

PAGERANK COMPUTATION AND THE STRUCTURE OF THE WEB: EXPERIMENTS AND ALGORITHMS

Arvind Arasu

Computer Science Department, Stanford University, CA 94305

Jasmine Novak

Computer Science Department, University of California at Berkeley

Andrew Tomkins & John Tomlin

IBM Almaden Research Center, San Jose, CA 95120

November 13, 2001

Abstract

In this paper we describe some computational experiments carried out with variants of the “PageRank” model due to Page et al. In particular we study the effect of exploiting the numerous small-scale substructures which appear repeatedly in the enormous web graph, and the use of an equation-solving approach rather than the eigensystem calculation presently favored. We also examine these algorithmic variations in the light of the large-scale (“bow tie”) structure of the web, and their implications for faster solution using partitioning and/or parallel procedures.

keywords: World Wide Web, PageRank, Linear Algebra, Sparse Matrices, Bow Tie Theory

Approximate Word Count: 7,500

1 Introduction

This paper describes some algorithmic approaches to “PageRank” computation and gives experimental results for some of these approaches. We do this in the context of seeking efficient methods which exploit both the small

scale and large scale structure of the web and which will scale well as the web expands even beyond its present enormous size.

The concept of PageRank, due to Page[19] and Page et al.[20] has proved extremely successful in practice, and it is not our intention to discuss the merits of the concept, discussed extensively in these and other references (e.g. Chakrabarti et al.[5] and Henzinger[12]). Our main concern is with the computational results of a series of experiments with variants of PageRank computation, carried out on a subset of the WWW with about 18 million nodes.

We begin by studying the convergence rate of the traditional iterative approach to PageRank computation, for various parameters. The convergence is slow enough to cast doubt on this well-established technique—after performing a fairly significant number of iterations of this computationally intensive operation (50), the average error per page is roughly 50% of the value of that page. In response, we study the reason for the slow convergence, which yields a characterization of the repetitive small-scale structure in the web graph. We show that web sites have connectivity properties with other web sites that allow fast convergence of PageRank-style algorithms, and that the network of linkages between sites is in some sense optimized for effective browsing. On the other hand, the structure within individual web sites is limiting from the perspective of PageRank, as well as from the perspective of an individual attempting to browse and discover the pages on the site.

Having understood the reason for slow convergence of the naive approaches used thus far, we evaluate more sophisticated approaches. We report results of experiments which compare an equation-solving procedure with the more conventional eigenvector calculation on this sample, and indicate the possibility of more efficient solution.

We then consider these results in the light of recent work on the large scale structure of the web - the “bow tie” structure described by Broder et al.[4]. We see that this structure is eminently exploitable to take advantage of the computational variants we have considered. We develop a partitioning approach to reduce the problem size, and show that this approach can be expected to result in significant savings.

The remainder of the paper is structured as follows. In Section 2, we give an overview of the necessary background, including PageRank, eigenvector computations, and graph theory. In Section 3 we study power iteration, and present our findings on the graph theoretical reasons for slow convergence of this method. In Section 4 we approach the problem as a system of equations rather than as an eigenvector computation and show faster convergence

rates. Next, in Section 5 we give a decomposition of the problem prompted by the bow tie structure mentioned above, and show that the decomposition can result in significant savings. Finally, in Section 6, we conclude.

2 Mathematical Formulation

In this section, we give enough of an overview of graph theory, eigenvector computations, and PageRank to make the paper self-contained.

2.1 Definitions from Graph Theory

We treat the HTML “points-to” relation on web pages as a directed graph, and represent it by two sets: $G = (V, E)$. The set V of vertices consists of the n pages, and the set E of edges consists of the directed hyperlinks between pages—the directed edge (i, j) exists if and only if page i has a hyperlink to page j . For convenience, we define d_i as the *out-degree* of page i ; that is, the number of hyperlinks on page i . We also define a *strongly connected component* of G to be a subset $V' \subset V$ of the vertices such that for all pairs of pages $i, j \in V'$, there exists a directed path from i to j in (V', E) . Intuitively, a strongly connected component is a set of pages each of which can reach all the rest, by following hyperlinks, without visiting pages outside the set. Finally, we will refer to a largest strongly connected component of a graph as *SCC*.

2.2 Definitions from Linear Algebra

For any $n \times n$ matrix M , the scalars λ and vectors x satisfying $Mx = \lambda x$ are called the (right) *eigenvalues* and *eigenvectors* respectively of M . There may be up to n distinct eigenvalues (and corresponding eigenvectors). Left eigenvalues μ and eigenvectors v may analogously be defined, satisfying¹ $\mu v^T = v^T M$. A *principal eigenvalue* of a matrix is an eigenvalue whose magnitude is maximal over all eigenvalues; a *principal eigenvector* is an eigenvector corresponding to a principal eigenvalue.

Now, recall that a Markov chain is simply a set of states and a set of transition probabilities—from any state, there is a probability distribution describing which state the chain will visit next. See [9] for more details about Markov chains.

An $n \times n$ matrix M is (row) *stochastic* if all entries are non-negative and the entries of each row sum to 1. Such a matrix can be thought of as an

¹We write v^T instead of v to imply that we consider a row vector, for convenience.

n -state Markov chain in which m_{ij} is the probability of transitioning to state j when the chain is in state i . The condition that all entries be non-negative is simply the requirement that probabilities be non-negative. The condition that each row sum to 1 is simply a rephrasing of the requirement that from any particular state of the Markov chain, the next state must be described as a probability distribution. Let the current state of the Markov chain be given by some vector v^T with non-negative entries summing to 1. The i^{th} entry of this vector is therefore the probability that the Markov chain is currently in state i . Then the next state of the chain can be computed from the current state v^T as $v^T M$.

