SIAM J. SCI. COMPUT. © 2010 Society for Industrial and Applied Mathematics
Vol. 32, No. 1, pp. 349–371

AN INNER-OUTER ITERATION FOR COMPUTING PAGERANK[*]

DAVID F. GLEICH[†], ANDREW P. GRAY[‡], CHEN GREIF[§], AND TRACY LAU[§]

Abstract. We present a new iterative scheme for PageRank computation. The algorithm is applied to the linear system formulation of the problem, using inner-outer stationary iterations. It is simple, can be easily implemented and parallelized, and requires minimal storage overhead. Our convergence analysis shows that the algorithm is effective for a crude inner tolerance and is not sensitive to the choice of the parameters involved. The same idea can be used as a preconditioning technique for nonstationary schemes. Numerical examples featuring matrices of dimensions exceeding 100,000,000 in sequential and parallel environments demonstrate the merits of our technique. Our code is available online for viewing and testing, along with several large scale examples.

Key words. damping factor, eigenvalues, inner-outer iterations, PageRank, power method, stationary schemes

AMS subject classifications. 65F10, 65F15, 65C40

DOI. 10.1137/080727397

1. Introduction.

PageRank [35] is a method for ranking Web pages whereby a page's importance (or rank) is determined according to the link structure of the Web. This model has been used by Google as part of its search engine technology. The exact ranking techniques and calculation methods used by Google today are no longer public information, but the PageRank model has taken on a life of its own and has received considerable attention in the scientific community in the last few years. PageRank is essentially the stationary distribution vector of a Markov chain whose transition matrix is a convex combination of the Web link graph and a certain rank-1 matrix. A key parameter in the model is the *damping factor*, a scalar that determines the weight given to the Web link graph in the model. Due to the great size and sparsity of the matrix, methods based on decomposition are considered infeasible; instead, iterative methods are used, where the computation is dominated by matrix-vector products. Detailed descriptions of the problem and available algorithms can be found in many references; see, for example, [12, 30].

In this paper we propose and investigate a new algorithm for computing the PageRank vector. We use the linear system formulation and apply stationary inner-outer iterations. The proposed technique is based on the observation that the smaller the damping factor is, the easier it is to solve the problem. Hence we apply an iterative scheme in which each iteration requires solving a linear system similar in its algebraic structure to the original one, but with a lower damping factor. In essence, what is proposed here is a simple preconditioning approach. We use a technique of inexact

[*]Received by the editors June 15, 2008; accepted for publication (in revised form) June 5, 2009; published electronically February 5, 2010.
http://www.siam.org/journals/sisc/32-1/72739.html

[†]Institute for Computational and Mathematical Engineering, Stanford University, 496 Lomita Mall Rm. 110, Stanford, CA 94305 (dgleich@stanford.edu). This author was supported by a Microsoft Live Labs fellowship.

[‡]Faculty of Medicine, The University of British Columbia, 317–2194 Health Sciences Mall, Vancouver, BC, V6T 1Z3, Canada (apgray@gmail.com).

[§]Department of Computer Science, The University of British Columbia, Vancouver, BC, V6T 1Z4, Canada (greif@cs.ubc.ca, tracylau@cs.ubc.ca). The work of the third author was supported by the Natural Science and Engineering Council of Canada (NSERC). The work of the fourth author was supported by an NSERC USRA internship.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

solves, whereby the inner iteration is solved only to a crude tolerance. The algorithm is just a few lines long, and can be implemented and parallelized in a straightforward fashion. We are not aware of any other PageRank algorithm that provides such an excellent combination of minimal memory requirements and fast convergence.

The remainder of the paper is structured as follows. In section 2 we provide a brief description of the PageRank problem. In section 3 we introduce the proposed algorithm. Section 4 is devoted to convergence results, and we provide some guidelines for the selection of the parameters involved. In section 5 we discuss how inner-outer iterations can be combined with other solvers as an acceleration scheme or as a preconditioner. Numerical examples for a few large problems are given in section 6. Finally, in section 7 we draw some conclusions.

2. The problem of computing PageRank.

The "raw" PageRank x_i of page i is defined as $x_i = \sum_{j\to i} \frac{x_j}{n_j}$, where $j \to i$ indicates that page j links to page i, and n_j is the out-degree of page j. Thus the problem in its basic form can be mathematically formulated as follows: find a vector $\mathbf{x}$ that satisfies $\mathbf{x} = \bar{P}\mathbf{x}$, where $\bar{P}$ is given by

$$\bar{P}_{ji} = \begin{cases} \frac{1}{n_i} & \text{if } i \to j, \\ 0 & \text{if } i \nrightarrow j. \end{cases}$$

Pages with no out-links (or, in graph terminology, out-edges) produce columns of all zeros in $\bar{P}$; hence $\bar{P}$ is not necessarily a stochastic matrix. Replacing $\bar{P}$ with

$$P = \bar{P} + \mathbf{u}\mathbf{d}^T, \qquad \mathbf{d}_i = \begin{cases} 1 & \text{if } n_i = 0, \\ 0 & \text{otherwise} \end{cases}$$

eliminates the zero columns. Here the vector $\mathbf{u} \geq 0$ is a probability vector, which we call the *dangling node vector*. Now the modified matrix P is a proper stochastic matrix. There are other alternatives to the correction $\mathbf{u}\mathbf{d}^T$ for dealing with dangling nodes, for example, the bounce-back model [30].

The ergodic theorem [21, Theorem 6.4.2] tells us that the stationary distribution is unique and is the limiting distribution starting from any initial distribution if the transition matrix is irreducible and aperiodic. In the case of PageRank, forming a convex combination of P with a certain rank-1 matrix achieves these desirable properties. We define

$$A = \alpha P + (1-\alpha)\mathbf{v}\mathbf{e}^T, \tag{2.1}$$

where $\alpha \in (0,1)$ is the damping factor, $\mathbf{e}$ is the vector of all ones, and $\mathbf{v}$ is a *personalization vector* or a *teleportation vector*. Just like the dangling node vector $\mathbf{u}$, $\mathbf{v}$ is a probability vector. See [7] for a discussion about the impact of setting $\mathbf{u} \neq \mathbf{v}$. It is common practice to set $\mathbf{u} = \mathbf{v}$ in the PageRank model [22, 26, 32].

The PageRank vector is defined as the vector satisfying $\mathbf{x} = A\mathbf{x}$, with A defined in (2.1). We note that, in general, for nonnegative $\mathbf{v}$, primitivity and irreducibility are not needed to ensure the uniqueness of the PageRank vector [37]. In terms of the model, it is assumed that in each time step, a "random" surfer either follows a link with probability α, or "teleports" with probability $1-\alpha$, selecting the new page from the probability distribution given by $\mathbf{v}$.

Selecting a damping factor significantly smaller than 1 allows for fast convergence of the power method, since α is in fact the contraction factor. (The scheme converges linearly.) In the original formulation of PageRank [35], the choice $\alpha = 0.85$ was

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

suggested, and the power method was applied. A higher value of α (i.e., close to 1) yields a model that is mathematically closer to the actual link structure of the Web, but makes the computation more difficult.

The choice of α has been explored in several papers, and seems to be application-dependent. Values that have been widely used in the Web search literature are $\alpha = 0.85,\ 0.9$, or 0.95 [26, 34]. Computing the limiting vector as $\alpha \to 1$ is explored in [11]. There are also papers that claim that the damping factor should be significantly smaller than 1. For example, in [2] it is claimed that α as small as 0.5 should be used, due to the presence of strongly connected components and the bow-tie structure of the Web; see [8] for arguments in favor of a small damping factor. PageRank-type algorithms are used in application areas other than Web search [33, 39, 46], where the value of α often has a concrete meaning and the graphs have a different structure that may not resemble the bow-tie structure of the Web link graph. It is thus useful to consider a variety of values, and in this paper we experiment with $0.85 \leq \alpha \leq 0.999$.

Notice that A is dense because $\mathbf{v}\mathbf{e}^T$ is typically dense, but it is never explicitly formed since matrix-vector products of A with $\mathbf{x}$ can be efficiently computed by imposing $\|\mathbf{x}\|_1 = 1$ (or, equivalently, $\mathbf{e}^T\mathbf{x} = 1$, since $\mathbf{x}$ is nonnegative):

$$A\mathbf{x} = \alpha P\mathbf{x} + (1-\alpha)\mathbf{v}. \tag{2.2}$$

When the power method is used, if the initial guess has a unit 1-norm, then so do all the iterates $\mathbf{x}$ throughout the iteration, and normalization does not seem necessary. Nonetheless, it is often carried out in practice for large scale problems in finite precision to improve the accuracy of the computation [45].

Since Brin and Page's original formulation of the PageRank problem, much work has been done by the scientific community to propose and investigate improvements over the power method. In [26] an extrapolation method is presented, which accelerates convergence by calculating and then subtracting off estimates of the contributions of the second and third eigenvectors. Analysis of the spectral structure of the PageRank matrix and interesting results and observations about the sensitivity of the eigenvalue problem are given in [15, 37]. An Arnoldi-type technique is proposed in [19]. Other methods have been considered; see the book [30], which offers a comprehensive survey of PageRank, the review papers [4, 29], which contain many additional results and useful references, and the books [5, 40], which include overviews of nonnegative matrices and Markov chains.

3. An inner-outer stationary method.

We now present our new algorithm. As is pointed out in [1, 18], since $\mathbf{e}^T\mathbf{x} = 1$, from (2.2) it follows that the eigenvalue problem $A\mathbf{x} = \mathbf{x}$ can be reformulated as a linear system:

$$(I - \alpha P)\mathbf{x} = (1-\alpha)\mathbf{v}. \tag{3.1}$$

Suppose β is a positive scalar. The same PageRank problem with β as a damping factor, namely $(I - \beta P)\mathbf{x} = (1-\beta)\mathbf{v}$, is easier to solve than the original problem if β is smaller than α. This is well understood, and one way to see it is by observing that other than the eigenvalue 1, all the eigenvalues of the matrix of (2.1) are contained in a circle on the complex plane whose radius is equal to the damping factor [10, 15].

