Modeling the Web and the Computation of PageRank

Kristen Thorson

April 15, 2004

Table of Contents

1 Introduction

Information retrieval on the web is much more challenging than traditional information retrieval (IR) systems. The web has a highly volatile structure: every day pages are added, deleted, and modified. The size of the web is on the order of more than a billion pages, and many of those pages contain redundant or incorrect information. A search tool on the web must be able to distinguish high-quality pages from low-quality pages. In addition, users of web search tools also present a challenge to IR researchers and developers. The average web IR user enters very short queries, does not make use of system feedback to revise the query, seldom performs a search using advanced search options, and generally views only the top 10-20 documents returned by the search [5].

A simple textual search of a large set of web pages will most often return skewed results. The underlying commercial structure of the web and the nature of the average web surfer encourages the spamming of web search engines (artificially increasing the likelihood a certain page will be returned by a query search) [2]. Ideally, a search engine should be immune to spamming and pages should be ranked by query relevance. Determining a page's relevance to query terms is a complex problem for an IR system, but the inherent hyperlink structure of the web may be used to generate an approximation of relevancy. This idea is implemented in the PageRank algorithm, first devised by Sergey Brin and Larry Page at Stanford University in 1997, and implemented by Google, a highly successful web search engine.

In the case of Google and PageRank, performing a user search requires two general steps:

1. perform text query: group and locally rank pages according to traditional IR methods and
2. merge these results with the global PageRank scores to order the documents returned by the search [2].

Traditional IR methods involve a textual search of the pages in the web where query terms

are matched to the documents and a relevancy set is created. The relevancy set is the group of pages most closely related to the query terms. One stalwart of traditional IR methods is Latent Semantic Indexing (LSI), a mathematical algorithm able to define semantic associations between words [5]. A Google query also utilizes page formatting information to create the relevancy set. Factors such as position on the page, font, size, and capitalization influence rank within the relevancy set.

Brin and Page note that web documents are notoriously poor at self-description. Many sites only contain a company or site name in the title. Brin and Page stress that anchor text should be associated both with the originating page and the linked page, because links often provide more complete and concise information about documents than text contained in the actual document. This method also allows Google to perform searches on documents that cannot be indexed by a text-based search engine, like images, programs, and databases [2].

Once a relevancy set is created containing documents relevant to the query terms, the Google system merges relevancy set rankings with PageRank scores and displays the results. The PageRank values are pre-calculated and stored for all pages known to the IR system. This means every page in the web has a PageRank score that is completely independent of query terms. A search that returns PageRank scores is reporting the importance hierarchy of pages containing the query terms.

Other popular web IR systems include HITS, which creates two rankings, a hub score and an authority score, and SALSA, which attempts to utilize the strengths of both PageRank and HITS. The focus of this paper is on PageRank, an algorithm introduced in 1998 by Brin and Page. Much research has been devoted to improving the computation of PageRank while maintaining the same basic mathematical model. In this paper, we also consider the adaptive algorithms introduced by Kamvar et al. in [3]. Finally, we present the results of small-scale (compared to the size of the web) experiments on the PageRank and adaptive PageRank algorithms running *Matlab 6.0 R12 student version* on an AMD

Athlon 64 3200+ 2GHz processor with 1 GB RAM.

2 The Hyperlink Structure of the Web

A set of pages in the web may be modeled as nodes in a directed graph. The edges between nodes represent links between pages. A graph of a simple 6-page web is depicted in Figure 1 below. The directed edge from node two to node three signifies that page two links to page three. However, page three does not link to page two, so there is no edge from node three to node two.

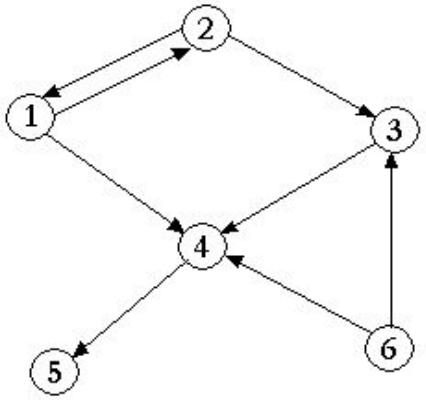


Figure 1: Graph modeling 6-node web

The PageRank thesis constructs page importance hierarchies based upon the link structure of the web. An inlink from a page may be seen as a recommendation by the author. Generally, more important pages will have more inlinks. Inlinks from important pages will also have a greater effect on PageRank for a particular page than inlinks from marginal pages. The calculation of PageRank is recursive, building the rank for a particular page based on the ranks of the pages that link to it.

To calculate PageRank, we must begin by building a mathematical model of the link structure of the web. We can construct an adjacency matrix $\boldsymbol{L}$ from the graph of the web where

$$\boldsymbol{L}_{ij} = \begin{cases} 1, & \text{if there exists an edge from node } i \text{ to node } j, \\ 0, & \text{otherwise.} \end{cases}$$

All entries of $\boldsymbol{L}$ are then either 0 or 1. An entry of 1 in the second row, third column would indicate that page two links to page three. For the 6-node graph in Figure 1, the adjacency matrix $\boldsymbol{L}$ is

$$\boldsymbol{L} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The matrix $\boldsymbol{L}$ is used directly in computation by HITS, which was developed during the same period as PageRank [4]. The HITS thesis constructs page ranking based on number of inlinks and outlinks, while the PageRank thesis constructs page ranking based on the *importance* of inlinks and outlinks. The underlying assumption in the PageRank thesis is that a web user is more likely to visit more important pages.

Consider a theoretical web user, navigating through a series of web pages. As the user views a page, he or she may follow any one of the page's outlinks to another page in the web. Each row, i, of the matrix $\boldsymbol{L}$ represents the outlinks from the corresponding page. The row may be reconstructed as a probability distribution for the movement of the web user, so that each entry in a row instead signifies the conditional probability that a user currently visiting page i would next visit page j.

Consider row one from the adjacency matrix $\boldsymbol{L}$ above. The matrix reveals that page one has two outlinks to pages two and four. Assuming that a web user is equally likely to follow all outlinks on any given page, we can then construct a new matrix $\boldsymbol{P}$ where $\boldsymbol{P}_{ij}$ is the probability of moving from node i to node j. Constructing $\boldsymbol{P}$ from $\boldsymbol{L}$, we get the following matrix:

$$\boldsymbol{P}=\begin{bmatrix} 0 & .5 & 0 & .5 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The accuracy of the PageRank score for the web may be improved by analyzing web usage logs to determine whether the probability of moving from node *i* to node *j* is the same for each outlink. For example, if usage logs show that a user is more likely to follow a link from page one to page two than from page one to page four, the first row of $\boldsymbol{P}$ may be

$$\boldsymbol{P}_1=\begin{bmatrix} 0 & .6667 & 0 & .3333 & 0 & 0 \end{bmatrix}.$$

For simplicity, we will assume throughout this paper that a user is equally likely to move from node *i* to each outlinked node, so the formal method for filling in the entries of $\boldsymbol{P}$ is

$$\boldsymbol{P}_{ij}=\begin{cases} \dfrac{1}{|O_i|}, & \text{if there exists an edge from node } i \text{ to node } j, \\ 0, & \text{otherwise}, \end{cases}$$

where $|O_i|$ is the cardinality of O_i, the set of outlinked nodes from node *i*. The construction of the transition probability matrix $\boldsymbol{P}$ from the network graph in this manner is the first step towards the mathematical model defining PageRank.

3 Markov Chain Model of the Random Surfer

We may assign PageRank to pages based on the information given in the matrix $\boldsymbol{P}$. Suppose a "random surfer" navigates through a series of pages, successively following links at random. The PageRank of a particular page may be defined as the long-term probability that the surfer will end up at that page, regardless of starting position [7]. Equivalently, if millions of users are surfing the web simultaneously, a page's PageRank is the percentage of viewers expected to be on that page at any given time. We recognize this probability vector as the stationary vector for a Markov Chain.

Consider the following vector:

$$m = \begin{bmatrix} 0 \\ .60 \\ .20 \\ .20 \end{bmatrix}.$$

The vector *m* could be a probability distribution for a set of four pages indicating that a user is 60% likely to visit page two, 20% likely to visit pages three and four, and not likely to visit page one at all from the current location. The vector *m* is called a discrete probability vector if all entries are nonnegative and the column sum is 1 or

$$\| m \|_1 = \sum m_i = 1.$$

A stochastic matrix $\boldsymbol{P}$ is a non-negative matrix composed entirely of probability vectors so that $\boldsymbol{P}e = e$ (where *e* is a vector of all ones) and $\boldsymbol{P} \geq 0$ ($P_{ij} \geq 0$ for all *i, j*). The condition $\boldsymbol{P}e = e$ states that the sum of each row is equal to 1. A Markov Chain is then defined as a sequence of probability vectors $m_0, m_1, m_2, m_3, \ldots$ such that

$$m_1^T = m_0^T \boldsymbol{P}, \quad m_2^T = m_1^T \boldsymbol{P}, \quad m_3^T = m_2^T \boldsymbol{P}, \ldots$$

and

$$m_{k+1}^T = m_k^T \boldsymbol{P}, \quad \text{for } k = 0, 1, 2, \ldots$$

An entry of the state vector m_k describes the probability that a user is visiting a particular page at time step *k*. Under certain assumptions, subsequent iterations of $m_{k+1}^T = m_k^T \boldsymbol{P}$ will converge to a stationary vector *q,* independent of m_0, such that

$$q^T \boldsymbol{P} = q^T.$$

The stationary vector *q* may be interpreted as the long-term probability distribution for the pages in the web [6]. Suppose an initial state vector for a set of four web pages is

$$m_0 = \begin{bmatrix} 0 \\ .60 \\ .20 \\ .20 \end{bmatrix}$$

and eventually converges to

$$q = \begin{bmatrix} .10 \\ .50 \\ .15 \\ .25 \end{bmatrix}.$$

Initially, a user is 60% likely to be at page two. The vector q reports that a user is 50% likely to be visiting page two at some distant future time independent of the current location. The vector q could also be interpreted as "at any given time, half of all surfers are visiting page two."