If the Markov chain is in a *steady state* then taking another step does not change the state; that is, $v^T = v^T M$. We see that the vector v^T is a (left) eigenvector of M corresponding to the eigenvalue 1. Stochastic matrices always have a maximal (principal) eigenvalue 1; therefore, the steady state of a Markov chain corresponding to a stochastic matrix M is simply the principal (left) eigenvector of M .

2.3 PageRank and Static Ranking

PageRank is a static ranking of web pages initially presented in [19], and used as the core of the Google search engine (<http://www.google.com>). It is the most visible link-based analysis scheme, and its success has caused virtually every popular search engine to incorporate link-based components into their ranking functions.

We begin with a definition of *static ranking* schemes. The simplest implementation of a static ranking scheme for web search is the following. All pages to be indexed are ordered from best to worst—this ordering is the “static ranking” itself. When a query arrives, and some fixed number of pages (say, 10) must be returned, the index returns the “best” 10 pages that satisfy the query, where best is determined by the static ranking. This simple scheme allows highly efficient index structures to be built. The scheme can then be augmented by incorporating other parameters.

None of the descriptions of PageRank calculation we have seen (e.g [5], [12], [19],[20]) are completely consistent in detail, but the general approach is clear enough. We define a matrix P such that²

²Some descriptions allow for multiple links between i and j by defining $p_{ij} = n_j/d_i$, where n_j is the number of links from i to j . We make the simplifying, but non-essential, assumption that such multiple links are coalesced, and that all $n_j = 1$.

$$p_{ij} = \begin{cases} d_i^{-1} & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \quad i, j = 1, \dots, n$$

If we assume an imaginary web surfer visiting a particular page follows each link on the page with equal probability, then the matrix P corresponds to a Markov chain model of the motion of this surfer through the web graph. This follows since p_{ij} then represents the transition probability that the surfer in state i (at page i) will move to state j (page j). If we let x_i be the probability that the surfer is in state i then by elementary theory (see e.g. [9]) for any initial vector x representing a distribution over possible locations of the surfer:

$$x^T P^k \rightarrow v^T \quad \text{as } k \rightarrow \infty$$

where v is the vector of probabilities representing the *steady state* of the system, with $v^T = v^T P$, as discussed in section 2.2.

The matrix P is said to be *reducible* if the surfer can get “trapped” in one section of the web; or more formally, if P can be permuted to the form:

$$\begin{pmatrix} P_{11} & 0 \\ P_{21} & P_{22} \end{pmatrix} \quad (1)$$

where the upper right partition is all zeros. It is easy to see that the values of v_i corresponding to the partition P_{22} are all zero, and so the surfer will get “trapped” in P_{11} . Particular, and frequent, examples of this phenomenon on the web are pages with zero out-degree, in which case we have $p_{ii} = 1$ (and $p_{ij} = 0$ for $j \neq i$). In Markov chain terminology, these are *absorbing states*.

Let us now consider the “ideal” definition of PageRank[19]—that is page i has rank x_i as a function of the rank of the pages which point to it:³

$$x_i = \sum_{(j,i) \in E'} d_j^{-1} x_j \quad (2)$$

where the set of edges E' is the set E with any self loops removed, and the out-degrees d_j modified if necessary to account for this.

This recursive definition gives each page a fraction of the rank of each page pointing to it—inversely weighted by the number of links out of that page. We may write this in matrix form as:

$$x = Ax \quad (3)$$

³This definition is often interpreted to mean that the “importance” of a page depends on the importance of pages pointing to it.

or

$$x = BD^{-1}x \tag{4}$$

where $D = \text{diag}(d_1, \dots, d_n)$ and B is a zero-one matrix with entries:

$$b_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E' \\ 0 & \text{otherwise} \end{cases} \quad i, j = 1, \dots, n \tag{5}$$

Note that $A = BD^{-1}$ is the transpose of the transition probability matrix P . At this point we also note that P excludes any self-loops in the web graph (other than those of absorbing states) from any kind of page ranking calculation, as they confer spurious PageRank.

We now have 3 very closely related, but distinct representations of the web:

- The directed web graph $G = (V, E)$.
- The transition matrix P
- The coefficient matrix A of the equations defining the ideal PageRank computation.

All three of these representations are useful in discussing PageRank calculations, but it is also important to keep them distinct—in particular to remember that the coefficient matrix A is the *transpose* of P , not P itself, while self-loops, and perhaps other forms of “spam” in G , are ignored.

Now, let us look at the ideal model (3). The PageRank vector x is clearly the *principal eigenvector* corresponding to the *principal eigenvalue* (with value 1) if this is nonzero (see e.g. [10]). Unfortunately, the real structure of the web, with many pages having zero in-degree (and others with zero out-degree) means that the transition matrix will have the structure (1)—in fact will be even more highly reducible—and hence the coefficient matrix A will have the structure:

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \tag{6}$$

and the eigenvector corresponding to the principal eigenvalue 1 will contain a great many zeros.