Inspired by the fact that the original problem is easier to solve when the damping factor is small, let us consider the following stationary iteration for solving (3.1):

$$(I - \beta P)\mathbf{x}_{k+1} = (\alpha - \beta)P\mathbf{x}_k + (1-\alpha)\mathbf{v}, \qquad k = 0, 1, 2, \ldots, \tag{3.2}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

where $0 < \beta < \alpha$. In what follows, we will refer to (3.2) as the *outer iteration*. As an initial guess we may take $\mathbf{x}_0 = \mathbf{v}$, the original teleportation vector, though other choices are possible, too. Applying (3.2) cannot be done directly since solving a linear system with $I - \beta P$ is still computationally difficult, even when β is small. Therefore, we compute or approximate $\mathbf{x}_{k+1}$ using an inner Richardson iteration as follows. Setting the right-hand side of (3.2) as

$$\mathbf{f} = (\alpha - \beta)P\mathbf{x}_k + (1-\alpha)\mathbf{v}, \tag{3.3}$$

we define the inner linear system as

$$(I - \beta P)\mathbf{y} = \mathbf{f}, \tag{3.4}$$

and apply the *inner iteration*

$$\mathbf{y}_{j+1} = \beta P \mathbf{y}_j + \mathbf{f}, \qquad j = 0, 1, 2, \ldots, \ell - 1, \tag{3.5}$$

where we take $\mathbf{y}_0 = \mathbf{x}_k$ as the initial guess and assign the computed solution $\mathbf{y}_\ell$ as the new $\mathbf{x}_{k+1}$. (The stopping criterion that determines ℓ will be discussed below.) As long as we have not converged to the desired PageRank vector (using a convergence criterion related to the original, outer linear system (3.1)) we repeat the procedure, incrementing k and then using $\mathbf{x}_k$ as our new inner initial guess $\mathbf{y}_0$.

The outer iteration (3.2) is associated with the splitting

$$I - \alpha P = M_O - N_O, \quad M_O = I - \beta P, \quad N_O = (\alpha - \beta)P, \tag{3.6}$$

and the corresponding outer iteration matrix is given by

$$T_O = M_O^{-1} N_O = (\alpha - \beta)(I - \beta P)^{-1} P. \tag{3.7}$$

The inner iteration (3.5) is aimed at solving $M_O \mathbf{y} = \mathbf{f}$, as stated in (3.4), and is associated with the splitting

$$M_O = I - \beta P = M_I - N_I, \quad M_I = I, \quad N_I = \beta P. \tag{3.8}$$

The corresponding inner iteration matrix is

$$T_I = M_I^{-1} N_I = \beta P. \tag{3.9}$$

To terminate the iterations, we use the 1-norm of the residuals of the outer system (3.1) and the inner system (3.4) as stopping criteria. For the outer iteration (3.2) we apply

$$\|(1-\alpha)\mathbf{v} - (I - \alpha P)\mathbf{x}_{k+1}\|_1 < \tau,$$

and for the inner iteration (3.5) we use

$$\|\mathbf{f} - (I - \beta P)\mathbf{y}_{j+1}\|_1 < \eta.$$

The resulting *inner-outer iteration* based on the iterative formulas given in (3.2) and (3.5) is presented in Algorithm 1. The parameters η and τ are the inner and outer tolerances, respectively, given in the two inequalities above. We note that for the purpose of illustrating how the computation can be efficiently done, the roles of $\mathbf{x}$ and $\mathbf{y}$ are altered from the notation used in the text.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 1. Basic inner-outer iteration.

```
Input: $P, \alpha, \beta, \tau, \eta, \mathbf{v}$
Output: $\mathbf{x}$
1: $\mathbf{x} \leftarrow \mathbf{v}$
2: $\mathbf{y} \leftarrow P\mathbf{x}$
3: while $\|\alpha\mathbf{y} + (1-\alpha)\mathbf{v} - \mathbf{x}\|_1 \geq \tau$
4:     $\mathbf{f} \leftarrow (\alpha - \beta)\mathbf{y} + (1-\alpha)\mathbf{v}$
5:     repeat
6:         $\mathbf{x} \leftarrow \mathbf{f} + \beta\mathbf{y}$
7:         $\mathbf{y} \leftarrow P\mathbf{x}$
8:     until $\|\mathbf{f} + \beta\mathbf{y} - \mathbf{x}\|_1 < \eta$
9: end while
10: $\mathbf{x} \leftarrow \alpha\mathbf{y} + (1-\alpha)\mathbf{v}$
```

Lines 1 and 2 of Algorithm 1 initialize $\mathbf{x} = \mathbf{v}$ and $\mathbf{y} = P\mathbf{x}$. The right-hand side of (3.2), defined as $\mathbf{f}$ in (3.3), is computed in line 4. The inner iteration described in (3.5) is defined in lines 5–8. We use a repeat-until clause here since we require at least one inner iteration to be performed. Upon exit from the inner loop, the vector $\mathbf{x}$ holds the value denoted by $\mathbf{y}_{j+1}$ in (3.5).

To stop the algorithm, line 3 tests the residual of the outer linear system (3.1), and for the inner system (3.4), the stopping criterion is in line 8. The vector $(I - \beta P)\mathbf{y}_{j+1}$ is given in the algorithm by $\mathbf{x} - \beta\mathbf{y}$, since $\mathbf{x}$ at this point holds $\mathbf{y}_{j+1}$ and $\mathbf{y}$ admits the value of $P\mathbf{x}$ in line 7 (and initially in line 2). Line 10 uses the computed values to take a single power method step at the end of the algorithm: since $\mathbf{y} = P\mathbf{x}$ is already computed, we might as well use it. (See [45] for possible benefits of using a power method step at the end of a given algorithm.) Upon exit from the algorithm, $\mathbf{x}$ holds the desired approximation to the PageRank vector.

4. Convergence analysis.

The damping parameter α is assumed to be given as part of the model, and τ is a value typically provided by the user. Thus the challenge is to determine values of β and η that will accelerate the computation.

We note that we do not consider $\beta = 0$ or $\beta = \alpha$, since setting β to be one of these values would simply lead back to the power method. If $\beta = 0$, then the inner iteration is meaningless regardless of η, and the outer iteration is equivalent to the power method. If $\beta = \alpha$, then the number of outer iterations is small (one iteration if $\eta = \tau$), and it would be the inner iterations this time that exactly reproduce power iterations.

A value of η very close to zero (that is, very strict) may result in spending a long computational time performing inner iterations, just to compute a single outer iterate at a time. This may lead to slow convergence overall. Setting η very loose, on the other hand, may result in inner iterates that do not sufficiently approximate the exact solution of (3.4), and hence do not yield sufficient progress towards the desired PageRank vector.

For β, the outer iterations converge faster if β is close to α, but the inner iterations converge faster if β is close to zero. This is intuitively clear, and we will provide analytical justification in section 4.2.

Our goal is, then, to find intermediate choices of β and η that reduce the overall computational work as much as possible.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

4.1. Convergence of the outer iterations. Let us examine the outer splitting. If we write $A(\alpha) = \alpha P + (1-\alpha)\mathbf{v}\mathbf{e}^T$ and $A(\beta) = \beta P + (1-\beta)\mathbf{v}\mathbf{e}^T$ and notice that $A'(\alpha) = A'(\beta) = P - \mathbf{v}\mathbf{e}^T$ (clearly, by linearity the derivative does not depend on the damping factor), we can write

$$A(\alpha) = A(\beta) + (\alpha - \beta)A'.$$

Thus we can interpret the outer iteration as a procedure in which we locally solve the PageRank problem for β rather than for α, and correct for the remainder, which is a scalar multiple of the derivative of A. The convergence analysis that follows explains why this methodology is effective.

Since $M_O = I - \beta P$ is a diagonally dominant M-matrix, convergence is guaranteed by known results for regular splittings [42, Theorem 3.32]. Asymptotic convergence analysis can be performed by observing that if λ_i is an eigenvalue of P, then

$$\mu_i = \frac{(\alpha-\beta)\lambda_i}{1-\beta\lambda_i} \tag{4.1}$$

is an eigenvalue of T_O. Since $|\lambda_i| \le 1$, we get

$$|\mu_i| = \left|\frac{(\alpha-\beta)\lambda_i}{1-\beta\lambda_i}\right| \le \frac{(\alpha-\beta)|\lambda_i|}{1-\beta|\lambda_i|} = \frac{\alpha-\beta}{|\lambda_i|^{-1}-\beta} \le \frac{\alpha-\beta}{1-\beta},$$

with equality holding for $\lambda_1 = 1$, so $\rho(T_O) = \frac{\alpha-\beta}{1-\beta} < 1$.

We note that μ_i in (4.1) indicates that all modes of the outer error decay if a good choice of β is made. The decay is effective also for terms associated with the eigenvalues of P on the unit circle, as can be seen by evaluating $|\frac{(\alpha-\beta)e^{\imath\theta}}{1-\beta e^{\imath\theta}}|$. For example, in the case $\theta = \pi$, which corresponds to the situation of an eigenvalue -1 (this happens if the graph has a cycle of length 2, which commonly occurs), the factor $\frac{\alpha-\beta}{1+\beta}$ could be quite small; for instance, if $\alpha \approx 1$ and $\beta = 1/2$, this factor is approximately 1/3, which guarantees a rapid reduction of error associated with this eigenvalue.

Similarly to what can be done in the case of applying the power method for computing PageRank [6], we can obtain strong convergence results that hold for each iteration and not only asymptotically.

LEMMA 4.1. *Given $0 < \alpha < 1$, if the inner iterations are solved exactly, the scheme converges for any $0 < \beta < \alpha$. Furthermore,*

$$\|\mathbf{x}_{k+1} - \mathbf{x}\|_1 \le \frac{\alpha-\beta}{1-\beta}\ \|\mathbf{x}_k - \mathbf{x}\|_1$$

and

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_1 \le \frac{\alpha-\beta}{1-\beta}\ \|\mathbf{x}_k - \mathbf{x}_{k-1}\|_1,$$

and hence the contraction factor $\frac{\alpha-\beta}{1-\beta}$ indicates that the closer β is to α, the faster the outer iterations converge.