The PageRank thesis posits that a page is important if other important pages link to it. The importance z_i of a page is calculated as

$$z_i = \sum_j \boldsymbol{P}_{ji} z_j.$$

Thus, the PageRank of page i is the sum of the PageRanks of the pages that link to page i, multiplied by the respective transition probabilities of $\boldsymbol{P}$. If a particular page that links to page i has a high PageRank, this will affect z_i more than an inlinking page with low PageRank. The PageRanks of a web of n pages are given by

$$\begin{aligned} z_1 &= P_{11} z_1 + P_{21} z_2 + P_{31} z_3 + \cdots + P_{n1} z_n \\ z_2 &= P_{12} z_1 + P_{22} z_2 + P_{32} z_3 + \cdots + P_{n2} z_n \\ z_3 &= P_{13} z_1 + P_{23} z_2 + P_{33} z_3 + \cdots + P_{n3} z_n \\ &\vdots \\ z_n &= P_{1n} z_1 + P_{2n} z_2 + P_{3n} z_3 + \cdots + P_{nn} z_n. \end{aligned}$$

In matrix terms, this may be written as $z = \boldsymbol{P}^{\boldsymbol{T}} z$ or $z^T = z^T \boldsymbol{P}$. Recognize that z is a left-hand eigenvector of $\boldsymbol{P}$ corresponding to $\lambda = 1$ [5]. Thus computing PageRank, the stationary vector of a Markov Chain, is computationally equivalent to finding the eigenvector associated with a known eigenvalue, $\lambda = 1$.

A Markov Chain is irreducible if every state is reachable from every other state. If $\boldsymbol{P}$ is positive ($\boldsymbol{P} > 0$ if $\boldsymbol{P}_{ij} > 0$ for all i, j), then each state is reachable from every other state. Since all entries are greater than zero, given any initial state, any other state may be reached in only one step. Thus, if $\boldsymbol{P}$ is positive, it is also irreducible. An irreducible,

stochastic matrix ***P*** is guaranteed to have a stationary vector by the Perron-Frobenius Theorem:

> If a square matrix ***A*** is positive, stochastic, and irreducible, then it has a simple eigenvalue equal to 1, with a corresponding positive eigenvector. All other eigenvalues are smaller in modulus.

The dominant eigenvalue of ***A*** is labeled $\lambda_1 = 1$, while all other eigenvalues are given the notations $\lambda_2, \lambda_3, \lambda_4, \ldots \lambda_n$ [8]. This theorem guarantees that a unique PageRank vector exists, provided that ***P*** meets the necessary hypotheses. Furthermore, we may take advantage of any computational algorithm designed to find the eigenvector associated with a known eigenvalue.

4 Modeling the Human Surfer

As it stands, our Markov matrix ***P*** does not satisfy $\boldsymbol{P} > 0$, and may not be irreducible or stochastic. Fortunately, these mathematical technicalities coincide with web modelling issues.

The random surfer model does not closely model the movement of a human surfer, in that a human user always has the option of randomly jumping to another page in the web. Should a user come to a page with no outlinks (such is the case with page five in our 6-node web), he or she will presumably not remain on that page forever. At this point, to continue surfing, the user is forced to jump to another page in the web. This ability is always present however, since at any point in time the user may manually enter a URL. Therefore, every page in the web is implicitly linked to one another through the ability of the user to randomly jump to another page.

Since the human user will presumably not remain on a page with no outlinks indefinitely, we must adjust ***P*** accordingly. We assume that the user is equally likely to jump to any other page in the web. For a matrix ***P***, construct $\bar{\boldsymbol{P}}$ by replacing the entries in

a row of zeros with $\frac{1}{n}e^T$, if such a row exists, where n is the order of $\boldsymbol{P}$. Fortunately, this also makes $\bar{\boldsymbol{P}}$ stochastic, since every row of $\bar{\boldsymbol{P}}$ is a probability vector.

Brin and Page add an adjustment matrix $\boldsymbol{E}$ to $\bar{\boldsymbol{P}}$, which directly links every page in the web. The adjustment matrix $\boldsymbol{E}$ is constructed as $\frac{1}{n}ee^T$, where n is the order of $\boldsymbol{P}$. Even though the user always has the option of jumping to any other page in the web, he or she will not always choose to do so. Therefore, we must introduce another factor, α. Google reportedly uses an $\alpha = .85$, which indicates that the model used by Google assumes that approximately 85% of the time, a user will simply follow successive links on pages in the web. However, 15% of the time, the user will choose instead to jump to another page in the web. Thus, we construct our new matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} = \alpha \bar{\boldsymbol{P}}^{\boldsymbol{T}} + (1-\alpha)\boldsymbol{E}.$$

The introduction of $\boldsymbol{E}$ also serves to make $\boldsymbol{A}$ irreducible, since $\boldsymbol{A}$ is now positive. Recall from the Perron-Frobenius theorem that a positive, stochastic, irreducible matrix is guaranteed to have a positive eigenvector. We can formally construct $\boldsymbol{A}$ from $\bar{\boldsymbol{P}}$ as

$$\boldsymbol{A} = \alpha \bar{\boldsymbol{P}}^{\boldsymbol{T}} + \frac{1-\alpha}{n}\boldsymbol{e}\boldsymbol{e}^{\boldsymbol{T}},$$

where α is the probability the user will choose to follow a link on the page, and $(1 - \alpha)$ is the probability that the user will opt to jump randomly to another page in the web. Note that the original $\boldsymbol{P}$ has been transposed to conform with usual convention of finding right, instead of left, eigenvectors.

5 Computation and the Power Method

Finding the eigenvector for a given eigenvalue and matrix can be a complex computation. There are several available methods for eigenvector calculation; the power method is often the method of choice, due to issues of storage, computation time, and complexity [5]. The power method is an iterative method that finds the vector x^{k+1} from

x^k as $x^{k+1} = \boldsymbol{A}\, x^k$ until x^{k+1} converges within a desired tolerance. When x^{k+1} converges, that vector is the eigenvector for the given matrix and dominant eigenvalue (which in this case is 1). The PageRank algorithm is an application of the power method:

function PageRank($\boldsymbol{A}$, z^0)
repeat
$z^{k+1} = \boldsymbol{A}\, z^k$;
$\delta = \left\| z^{k+1} - z^k \right\|_1$;
until $\delta < \epsilon$;
return z^{k+1} ;

where δ is the change from the k^{th} iteration to the $k+1^{th}$ iteration and ϵ is the desired convergence threshold. Notice that the power method requires repeated matrix-vector products, which can be implemented efficiently using specialized algorithms designed for compressed matrices.

The irreducibility of $\boldsymbol{A}$ implies that $\lambda_1(\boldsymbol{A}) = 1$. In addition, we can show that $\lambda_2(\boldsymbol{A}) \le \alpha < 1$ and that choosing an α farther from 1 will speed convergence of the power method.

Theorem: The power method on $\boldsymbol{A}$ converges at a geometric rate of $|\lambda_2|$.

Proof: Assume $\boldsymbol{A}$ has eigenvalues $\{ \lambda_1 = 1, \lambda_2, \lambda_3, ..., \lambda_n \}$ ordered by magnitude and $| \lambda_k | < 1$ if $k \neq 1$. Assume there is a full basis of eigenvectors given by $\boldsymbol{A} v_k = \lambda_k v_{k.}$. Since any vector may be expressed as a linear combination of eigenvectors, given a starting vector x_0,

$$x_0 = \sum_{k=1}^{n} \theta_k v_k ,$$

for some constants θ_k. Successive iterations of the power method give:

$$
\begin{aligned}
x_1 = \boldsymbol{A}x_0 &= \boldsymbol{A}\left(\sum_{k=1}^{n}\theta_k v_k\right) \\
&= \sum_{k=1}^{n}\theta_k \boldsymbol{A}v_k \\
&= \sum_{k=1}^{n}\theta_k \lambda_k v_k \\
x_2 = \boldsymbol{A}x_1 &= \boldsymbol{A}x_1 = \boldsymbol{A}\left(\sum_{k=1}^{n}\theta_k \lambda_k v_k\right) \\
&= \sum_{k=1}^{n}\theta_k \lambda_k \boldsymbol{A}v_k \\
&= \sum_{k=1}^{n}\theta_k \lambda_k^2 v_k \\
x_3 = \boldsymbol{A}x_2 &= \boldsymbol{A}\left(\sum_{k=1}^{n}\theta_k \lambda_k^2 v_k\right) \\
&= \sum_{k=1}^{n}\theta_k \lambda_k^2 \boldsymbol{A}v_k \\
&= \sum_{k=1}^{n}\theta_k \lambda_k^2 \boldsymbol{A}v_k \\
&= \sum_{k=1}^{n}\theta_k \lambda_k^3 v_k \\
&\vdots \\
x_j = \boldsymbol{A}^j x_0 &= \sum_{k=1}^{n}\theta_k \lambda_k^j v_k \\
&= \theta_1 v_1 + \sum_{k=2}^{n}\theta_k \lambda_k^j v_k
\end{aligned}
$$

Since $|\lambda_k| < 1$ for $k > 1$,

$$\lim_{j\to\infty}\lambda_k^j = 0$$

and

$$x_j \to \theta_1 v_1 .$$

The power method eventually converges to a multiple of the desired eigenvector v_1. If λ_2 is the second largest eigenvalue, then the rate of convergence depends upon how fast λ_2^j converges to zero. ■

The value of α affects the size of $|\lambda_2|$, and hence, how fast the power method converges. Suppose the stochastic matrix $\bar{\boldsymbol{P}}$ has eigenvalues $\lambda(\bar{\boldsymbol{P}}) = \{1, \lambda_2, \lambda_3, \ldots, \lambda_n\}$ arranged in decreasing size so that $1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n$. To force irreducibility and model the movements of a human surfer who may make non-link jumps to other pages in the web, the PageRank thesis constructs the matrix $\boldsymbol{A}$ such that

$$\boldsymbol{A} = \alpha \bar{\boldsymbol{P}}^{\boldsymbol{T}} + \frac{1-\alpha}{n} e e^T.$$

Theorem: The eigenvalues of $\boldsymbol{A}$ are then given by $\lambda(\boldsymbol{A}) = \{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$.