To get around this difficulty, Page[19] proposed an “actual PageRank model”:

$$x_i = (1 - \alpha) + \alpha \sum_{(j,i) \in E'} d_j^{-1} x_j \quad \forall i \tag{7}$$

or in matrix terms:

$$x = (1 - \alpha)e + \alpha Ax \tag{8}$$

where e is the vector of all 1's, and α ($0 < \alpha < 1$) is a parameter. Unless stated otherwise we use a value of 0.9 for α , but Page et al [20] report using a value of 0.85. This modification clearly overcomes the problem of identically zero PageRank—we may think of (7) as “seeding” each page with a rank of $(1 - \alpha)$.

Page et al[20] and Chakrabarti et al[5] then proceed to obtain an analogous eigenvalue problem again, as follows. Let us suppose that in addition to following links out of a page with probability p_{ij} a surfer makes a “random jump” every so often to some other page with uniform probability $1/n$. Let us suppose the surfer follows some link with probability α and makes the random jump with probability $(1 - \alpha)$. For convenience, we define $M = (1 - \alpha)E/n + \alpha A$, where E is ee^T , the matrix of all 1's. Then the modified transition probabilities are given by M , and the actual PageRank calculation becomes in matrix form:

$$x = Mx \tag{9}$$

With minor variations (e.g. excluding a random jump from a page to itself), this seems to be the standard model. It can be solved by application of the Power Iteration method (see [10]) as discussed in the next section.

It is easy to show that solving (8) and (9) are equivalent. If we scale the eigenvector obtained from (9) so that $e^T x = n$ we immediately obtain (8). Conversely, taking any solution x of (8) and noting that $e^T A = e^T$, we see that $e^T x = n$ and (9) follows.

Two observations are appropriate here. Firstly, much of the reformulation is necessary because of special small-scale structure in the web graph—zero in-degree and out-degree nodes in particular. Secondly, although we can preserve actual PageRank computation as an eigenvalue calculation, this can equivalently be accomplished by the solution of linear equations.

3 Power Iteration

As indicated in section 2, the PageRank computation is equivalent to computing the principal eigenvector of the matrix $M = (1 - \alpha)E/n + \alpha A$, as defined above. *Power iteration* is the most straightforward technique for computing the principal eigenvector of a matrix, and the technique commonly used to compute PageRank. In this section, we study the compu-

tation of PageRank using power iteration. We also provide characterizations of the microscopic structure of the web to explain the convergence properties of power iteration. These characterizations have implications for the problem of uniformly sampling web pages using random walks over the webgraph—currently the best known technique for generating uniform samples of webpages [13, 1, 14, 2].

In power iteration, an arbitrary vector is repeatedly multiplied with the given matrix until it provably converges to the principal eigenvector. The power iteration algorithm for PageRank computation is given below:

1. $s \leftarrow$ a random $n \times 1$ vector, typically e_n .
2. $r \leftarrow M \times s$.
3. If $\|r - s\| < \epsilon$, end. r is the PageRank vector.
4. $s \leftarrow r$. Goto step 2.

3.1 Implementation Issues

Implementation of this algorithm raises many system issues. The goal of this paper is not to give a detailed description of these issues; the interested reader is referred to [11]. Given the scale of the web, it is obvious that neither the vectors r , s nor the matrix M can be assumed to fit in the memory. So we have to use secondary memory techniques for implementing the matrix product of step 2 and the error computation of step 3 of the power iteration. Note that although the matrix M is “dense”, one need not explicitly materialize it. We can efficiently implement $M \times s$ using equation (8), which involves multiplication using the sparse matrix A . In [11], it is discussed how one can implement a sparse matrix vector multiplication using a “nested loop join” like technique.

3.2 Convergence of Power Iteration

Theoretically, the convergence of power iteration for any given matrix depends on the *eigenvalue gap*, the difference in the moduli of the largest two eigenvalues of the matrix. The larger the eigenvalue gap, the faster the convergence of power iteration. The eigenvalue gap for stochastic matrices also has a neat graph theoretic characterization in terms of *expanders*[18]. Without giving precise definitions, a graph is an expander if the neighborhood of every subset of nodes of the graph is greater than some multiple of the size of the subset. Expanders have the desirable property that the power iteration

converges quickly to the stationary state—in our case the PageRank. Thus stochastic matrices that have a large eigenvalue gap correspond to expander graphs and vice versa. If the webgraph were an expander, power iteration would converge quickly. In subsequent sections we study the convergence of the PageRank computation in practice and examine the structural insights that these experiments have to offer.

3.3 Power Iteration Experiments

The experiments described in this and the following sections were done using a snapshot of the webgraph, crawled in 1998, available at the Stanford WebBase repository [15]. The webgraph contains around 25 million URLs and around 250 million links between these URLs. The largest strongly connected component (SCC) in this webgraph, with about 15 million pages, was used for all the experiments. As shown in [4] the largest strongly connected component forms a significant fraction of the entire web. We restrict our attention to the SCC because it allows experimentation with all possible values of α including $\alpha = 1$. In the following experiments, the vector s of the power iteration was initialized to a vector of all 1s.

A common measure of convergence for power iteration is the norm of the *residual vector*, which is the difference of two successive vectors generated by the power iteration. We consider two different norms, the L_2 norm $\|v\|_2 = (\sum_i v_i^2)^{1/2}$ and the L_∞ norm $\|v\|_\infty = \max_i |v_i|$. Note that this error measure is not the same as the *absolute error*—the difference between the principal eigenvector (that we are trying to compute) and the current vector. In section 4 we measure convergence with respect to absolute error.

