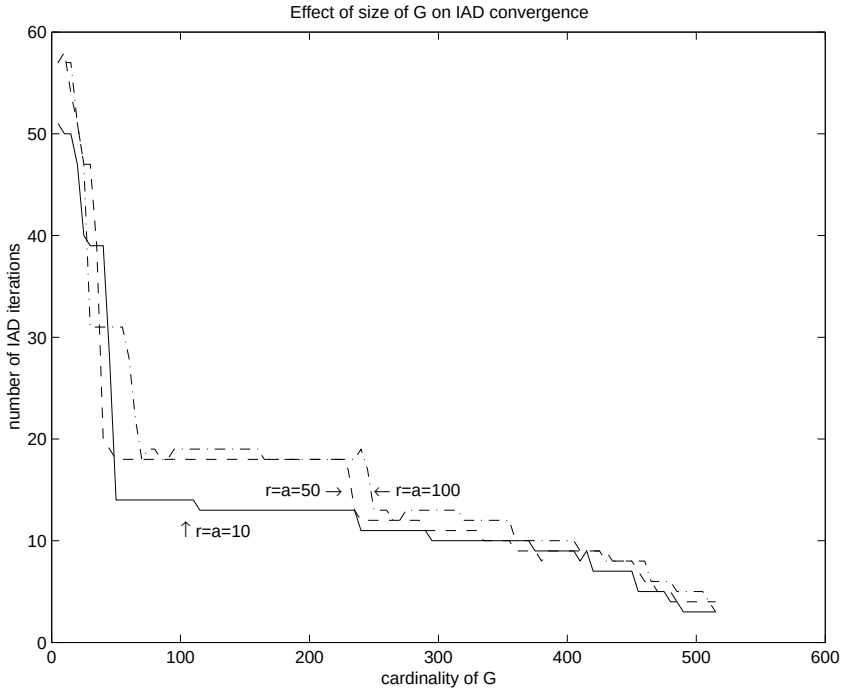

Figure 1: Dropoff point ≈ 40 for `mathworks.dat` for element-updating experiments

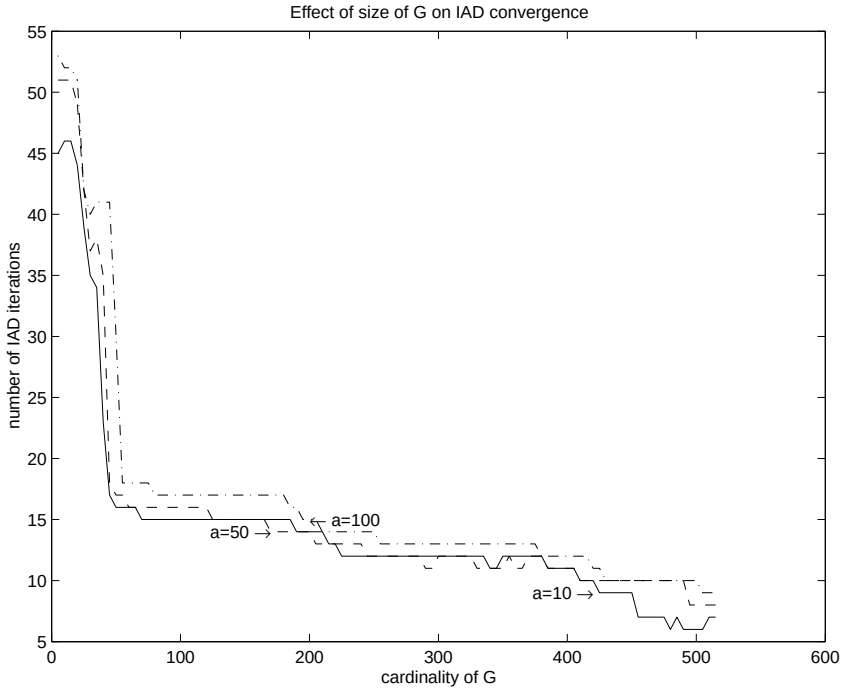

Figure 2: Dropoff point ≈ 40 for `mathworks.dat` for state-updating experiments

References

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

[2] Andrei Broder, Ravi Kumar, and Marzin Maghoul. Graph structure in the web. In *The Ninth International WWW Conference*, May 2000.

[3] Steven Campbell and Carl D. Meyer. *Generalized Inverses of Linear Transformations.* Pitman, San Francisco, 1979.

[4] Steve Chien, Cynthia Dwork, Ravi Kumar, and D. Sivakumar. Towards exploiting link evolution.

[5] Grace E. Cho and Carl D. Meyer. Comparison of perturbation bounds for the stationary distribution of a Markov chain. *Linear Algebra and its Applications*, 335(1–3):137–150, 2001.

[6] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[7] Robert E. Funderlic and Carl D. Meyer. Sensitivity of the stationary distribution vector for an ergodic Markov chain. *Linear Algebra and its Applications*, 76:1–17, 1986.

[8] Robert E. Funderlic and Robert J. Plemmons. Updating **lu** factorizations for computing stationary distributions. *SIAM Journal on Algebraic and Discrete Methods*, 7(1):30–42, 1986.

[9] Gene H. Golub and Carl 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*, 17:273–281, 1986.

[10] Winfried K. Grassmann, M. I. Taskar, and Daniel P. Heyman. Regenerative analysis and steady state distributions for Markov chains. *Operations Research*, 33(5):1107–1116, 1985.

[11] Ilse C. F. Ipsen and Carl D. Meyer. Uniform stability of Markov chains. *SIAM Journal on Matrix Analysis and Applications*, 15(4):1061–1074, 1994.

[12] John G. Kemeny and Laurie J. Snell. *Finite Markov Chains.* D. Van Nostrand, New York, 1960.

[13] Amy N. Langville and Carl D. Meyer. Updating pagerank using the group inverse and stochastic complementation. Technical Report crsc02-tr32, North Carolina State University, Mathematics Department, CRSC, 2002.

[14] Ronny Lempel and S. Moran. The stochastic approach for link-structure analysis (salsa) and the tkc effect. In *The Ninth International WWW Conference*, May 2000.

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

[16] Carl D. Meyer. Sensitivity of the stationary distribution of a Markov chain. *SIAM Journal on Matrix Analysis and Applications*, 15(3):715–728, 1994.

[17] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra.* SIAM, Philadelphia, 2000.

[18] Carl D. Meyer and James M. Shoaf. Updating finite Markov chains by using techniques of group matrix inversion. *Journal of Statistical Computation and Simulation*, 11:163–181, 1980.

[19] Cleve Moler. The world's largest matrix computation. *Matlab News and Notes*, pages 12–13, October 2002.

[20] Eugene Seneta. Sensivity analysis, ergodicity coefficients, and rank-one updates for finite Markov chains. In William J. Stewart, editor, *Numerical Solutions of Markov Chains*, pages 121–129, 1991.

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

[22] William J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.