Proof. By definition,

$$(I - \beta P)\mathbf{x} = (\alpha - \beta)P\mathbf{x} + (1-\alpha)\mathbf{v}.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Subtracting the above equation from (3.2), we get $\mathbf{x}_{k+1} - \mathbf{x} = T_O(\mathbf{x}_k - \mathbf{x})$, which reads

$$\mathbf{x}_{k+1} - \mathbf{x} = (\alpha - \beta)(I - \beta P)^{-1} P(\mathbf{x}_k - \mathbf{x}).$$

Taking 1-norms and using the triangular inequality, we have

$$\|\mathbf{x}_{k+1} - \mathbf{x}\|_1 \leq (\alpha - \beta)\|(I - \beta P)^{-1}\|_1 \|P\|_1 \|\mathbf{x}_k - \mathbf{x}\|_1.$$

The matrix P is column stochastic. Also, $I - \beta P$ is a diagonally dominant M-matrix, and hence its inverse is nonnegative. It follows that $\|P\|_1 = 1$ and $\|(I - \beta P)^{-1}\|_1 = \frac{1}{1-\beta}$. (The latter result can also be obtained by using the technique of [25, Lemma 5].) This gives the first inequality in the statement of the lemma.

To obtain the second inequality, set (3.2) for $k - 1$, subtract it from the same equation with k, and take norms in the same fashion.

Denote the contraction factor by $g_\alpha(\beta) = \frac{\alpha - \beta}{1 - \beta}$; then $g'_\alpha(\beta) = \frac{\alpha - 1}{(1-\beta)^2}$ is negative for $\alpha < 1$, and hence $g_\alpha(\beta)$ is monotonically decreasing. Therefore, the closer β is to α the faster the outer iterations converge, as claimed. $\square$

4.2. Convergence of the inner iterations. Let us consider the rate of convergence of the inner iterations (3.5) and their dependence on the parameters α and β. From (3.9) it trivially follows that $\rho(T_I) = \beta$, and hence asymptotically, the error is reduced by a factor of approximately β in each inner iteration. But as we showed in the convergence analysis for the outer iteration, we can show convergence (with a contraction factor β in this case) that is not only asymptotic. The proof of the following lemma is straightforward and follows by taking 1-norms, applying the triangular inequality, and using $\|P\|_1 = 1$.

LEMMA 4.2. *For the inner iterations, if* $\mathbf{y}$ *is the solution of the inner system* (3.4), *then*

$$\|\mathbf{y}_{j+1} - \mathbf{y}\|_1 \leq \beta \|\mathbf{y}_j - \mathbf{y}\|_1$$

and

$$\|\mathbf{y}_{j+1} - \mathbf{y}_j\|_1 \leq \beta \|\mathbf{y}_j - \mathbf{y}_{j-1}\|_1.$$

What is also interesting is the connection between a given inner iterate and the solution of the *outer* problem (i.e., the PageRank vector). Define

$$\mathbf{c}_j = \mathbf{y}_j - \mathbf{x}$$

to be this error. By the linear system formulation, $\mathbf{x} = \alpha P\mathbf{x} + (1 - \alpha)\mathbf{v}$. Subtracting this from (3.5) and using the definition of $\mathbf{e}_k$ in (4.2) we get

$$\begin{aligned}\mathbf{c}_{j+1} &= \beta P\mathbf{y}_j - \beta P\mathbf{x}_k + \alpha P\mathbf{e}_k \\ &= \beta P\mathbf{c}_j + (\alpha - \beta)P\mathbf{e}_k.\end{aligned}$$

Since $\mathbf{c}_0 = \mathbf{e}_k$, the error at the jth iterate can be expressed as a jth degree polynomial in βP, multiplied by $\mathbf{c}_0$. It also follows (either from the above equality or from Lemma 4.2) that

$$\|\mathbf{c}_{j+1} - \mathbf{c}_j\|_1 \leq \beta \|\mathbf{c}_j - \mathbf{c}_{j-1}\|_1.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

As we have mentioned, in practice we will want to accelerate convergence by solving the inner iteration inexactly. Analysis of inner-outer stationary schemes for solving linear systems, giving insight on possible choices of parameters and estimates of the overall computational work, has been presented in a number of papers or books, of which we mention [16, 17, 20, 23]. Analysis of inner-outer iterations for Krylov subspace solvers can be found in [38]. Below we derive bounds on convergence of the inexact iteration. We do so by using difference inequalities, adopting an approach similar to the one taken in [16, section 2]. Given a linear system $Bx = g$ and a splitting $B = M - N$, consider a stationary scheme based on inexact solves as follows:

$$M\mathbf{x}_{k+1} = N\mathbf{x}_k + \mathbf{g} + \mathbf{s}_k,$$

where $\mathbf{s}_k$ denotes a nonzero residual that measures the inexactness in the computation. Let

$$\mathbf{e}_k = \mathbf{x}_k - \mathbf{x} \tag{4.2}$$

be the error in the kth iteration. Then

$$\mathbf{e}_{k+1} = T\mathbf{e}_k + M^{-1}\mathbf{s}_k, \tag{4.3}$$

where $T = M^{-1}N$. Suppose

$$\|\mathbf{s}_k\| \le \zeta\|\mathbf{e}_k - \mathbf{e}_{k-1}\| \tag{4.4}$$

for some ζ. Note that the bound in (4.4) is computable since $\mathbf{e}_k - \mathbf{e}_{k-1} = \mathbf{x}_k - \mathbf{x}_{k-1}$. In our algorithms we terminate the iteration by examining the residual rather than the difference of errors.

Combining (4.3) and (4.4) we have

$$\|\mathbf{e}_{k+1}\| \le \|T\|\,\|\mathbf{e}_k\| + \zeta\|M^{-1}\|\,\|\mathbf{e}_k - \mathbf{e}_{k-1}\|.$$

Defining $\rho = \|T\|$ and $\sigma = \zeta\|M^{-1}\|$ and applying the triangular inequality, the resulting inequality

$$\|\mathbf{e}_{k+1}\| \le \rho\|\mathbf{e}_k\| + \sigma(\|\mathbf{e}_k\| + \|\mathbf{e}_{k-1}\|)$$

involves a three-term relation. Let ν_k be the solution of the recurrence relation

$$\nu_{k+1} = \rho\nu_k + \sigma(\nu_k + \nu_{k-1}),$$

with $\nu_1 = \|\mathbf{e}_1\|$ and $\nu_2 = \|\mathbf{e}_2\|$. Then

$$\|\mathbf{e}_k\| \le \nu_k = a_1\xi_+^k + a_2\xi_-^k,$$

where

$$\xi_\pm = \frac{\rho+\sigma}{2}\left(1 \pm \sqrt{1 + \frac{4\sigma}{(\rho+\sigma)^2}}\right)$$

and

$$a_1 = \frac{2(\nu_2 - \nu_1\xi_-)}{(\rho+\sigma)^2 s(s+1)}\ , \qquad a_2 = \frac{2(\nu_2 - \nu_1\xi_+)}{(\rho+\sigma)^2 s(s-1)}\ ,$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

with $s = \sqrt{1 + 4\sigma/(\rho+\sigma)^2}$. Since $\xi_- < 0$, we have that $a_1 \geq 0$ and $\xi_+ > |\xi_-|$. It follows that

$$\|\mathbf{e}_k\| \leq a_1 \xi_+^k + |a_2||\xi_-|^k \leq (a_1 + |a_2|)\xi_+^k. \tag{4.5}$$

Hence we have a bound of the type $\|\mathbf{e}_k\| \leq \vartheta \xi_+^k$, where ϑ is independent of k. The bound (4.5) motivates us to find ζ such that $\xi_+ < 1$. Using 1-norms, since $\|M_O^{-1}\|_1 = \frac{1}{1-\beta}$ and $\|T_O\|_1 = \frac{\alpha-\beta}{1-\beta}$, we can set $\rho = \frac{\alpha-\beta}{1-\beta}$ and $\sigma = \frac{\zeta}{1-\beta}$. Defining $\gamma = \alpha - \beta$ and using $\sqrt{1+x} \lesssim 1 + \frac{x}{2}$ for $x \ll 1$, we obtain

$$\xi_+ = \frac{\rho+\sigma}{2}\left(1 + \sqrt{1 + \frac{4\sigma}{(\rho+\sigma)^2}}\right) \lesssim \frac{\gamma+\zeta}{1-\beta} + \frac{\zeta}{\gamma+\zeta}.$$

Requiring $\xi_+ < 1$ yields a quadratic inequality for which it is possible to see that the closer α is to 1, the smaller ζ must be.

In Figure 4.1 we experimentally examine how ξ_+ relates to β and ζ. The graphs show that convergence is expected for a large range of values of these parameters. It is worth stressing that our analysis provides only sufficient (not necessary) conditions for convergence.

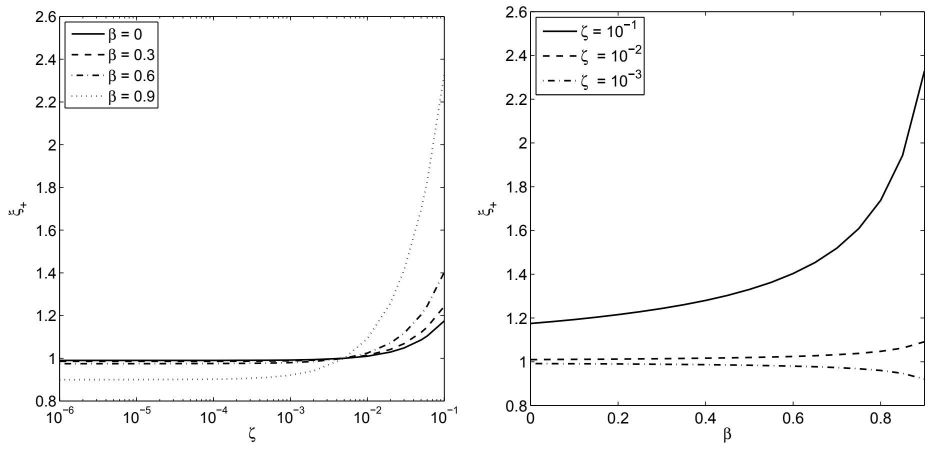

FIG. 4.1. *ξ_+ for $\alpha = 0.99$, and various values of ζ and β. A sufficient condition for the algorithm to converge is $\xi_+ < 1$.*

5. Inner-outer acceleration and preconditioning.

The simple mechanism of the inner-outer iterations allows for seamlessly incorporating it into other solvers as either an acceleration scheme or a preconditioning technique. In this section we discuss how inner-outer iterations can be combined with the power method, the Gauss–Seidel scheme, and Krylov subspace solvers.