Figure 1 shows the convergence of power iteration for $\alpha = 0.99$ with respect to the L_2 norm. The Figure 2 shows the same in log scale as one of the 4 curves.

One can easily see that the convergence of power iteration is very slow (although it is rapid in the first few iterations). After about 50 iterations, the L_2 error is about 2800. Thus the average error per page (there are around 15 million pages) is approximately 0.5—unacceptably high as a fraction of the average PageRank per page, 1.0. Moreover, from the L_∞ error we observe that there exist pages whose PageRank changes (within one iteration) by as much as 2000. This clearly suggests that the convergence of the power iteration is *very* slow for values of α close to 1. Indeed, for the case $\alpha = 1$, the “ideal PageRank”, the power iteration practically stops converging after about 40 iterations. Thus we observe that the entire webgraph is not a good expander. This suggests that a plain power iteration is likely to be laborious

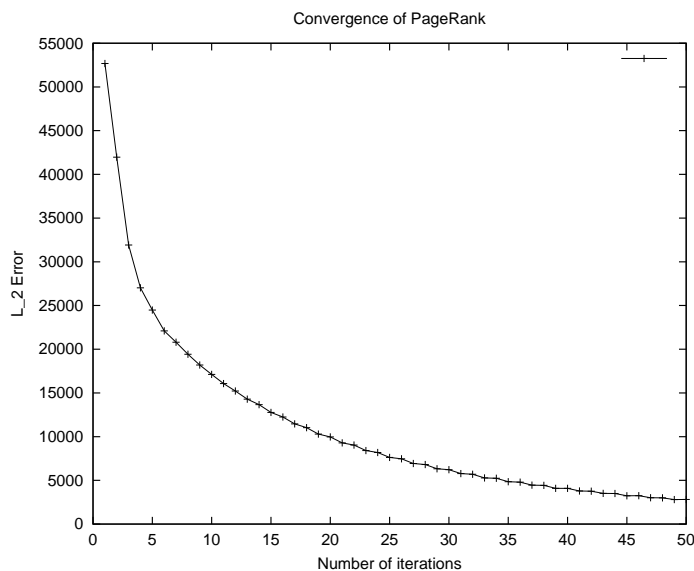


Figure 1.
Convergence of PageRank for power iteration with $\alpha = 0.99$

for PageRank computation. We study more sophisticated linear algebraic techniques in section 4.

We consider next the effect of the parameter α on power iteration. Figure 2 shows the log of the error after each power iteration for various values of α .

Figure 2 clearly shows that the parameter α plays an important role in the convergence of power iteration. The power iteration converges much faster for smaller values of α . Intuitively, this is what we might expect from the definition of the expander graph. A small value of α implies that the *random surfer* would make more random jumps, the effect of which is to implicitly add more “edges” out of each subgraph (corresponding to the random jumps). However, using a smaller value of α has implications on the quality of the PageRank ranking scheme. Intuitively, by making random jumps more often we are using less information in the links. As an extreme case, consider what happens when $\alpha = 0$.

Alternately, one can consider the effect of α as follows. In the “web surfer” interpretation of PageRank, the surfer follows an outlink with probability α , and jumps uniformly to a random location with probability $1 - \alpha$.

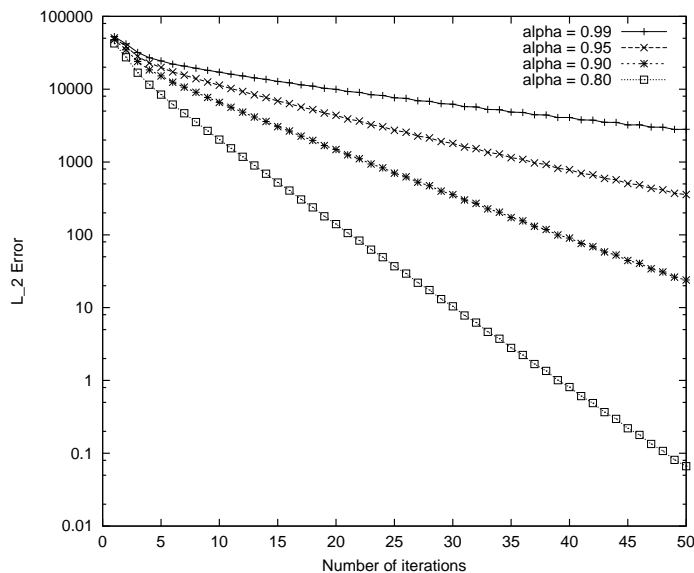


Figure 2.
Convergence for four different values of α

Thus, the expected number of steps until a uniform jump is $1/(1 - \alpha)$. As α decreases, this value drops rapidly—with $\alpha = 0.8$, for instance, the expected path length is 5. By elementary probability, the number of paths that will be significantly greater than this expectation is small. Therefore, for small α , the actual PageRank model, far from approximating the principal eigenvector of matrix A , behaves like a random walk of five steps from the uniform distribution. Clearly this walk is straightforward to compute. We therefore expect the behaviour of Figure 2, in which a decrease in α has a significant and analytical impact on convergence.