Proof: (Based on the proof by Langville and Meyer in [4]) Suppose (λ, x) is an eigenpair of $\boldsymbol{A}$. Then, $\boldsymbol{A}x = \lambda x$, which can be expressed as $(\boldsymbol{A} - \lambda \boldsymbol{I})x = 0$. This implies that $(\boldsymbol{A} - \lambda \boldsymbol{I})$ is singular, and therefore $\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$. We will compute this determinant and express it as a polynomial in λ, whose roots are the eigenvalues of $\boldsymbol{A}$.

$$\begin{aligned}
det\left(\boldsymbol{A} - \lambda \boldsymbol{I}\right) &= det\left(\alpha \bar{\boldsymbol{P}} + \frac{1-\alpha}{n} e e^T - \lambda \boldsymbol{I}\right) \\
&= det\left(\alpha \bar{\boldsymbol{P}} - \lambda \boldsymbol{I} + (\alpha - \lambda)\frac{1-\alpha}{(\alpha-\lambda)n} e e^T\right) \\
&= det\left(\alpha \bar{\boldsymbol{P}} - \lambda \boldsymbol{I} + \alpha\frac{1-\alpha}{(\alpha-\lambda)n} e e^T - \lambda\frac{1-\alpha}{(\alpha-\lambda)n} e e^T\right) \\
&= det\left(\alpha \bar{\boldsymbol{P}} - \lambda \boldsymbol{I} + \alpha\frac{1-\alpha}{(\alpha-\lambda)n} \bar{\boldsymbol{P}} e e^T - \lambda\frac{1-\alpha}{(\alpha-\lambda)n} e e^T\right).
\end{aligned}$$

Note that $\bar{\boldsymbol{P}} e = e$ since $\bar{\boldsymbol{P}}$ is stochastic. So,

$$\begin{aligned}
det\left(\boldsymbol{A} - \lambda \boldsymbol{I}\right) &= det\left(\left(\alpha \bar{\boldsymbol{P}} - \lambda \boldsymbol{I}\right)\left(\boldsymbol{I} + \frac{1-\alpha}{(\alpha-\lambda)n} e e^T\right)\right) \\
&= det\left(\alpha \bar{\boldsymbol{P}} - \lambda \boldsymbol{I}\right) det\left(\boldsymbol{I} + \frac{1-\alpha}{(\alpha-\lambda)n} e e^T\right).
\end{aligned}$$

At this point, let $\boldsymbol{B} = \boldsymbol{I} + \frac{1-\alpha}{(\alpha-\lambda)n} e e^T$ and recall that the determinant is the product

of the eigenvalues:

$$det(\boldsymbol{B}) = \prod \lambda(\boldsymbol{B}).$$

Fortunately, the eigenvalues of $\boldsymbol{B}$ are easily recognized. First note that

$$\boldsymbol{B}e = \left(1 + \frac{1-\alpha}{\alpha-\lambda}\right)e = \frac{1-\lambda}{\alpha-\lambda}e$$

describes one eigenpair, $\left(\frac{1-\alpha}{\alpha-\lambda}, e\right)$. Any $w \perp e$ is also an eigenvector with $\lambda = 1$, because $\boldsymbol{B}w = w$. Therefore, all other eigenvalues are 1 and

$$\lambda(\boldsymbol{B}) = \left\{\frac{1-\lambda}{\alpha-\lambda}, 1, 1, \ldots, 1\right\} \quad \Rightarrow \quad det(\boldsymbol{B}) = \frac{1-\lambda}{\alpha-\lambda}.$$

Returning now to the original expression:

$$\begin{aligned} det\left(\boldsymbol{A} - \lambda \boldsymbol{I}\right) &= det\left(\alpha \bar{\boldsymbol{P}} - \lambda \boldsymbol{I}\right) det(\boldsymbol{B}) \\ &= det\left(\alpha \bar{\boldsymbol{P}} - \lambda \boldsymbol{I}\right)\left(\frac{1-\lambda}{\alpha-\lambda}\right) \\ &= \left(\alpha-\lambda\right)\left(\alpha\lambda_2-\lambda\right)\left(\alpha\lambda_3-\lambda\right)\ldots\left(\alpha\lambda_n-\lambda\right)\frac{1-\lambda}{\alpha-\lambda} \\ &= \left(1-\lambda\right)\left(\alpha\lambda_2-\lambda\right)\left(\alpha\lambda_3-\lambda\right)\ldots\left(\alpha\lambda_n-\lambda\right). \end{aligned}$$

The roots of this polynomial are

$$\lambda(\boldsymbol{A}) = \{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}.$$

The primary importance of this result is that the power method applied to $\boldsymbol{A}$ will converge at a rate given by

$$\left|\frac{\lambda_2(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right| \leq \left|\frac{\alpha}{1}\right| = \alpha.$$

■

Since, $|\lambda_2| \leq \alpha$ and the rate of convergence of the power method depends on the size of the second largest eigenvalue, the choice of α will determine the rate of convergence of the power method. A change in α will greatly reduce the total number of

iterations needed to perform the PageRank algorithm, and may also drastically affect the PageRanks of pages in the web. A higher value of α will place more weight on the link structure of the web, but will cost more in terms of computation time.

We ran the power method using various values of α on a 6012 x 6012 matrix modelling the hollins.edu web with a convergence tolerance of 10e-8. Running the code of Appendix A in *Matlab 6.0 R12 student version* produces the following results:

α	***Iterations***
.99	1283
.95	255
.85	84
.75	49
.5	22

Table 1: Effect of α on number of iterations

Note that a change in α will have serious effects on the number of iterations needed for the power method to converge to the desired tolerance (see Appendix C for graphs). In fact, geometric convergence allows us to estimate the number of iterations by $\frac{-6}{\log \alpha}$. Performing PageRank calculations on the hollins.edu domain produces differing PageRanks according to the value of α (See Appendix B).

6 PageRank Implementation

It is important to note that the implementation of PageRank in an IR system requires several steps. At the outset, query terms must be matched to pages in the web and a relevancy set created. The relevancy set is then sorted by PageRank values. In most cases, PageRank seems to serve as a good approximation for query relevance, and advanced methods of text searching can improve the search quality of the IR system. For example, Google assigns weights to terms in pages in the web. Words in comparatively

large or bold font (such as article titles) are weighted higher. Outside information known as meta data includes information about the reputation of the source, frequency of updates, and usage statistics. The text of an outgoing link is associated both with the linking page and the linked page since the text of an inlink often provides more accurate information about page content than the page itself [2]. The implementation of query parsing and text searching is beyond the scope of this paper, but plays a very important role in the accuracy of Google's answer to a query.

The resulting PageRank values for the web are greatly affected by the choice of α and by the chosen convergence tolerance. Google reportedly uses a value of α = .85, but the convergence test is proprietary information. In 1998, Brin and Page claimed that a web of approximately 322 million pages converged in about 52 iterations and a web half that size converged in about 45 iterations. Our experiments indicate that the number of iterations required to reach convergence depends primarily on the hyperlink structure of the web, rather than its size (see Appendix E).

7 PageRank Example

Consider the small 6-node web from Figure 1. Recall that the transition probability matrix $\boldsymbol{P}$ for this graph was

$$\boldsymbol{P} = \begin{bmatrix} 0 & .5 & 0 & .5 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Note that row five does not have any outlinks, creating a row of zeros. Every entry in $\boldsymbol{P}$ must be replaced with $\frac{1}{n} e^{\mathrm{T}}$. This yields the matrix $\bar{\boldsymbol{P}}$ as

$$\bar{\boldsymbol{P}} = \begin{bmatrix} 0 & .5 & 0 & .5 & 0 & 0 \\ .5 & 0 & .5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ .1667 & .1667 & .1667 & .1667 & .1667 & .1667 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Recall that $\bar{\boldsymbol{P}}$ is now stochastic but may not be irreducible. With $\alpha = .85$, the adjusted matrix $\boldsymbol{A}$ is constructed from $\bar{\boldsymbol{P}}$ as

$$\boldsymbol{A} = \begin{bmatrix} .025 & .45 & .025 & .025 & .1667 & .025 \\ .45 & .025 & .025 & .025 & .1667 & .025 \\ .025 & .45 & .025 & .025 & .1667 & .025 \\ .45 & .025 & .875 & .025 & .1667 & .875 \\ .025 & .025 & .025 & .875 & .1667 & .025 \\ .025 & .025 & .025 & .025 & .1667 & .025 \end{bmatrix}.$$

After 24 iterations and a tolerance of 10e-8, the PageRank scores for this web converge to

$$z^T = \begin{bmatrix} .1179706 & .1179706 & .1179706 & .2759037 & .3023513 & .0678331 \end{bmatrix}.$$

These PageRank scores rank pages four and five highest, meaning they have the greatest importance. Note that page five has no outlinks, and only one inlink. Since page four has three inlinks, it has a high PageRank. Page five gets a high PageRank due to the one inlink from page four. The single inlink from an important page makes page four the most important page, even though other pages in the web have more inlinks. This is exactly the desired effect of the PageRank algorithm.

Appendix D displays the PageRank vector at each iteration. Note that some pages in the web seem to converge to the final PageRank values quickly, while other pages take many more iterations before they begin to approach their final PageRank values.

8 Strengths and Weaknesses of PageRank

One of PageRank's strengths is query independence, since PageRank is calculated for all pages in the web, independent of any query. However, this facet may also be

considered a weakness due to the volatile nature and vast size of the web. The web contains so many documents that performing PageRank calculations must be done utilizing distributed methods of computing, so that portions of the calculations are completed on separate computers. Prior to calculating PageRank for the known web, a web crawl must be done to build the transition probability matrix. Both of these tasks are extremely time-consuming, taking several days even on state-of-the art distributed computing clusters. For this reason, Google recomputes PageRank only once every month. For less volatile mediums, calculating PageRank once a month may produce reasonably accurate results, but the structure and content of the web changes daily, sometimes in very drastic ways. Much of the research surrounding the PageRank algorithm today focuses on the issue of updating the PageRank vector, so that it may be updated more frequently (and thus a more accurate representation of the current web) at a lower computational cost.