5.1. Inner-outer power iterations.

The convergence analysis performed in section 4 indicates that the outer iteration yields convergence rates that depend strongly on β, and certain choices of this parameter can yield significant gains in the initial iterates, compared to the power method. When $\mathbf{y}_0 = \mathbf{x}_k$ is sufficiently close to $\mathbf{x}$, the inner iterations rapidly converge, and the scheme more closely resembles the power method. Thus we can incorporate the following improvement to the power method: apply the inner-outer scheme, and once the inner iterations start converging quickly, switch back to the power method.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

The modified scheme is presented in Algorithm 2. We replace the *repeat-while* inner-iteration with a *for* loop, so as to count the number of inner iterations. The "power($\alpha\mathbf{y} + (1-\alpha)\mathbf{v}$)" clause in line 9 means to apply the power method with $\alpha\mathbf{y} + (1-\alpha)\mathbf{v}$ as an initial guess, until convergence. The value of i_m is small and determines the point, in terms of inner iteration count, where we switch to the power method. If $i_m = 1$, then switching to the power method is an optimization for subsequent iterations that saves the need to compute $\mathbf{f}$ and check the stopping criterion in line 8 of Algorithm 1. This optimization saves touching an extra vector in memory and a 1-norm computation for those iterations after switching to the power method. In our numerical experiments we adopt this version of the accelerated algorithm, i.e., we take $i_m = 1$.

Algorithm 2. Inner-outer power iterations.

```
Input: $P, \alpha, \beta, \tau, \eta, \mathbf{v}, i_m$
Output: $\mathbf{x}$
1: $\mathbf{x} \leftarrow \mathbf{v}$
2: $\mathbf{y} \leftarrow P\mathbf{x}$
3: while $\|\alpha\mathbf{y} + (1-\alpha)\mathbf{v} - \mathbf{x}\|_1 \geq \tau$
4:       $\mathbf{f} \leftarrow (\alpha - \beta)\mathbf{y} + (1-\alpha)\mathbf{v}$
5:       for $i = 1, \ldots$  repeat
6:             $\mathbf{x} \leftarrow \mathbf{f} + \beta\mathbf{y}$
7:             $\mathbf{y} \leftarrow P\mathbf{x}$
8:       until $\|\mathbf{f} + \beta\mathbf{y} - \mathbf{x}\|_1 < \eta$
9:       if $i \leq i_m$,  $\mathbf{x}$=power($\alpha\mathbf{y} + (1-\alpha)\mathbf{v}$); return
10: end while
11: $\mathbf{x} \leftarrow \alpha\mathbf{y} + (1-\alpha)\mathbf{v}$
```

It is useful to consider the computational cost involved in applying this scheme. Let us ignore operations between scalars. The quantity $\alpha\mathbf{y} + (1-\alpha)\mathbf{v}$ is computed once per outer iteration and is used in lines 3 and 4 of Algorithm 2. It is also used upon entering line 9 to execute the power method until convergence. The vector $\mathbf{f} + \beta\mathbf{y}$ is computed once per inner iteration and is used in lines 6 and 8. Once the above mentioned vectors have been computed, lines 3 and 4 involve two additional SAXPY operations and a 1-norm computation. The operations inside the inner loop include a computation of $P\mathbf{x}$ (line 7 in Algorithm 2), which typically accounts for the most computationally costly component. The computation of $P\mathbf{x}$ involves more than a mere matrix-vector product, because, for dealing with dangling nodes, our matrix is in fact of the form $P = \bar{P} + \mathbf{v}\mathbf{d}^T$, as explained in section 2. We note that in large scale settings, it may be helpful to perform an extra normalization operation, to avoid loss of accuracy; see [45].

5.2. Inner-outer Gauss–Seidel iterations. The performance of Gauss–Seidel applied to the PageRank linear system $(I - \alpha P)\mathbf{x} = (1-\alpha)\mathbf{v}$ is considered excellent, given its modest memory requirements. It often converges in roughly half the number of power method iterations. However, from a practical point of view, two pitfalls of the method are that it requires the matrix P by rows (i.e., the graph by in-edges) and it does not parallelize well.

We can accelerate Gauss–Seidel using inner-outer iterations as follows. Our Gauss–Seidel codes implement the dangling correction to the matrix $\bar{P}$ implicitly

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

and match those in the `law` toolkit [43]. To convert the inner-outer method to use Gauss–Seidel, we replace the inner Richardson iteration (3.5) with a Gauss–Seidel iteration. In our pseudocode, presented in Algorithm 3, the gssweep($\mathbf{x}, A, \mathbf{b}$) function implements $\mathbf{x}_{k+1} = M_{GS}^{-1}(N_{GS}\mathbf{x}_k + \mathbf{b})$ in-place for a Gauss–Seidel splitting of the matrix A. The Gauss–Seidel-pr function in line 7 of the algorithm refers to the standard "Gauss–Seidel for PageRank" scheme. This algorithm requires only two vectors of storage because $\mathbf{x}$ is always updated in-place. Details on the performance of this algorithm are provided in section 6.

Algorithm 3. Inner-outer/Gauss–Seidel iteration.

```
Input: $P, \alpha, \beta, \tau, \eta, \mathbf{v}$
Output: $\mathbf{x}$
1: $\mathbf{x} \leftarrow \mathbf{v}$
2: $\mathbf{y} \leftarrow P\mathbf{x}$
3: while $\|\alpha\mathbf{y} + (1-\alpha)\mathbf{v} - \mathbf{x}\|_1 \geq \tau$
4:      $\mathbf{f} \leftarrow (\alpha-\beta)\mathbf{y} + (1-\alpha)\mathbf{v}$
5:      for $i = 1, \ldots$
6:           $\mathbf{x}, \delta \leftarrow$ gssweep($\mathbf{x}, I - \beta P, \mathbf{f}$) $\{\delta = \|\mathbf{x}_{i+1} - \mathbf{x}_i\|_1\}$
7:      until $\delta < \eta$
8:      if i=1, Gauss–Seidel-pr($\mathbf{x}, I - \alpha P, (1-\alpha)\mathbf{v}$); break
9:      $\mathbf{y} \leftarrow P\mathbf{x}$
10: end repeat
```

5.3. Preconditioning for nonstationary schemes. Thus far we have examined the inner-outer algorithm as a stationary iteration. In this section we switch our viewpoint and examine it as a preconditioner for a nonstationary iteration. As such, we will be mainly interested in how well the eigenvalues of the preconditioned matrix are clustered.

Consider an approximation of $(I - \beta\bar{P})^{-1}$ as a preconditioner for the PageRank linear system $(I - \alpha\bar{P})$. Gleich, Zhukov, and Berkhin [18] and Del Corso, Gullí, and Romani [13] examined the behavior of Krylov subspace methods on the system

$$(I - \alpha\bar{P})\mathbf{y} = (1-\alpha)\mathbf{v}, \tag{5.1}$$

and concluded, as is generally expected, that preconditioning is essential for the linear system formulation of the PageRank problem. Their preconditioners were incomplete factorizations or factorizations of diagonal blocks of the matrix, both of which modify the matrix data structure.

The system in (5.1) is different from the system in (3.1) because we use $\bar{P}$ instead of P. But the solutions of the two systems are proportional if $\mathbf{u} = \mathbf{v}$. (Recall from section 2 that $\mathbf{u}$ is the dangling nodes vector and $\mathbf{v}$ is the personalization vector.) In case of equality, we have $\mathbf{x} = \mathbf{y}/\|\mathbf{y}\|_1$. A proof is given in [30, Theorem 7.2.1], and for completeness we provide the main details below. Indeed, $(I - \alpha P)\mathbf{x} = [I - \alpha(\bar{P} + \mathbf{v}\mathbf{d}^T)]\mathbf{x}$, and therefore $(I - \alpha\bar{P})\mathbf{x} = (1 - \alpha + \gamma)\mathbf{v}$, where $\gamma = \alpha\mathbf{d}^T\mathbf{x}$. Suppose $\varphi = \frac{1-\alpha}{1-\alpha+\gamma}$. Then, rescaling the above system by φ, we get

$$(I - \alpha\bar{P})\underbrace{(\varphi\mathbf{x})}_{\mathbf{y}} = (1-\alpha)\mathbf{v}.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Since $I - \alpha\bar{P}$ is nonsingular, we must have a unique solution. We have $\mathbf{y} = \varphi\mathbf{x}$, and hence $\mathbf{y}$ must be positive. Using $\mathbf{e}^T\mathbf{x} = 1$ it follows that $\varphi = \mathbf{e}^T\mathbf{y} = \|\mathbf{y}\|_1$, and hence $\mathbf{x} = \mathbf{y}/\|\mathbf{y}\|_1$ as claimed.

We switch to the alternate formulation with $\bar{P}$ for two primary reasons: it converged more often in our experience (we have no insight into why this occurs); and the behavior of Krylov methods has been empirically studied more often on this formulation of the system. However, this setting is more restrictive than what has been discussed and analyzed in previous sections, and it remains to be tested to what extent it applies for the general case where the dangling node vector is not related to the personalization vector. That said, the PageRank problem formulation with $\bar{P}$, as in (5.1), has received considerable attention in the literature [22, 32]. We note also that there are models such as GeneRank [33], in which the initial matrix is assumed to be column stochastic and the formulation is equivalent to the one we consider in this section.

For β near α, the preconditioned system is as difficult to solve as the original linear system. We thus consider a Neumann series approximation, which is practically equivalent to a Richardson approach as in the basic inner-outer iteration:

$$(I - \beta\bar{P})^{-1} \approx I + \beta\bar{P} + (\beta\bar{P})^2 + \cdots + (\beta\bar{P})^m.$$

Since $\beta < 1$ and $\|\bar{P}\|_1 = 1$, this approximation converges as $m \to \infty$, and it gives rise to an implementation that uses only matrix-vector multiplies with $\bar{P}$ and avoids costly (or even impossible) modification of the matrix structure. Details on the performance of this approach are provided in section 6.

6. Numerical experiments.

We have implemented the power method, inner-outer iteration (Algorithms 1 and 2), Gauss–Seidel method, and Gauss–Seidel inner-outer method (Algorithm 3) in pure MATLAB, MATLAB `mex`, and C++.