3.4 Convergence properties of the SCC

The above experiments suggest a picture of connectivity in the SCC. The rapid convergence during early rounds starting from the uniform distribution suggests that most regions of the graph have good expansion and allow the system to move rapidly towards steady state. However the slow convergence of the process with values of α close to 1, after the first few rounds, suggests that there are regions of the SCC that cause the power iteration to “bog down.” We wish to understand the nature of these regions to explain

and perhaps improve the performance of power iteration. Further, given that surfing is a common and important approach to web exploration, we also wish to understand the sociological implications of these self-contained regions of the graph.

We can naturally partition the graph into links inside web sites, and links across web sites, and then ask whether either set of links in particular is responsible for the slow convergence. To answer this question, we consider the convergence of power iteration on the *host graph*, in which each vertex corresponds to an entire web site. Any edge between pages in different sites is represented as an edge between the corresponding site vertices in the host graph. We consider two variants: in the first we allow multiple edges between sites if multiple pages on one site contained links to the same destination site. In the second, we collapse all such hyperlinks into a single edge. Figure 3 shows the convergence of power iteration for these two variants of the hostgraph against the convergence for the entire SCC. The error values are normalized for comparison purposes.

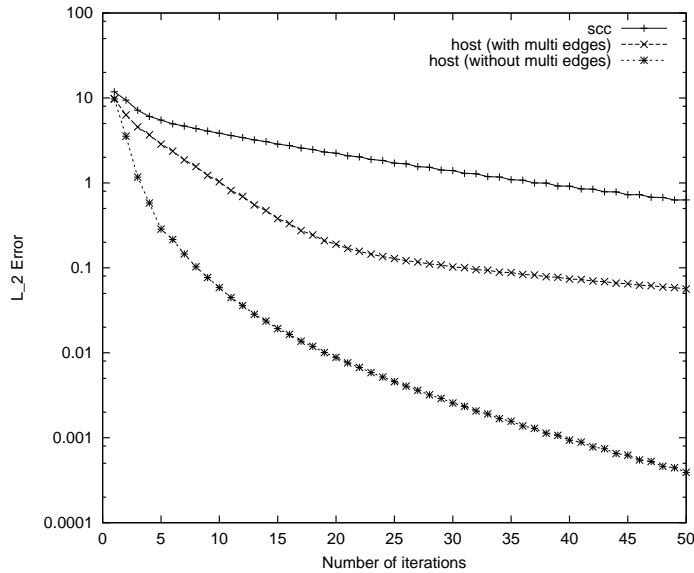


Figure 3.
Convergence of Host graph vs web graph

Note that the error measure in Figure 3 is plotted in log scale. Clearly, error rates are orders of magnitude smaller in the host graph than in the

whole SCC. Furthermore, using ARPACK[17], we were able to compute numerically the second eigenvalue of the host graph, which is 0.9896. Thus, the eigenvalue gap is larger than 1%, implying quite rapid convergence. On the other hand, the eigenvalue gap for the SCC was too small for us to complete computation of the second eigenvalue in reasonable time, as we might have expected from the extremely slow convergence rates of power iteration previously observed.

Thus, connectivity between web sites is strong and robust. In fact, because expander graphs have logarithmic diameter, a surfer jumping from site to site could arrive at any site of interest within a few steps. However, once we shift from the granularity of the host graph back to the granularity of the web graph itself, the complex structure within individual sites causes the surfer to become “stuck” within a local region.

Certainly, the majority of web sites do not contain such “surfer traps,” so the remaining problem is to identify particular sites that slow convergence, and deal with them using special-purpose techniques. Our initial attempts along this line have been promising, but we do not have conclusive evidence.

We now turn to alternate techniques for improving the performance of naive computations of PageRank.

4 Eigensystem or Equations?

We think it is probably fair to say that, all other things being equal, most numerical analysts would prefer to solve a set of equations rather than find an eigenvector. While the conventional method of computing page rank has involved finding the eigenvector specified by (9), we might equally well consider solving the set of equations (8), which we can write more conventionally as:

$$(I - \alpha A)x = (1 - \alpha)e \quad (10)$$

For very large sparse matrices there are a large number of iterative solution methods (see e.g. [10], [22]), some of which we might hope would converge more quickly than the power method for eigenvectors.

We have experimented with two iterative methods, and rather unsystematically with a third, comparing the convergence with the power method for (9). Our first experiment used the simple Jacobi iteration for solving (10), where denoting the rank vector at each iteration k by $x^{(k)}$ we compute:

$$x_i^{(k+1)} = (1 - \alpha) + \alpha \sum_{(j,i) \in E'} a_{ij} x_j^{(k)} \quad \forall i \quad (11)$$

Note that this is very similar to the power iteration step,

$$x_i^{(k+1)} = \frac{(1-\alpha)}{n} \sum_{j=1}^n x_j^{(k)} + \alpha \sum_{(j,i) \in E'} a_{ij} x_j^{(k)} \quad \forall i \quad (12)$$

and indeed when we start both processes with the initial iterate $x^{(0)} = e$ (all 1's), then the first step will be identical. Since the matrix is (column) stochastic then this should continue in subsequent iterations as $\sum_i x_i^{(k)}$ will remain constant, ignoring rounding error, and we observe this to be the case in practice. However, we are not restricted to using the Jacobi iteration. The next obvious step is to use the *Gauss-Seidel* iteration:

$$x_i^{(k+1)} = (1-\alpha) + \alpha \sum_{j<i} a_{ij} x_j^{(k+1)} + \alpha \sum_{j>i} a_{ij} x_j^{(k)} \quad \forall i \quad (13)$$

which uses the most recent values $x_j^{(k+1)}$ wherever possible. To illustrate the advantage of this method we plot the convergence of the two methods using the sample 15 million node SCC of the previous sections. In this experiment we first computed the eigenvector (for $\alpha = 0.9$) using the ARPACK software[17] to full single precision accuracy—where the eigenvector is normalized so that $\|x\|_2 = 1$. We then ran the Power Iteration and Gauss-Seidel methods, at each step computing the deviation of the current iterate $x^{(k)}$ from the ARPACK solution x , by normalizing $x^{(k)}$ so that $\|x^{(k)}\|_2 = 1$, and then evaluating $\|x^{(k)} - x\|_2$. The results are shown in Figure 4.