PageRank is a recursive calculation by nature: the PageRank for page *i* depends on the PageRanks of all pages that link to page *i*. In general, a page will receive a high PageRank due to the high PageRank of an inlinking page. However, a sufficient number of links from pages with low PageRank can produce the same effect as one inlink from a page with high PageRank. Intentionally setting up such a scenario would be considered spamming the search engine.

Though the PageRank algorithm is notoriously tough to spam, it is not impervious. Link farms are networks of pages that all link to one another, and link to a separate page. The PageRanks of the network of pages may then sum to the equivalent of an inlink from one page with a high PageRank. An unintentional example of a link farm may be seen in the Appendix B results of the hollins.edu domain. The ten highest ranking results for $\alpha = .85$ are:

Order	PageRank	Page
1	0.01987875	http://www.hollins.edu/
2	0.00928762	http://www.hollins.edu/admissions/visit/visit.htm
3	0.00861039	http://www.hollins.edu/about/about_tour.htm
4	0.00806503	http://www.hollins.edu/htdig/index.html
5	0.00802657	http://www.hollins.edu/admissions/info-request/info-request.cfm
6	0.00716464	http://www.hollins.edu/admissions/apply/apply.htm
7	0.00658278	http://www.hollins.edu/academics/library/resources/web_linx.htm
8	0.00598921	http://www.hollins.edu/admissions/admissions.htm
9	0.00557174	http://www.hollins.edu/academics/academics.htm
10	0.00445247	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld001.htm

Page ten in the above rankings is the introduction page of a slideshow on Greek sculpture. The page's high PageRank is due to the fact that all subsequent slides link to the previous and next slides, as well as the introduction slide. Our hypothetical web surfer randomly clicking links would get "stuck" in the slide show, since there are no escape links to the rest of the web. Introducing $\boldsymbol{E}$, parameterized by $1-\alpha$, moderates the importance of the link structure in the calculation of PageRank. When a high value, say $\alpha = .99$, is used (see Appendix B), the slide show introduction rises to first in PageRank.

Langville and Meyer report that Google uses an alternative construction of the matrix $\boldsymbol{A}$ in order to strengthen PageRank's defense against this form of spamming [4]. The "personalization vector" v is a non-uniform vector used in replacement of e in the construction of the adjustment matrix $\boldsymbol{E}$. In 1998, Brin and Page introduced the idea of personalized PageRank based on a nonuniform distribution for $\boldsymbol{E}$ [7]. This version of the adjustment matrix was intended to be personalized for individuals or groups. For example, a user who surfs the web primarily for news articles may find the results of a query more helpful if the PageRanks are biased towards news sites. The idea of personalized PageRank proved to be very costly in implementation, since multiple PageRank vectors would need to be calculated for the entire web. However, this alternative to a uniform distribution of $\boldsymbol{E}$ has allowed Google to bias PageRank against link farms.

9 PageRank Convergence and the Adaptive PageRank Algorithm

Many researchers are interested in reducing the prohibitive computational cost and time of calculating PageRank, so that it may be computed more frequently and thus more accurately reflect the current web. Kamvar, Haveliwala, and Golub note that different pages in the web converge to their respective PageRanks at differing rates. In particular, those pages with lower PageRanks tend to converge very quickly, while pages with higher PageRanks tend to take more iterations to converge [3].

In January, 2001, Kamvar et al. measured the rates of convergence to final PageRank for the approximately 80 million nodes in the stanford.edu web. They note that most pages converge to final PageRank in 15 or fewer iterations of the power method (see Table 2).

	Number of Pages	***Average PageRank***
$t_i \leq 15$	227597	2.6642e-06
$t_i > 15$	54306	7.2487e-06
Total	281903	3.5473e-06

Table 2: Effect of PageRank on number of iterations to converge in the stanford.edu web

That most pages converge to their final PageRank in few iterations suggests much computational time may be saved by eliminating the calculation of subsequent iterations for those nodes. Using Algorithm 1, we generate z^{k+1} from z^k as $z^{k+1} = \boldsymbol{A} z^k$. At any given step, this algorithm may involve redundant computations for pages that have already converged.

We can begin to eliminate the redundant computations by monitoring for nodes that appear to have converged. Let *C* be the set of *n* - *m* pages that have converged for the iteration z^k, and let *N* be the set of *m* pages that have not converged. We may reorder the current iteration of the PageRank vector as

$$z^k = \begin{bmatrix} z_N^k \\ z_C^k \end{bmatrix}.$$

The matrix $\boldsymbol{A}$ may also be split into two submatrices. Let $\boldsymbol{A_N}$ be the m x n submatrix corresponding to the nodes that have not converged, and $\boldsymbol{A_C}$ the $(n - m)$ x n submatrix corresponding to those nodes that have converged [3]. The next iteration of the power method in this case would be

$$\begin{bmatrix} z_N^{k+1} \\ z_C^{k+1} \end{bmatrix} = \begin{bmatrix} A_N \\ A_C \end{bmatrix} \begin{bmatrix} z_N^k \\ z_C^k \end{bmatrix}.$$

The subvector z_C^k is the vector of nodes that have already converged, so the next iteration of this subvector, z_C^{k+1}, need not be calculated [3]. The next iteration of the algorithm may be simplified to

$$\begin{aligned} z_N^{k+1} &= A_N z^k \\ z_C^{k+1} &= z_C^k \end{aligned}.$$

Thus, the PageRanks for those pages already converged to within the given tolerance are not recomputed. Given a large enough matrix, the time saved by eliminating the recalculation of the pages with converged PageRanks will outweigh the overhead involved with maintaining the sets C and N. Kamvar et al. [3] give the formal algorithm for Adaptive PageRank as:

function AdaptivePageRank($\boldsymbol{A}$, z^0)

repeat

$z_N^{k+1} = \boldsymbol{A_N} z^k$;

$z_C^{k+1} = z_C^k$;

[N, C] = detectConverged(z^k , z^{k+1} , ϵ);

periodically, $\delta = \left\| \boldsymbol{A} z^k - z^k \right\|_1$;

until $\delta < \epsilon$;

return z^{k+1} .

Note that the submatrix $\boldsymbol{A_C}$ is not actually used in the new computation. To reorder the matrix $\boldsymbol{A}$ or completely construct the submatrix $\boldsymbol{A_N}$ would be very costly in terms of

computation time.

Though the matrix $\boldsymbol{A}$ has not become smaller in dimension, the matrix needed for the computation of the next iteration is more sparse than the original matrix $\boldsymbol{A}$. Sparse matrices are matrices with many zero entries, and thus storage requirements and computation time may be reduced by storing only the nonzero entries and their locations. Since each nonzero in the matrix corresponds to a link, and most pages have very few links relative to the enormous size of the web, we expect any matrix generated from a web graph to be extremely sparse. See, for example, the sparsity plot of the hollins.edu web in Appendix F.

10 Filter-Based and Modified Adaptive PageRank

Kamvar et al. also introduce a Filter-Based Adaptive method and a Modified Adaptive method both based on the Adaptive PageRank algorithm [3]. The filter based Adaptive method completely reforms the matrix $\boldsymbol{A}$ as

$$\boldsymbol{A}' = \begin{bmatrix} \boldsymbol{A}_N \\ 0 \end{bmatrix},$$

so that all entries of of the submatrix $\boldsymbol{A}_C$ are changed to zero. This makes $\boldsymbol{A}'$ even more sparse than $\boldsymbol{A}$, which greatly decreases computation time, since non-zero entries slow matrix multiplication. The cost of matrix multiplication is more directly given by the nonzero entries of the matrix, rather than the dimensions of the matrix [3]. However, creating this new matrix $\boldsymbol{A}'$ is very costly (about the same cost as one full power iteration), so it should not be done too often. Kamvar et al. formally define the matrix $\boldsymbol{A}'$ as

$$\boldsymbol{A}_{ij}' = \begin{cases} 0 & \text{if } i \in C, \\ \boldsymbol{A}_{ij} & \text{otherwise.} \end{cases}$$

This constructs $\boldsymbol{A}'$ as a sparse matrix with non-zero entries corresponding only to nonconverged nodes. The PageRanks of the converged nodes are constructed as

$$z'^{k}_{i} = \begin{cases} z_i^k & \text{if } i \in C, \\ 0 & \text{otherwise.} \end{cases}$$

Each iteration of the Filter-Based Adaptive PageRank multiplies the matrix $\boldsymbol{A}'$ by the previous iteration's PageRank vector. Since the converged nodes have been "zeroed out" of $\boldsymbol{A}'$, the PageRanks of those nodes must be added to the PageRanks of the nonconverged nodes for the current iteration. The Filter-Based Adaptive PageRank algorithm [3] is:

function filterAPR($\boldsymbol{A}$, z^0)
repeat
$z_N^{k+1} = \boldsymbol{A}' z^k + z'$;
periodically,
[N, C] = detectConverged(z^k, z^{k+1}, ϵ);
$\boldsymbol{A}'$ =filter($\boldsymbol{A}'$, N, C);
z' =filter(z^k, C);
periodically, $\delta = \left\| \boldsymbol{A} z^k - z^k \right\|_1$;
until $\delta < \epsilon$;
return z^{k+1};

Even though the dimensions of $\boldsymbol{A}'$ are the same as those of $\boldsymbol{A}$, the cost of performing matrix multiplication is lower, since $\boldsymbol{A}'$ is much more sparse. The goal of the Filter-Based Adaptive PageRank algorithm is to periodically increase the sparsity of $\boldsymbol{A}$ to lower the average computation cost of each iteration [3].