Using the data enumerated in Table 6.1, we evaluated these implementations in a wide range of experimental situations. The initial guess, the dangling node vector $\mathbf{u}$, and the teleportation distribution $\mathbf{v}$ for every method is the uniform distribution, $\mathbf{x}_0 = (1/n)\mathbf{e}$, where $\mathbf{e}$ is a vector of all ones. We use these specific choices since they are commonly used. When we state a speedup, we use relative percentage gain

$$\frac{v_r - v_t}{v_r} \cdot 100\% \ , \tag{6.1}$$

where v_r is a reference performance and v_t is a test performance. The reference performance is either the power method or the Gauss–Seidel method. All solution vectors satisfy the residual tolerance

$$\|\alpha P\mathbf{x}_k + (1-\alpha)\mathbf{v} - \mathbf{x}_k\|_1 < \tau,$$

where τ is specified in the experiment description. Thus we are able to directly compare all the methods in terms of total work and time. Parallel, large scale (`arabic-2005`, `sk-2005`, and `uk-2007`), and C++ tests were run on an 8-core (4 dual-core chips) 2.8 GHz Opteron 8220 computer with 128 GB of RAM. MATLAB `mex` tests were run on a 4-core (2 dual-core chips) 3.0 GHz Intel Xeon 5160 computer with 16 GB of RAM.

6.1. Implementations.

Due to the great size of Web link graphs, matrix storage is a primary consideration in the design of PageRank solvers. Web crawlers

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Table 6.1
Dimensions and number of nonzeros of a few test matrices.

Name	Size	Nonzeros	Average nonzeros per row
`ubc-cs-2006`	51,681	673,010	13.0
`ubc-2006`	339,147	4,203,811	12.4
`eu-2005`	862,664	19,235,140	22.3
`in-2004`	1,382,908	16,917,053	12.2
`wb-edu`	9,845,725	57,156,537	5.8
`arabic-2005`	22,744,080	639,999,458	28.1
`sk-2005`	50,636,154	1,949,412,601	38.5
`uk-2007`	105,896,555	3,738,733,648	35.3

collect graphs by storing the out-edges of each node and thus solvers that work with only out-edges are the most natural. Storing out-edges is equivalent to storing the nonzero structure $\bar{P}$ by columns. Furthermore, storing the out-edges makes the simple random-walk normalization of the matrix entries possible by storing only the matrix nonzero structure. That is, it is sufficient to store only the indices of adjacent vertices, equivalently the nonzero row numbers for a column, rather than store integer or floating point nonzero values as well.

Both the power method and the inner-outer iteration work only with out-edges because they require only (stochastic) matrix-vector products. In contrast, standard Gauss–Seidel-based methods require access to the matrix $\bar{P}$ by rows or the in-edges of the graph. To get rows of the matrix $\bar{P}$ when storing just the matrix nonzero structure requires an extra length-n vector to store the out-degree counts for each node. While this vector could be a 4-byte integer for all our matrices, we observed a performance advantage when storing the inverse out-degree in a vector of 8-byte doubles. We also observed a slight performance increase when storing the graph by in-edges for the power method. This advantage was more pronounced in the multicore algorithms and we discuss why in section 6.4. We implemented our C++ codes to work with the matrix stored by either out- or in-edges and we always state which version we used for a particular performance time. The C++ codes work only with Web graphs compressed into `bvgraph` data structures [9]. Our experience indicates that manipulating these data structures causes a 2–4x slowdown, but yields enormous memory savings (e.g., running the power method on `uk-2007` takes 2.9 GB of RAM). Due to speed considerations, our MATLAB implementations store $\bar{P}^T$ instead of $\bar{P}$ — this choice corresponds to storing the graph by in-edges.

6.2. Inner-outer parameters. Results for the inner-outer method applied to the `in-2004` matrix are presented in Figure 6.1. On the left-hand graph we see that, for too loose an inner tolerance η, there is only a narrow range of values of β for which the inner-outer method converges much more quickly than the power method; see the convergence graph for $\eta = 0.1$. When η is very strict the overall computational work is large for almost all values of β, due to a large number of inner iterations; see the graph for $\eta = 10^{-5}$. Significant gains are observed for *moderate* values of η; see, for example, the graph for $\eta = 0.01$. In this case, the performance of the scheme is not sensitive to the choice of β. We have observed a similar behavior for our other test matrices: choosing $\eta \approx 10^{-2}$ and $0.4 \lesssim \beta \lesssim 0.8$ has in most cases led to accelerated convergence.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

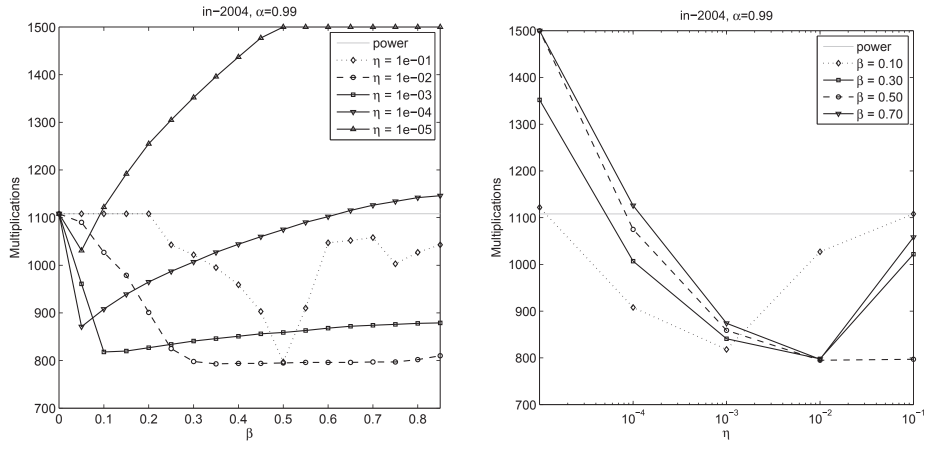

FIG. 6.1. *Total number of matrix-vector products required for convergence of the inner-outer scheme, for the* `in-2004` *matrix. ($\tau = 10^{-7}$, $\alpha = 0.99$, β, and η varied. The iteration limit is* 1500 *and causes the ceiling on the left figure.)*

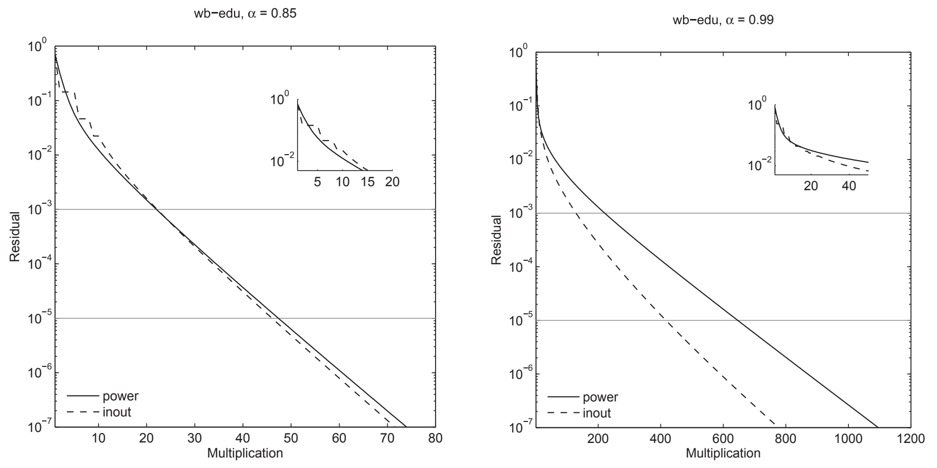

FIG. 6.2. *Convergence of the computation for the* $9,845,725 \times 9,845,725$ `wb-edu` *matrix ($\tau = 10^{-7}$, $\beta = 0.5$, and $\eta = 10^{-2}$ in the inner-outer method). The interior figures highlight the first few iterations.*

The right-hand graph in Figure 6.1 shows a similar behavior for various fixed values of β, with η varying. The conclusion is that moderate values of both β and η often significantly reduce the overall computational work and are fairly insensitive to small perturbations. Our experiments use the choice $\eta = 10^{-2}$ and $\beta = 0.5$.

In Figure 6.2 we plot the norms of the residuals for both methods run on the large `wb-edu` matrix for two values of α. The gains in the inner-outer method are often made in the *initial* iterates, and the gains are most significant when $\alpha = 0.99$ and the outer tolerance is loose. This can be observed in Figure 6.2, where for $\tau = 10^{-3}$ we have a relative gain of 41%. When the outer tolerance is stricter (10^{-7}), the inner-outer scheme achieves a relative gain of less than 3% for $\alpha = 0.85$ (72 matrix-vector products compared to 74 for the power method), which is marginal. On the other

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Table 6.2

Total number of matrix-vector products and wall-clock time required for convergence to three different outer tolerances τ, and the corresponding relative gains defined by (6.1). *The parameters used here are $\alpha = 0.99$, $\beta = 0.5$, $\eta = 10^{-2}$. For the first five graphs, we used the MATLAB* `mex` *codes, and for the final three large graphs we used the C++ codes, where the graph is stored by out-edges and the times refer to the performance of an 8-core parallel code.*

Tol.	Graph	Work (mults.)			Time (secs.)		
		Power	In/out	Gain	Power	In/out	Gain
10^{-3}	`ubc-cs-2006`	226	141	37.6%	1.9	1.2	35.2%
	`ubc`	242	141	41.7%	13.6	8.3	38.4%
	`in-2004`	232	129	44.4%	51.1	30.4	40.5%
	`eu-2005`	149	150	−0.7%	26.9	28.3	−5.3%
	`wb-edu`	221	130	41.2%	291.2	184.6	36.6%
	`arabic-2005`	213	139	34.7%	779.2	502.5	35.5%
	`sk-2005`	156	144	7.7%	1718.2	1595.9	7.1%
	`uk-2007`	145	125	13.8%	2802.0	2359.3	15.8%
10^{-5}	`ubc-cs-2006`	574	432	24.7%	4.7	3.6	22.9%
	`ubc`	676	484	28.4%	37.7	27.8	26.2%
	`in-2004`	657	428	34.9%	144.3	97.5	32.4%
	`eu-2005`	499	476	4.6%	89.3	87.4	2.1%
	`wb-edu`	647	417	35.5%	850.6	572.0	32.8%
	`arabic-2005`	638	466	27.0%	2333.5	1670.0	28.4%
	`sk-2005`	523	460	12.0%	5729.0	5077.1	11.4%
	`uk-2007`	531	463	12.8%	10225.8	8661.9	15.3%
10^{-7}	`ubc-cs-2006`	986	815	17.3%	8.0	6.8	15.4%
	`ubc`	1121	856	23.6%	62.5	49.0	21.6%
	`in-2004`	1108	795	28.2%	243.1	179.8	26.0%
	`eu-2005`	896	814	9.2%	159.9	148.6	7.1%
	`wb-edu`	1096	777	29.1%	1442.9	1059.0	26.6%
	`arabic-2005`	1083	843	22.2%	3958.8	3012.9	23.9%
	`sk-2005`	951	828	12.9%	10393.3	9122.9	12.2%
	`uk-2007`	964	857	11.1%	18559.2	16016.7	13.7%

hand, when $\alpha = 0.99$ the inner-outer stationary method achieves a substantial relative gain of 29%: 319 fewer matrix-vector products than 1096.