The Gauss-Seidel method clearly converges much faster than the Power or Jacobi methods.

We also experimented with the well known “Successive Over-Relaxation” (SOR) method, which uses a relaxation parameter ω and iterates:

$$x_i^{(k+1)} = \omega \left\{ (1-\alpha) + \alpha \sum_{j<i} a_{ij} x_j^{(k+1)} + \alpha \sum_{j>i} a_{ij} x_j^{(k)} \right\} + (1-\omega) x_i^{(k)} \quad \forall i \quad (14)$$

We found that values of ω greater than 1 degraded convergence quite significantly, and values less than 1 (we tried 0.9 and 0.8) produced minor degradations. Without more sophisticated analysis, it appears that SOR has little to offer over Gauss-Seidel in this application. However, we have barely begun to investigate whole classes of promising iterative methods for solving large sparse systems of linear equations—including, for example, conjugate gradient or Lanczos methods (see [10]).

The idea of equation-solving for PageRank appears to have significant computational advantages, and few disadvantages, when considered in isolation. We shall see in the following section that it has even more significant advantages for the larger picture.

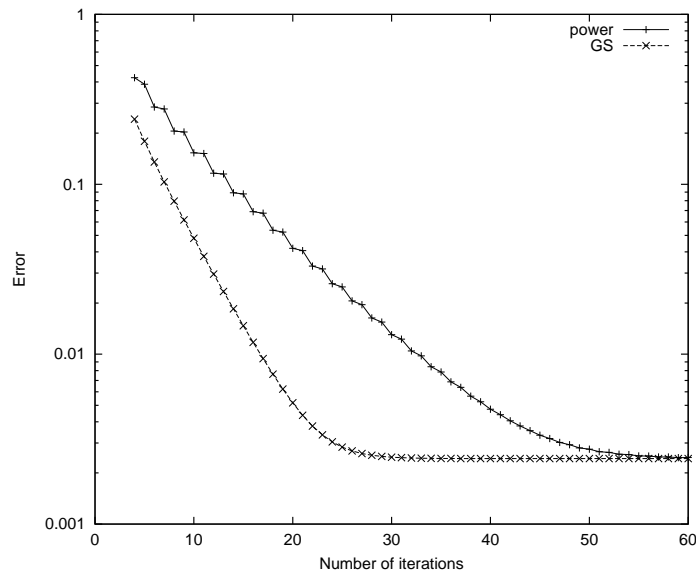
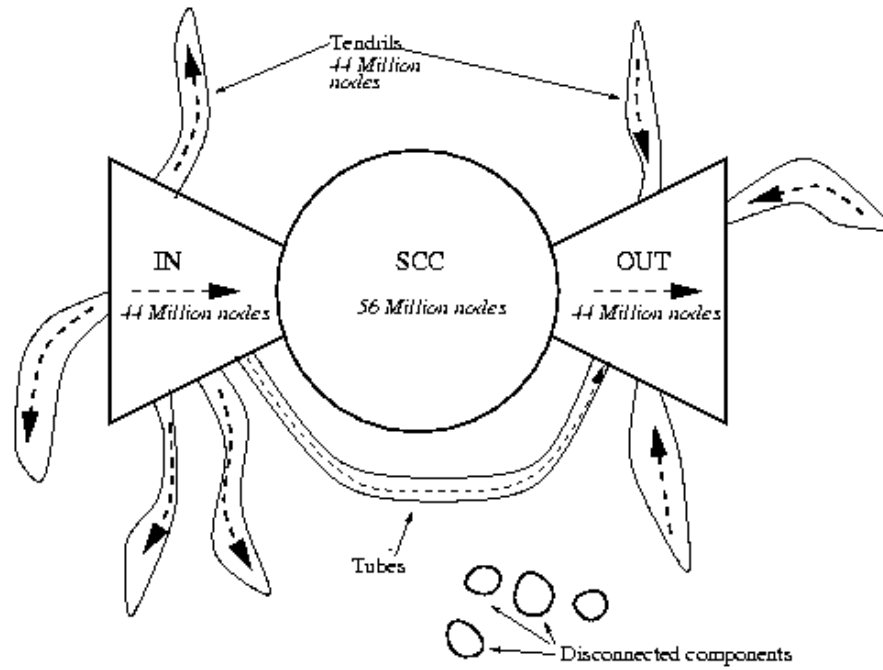


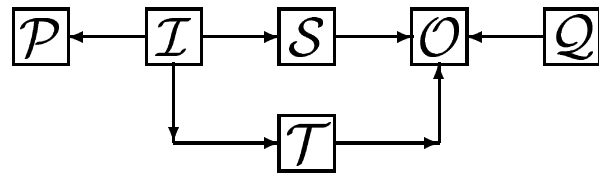
Figure 4.
 Logarithmic convergence of PageRank for Power Iteration
 and Gauss-Seidel with $\alpha = 0.9$