The Modified Adaptive PageRank algorithm is more refined than the first two algorithms, but it also requires the most overhead, which can slow computation time. Kamvar et al. further reduce redundant computation of PageRank by eliminating the recomputation of the pages in N due to inlinks from C. The matrix $\boldsymbol{A}$ is then reordered as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{NN} & \boldsymbol{A}_{NC} \\ \boldsymbol{A}_{CN} & \boldsymbol{A}_{CC} \end{bmatrix},$$

where $\boldsymbol{A}_{NN}$ are the entries corresponding to links within pages that have not converged to the desired tolerance. Similarly, $\boldsymbol{A}_{NC}$ are the entries corresponding to links from pages that have converged to pages that have not converged, and so on [3]. Since the PageRank of a given page is the sum of the PageRanks of *inlinking* pages, we need only perform calculations on nonconverged pages that have inlinks. Thus, we need only calculate PageRank for the matrices $\boldsymbol{A}_{NN}$ and $\boldsymbol{A}_{NC}$. The algorithm for the Modified Adaptive PageRank [3] is given as

function modifiedAPR($\boldsymbol{A}$, z^0)
repeat
$z_N^{k+1} = \boldsymbol{A}_{NN} z_N^k + y$;
$z_C^{k+1} = z_C^k$;
periodically,
[N, C] = detectConverged(z^k, z^{k+1}, ϵ);
$y = \boldsymbol{A}_{NC} z_C^k$;
periodically, $\delta = \left\| \boldsymbol{A} z^k - z^k \right\|_1$;
until $\delta < \epsilon$;
return z^{k+1} ;

It is not necessary to explicitly reorder the entire matrix to form the submatrices $\boldsymbol{A}_{NN}$, $\boldsymbol{A}_{CN}$, $\boldsymbol{A}_{NC}$, and $\boldsymbol{A}_{CC}$. Kamvar et al. lastly introduce a Filter-Based Modified Adaptive PageRank algorithm that combines the two methods.

11 Implementation Issues

The implementation of Adaptive PageRank using a high-level language such as *Matlab* is somewhat limited by the performance of included functions. An optimal implementation of both the PageRank and Adaptive PageRank algorithms would require a low-level language such as C++, which as a compiled language is more efficient than code written for *Matlab*. Code written and run using *Matlab* must be interpreted at runtime, meaning the computer must read each line, and execute the code accordingly, with little or no chance to optimize for efficiency. Compiled code is optimized and converted by a compiler into code for interpretation. The efficiency of code written for *Matlab* is reliant upon the efficiency of the internal functions included with the *Matlab* installation. For example, to compute the one-norm of a vector, the *Matlab* sum(abs(*x*)) command is mysteriously many times more efficient than the mathematically equivalent norm(*x*,1) function. Calculations involving submatrices in the adaptive algorithm require special

consideration. Column based submatrices are more efficiently extracted than square or row based submatrices due to sparse matrix storage design issues.

For Brin and Page's PageRank algorithm, the adjustment matrix $\boldsymbol{E}$ is added to the stochastic matrix $\bar{\boldsymbol{P}}$. If implemented in this way, we must then perform matrix multiplication on a completely full matrix. Since all entries of $\boldsymbol{E}$ are assumed to be uniform, it is highly inefficient to destroy the sparsity of $\boldsymbol{P}$ in this manner. Thus, matrix multiplications are performed on the sparse matrix $\boldsymbol{P}$, but multiplication by $\boldsymbol{E}$ may be done implicitly. Similarly, zero rows in $\boldsymbol{P}$ may be adjusted implicitly in the matrix-vector multiplication routine, again with the aim of preserving sparsity.

Kamvar et al. note that even using a filtering-based algorithm that does not completely reorder $\boldsymbol{A}$, the overhead associated with filtering out nonconverged nodes in $\boldsymbol{A}$ can outweigh the time saved by performing calculations only on those nonconverged nodes. Also, they note that some nodes seem to converge immediately, then change as other related nodes converge to their final PageRank. Thus, they introduce phases of "pruning." Using this scheme, all nodes are included in the initial computations, and converged nodes are filtered out until all nodes converge to the desired tolerance. The tolerance level is lowered, and the adaptive method is run again on the entire matrix, pruning out converged nodes at the current tolerance level. This technique is utilized in the code we used to calculate the Adaptive PageRanks.

12 Adaptive PageRank Example

The adaptive PageRank algorithm produces the same PageRank vector results as the PageRank example of Section 7. The number of iterations required and the interim PageRank vectors for each iteration are different for the adaptive method.

The power method and the adaptive method were both run on the hollins.edu web.

Appendix G is a graphical comparison of the convergence pattern of both algorithms. Initially, the algorithms have a similar pattern, but the adaptive method takes more iterations to converge to a tolerance of 10e-3. This is because the adaptive method requires all nodes to converge to 10e-3 before lowering the tolerance threshold and performing the algorithm on the entire matrix. Each successive pruning stage requires a few more iterations, and the end result is that the adaptive method requires more iterations to converge to the final tolerance of 10e-8. However, the general slope of the convergence curve is not appreciably altered.

As discussed previously, the adaptive method introduces an extra layer of overhead. For example, the sets *C* and *N* must be created for each phase and adjusted as some nodes converge. Due to this extra overhead the adaptive method is much less efficient than the power method for small webs. Using datasets contributed by Langville, Kamvar, and Massey, we can compare the speed results of the PageRank versus the Adaptive PageRank algorithms for various sized webs (see Appendix E).

The adaptive algorithm only begins to show improvement over the PageRank algorithm for the two largest webs in the dataset. The experimental results show at best an 11% improvement in computation time for a web of 685,230 pages. Kamvar et al. report a 20% improvement for a web of approximately 80 million nodes using the Modified Adaptive PageRank method. For small webs, the adaptive approach is generally slower than the original PageRank algorithm, though experiments indicate that for larger webs, some form of the adaptive method would result in a significant reduction in computation time.

For every dataset in the experiment, the adaptive method required more iterations than the PageRank algorithm. Kamvar et al. report similar results [3]. In general, the adaptive method seems to require more iterations, but the average cost in computation time for each iteration is lower for large webs. This is good news for an implementation of the Adaptive PageRank method on the entire web, since the current web contains

billions of pages. Suppose the average computation time for PageRank using the original power method is three days. A 20% improvement would save more than 14 hours of computation time. In addition, experiments indicate that the time saved with the adaptive method may increase significantly for a web of several billion pages, compared to a web of only 80 million.

13 Conclusion

The core of the Google search engine is the PageRank algorithm, which defines page importance hierarchies for all pages in the web. PageRank is used in conjunction with many forms of advanced IR tools, and provides a fast, reasonable response to query terms. The nature of the PageRank algorithm and the philosophy behind Google make the search engine resistant to spamming.

The inherent link structure of the web lends itself to the construction of a Markov model of the web and web users. By using known algorithms to calculate the eigenvector of the transition probability matrix when $\lambda = 1$, we can find the stationary vector of this Markov Chain. This stationary vector is the vector of PageRanks for all pages in the web.

Much research is devoted to many aspects of the Google search engine. Topics include PageRank updating techniques, anti-spamming, personalization, and accelerating convergence. For large scale webs, some form of the adaptive PageRank method shows a definite improvement in computation time, though it generally requires more iterations. The results show that for a web as large as the current web, the adaptive methods will produce a lower average cost of computation per iteration, despite the extra overhead.

14 Appendix A: *Matlab* code for PageRank experiments

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% [U,P] = loaddat(fname,ftype)
%% loads .dat file formatted like
%%    ftype = 1 : hollins.dat, mathworks.dat
%%          = 2 : stanford.dat, stanberk.dat
%%          = 3 : .dat files from Langville (linktiny10, linkjordan, etc)
%% returns sparse transition matrix with P(i,j) = prob(i ==> j)
%% U contains the urls when ftype = 1
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [U,P] = loaddat(fname,ftype)

tic;
f = fopen(fname,'r');
U = cell(0,0);

if (ftype == 1)
  n = fscanf(f,'%d',1);
  nlink = fscanf(f,'%d',1);

  U = cell(n,1);
  ij = zeros(nlink,2);

  for i=1:n
    a = fscanf(f,'%d',1);
    s = fscanf(f,'%s',1);
    U{i} = s;
  end

  for i=1:nlink
    ij(i,1) = fscanf(f,'%d',1);
    ij(i,2) = fscanf(f,'%d',1);
  end

  P = (sparse(ij(:,1),ij(:,2),1,n,n) ~= 0);   %% 0-1 matrix
elseif (ftype == 2)
  P = spconvert(load(fname));
  n = max(size(P)); P(n,n) = 0;      % make the matrix square
elseif (ftype == 3)
  j = fscanf(f,'%s',7);
  n = fscanf(f,'%i',1);
  j = fscanf(f,'%s',5);
  P = sparse(n,n);

  for i=1:n
    j = fscanf(f,'%s',7);
    j = fscanf(f,'%i',1);
    if j >= 1
      for m=1:j
        k = fscanf(f,'%i',1);
        P(i,k+1) = 1/j;            % adjusts indexing to start at 1 instead of 0
      end
    end
  end
end

%% normalize

P = P';        % much faster if working with columns

for i=1:n
  k = sum(P(:,i));
  if (k ~= 0)
    P(:,i) = P(:,i) / k;
  end
end

%% record indices of zero rows (not returned)
zr = find(sum(P,1) == 0);