From Table 6.2 we can see that for the other matrices the savings range from 9% to 28% for $\tau = 10^{-7}$, and from −1% to 44% for $\tau = 10^{-3}$. The `eu-2005` matrix shows one case where the overhead in the initial inner iterations causes convergence to slow down. Even though it has a one iteration disadvantage, it seems the improved convergence of the power method after the inner iterations accelerates the method as the residual continues to decrease. Comparing the savings in iteration counts to the savings in CPU time confirms that the overhead per iteration introduced by the inner-outer method (section 5) is small compared to the overall computational gain, and thus it pays off.

For all the graphs, the inner iteration counts per iteration decrease monotonically down to a single iteration quickly. For `eu-2005` with $\alpha = 0.99$, it takes 24 inner iterations within 9 outer iterations, until the inner iterates start converging immediately, at which point we switch to the power method. We observed similar performance for the other graphs.

6.3. Inner-outer Gauss–Seidel. We present our comparison for the Gauss–Seidel method in Table 6.3. Rather than matrix-vector multiplications, the results

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

TABLE 6.3
Results from the Gauss–Seidel inner-outer method compared with the Gauss–Seidel method. Total number of Gauss–Seidel sweep iterations (equivalent in work to one matrix-vector multiply) and wall-clock time required for convergence, and the corresponding relative gains defined by (6.1). *The parameters used here are $\beta = 0.5$ and $\eta = 10^{-2}$, and we used MATLAB* `mex` *codes. The convergence tolerance was* 10^{-7}.

α	Graph	Work (sweeps.)			Time (secs.)		
		GS	In/out	Gain	GS	In/out	Gain
0.99	`ubc-cs-2006`	562	492	12.5%	2.9	2.7	7.0%
	`ubc`	566	503	11.1%	19.5	18.0	7.7%
	`in-2004`	473	469	0.8%	65.9	67.3	−2.2%
	`eu-2005`	439	462	−5.2%	56.6	60.4	−6.6%
	`wb-edu`	450	464	−3.1%	357.9	380.0	−6.2%
0.999	`ubc-cs-2006`	4430	3576	19.3%	19.8	16.8	14.9%
	`ubc`	4597	3646	20.7%	141.6	113.8	19.7%
	`in-2004`	3668	3147	14.2%	451.1	391.0	13.3%
	`eu-2005`	3197	3159	1.2%	354.8	352.4	0.7%
	`wb-edu`	3571	3139	12.1%	2532.5	2249.0	11.2%

are measured in sweeps through the matrix. The work for a single sweep is roughly equivalent to a single matrix-vector multiplication. For $\alpha = 0.99$, the inner-outer iteration accelerates only two of our smallest test graphs. Increasing α to 0.999 and using a strict τ shows that the inner-outer method also accelerates Gauss–Seidel-based codes.

We have not invested effort in optimizing the scheme in this case; our experiments are intended only to show that the inner-outer idea is promising in combination with other high-performance PageRank techniques. We believe that an analysis of the sort that we have performed for the Richardson iteration in the previous sections may point out a choice of parameters that could further improve convergence properties for the inner-outer scheme combined with Gauss–Seidel.

6.4. Parallel speedup. Parallel PageRank algorithms take many forms [18, 28, 32, 36]. Our implementation substitutes OpenMP shared memory operations for the linear algebra operations norm, AXPY, and the matrix-vector multiply. We implemented two versions of the parallel code to manipulate the graph stored by out-edges (the natural order) or by in-edges (the Gauss–Seidel order).

Boldi and Vigna's `bvgraph` structure [9] efficiently iterates over the edges emanating from a vertex for a fixed ordering of the nodes. These could be either out- or in-edges, depending on how the graph is stored. To implement the parallel matrix-vector multiply for p processors, we make one pass through the file and store $p-1$ locations in the structure that roughly divide the edges of the graph evenly between p processors. When the graph is stored by out-edges, the serial matrix-vector operation is

```
xi=x[i]/degree(i); for (j in edges of i) { y[j]+=xi; },
```

which has writes to arbitrary and possibly overlapping locations in memory. In the OpenMP version, the update of $\mathbf{y}$ becomes the atomic operation

```
xi=x[i]/degree(i); for (j in edges of i) { atomic(y[j]+=xi); }.
```

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

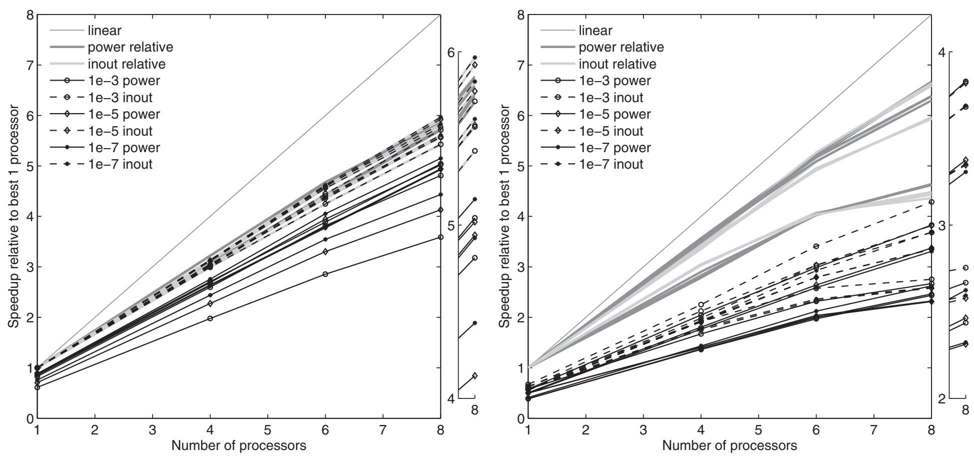

FIG. 6.3. *Parallel scalability for the three large graphs* `arabic-2005`*,* `sk-2005`*, and* `uk-2007` *with the matrix stored by out-edges (left) and in-edges (right). Light gray or dark gray lines are the relative speedup compared with the same algorithm on one processor. Black lines are the true speedup compared with the best single processor code. At the right of each subfigure is a small enlargement of the* 8 *processor results. The parameters were* $\alpha = 0.99, \beta = 0.5$*, and* $\eta = 10^{-2}$*. In this plot, speedup is the ratio* v_r/v_t*.*

When the graph is stored by in-edges, the original serial update works without modification as the processors never write to the same location in the vector $\mathbf{y}$.

We evaluated a few variants of the matrix-vector multiplication when the graph is stored by out-edges and found that the atomic operation variant has similar performance to storing a separate $\mathbf{y}$ vector for each processor and aggregating the vectors at the end of the operation. Variants that replaced separate $\mathbf{y}$ vectors with separate hash tables were slower in our tests.

Figure 6.3 demonstrates the scalability of the codes when storing the graph by out- and in-edges. In the figure, we distinguish between relative and true speedup. Relative speedup is the time on p processors compared with the time on 1 processor for the same algorithm. True speedup is the time on p processors compared with the best time on 1 processor. The higher relative speedup of methods based on the in-edges of the graph (6–7x) compared to out-edge methods (5–6x) demonstrates that in-edge algorithms are more scalable. In light of the atomic operation in out-edge methods, this parallelization difference is not surprising. No consistent differences appear when comparing the relative speedup of the inner-outer method and the power method. Consequently, we assume that these methods parallelize similarly and compare true speedup.

For an out-edge graph, the true speedup and relative speedup are similar. In contrast, the relative speedup for an in-edge graph is much higher than the true speedup. Gauss–Seidel causes this effect, since it is the fastest method on an in-edge graph and so the true speedup of most methods starts at around 0.5, a 50% slowdown. With one exception, the inner-outer methods (dashed lines) all demonstrate a higher true speedup than the power method.

6.5. Preconditioning. We evaluate $(I - \beta\bar{P})^{-1}$ or its Neumann series approximation as a preconditioner (see section 5.3) by examining eigenvalue clustering and matrix-vector multiplies.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

To understand the preconditioner's effect on the convergence of Krylov subspace methods, we look at the clustering of eigenvalues of the matrix $(I-\beta\bar{P})^{-1}(I-\alpha\bar{P})$. Let $\lambda_i,\ i=1,\dots,n$ be the eigenvalues of $\bar{P}$. If we solved the preconditioned iteration defined by $(I-\beta\bar{P})^{-1}$ exactly, then the eigenvalues of the matrix $\bar{P}$ undergo a Möbius transform to the eigenvalues of the preconditioned system

$$p(\lambda)=\frac{1-\alpha\lambda}{1-\beta\lambda}. \tag{6.2}$$

When we precondition with only a finite number of terms, the modification of the eigenvalues is no longer a Möbius transform but the polynomial

$$p_m(\lambda)=(1-\alpha\lambda)(1+\beta\lambda+\cdots+(\beta\lambda)^m). \tag{6.3}$$

Of course as $m\to\infty$, we recover the exact preconditioned system.

To illustrate the spectral properties of the preconditioned system, Figure 6.4 shows the behavior of the preconditioned eigenvalues of two test matrices, for a variety of choices of β and m. In these examples, our preconditioner concentrates the eigenvalues around $\lambda=1$. Solving the system exactly appears unnecessary as we see a strong concentration even with $m=2$ or $m=4$ terms of the Neumann series for the values of β we tested.

Gleich, Zhukov, and Berkhin [18] and Del Corso, Gullí, and Romani [13] explored the performance of preconditioned BiCG-STAB on the PageRank system. We have modified the MATLAB implementation of BiCG-STAB to use the 1-norm of the residual as the stopping criterion.