5 Large Scale Structure of the Web and its Implications

So far we have considered only small-scale structure of the web graph and its associated matrices. When we consider the large-scale structure of the web, the arguments for using an equation-solving approach become even stronger. As is now well known, the graph structure of the web may be characterized by the “bow tie” (see Broder et al[4])



Ignoring the disconnected components, which can be dealt with separately, this corresponds to an aggregated graph structure:



where \mathcal{I} , \mathcal{S} , \mathcal{O} represent the “IN”, “strongly connected (SCC)”, and “OUT” segments of the web and \mathcal{P} , \mathcal{Q} , \mathcal{T} represent the “in tendril”, “out tendril” and “tube” nodes. This structure immediately induces (under symmetric

There may also be nodes (pages) which do not fit into any of the subgraphs previously defined. These anomalous nodes are still part of the weakly connected main component, but may only be connected to the bow tie components already identified by edges directed out from \mathcal{Q} and/or by edges directed in to \mathcal{P} . These anomalous nodes may however be connected among themselves arbitrarily. This new subgraph of anomalous nodes can then form a new diagonal block (or blocks) on the diagonal somewhere between \mathcal{P} and \mathcal{Q} in (15). Given the possibilities, it seems best then to concentrate on the more general structure (16).

The structure (16), where each diagonal block is irreducible (corresponding to a strongly connected component of G), may be extracted by Depth First Search (DFS), e.g. Tarjan’s algorithm [21] (see also [7], [8]), which may find a very large number of diagonal blocks - indeed every “dead-end” page with no outlinks corresponds to a degenerate 1 by 1 diagonal block. The largest diagonal block corresponds to the SCC, and this is the only really efficient way that we know of to find it. We may therefore consider the finding of the entire detailed block structure (16) equivalent to the work involved in just the first step of finding the SCC for the bow tie.

At this point it is appropriate to mention the following properties of the DFS approach to finding the block triangular structure:

1. The DFS method is *linear* in the size of A . More precisely it is linear in the number of nodes plus number of edges. Thus we expect the DFS process to take about the same time as an iteration (or a few iterations) of the iterative methods we have discussed for the whole matrix A .
2. The number of strong components found will in general be very large. For efficiency and manageability we should aggregate many of the adjacent smaller diagonal blocks, so that the A_{i_i} are as few in number as will permit them to each fit in memory, whenever possible, for reasons we shall see below.
3. The actual partitioning of A is also a linear process, requiring only a single pass of the matrix.

The block triangular structure has obvious possibilities for parallel processing—whether we seek an eigenvector by the power method or are solving a system of equations—with submatrices being assigned to processors and passing submatrix-vector multiples to build up a complete matrix-vector product for, e.g., power iterations or Jacobi. The situation becomes slightly more complicated, but still straightforward for Gauss-Seidel.

When an equation solving method is used we have an opportunity to use not only parallelism but, quite independently, to use *partitioning*. While the eigenvector calculation must be rescued by the notional random jumps, which produce a *fully dense* matrix in (9) we are free to fully exploit the structure in (15), or more generally (16). Conformably partitioning x with the matrix so that

$$x^T = (x_1^T, x_2^T, x_3^T, \dots, x_N^T)$$

equation (10) can be solved by solving a sequence of smaller problems, (see [8]) beginning with:

$$(I - \alpha A_{NN})x_N = (1 - \alpha)e \quad (17)$$

followed by:

$$(I - \alpha A_{N-1, N-1})x_{N-1} = (1 - \alpha)e + \alpha A_{N-1, N}x_N \quad (18)$$

leading to the general step:

$$(I - \alpha A_{ii})x_i = (1 - \alpha)e + \alpha \sum_{j=i+1}^N A_{ij}x_j \quad (19)$$

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

Note that using this partitioning approach, there is still some opportunity for parallelism (e.g. in computing the off-diagonal matrix vector products $A_{ij}x_j$), and that when some of the off-diagonal blocks are zero, we may compute some of the x_i in parallel also. For example, if we were to actually take the structure in the form of (15), $x_{\mathcal{I}}$ and $x_{\mathcal{Q}}$ could be computed completely in parallel, as could the pairs $x_{\mathcal{S}}, x_{\mathcal{T}}$ and $x_{\mathcal{P}}, x_{\mathcal{O}}$.

However, it must be emphasized that the real benefit from the equation solving approach is not so much the parallelism as the partitioning into smaller subproblems (19). Generally speaking, we expect the solution of the subproblems by iterative methods to take fewer iterations individually than is required for the huge matrix A , leading to less arithmetic—especially since the off-diagonal matrix-vector products ($A_{ij}x_j$, etc.) need be performed only once, rather than at every iteration. The other payoff, perhaps *the* payoff, is in reduced I/O.