P = P';        % flip back so row sum is one

fprintf(1,'matrix size = %d, nnz = %d, zero cols = %d\n',n,nnz(P),length(zr));
fprintf(1,'  took %f seconds\n',toc);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% function [x,chistory] = powermethod(P,alpha,tol,mu)
%%
%% efficient power method to compute stationary vector
%% the operator is:
%%    A = alpha * P' + (1-alpha)ve'
%% (personalization vector v=e/n is hard coded)
%% implicitly applies ve' and zero row corrections
%% applies shifts in mu
%% convergence history is the 1-norm difference of successive iterates
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,chistory] = powermethod(P,alpha,tol,mu)

tic;
n = size(P,1);
x = ones(n,1) / n;
nmu = length(mu);

iteration = 0;
while (1)
  iteration = iteration + 1;
  x0 = x;
  x = alpha * (x'*P)';
  x = x + (1-sum(x))/n;

  if (iteration <= nmu)
    x = x - mu(iteration)*x0;
    x = x/sum(x);
  end

  % tracking convergence is noticably time consuming
  % one norm suggested by golub, norm(,1) is inefficient
  change = sum(abs(x-x0));
  chistory(iteration) = change;
  if (change < tol)
    break;
  end
end

fprintf(1,'power method took %d iterations to converge to a tolerance of %e\n',iteration,tol);
fprintf(1,'  elapsed time = %f\n',toc);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% function [x,chistory] = adaptive(P,alpha,tol,mu,ipp,ppr,tl)
%%
%% best matlab implementation of Kamvar's adaptive method
%%   (expect better results with specialized compiled code)
%% ipp iterations per phase, and ppr phases per restart
%%   prunes converged links after every phase
%%   lowers threshold every restart (up to tl threshold levels)
%%   (ppr=1 then ordinary power method)
%% the operator is:
%%     A = alpha * P' + (1-alpha)ve'
%% (personalization vector v=e/n is hard coded)
%% implicitly applies ve' and zero row corrections
%% applies shifts in mu
%% convergence history is 1-norm difference of successive full-matrix iterates
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,chistory] = adaptive(P,alpha,tol,mu,ipp,ppr,tl)

tic;
n = size(P,1);
x = ones(n,1) / n;
nmu = length(mu);

iteration = 0;
avgk = 0;
lt = -2;                                % log10(threshold)
dt = (log10(tol)-lt) / tl;              % change in threshold each restart

while (1)
  if (mod(iteration,ipp*ppr) == 0)      % restart
    lt = lt + dt;
    threshold = max(tol,10^lt);
    k = n;                              % number of non-converged
  elseif (mod(iteration,ipp) == 0)      % new phase
    relch = abs((x-x0) ./ x0);          % relative change
    C = find(relch < threshold);
    N = find(relch >= threshold);
    k = length(N);
    Pn = P(:,N);                        % extracting cols is most efficient
    sxn = sum(x(N));
  end

  if (mod(iteration,ipp) == 0)
    fprintf(1,'%12d %e\n',k,threshold);
  end

  iteration = iteration + 1;
  avgk = avgk + k;

  if (k==n)                             % do regular power iteration
    x0 = x;
    x = alpha * (x'*P)';
    x = x + (1-sum(x))/n;

    if (iteration <= nmu)
      x = x - mu(iteration)*x0;
      x = x/sum(x);
    end

    change = sum(abs(x-x0));            % one norm suggested by golub
    chistory(iteration) = change;
    if (change < tol)
      break;
    end
  else
    if (mod(iteration,ipp)==0)          % store for pruning
      x0 = x;
    end
    xn = alpha * (x'*Pn)';
    % without knowing exactly which columns were zero to apply ve' implicitly,
    % we approximate that sum(x) should remain constant
    x(N) = xn + (sxn-sum(xn))/k;
  end
end

avgk = avgk / iteration;
```

```
fprintf(1,'adaptive method took %d iterations to converge to a tolerance of %e\n',iteration,tol);
fprintf(1,'  average size of non-converged set is %.0f out of %.0f\n',avgk,n);
fprintf(1,'  elapsed time = %f\n',toc);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% x = testpr(U,P,alpha,tol,nmu,ipp,ppr,tl)
%%
%% example (tested on stanberk.dat)
%%   testpr(U,P,.85,1e-8,0,8,3,4)  ==> 92 (17.5), 99 (16)
%%   testpr(U,P,.85,1e-8,4,8,3,4)  ==> 72 (13.8), 103 (16.2)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function x = testpr(U,P,alpha,tol,nmu,ipp,ppr,tl)

% do some heuristic shifts to speed convergence
% pick some shifts in [-alpha,alpha], skewed toward the left (away from 1)
mu = zeros(0);
if (nmu > 1)
  mu = alpha*((2.^[0:nmu-1] - 1)*2/(2^(nmu-1)-1) - 1)
end


[x,ch] = powermethod(P,alpha,tol,mu);
[x1,ch1] = adaptive(P,alpha,tol,mu,ipp,ppr,tl);

semilogy(ch); hold on; semilogy(ch1); hold off;