To compare against the power method and Gauss–Seidel, the normalized solution vectors $\mathbf{x}=\mathbf{y}/\|\mathbf{y}\|_1$ always satisfy

$$\|\alpha P\mathbf{x}+(1-\alpha)\mathbf{v}-\mathbf{x}\|_1\le\tau.$$

This criterion is equivalent to taking one step of the power method and checking the difference in iterations. Consequently, all the solution vectors tested are at least as accurate as the vectors computed in the power method with tolerance τ. In practice, we did not test the previous solution criteria at every iteration and instead modified the MATLAB BiCG-STAB function to terminate the computation when the 1-norm of the residual was less than $(\sqrt{1-\alpha})\tau$. Empirically, using $(\sqrt{1-\alpha})\tau$ as the tolerance to BiCG-STAB yielded $\mathbf{y}$'s that satisfied our actual tolerance criteria without a full rewrite of the residual computations.

BiCG-STAB diverges or breaks down on `in-2004` without preconditioning for $\alpha=0.99$; see Table 6.4. This matches observations by Del Corso, Gullí, and Romani [13] that Krylov methods often have convergence problems. Adding the preconditioner with $m=2$ and $\beta\ge 0.5$ avoids these break-down cases. The remainder of the table shows that preconditioning accelerates convergence in many cases for "reasonable" parameter choices. Among the methods discussed, BiCG-STAB with this preconditioner converges in the fewest number of matrix-vector multiplications on the `in-2004` graph with $\alpha=0.99$ and $\alpha=0.999$. However, the cost per iteration is higher.

6.6. Other applications. The IsoRank algorithm [39] is a heuristic to solve the network alignment problem. Given two graphs, A and B, the goal in the network alignment problem is to find a match for each vertex of graph A in B and vice versa. The resulting matching should maximize the number of cases where i in A is mapped to j in B, i' in A is mapped to j' in B, and both of the edges $(i,i')\in A$ and $(j,j')\in B$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

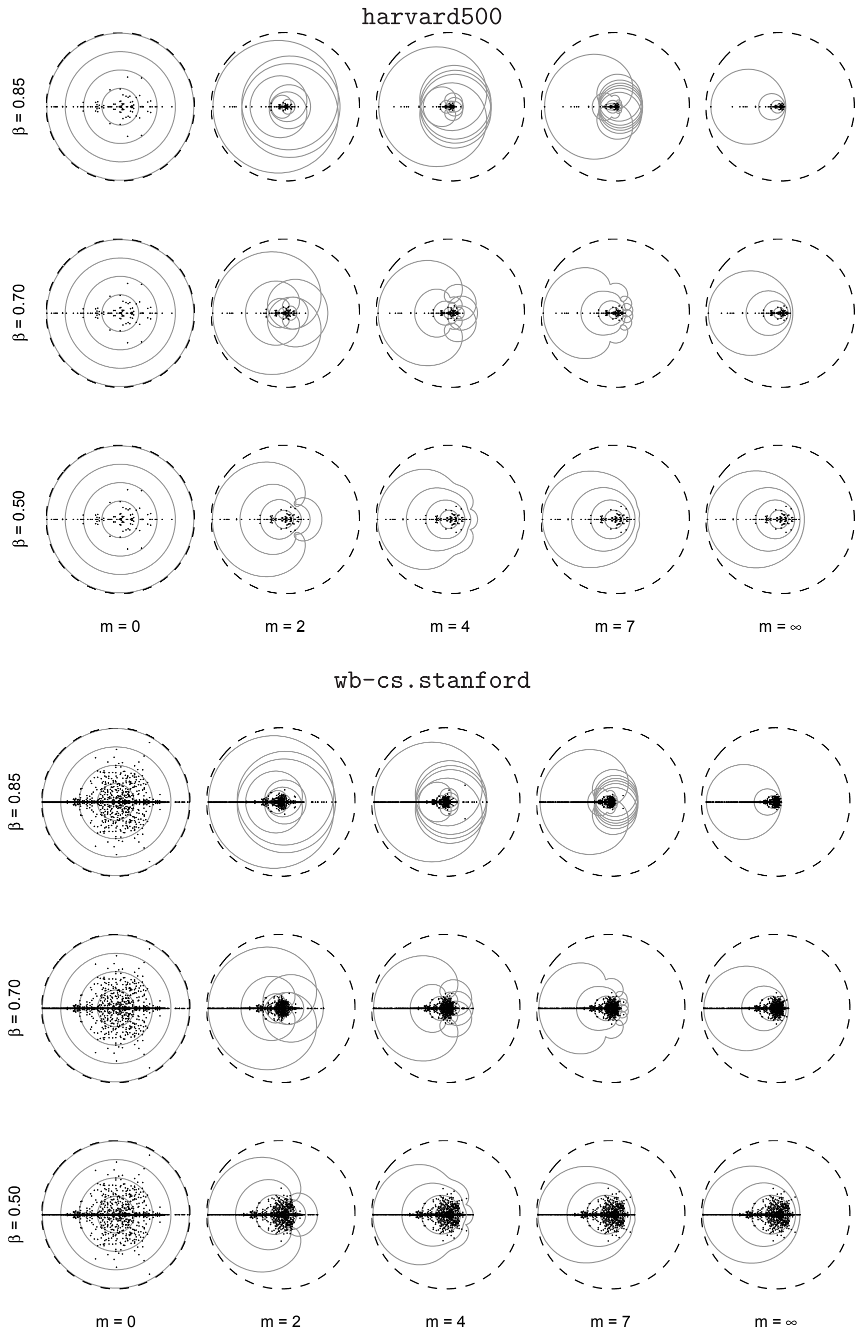

FIG. 6.4. *For the matrices* `harvard500` *and* `wb-cs.stanford` *with* $\alpha = 0.99$, *this figure plots the eigenvalues of the preconditioned matrix* $(I - \beta\bar{P})^{-1}(I - \alpha\bar{P})$ *and approximations based on Neumann series. Each dashed circle encloses a circle of radius* 1 *in the complex plane centered at* $\lambda = 1$, *and hence the scale is the same in each small figure. Gray lines are contours of the function* $p_m(\lambda)$ *defined in* (6.3), *which is the identity matrix for* $m = 0$ *(i.e., no preconditioning) and the exact inverse when* $m = \infty$. *The dots are the eigenvalues of the preconditioned system,* $p_m(\lambda_i)$. *Interlacing contours of* $p_m(\lambda)$ *demonstrate that this function is not* 1–1 *for* $\lambda : |1 - \lambda| \leq 1$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

TABLE 6.4
Matrix-vector products required for BiCG-STAB on `in-2004`*, including preconditioning and residual computations, to converge on the system* $(I - \alpha\bar{P})$ *with preconditioner* $\sum_{k=0}^{m}(\beta P)^k$*. A dash indicates that the method made progress but did not converge to a tolerance of* $(\sqrt{1-\alpha})10^{-7}$ *in the maximum number of iterations required for the power method (*100 *for* $\alpha = 0.85$*,* ≈ 1500 *for* $\alpha = 0.99$*, and* ≈ 15000 *for* $\alpha = 0.999$*), and an* $\times$ *indicates that the method diverged or broke down. When* $m = 0$*, there is no preconditioning and the results are independent of* β*.*

	α											
	0.85				0.99				0.999			
	β				β				β			
m	0.25	0.50	0.75	0.85	0.25	0.50	0.75	0.85	0.25	0.50	0.75	0.85
0	102	102	102	102	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
2	128	88	76	76	1140	672	508	500	$\times$	6276	3972	2772
4	186	120	84	78	1584	786	438	414	$\times$	5178	2358	2112
7	—	207	108	72	2565	1053	621	441	$\times$	9567	2709	1449
25	—	—	—	81	—	—	1809	1026	—	20385	7911	2754

exist. This objective alone is NP-hard. Often there are weights for possible matches (e.g., V_{ji} for i in A and j in B) that should bias the results towards these matchings, and hence the objective also includes a term to maximize these weights.

Let P and Q be the uniform random-walk transition matrices for A and B, respectively. Also, let the weights in V be normalized so that $\mathbf{e}^T V \mathbf{e} = 1$ and $V_{ij} \geq 0$. IsoRank uses the PageRank vector

$$\mathbf{x} = \alpha(P \otimes Q)\mathbf{x} + (1-\alpha)\mathbf{v},$$

where the teleportation vector $\mathbf{v} = \text{vec}(V)$ encodes the weights and α indicates how much emphasis to place on matches using the weights' information. Thus the IsoRank algorithm is a case when $\mathbf{v}$ is not uniform, and α has a more concrete meaning. For a protein-matching problem, it is observed experimentally in [39] that values of α between 0.7 and 0.95 yield good results.

We look at a case when A is the 2-core of the undirected graph of subject headings from the Library of Congress [41] (abbreviated LCSH-2) and B is the 3-core of the undirected Wikipedia category structure [44] (abbreviated WC-3). One of the authors previously used these datasets in analyzing the actual matches in a slightly different setting [3]. The size of these datasets is reported in Table 6.5. For this application, the weights come from a text-matching procedure on the labels of the two graphs.

TABLE 6.5
The size of non-Web datasets. The product graph is never formed explicitly.

Dataset	Size	Nonzeros
LCSH-2	59,849	227,464
WC-3	70,509	403,960
Product graph	4,219,893,141	91,886,357,440

In this experiment, we do not investigate all the issues involved in using a heuristic to an NP-hard problem and focus on the performance of the inner-outer algorithm in a non-Web ranking context. Without any parameter optimization (i.e., using $\beta = 0.5$ and $\eta = 10^{-2}$), the inner-outer scheme shows a significant performance advantage, as demonstrated in Table 6.6.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Table 6.6
The performance of the inner-outer ($\beta = 0.5$, $\eta = 10^{-2}$) and power iterations on non-Web data with $\alpha = 0.95$, $\tau = 10^{-7}$, and $\mathbf{v}$ sparse and nonuniform. The computations were done in pure MATLAB.

Inner-outer	188 mat-vec	36.2 hours
Power	271 mat-vec	54.6 hours

7. Conclusions. We have presented an inner-outer iterative algorithm for accelerating PageRank computations. Our algorithm is simple, fast, and introduces minimal overhead. It is tailored to the PageRank problem, exploiting the fact that all the eigenvalues of the matrix except the eigenvalue 1 are enclosed within a circle in the complex plane whose radius equals the damping factor, and hence it is beneficial to iterate with a small damping factor. Because no permutations, projections, orthogonalizations, or decompositions of any sort are involved, programming the algorithm is completely straightforward, and it is highly parallelizable.