The enormous size of A makes it impractical to solve in main memory on standard work stations. Therefore the matrix (and perhaps even the vectors $x^{(k)}$) must be buffered into memory. For power iterations, the entire matrix A must be buffered every iteration, and we have seen that convergence of

Strong Component Size	Number of This Size
8,188,380	1
108,131	1
30,000 - 40,000	7
20,000 - 29,000	34
10,000 - 19,999	68
1,000 - 9,999	808
2 - 1,000	221,149
1	4,771,701

Table 1.
Size and Number of Strong Components

this method is at best inferior to Gauss-Seidel. When we solve the sequence of sets of equations (19) we expect the largest of the subproblems to be of the order of the SCC, and many of the A_{ii} matrices may now be able to fit in memory. To illustrate this, we show in Table 1 the size of the strong components for the web graph of an experimental (and incomplete) crawl of the IBM intranet, with 20,813,497 pages and 333,203,246 links. We see that there is a rapid drop-off in size from the largest SCC of about 8 million pages to the next largest of about 100,000 and then many much smaller components. This indicates that our suggested strategy of aggregating adjacent components (i.e. diagonal blocks in (16)) can be expected to successfully produce many A_{ii} submatrices which fit in main memory—perhaps for all but the largest SCC. In this situation the total amount of buffering (I/O) required is considerably reduced, which should have even more impact on elapsed solution time than the reduced amount of arithmetic; an important consideration if we wish to keep our PageRank data as current as possible for a rapidly expanding web.

6 Conclusion

We have studied web-scale computation of the PageRank static ranking function, incorporating both algorithmic techniques drawn from numerical analysis, and particular structure in the problem instance drawn from web characterization. Algorithmically, we approach the problem using equation solving rather than eigensystem techniques. This approach yields significant performance improvements over power iteration. Further, it allows exploitation of the well-known “bow tie” characterization of the web to partition

the problem into independent subproblems. Finally, our own characterization of the expansion properties of the graph as a whole versus the graph of inter-site connectivity explains the poor performance of power iteration, and suggests a new direction for further work. Namely, in exploiting the convergence properties of the expander graph that links sites together, while developing specialized techniques to deal with individual site subgraphs of small eigenvalue gap.

At present our experiments are limited to a (by no means trivial) WebBase [15] subgraph of the net and the IBM intranet. We await the results of larger crawls being carried out by or colleagues to test the methods we have discussed here on a full web scale.

7 Acknowledgements

The authors are indebted to Ravi Kumar and Sridhar Rajagopalan for valuable discussions in the course of this work. We also thank the Stanford University WebBase group for providing access to the web data, and Wang Lam for his assistance in processing the WebBase data.

References

- [1] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, D. Weitz, “Approximating Aggregate Queries about Web Pages via Random Walks,” VLDB 2000.
- [2] K. Bharat, A. Broder, “A Technique for Measuring the Relative Size and Overlap of Public Web Search Engines,” Proc. 7th International World Wide Web Conference, 1998.
- [3] S. Brin and L. Page, “The Anatomy of a Large-Scale Hypertextual Web Search Engine”, Proc. of WWW7, Brisbane, Australia, June 1998. See: <http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm>
- [4] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins and J. Wiener, “Graph Structure in the Web”, Proc. WWW9 conference, 309-320, May 2000. See also: <http://www9.org/w9cdrom/160/160.html>
- [5] S. Chakrabarti, B.E. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins, “Topic Distillation and Spectral Filtering”, *Artificial Intelligence Review* **13**: 409-435 (1999).

- [6] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, J. M. Kleinberg, "Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text," Proc. 7th International World Wide Web Conference, 1998.
- [7] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, McGraw-Hill (1990).
- [8] I.S. Duff, A.M. Erisman and J.K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford and New York (1986).
- [9] W. Feller, *An Introduction to Probability Theory and its Applications, Vol 1 (3rd edition)*, Wiley, NY (1968)
- [10] G.H. Golub and C.F. Van Loan, *Matrix Computations (3rd edition)*, Johns Hopkins University Press, Baltimore and London (1996).
- [11] T. H. Haveliwala, "Efficient Computation of PageRank," Technical Report, Stanford University, 1999.
- [12] M. R. Henzinger, "Hyperlink Analysis for the Web", *IEEE Internet Computing*, pp 45–50, Jan–Feb (2001).
- [13] M. Henzinger, A. Heydon, M. Mitzenmacher, M. Najork, "On near-uniform URL sampling," Proc. 9th International World Wide Web Conference, 2000.
- [14] M. Henzinger, A. Heydon, M. Najork, "Measuring Index Quality Using Random Walks on the Web," Proc. 8th International World Wide Web Conference, 1999.
- [15] J. Hirai, S. Raghavan, H. Garcia-Molina and A. Paepcke, "WebBase: A Repository of Web Pages", Proc. WWW9 conference, May 2000. See also: <http://www9.org/w9cdrom/296/296.html>
- [16] J. M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *JACM*, **46**(5) pp 604-632, 1999.
- [17] R. B. Lehoucq, D. C. Sorensen and C. Yang, *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Rice University, October 1997. see: <http://www.caam.rice.edu/software/ARPACK/>
- [18] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press (1995).

- [19] L. Page, "PageRank: Bringing Order to the Web", undated. See: <http://hci.stanford.edu/page/papers/PageRank/ppframe.htm>
- [20] L. Page, S. Brin, R. Motwani and T. Winograd "The PageRank Citation Ranking: Bringing Order to the Web", Stanford Digital Library working paper SIDL-WP-1999-0120 (version of 11/11/1999). See: <http://www-diglib.stanford.edu/cgi-bin/get/SIDL-WP-1999-0120>
- [21] R.E. Tarjan, "Depth-first Search and Linear Graph Algorithms", *SIAM J. Computing*, **1**, pp 146-160 (1972).
- [22] R.S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ (1962).