disp(norm(x-x1,'inf'));
[x,ii] = sort(-x); x = -x;    % descending order
if (length(U) > 0)
  U = U(ii);
end
for i=1:10
  if (length(U) > 0)
    fprintf(1,'%10.8f %s\n',x(i),char(U(i)));
  else
    fprintf(1,'%10.8f\n',x(i));
  end
end
```

Appendix B: PageRank Results

The Hollins link data was generated from a web crawl of the www.hollins.edu and www1.hollins.edu domains on Jan 15, 2004.

$\alpha = .85$

Order	PageRank	Page
1	0.01987875	http://www.hollins.edu/
2	0.00928762	http://www.hollins.edu/admissions/visit/visit.htm
3	0.00861039	http://www.hollins.edu/about/about_tour.htm
4	0.00806503	http://www.hollins.edu/htdig/index.html
5	0.00802657	http://www.hollins.edu/admissions/info-request/info-request.cfm
6	0.00716464	http://www.hollins.edu/admissions/apply/apply.htm
7	0.00658278	http://www.hollins.edu/academics/library/resources/web_linx.htm
8	0.00598921	http://www.hollins.edu/admissions/admissions.htm
9	0.00557174	http://www.hollins.edu/academics/academics.htm
10	0.00445247	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld001.htm
11	0.00438508	http://www.hollins.edu/grad/coedgrad.htm
12	0.00377793	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/tsld001.htm
13	0.00374159	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/index.htm
14	0.00352556	http://www.hollins.edu/alumnae/alumnae.htm
15	0.00339769	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/sld001.htm
16	0.00333746	http://www1.hollins.edu/classes/dance/website/home.htm
17	0.00333375	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld053.htm
18	0.00331817	http://www.hollins.edu/calendar
19	0.00322873	http://www.hollins.edu/calendar/index.html
20	0.00309215	http://www.hollins.edu/admissions/financial/finaid.htm
21	0.00291088	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/sld001.htm
22	0.00287267	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/tsld001.htm
23	0.00284728	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/index.htm
24	0.00280248	http://www.hollins.edu/calendar/null.htm
25	0.00267832	http://www.hollins.edu/cgi-bin/forumdisplay.cgi

---

$\alpha = .95$

Order	PageRank	Page
1	0.01815080	http://www.hollins.edu/
2	0.01039056	http://www.hollins.edu/admissions/visit/visit.htm
3	0.00959949	http://www.hollins.edu/about/about_tour.htm
4	0.00919491	http://www.hollins.edu/htdig/index.html
5	0.00898837	http://www.hollins.edu/admissions/info-request/info-request.cfm
6	0.00792486	http://www.hollins.edu/admissions/apply/apply.htm
7	0.00774263	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld001.htm
8	0.00702116	http://www.hollins.edu/admissions/admissions.htm
9	0.00659465	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/index.htm
10	0.00605345	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/tsld001.htm
11	0.00602382	http://www.hollins.edu/academics/academics.htm
12	0.00584354	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld053.htm
13	0.00583822	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/sld001.htm
14	0.00545755	http://www.hollins.edu/academics/library/resources/web_linx.htm
15	0.00522115	http://www.hollins.edu/grad/coedgrad.htm
16	0.00495969	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/sld001.htm
17	0.00495163	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/index.htm
18	0.00455365	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/tsld001.htm
19	0.00452059	http://www1.hollins.edu/faculty/saloweyca/clas%20395/DAGEO/sld001.htm
20	0.00441467	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/sld040.htm
21	0.00419394	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/index.htm
22	0.00408164	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Acropolis/sld001.htm
23	0.00394010	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/tsld002.htm
24	0.00389185	http://www.hollins.edu/admissions/financial/finaid.htm
25	0.00386187	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/tsld001.htm

$\alpha = .99$

Order	PageRank	Page
1	0.01304090	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld001.htm
2	0.01120217	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/index.htm
3	0.00991319	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld053.htm
4	0.00982378	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/tsld001.htm
5	0.00960742	http://www.hollins.edu/
6	0.00941925	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/sld001.htm
7	0.00805206	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/index.htm
8	0.00777246	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/sld001.htm
9	0.00716868	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/sld040.htm
10	0.00708249	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/tsld001.htm
11	0.00695790	http://www1.hollins.edu/faculty/saloweyca/clas%20395/DAGEO/sld001.htm
12	0.00678252	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/tsld002.htm
13	0.00662154	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/index.htm
14	0.00615101	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Acropolis/sld001.htm
15	0.00605808	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld002.htm
16	0.00604710	http://www.hollins.edu/admissions/visit/visit.htm
17	0.00592041	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/sld034.htm
18	0.00591465	http://www1.hollins.edu/faculty/saloweyca/clas%20395/DAGEO/index.htm
19	0.00583656	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/tsld001.htm
20	0.00558166	http://www.hollins.edu/about/about_tour.htm
21	0.00539421	http://www.hollins.edu/htdig/index.html
22	0.00535363	http://www1.hollins.edu/faculty/saloweyca/clas%20395/painting/sld001.htm
23	0.00530287	http://www1.hollins.edu/faculty/saloweyca/clas%20395/DAGEO/sld031.htm
24	0.00523053	http://www.hollins.edu/admissions/info-request/info-request.cfm
25	0.00522050	http://www1.hollins.edu/faculty/saloweyca/clas%20395/DAGEO/tsld001.htm

---

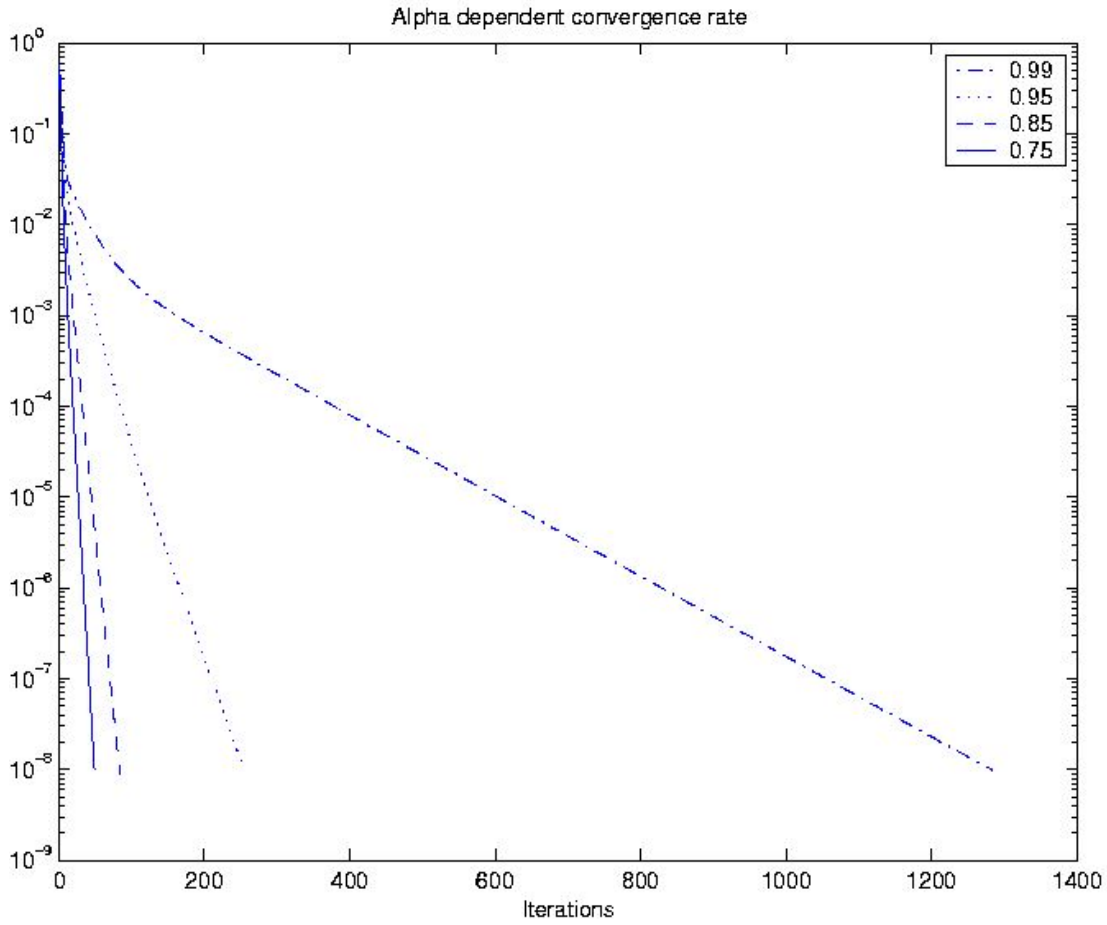
$\alpha = .75$

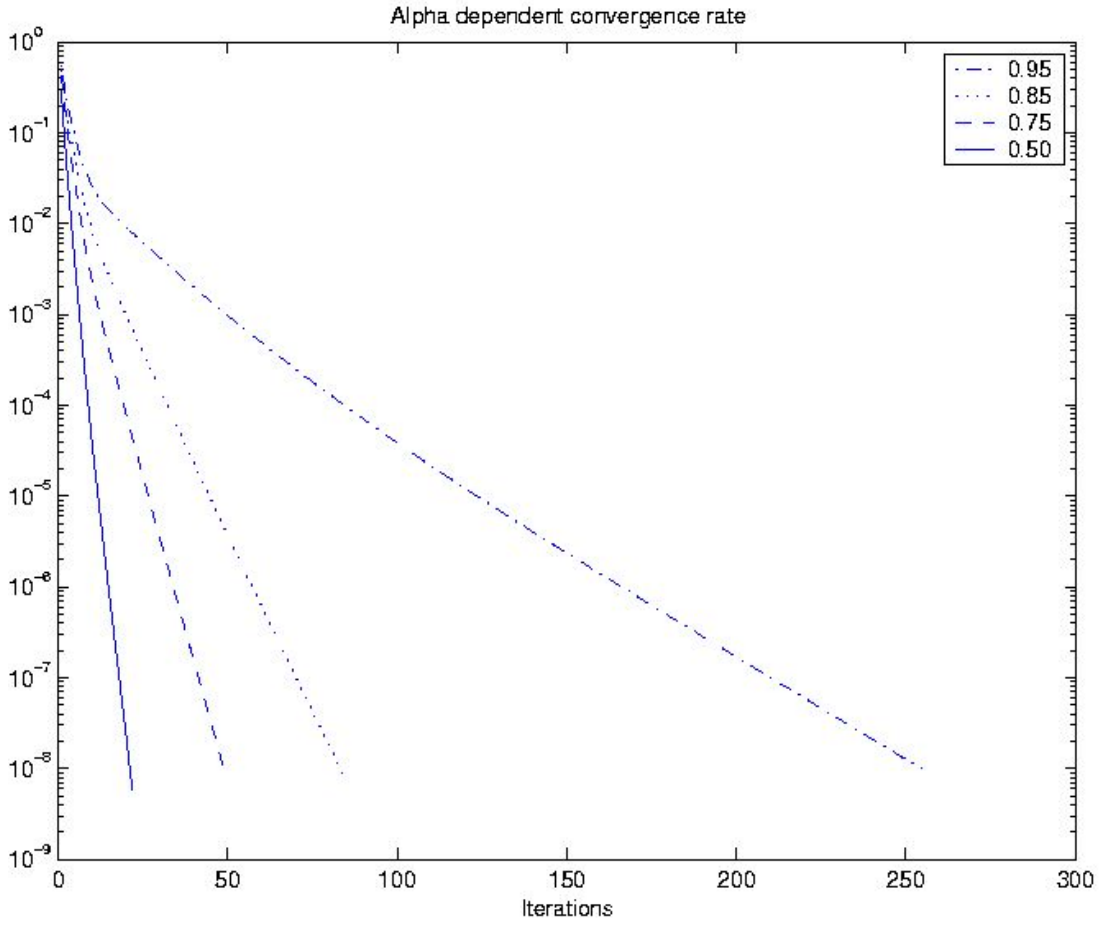
Order	PageRank	Page
1	0.01831690	http://www.hollins.edu/
2	0.00722917	http://www.hollins.edu/admissions/visit/visit.htm
3	0.00673192	http://www.hollins.edu/about/about_tour.htm
4	0.00624490	http://www.hollins.edu/admissions/info-request/info-request.cfm
5	0.00621035	http://www.hollins.edu/academics/library/resources/web_linx.htm
6	0.00617604	http://www.hollins.edu/htdig/index.html
7	0.00565717	http://www.hollins.edu/admissions/apply/apply.htm
8	0.00444695	http://www.hollins.edu/academics/academics.htm
9	0.00443701	http://www.hollins.edu/admissions/admissions.htm