There are a variety of PageRank optimizations which manipulate the Web link graph. For example, dangling node optimizations [14, 24, 30, 31] and matrix-vector product acceleration [27] further reduce the work performed in an iteration. Our inner-outer iterations, by their nature, can be integrated with these solution methodologies in a straightforward fashion.

The inner-outer algorithm is parameter-dependent, but an effective choice of the parameters can be made, as our analysis shows. We have provided a detailed convergence analysis and have shown analytically and experimentally that the proposed technique is effective for a large range of inner tolerances. Observing that the gains are often made in the initial iterates, we have also introduced a hybrid scheme that switches to the power method once the inner iterations start converging immediately.

In principle, any iterative solver that is effective for the linear system formulation of PageRank computation can have inner-outer iterations incorporated into it and possibly accelerate convergence. We have demonstrated this in detail for the Richardson-type scheme that is equivalent to applying the power method to the eigenvalue problem, a Gauss–Seidel method, and preconditioned BiCG-STAB iterations. For nonstationary schemes it is natural to think of our approach as a preconditioning technique.

Our scheme compares favorably with other widely used schemes, not only in terms of performance but also in terms of the availability of a straightforward, easily reproducible implementation. For example, consider the well-known quadratic extrapolation scheme [26]. Our inner-outer method has a few distinct advantages over it. It is simpler and has lower space requirements, since it involves only three working vectors ($\mathbf{x}, \mathbf{y}, \mathbf{f}$ in Algorithms 1 and 2). Also, the issue of how often to apply the acceleration scheme is less delicate, and is covered by our analysis.

Optimizing inner-outer iterations to make them work effectively with Gauss–Seidel and nonstationary schemes requires further fine-tuning. But as our results show, the performance of the algorithm is consistently strong. Given its simplicity and modest memory requirements, it is currently our solver of choice for PageRank solvers, among many solvers we have tested.

Future work may include investigating how to dynamically determine the parameters β and η, and exploring the performance of the algorithm as an acceleration technique for a variety of methods of PageRank computation. Also, the convergence of the inner-outer algorithm seems to be fairly insensitive to the presence of eigenvalues

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

of P not equal to 1 on the unit circle, but validating this analytically and numerically is subject to future investigation.

Finally, there are a number of applications based on random walks, which are similar in spirit to PageRank. They deal with social networks, peer-to-peer networks, and other problems, and many of them are extremely large. It will be interesting to explore the effectiveness of inner-outer iterations for large scale applications other than PageRank.

Our code is available to download and test at

http://www.stanford.edu/~dgleich/publications/2009/innout/.

REFERENCES

[1] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin, *PageRank Computation and the Structure of the Web: Experiments and Algorithms*, in Proceedings of the 11th International Conference on the World Wide Web, Honolulu, 2002, available online from http://www2002.org/CDROM/poster/173.pdf.

[2] K. Avrachenkov, N. Litvak, and K. S. Pham, *Distribution of PageRank mass among principle components of the web*, in Proceedings of the 5th Workshop on Algorithms and Models for the Web Graph, A. Bonato and F. C. Graham, eds., Lecture Notes in Comput. Sci. 4863, Springer-Verlag, 2007, pp. 16–28.

[3] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang, *Matching Wikipedia categories to the Library of Congress subject headings with network alignment*, submitted.

[4] P. Berkhin, *A survey on PageRank computing*, Internet Math., 2 (2005), pp. 73–120.

[5] A. Berman and R. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, revised reprint of the 1979 original, Classics Appl. Math. 9, SIAM, Philadelphia, 1994.

[6] M. Bianchini, M. Gori, and F. Scarselli, *Inside PageRank*, ACM Trans. Internet Technol., 5 (2005), pp. 92–128.

[7] P. Boldi, R. Posenato, M. Santini, and S. Vigna, *Traps and Pitfalls of Topic-Biased PageRank*, in Algorithms and Models for the Web-Graph, Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, Heidelberg, 2007.

[8] P. Boldi, M. Santini, and S. Vigna, *PageRank as a function of the damping factor*, in Proceedings of the 14th International Conference on the World Wide Web, Chiba, Japan, 2005, pp. 557–566.

[9] P. Boldi and S. Vigna, *The webgraph framework* I: *Compression techniques*, in Proceedings of the 13th International Conference on the World Wide Web, New York, 2004, pp. 595–602.

[10] A. Brauer, *Limits for the characteristic roots of a matrix.* IV: *Applications to stochastic matrices*, Duke Math. J., 19 (1952), pp. 75–91.

[11] C. Brezinski, M. Redivo-Zaglia, and S. Serra-Capizzano, *Extrapolation methods for PageRank computations*, C. R. Math. Acad. Sci. Paris., 340 (2005), pp. 393–397.

[12] K. Bryan and T. Leise, *The* \$25,000,000,000 *eigenvector: The linear algebra behind Google*, SIAM Rev., 48 (2006), pp. 569–581.

[13] G. M. Del Corso, A. Gullí, and F. Romani, *Comparison of Krylov subspace methods on the PageRank problem*, J. Comput. Appl. Math., 210 (2007), pp. 159–166.

[14] N. Eiron, K. S. McCurley, and J. A. Tomlin, *Ranking the web frontier*, in Proceedings of the 13th International Conference on the World Wide Web, New York, 2004, pp. 309–318.

[15] L. Eldén, *A Note on the Eigenvalues of the Google Matrix*, Report LiTH-MAT-R-04-01, Linköping University, Linköping, Sweden, 2003.

[16] H. C. Elman and G. H. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1645–1661.

[17] E. Giladi, G. H. Golub, and J. B. Keller, *Inner and outer iterations for the Chebyshev algorithm*, SIAM J. Numer. Anal., 35 (1998), pp. 300–319.

[18] D. Gleich, L. Zhukov, and P. Berkhin, *Fast Parallel PageRank: A Linear System Approach*, Yahoo! Research Technical Report YRL-2004-038, available online from http://research.yahoo.com/publication/YRL-2004-038.pdf, 2004.

[19] G. H. Golub and C. Greif, *An Arnoldi-type algorithm for computing PageRank*, BIT, 46 (2006), pp. 759–771.

[20] G. H. Golub and M. L. Overton, *The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems*, Numer. Math., 53 (1988), pp. 571–593.

[21] G. Grimmett and D. Stirzaker, *Probability and Random Processes*, 3rd ed., Oxford Univer-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

sity Press, Oxford, UK, 2001.

[22] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, *Combating Web spam with TrustRank*, in Proceedings of the 30th International Conference on Very Large Data Bases, Toronto, 2004, pp. 576–587.

[23] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, 1994.

[24] I. C. F. Ipsen and T. M. Selee, *PageRank computation, with special attention to dangling nodes*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 1281–1296.

[25] S. Kamvar and T. Haveliwala, *The Condition Number of the PageRank Problem*, Technical report, Stanford University, Stanford, CA, 2003, available online from http://dbpubs.stanford.edu:8090/pub/2003-36.

[26] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub, *Extrapolation methods for accelerating PageRank computations*, in Proceedings of the 12th International Conference on the World Wide Web, Budapest, 2003, pp. 261–270.

[27] C. Karande, K. Chellapilla, and R. Andersen, *Speeding up algorithms on compressed web graphs*, in Proceedings of the 2nd ACM International Conference on Web Search and Data Mining, Barcelona, 2009, pp. 272–281.

[28] G. Kollias, E. Gallopoulos, and D. Szyld, *Asynchronous iterative computations with Web information retrieval structures: The PageRank case*, in Parallel Computing: Current and Future Issues of High-End Computing, NIC Ser. 33, John von Neumann Institut für Computing, Jülich, Germany, 2006, pp. 309–316.

[29] A. N. Langville and C. D. Meyer, *A survey of eigenvector methods for Web information retrieval*, SIAM Rev., 47 (2005), pp. 135–161.

[30] A. Langville and C. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, Princeton, NJ, 2006.

[31] C. Lee, G. Golub, and S. Zenios, *A Fast Two-Stage Algorithm for Computing PageRank and Its Extensions*, Technical report SCCM-03-15, Stanford University, Stanford, CA, 2003.

[32] F. McSherry, *A uniform approach to accelerated PageRank computation*, in Proceedings of the 14th International Conference on the World Wide Web, Chiba, Japan, 2005, pp. 575–582.

[33] J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert, *GeneRank: Using search engine technology for the analysis of microarray experiments*, BMC Bioinformatics, 6 (2005), p. 233.

[34] M. A. Najork, H. Zaragoza, and M. J. Taylor, *Hits on the web: How does it compare?*, in Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, 2007, pp. 471–478.

[35] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Stanford Digital Libraries, 1999, available online from http://dbpubs.stanford.edu:8090/pub/1999-66.

[36] J. Parreira, D. Donato, S. Michel, and G. Weikum, *Efficient and decentralized PageRank approximation in a peer-to-peer web search network*, in Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, 2006, pp. 415–426.

[37] S. Serra-Capizzano, *Jordan canonical form of the Google matrix: A potential contribution to the PageRank computation*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 305–312.

[38] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.

[39] R. Singh, J. Xu, and B. Berger, *Pairwise global alignment of protein interaction networks by matching neighborhood topology*, in Research in Computational Molecular Biology, Lecture Notes in Comput. Sci. 4453, Springer-Verlag, Berlin, Heidelberg, pp. 16–31.

[40] W. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.

[41] *The Library of Congress Subject Headings*, http://id.loc.gov/authorities/search/. Accessed via http://lcsh.info, September 2008.

[42] R. Varga, *Matrix Iterative Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1962.

[43] S. Vigna, R. Posenato, and M. Santini, *Law* 1.3.1 *- Library of Algorithms for the Webgraph*, http://law.dsi.unimi.it/software/docs/.

[44] *Wikipedia XML Database Dump from April* 2*, 2007.* Accessed from http://en.wikipedia.org/wiki/Wikipedia:Database_download, April 2007.

[45] R. S. Wills and I. C. F. Ipsen, *Ordinal ranking for Google's PageRank*, SIAM J. Matrix Anal. Appl., 30 (2009), pp. 1677–1696.

[46] D. Zhou, J. Huang, and Schölkopf, *Learning from labeled and unlabeled data on a directed graph*, in Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 2005, pp. 1036–1043.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.