10	0.00321908	http://www.hollins.edu/grad/coedgrad.htm
11	0.00312996	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld001.htm
12	0.00290543	http://www1.hollins.edu/classes/dance/website/home.htm
13	0.00284957	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/tsld001.htm
14	0.00276598	http://www.hollins.edu/alumnae/alumnae.htm
15	0.00263547	http://www.hollins.edu/calendar/index.html
16	0.00262602	http://www.hollins.edu/cgi-bin/forumdisplay.cgi
17	0.00262042	http://www.hollins.edu/calendar
18	0.00262039	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/index.htm
19	0.00240315	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/sld001.htm
20	0.00235403	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld053.htm
21	0.00221787	http://www.hollins.edu/cgi-bin/Ultimate.cgi?action=intro&BypassCookie=true
22	0.00217753	http://www1.hollins.edu/faculty/saloweyca/clas%20395/BronzeAGe/tsld001.htm
23	0.00215283	http://www.hollins.edu/admissions/financial/finaid.htm
24	0.00213418	http://www.hollins.edu/undergrad/undergraduate.htm
25	0.00206775	http://www1.hollins.edu/faculty/saloweyca/clas%20395/kouroikorai/sld001.htm

$\alpha = .5$

Order	PageRank	Page
1	0.01279958	http://www.hollins.edu/
2	0.00436698	http://www.hollins.edu/academics/library/resources/web_linx.htm
3	0.00365657	http://www.hollins.edu/admissions/visit/visit.htm
4	0.00345085	http://www.hollins.edu/about/about_tour.htm
5	0.00318433	http://www.hollins.edu/admissions/info-request/info-request.cfm
6	0.00305074	http://www.hollins.edu/htdig/index.html
7	0.00297057	http://www.hollins.edu/admissions/apply/apply.htm
8	0.00234763	http://www.hollins.edu/academics/academics.htm
9	0.00198186	http://www.hollins.edu/admissions/admissions.htm
10	0.00194693	http://www.hollins.edu/cgi-bin/forumdisplay.cgi
11	0.00187523	http://www1.hollins.edu/classes/dance/website/home.htm
12	0.00166158	http://www1.hollins.edu/Docs/Academics/international_programs/forms.htm
13	0.00156837	http://www.hollins.edu/cgi-bin/Ultimate.cgi?action=intro&BypassCookie=true
14	0.00156778	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/tsld001.htm
15	0.00153113	http://www.hollins.edu/calendar/index.html
16	0.00148812	http://www1.hollins.edu/faculty/saloweyca/clas%20395/Sculpture/sld001.htm
17	0.00143111	http://www1.hollins.edu/docs/alumdev/Default.htm
18	0.00142132	http://www.hollins.edu/grad/coedgrad.htm
19	0.00140839	http://www.hollins.edu/calendar
20	0.00139081	http://www1.hollins.edu/faculty/clarkjm/Math152/math152.htm
21	0.00136820	http://www1.hollins.edu/Docs/alumdev/Default.htm
22	0.00136586	http://www1.hollins.edu/faculty/clarkjm/Math152/mth152schedulespring03.htm
23	0.00135915	http://www.hollins.edu/alumnae/alumnae.htm
24	0.00135536	http://www1.hollins.edu/Docs/AlumDev/Default.htm
25	0.00130541	http://www1.hollins.edu/classes/anth250/anth_250.htm

Appendix C: Graphs Comparing Number of Iterations for Varying α Values




Alpha dependent convergence rate
0.95
0.85
0.75
0.50
Iterations

Appendix D: Table of Iterates for PageRank Computation of 6-node Example Web

Convergence tolerance 10e-8
$\alpha = .85$

Iteration	**PageRank**					
1	[.1666667	.1666667	.1666667	.1666667	.1666667	.1666667]
2	[.1194444	.1194444	.1194444	.4027778	.1902778	.0486111]
3	[.1027199	.1027199	.1027199	.2455671	.3943171	.0519560]
4	[.1245176	.1245176	.1245176	.2559921	.2895937	.0808616]
5	[.1189457	.1189457	.1189457	.293518	.2836190	.0660258]
6	[.1157313	.1157313	.1157313	.2729571	.3146697	.0651794]
7	[.1187640	.1187640	.1187640	.2725381	.3015917	.0695782]
8	[.1182002	.1182002	.1182002	.2782911	.2993829	.0677255]
9	[.1176477	.1176477	.1176477	.2756845	.3039600	.0674126]
10	[.1180613	.1180613	.1180613	.2753624	.3023928	.0680610]
11	[.1180150	.1180150	.1180150	.2762189	.3018971	.0678390]
12	[.1179251	.1179251	.1179251	.2759010	.3025548	.0677687]
14	[.1179784	.1179784	.1179784	.2759441	.3022838	.0678369]
13	[.1179801	.1179801	.1179801	.2758199	.3023778	.0678619]
15	[.1179644	.1179644	.1179644	.2759073	.3023761	.0678235]
16	[.1179715	.1179715	.1179715	.2758912	.3023578	.0678366]
17	[.1179719	.1179719	.1179719	.2759088	.3023415	.0678340]
18	[.1179698	.1179698	.1179698	.2759048	.3023542	.0678317]
19	[.1179707	.1179707	.1179707	.2759019	.3023526	.0678335]
20	[.1179708	.1179708	.1179708	.2759044	.3023499	.0678333]
21	[.1179705	.1179705	.1179705	.2759040	.3023516	.0678329]
22	[.1179706	.1179706	.1179706	.2759035	.3023515	.0678331]
23	[.1179706	.1179706	.1179706	.2759038	.3023511	.0678331]
24	[.1179706	.1179706	.1179706	.2759038	.3023513	.0678331]

Appendix E: Comparison of Computation Time and Number of Iterations for PageRank and Adaptive PageRank for Webs of Varying Sizes

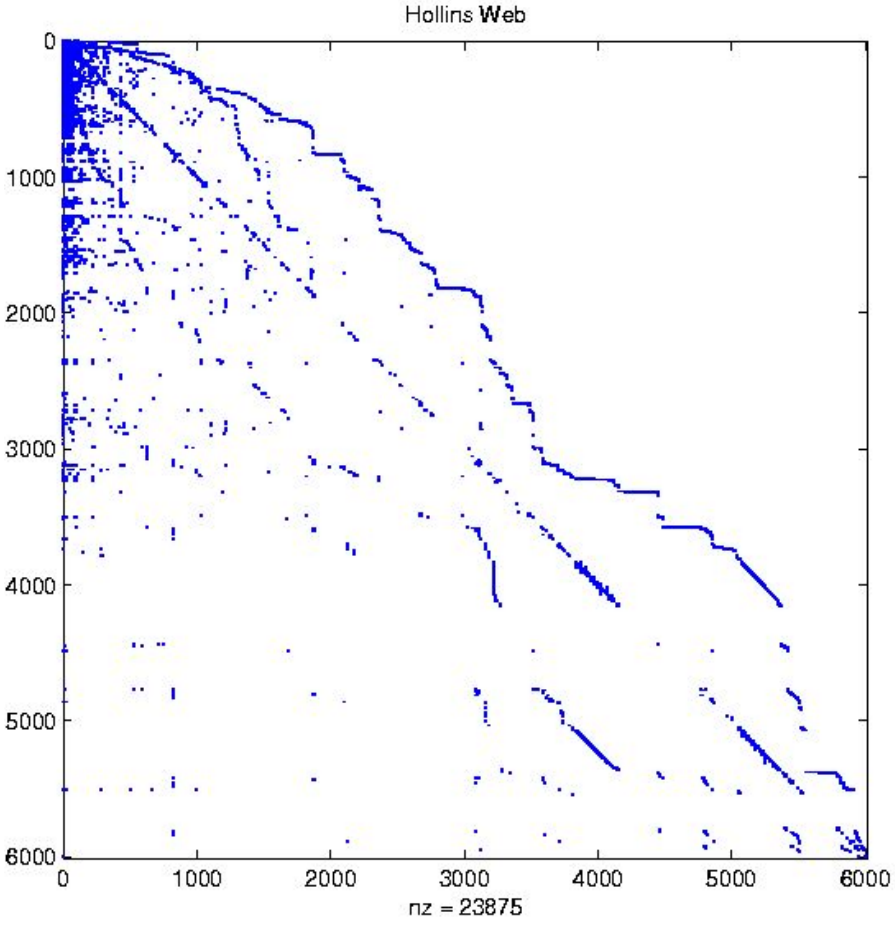
For both methods:
$\alpha = .85$
final tolerance = 10e-8

For the Adaptive PageRank method:
8 iterations per phase
3 phases per restart
4 threshold levels

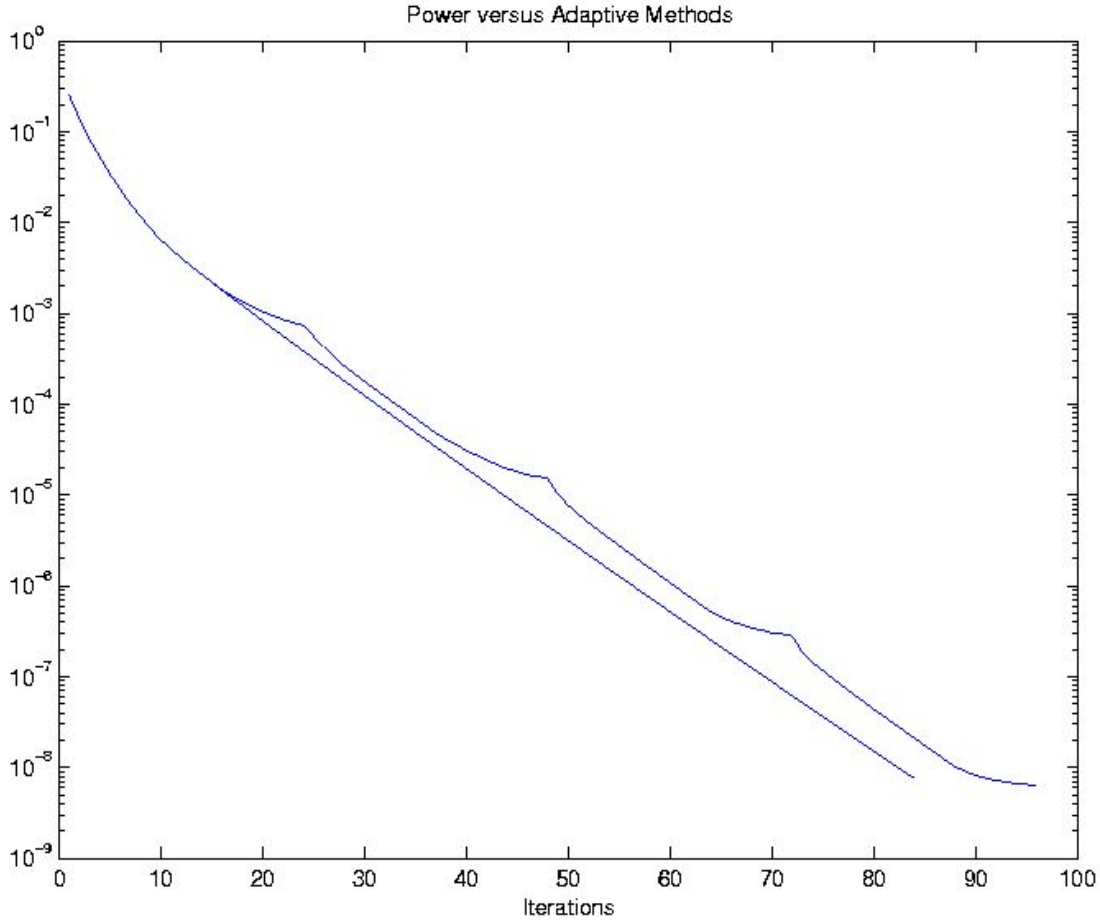
		PageRank		*Adaptive PageRank*		*Adaptive PageRank to PageRank time ratio*
dataset	*n*	*iterations*	*time*	*iterations*	*time*	
linktiny10	10	31	0.01	55	0.02	2
linktiny20	20	99	0.04	100	0.14	3.5
linkmovies	451	16	0.02	31	0.06	3
linkjordan	1885	79	0.07	99	0.21	3
linkjava	2810	84	0.08	99	0.25	3.13
mathworks	517	42	0.05	80	0.22	4.4
hollins	6012	84	0.11	97	0.32	2.91
stanford	281903	91	11.64	99	10.67	0.92
stanberk	685230	92	17.25	99	15.36	0.89

Appendix F: Matrix Graph of hollins.edu Web Structure

Each dot represents a nonzero in the adjacency matrix.



Appendix G: Graph Comparing Convergence Plots of Power and Adaptive Methods on the hollins.edu Web with $\alpha = .85$



15 References

[1]Arvid Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan. Searching the Web. *ACM Transactions on Internet Technology*, 2001.

[2]Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 1998.

[3]Sepandar Kamvar, Taher Haveliwala, and Gene Golub. Adaptive Methods for the Computation of PageRank. Technical Report, Computer Science Department, Stanford University, 2003.

[4]Amy N. Langville and Carl D. Meyer. Deeper Inside PageRank. *Internet Mathematics*, February 2004.

[5]Amy N. Langville and Carl D. Meyer. A Survey of Eigenvector Methods of Web Information Retrieval. *The SIAM Review,* December 2003.

[6]David C. Lay. Linear Algebra and Its Applications. Addison-Wesley, 2000.

[7]Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report, Computer Science Department, Stanford University, 1998.

[8]Prentice Hall. http://www.prenhall.com/divisions/esm/app/ph-linear/leon/html/perron.html, 1998.