Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 525313, 8 pages
http://dx.doi.org/10.1155/2013/525313

Research Article

The Extrapolation-Accelerated Multilevel Aggregation Method in PageRank Computation

Bing-Yuan Pu,[1,2] Ting-Zhu Huang,[1] Chun Wen,[1] and Yi-Qin Lin[3]

[1] *School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China*
[2] *Department of Basic Courses, Chengdu Textile College, Chengdu 611731, China*
[3] *Department of Mathematics and Computational Science, Hunan University of Science and Engineering, Yongzhou 425100, China*

Correspondence should be addressed to Bing-Yuan Pu; skypuby@163.com

Received 22 June 2013; Accepted 11 August 2013

Academic Editor: Masoud Hajarian

An accelerated multilevel aggregation method is presented for calculating the stationary probability vector of an irreducible stochastic matrix in PageRank computation, where the vector extrapolation method is its accelerator. We show how to periodically combine the extrapolation method together with the multilevel aggregation method on the finest level for speeding up the PageRank computation. Detailed numerical results are given to illustrate the behavior of this method, and comparisons with the typical methods are also made.

1. Introduction

The PageRank algorithm for assigning a rank of importance to web pages has been the key technique in web search [1]. The core of the PageRank computation consists in the principal eigenvector of a stochastic matrix representing the hyperlink structure of the web. Due to the large size of the web graph (over eight billion nodes [2]), computing PageRank is faced with the big challenge of computational resources, both in terms of the space of CPU and RAM required and in terms of the speed of updating in time; that is, as a new crawl is completed, it can be soon available for searching. Among all the numerical methods to compute PageRank, the Power method is one of the standard ways for its stable and reliable performances [1], whereas the low rate of convergence is its fatal flaw. Many accelerating techniques have been proposed to speed up the convergence of the Power method, including aggregation/disaggregation [3–5], vector extrapolation [6–10], multilevel [11–14], lumping [15, 16], Arnoldi-type [10, 17], and adaptive methods [4, 18]. To some extent, our work follows in this vein, that is, seeking a method to accelerate the convergence of PageRank computation. In this research, we introduce a multilevel aggregation method to accelerate the computation of PageRank, and the acceleration is performed by vector extrapolation, which aims at combining the old iteration sequences.

The remainder of this paper is organized as follows. Section 2 provides preliminary and related works on PageRank and some acceleration techniques. Section 3 describes our main acceleration algorithm. Section 4 covers our numerical results and comparisons among the main PageRank algorithms. Section 5 contains conclusion and points out directions for future research.

2. Preliminaries and Related Works

2.1. Background on PageRank. PageRank [19] is the heart of software, as Google claims on its webpage, and continues to provide the basis for all the web search tools. Link-based PageRank views the web as a directed graph, where each node represents a page, and each edge from node i to node j represents the existence of a link from page i to page j, and at this time, one can view page j as "important." Page and Brin [1] made another basic assumption: the amount of importance distributed to j by i is proportional to the importance of i and inversely proportional to the number of pages i is pointing to. This definition suggests an iterative

fixed-point computation for determining the importance for every page on the web. Each page has a fixed rank score π_i, forming the rank vector π. Page and Brin convert the computation of PageRank to the computation of stationary distribution vector of the linear system $\pi^T = \pi^T P$, where P is the adjacency matrix of the web graph G normalized so that each row sums to 1. For P to be a valid transition probability matrix, every node must have at least 1 outlink; that is, P should have no rows consisting of all zeros (pages without outlinks are called dangling nodes). Meanwhile, to guarantee the uniqueness of a unitary norm eigenvector corresponding to the eigenvalue 1, by the ergodic theorem for Markov chains [20], P should be aperiodic and irreducible. To solve these problems, Page and Brin have introduced a parameter α and transferred matrix P to a new one P': $P' = \alpha(P + dV^T) + (1 - \alpha)EV^T$, where E is the vector with all entries equal to 1, V is the column vector representing a uniform probability distribution over all nodes, $V = [1/n]_{n\times 1}$, $d = 1$, while page i is dangling and $d = 0$ otherwise, assuming the total page number is n. If viewing a surfer as a random walker on the web, the constant α can be explained below: a web surfer visiting a page will jump to any other page on the web with probability $(1 - \alpha)$, rather than following an outlink. Matrix P' is usually called the Google matrix. Then, PageRank vector π is the stationary distribution vector of P'; that is,

$$\pi^T = \pi^T P', \tag{1}$$

obviously, (1) can be rewritten as

$$\pi^T A = 0, \tag{2}$$

where $A = I - P'$, I is an identity matrix, and A is a singular M-matrix with the diagonal elements are negative column sums of its off-diagonal elements. So what remains to do is to solve the linear system (1) or its homogeneous one (2), corresponding to an irreducible Markov chain.

2.2. The Vector Extrapolation-Acceleration of PageRank. The Power method is the standard algorithm for PageRank, that is, giving the initial uniform distribution $x^{(0)}$ of the system (1), to compute successive iterates $x^{(k)} = x^{(k-1)}P'$, until convergence, that is, when $\lim_{k\to\infty} x^{(k)}$ exists, which is exactly the PageRank vector. As mentioned in Section 1, although the Power method is a stable and reliable [1] or even a fast iteration algorithm [21], accelerating the computation is still important, since every search engine crawls a huge number of pages, and each matrix multiplication in iteration is so expensive, requiring considerable resources both in terms of CPU and RAM, and hence, the rate of convergence deteriorates as the number of pages grows larger. Now there are many acceleration algorithms for PageRank computation, and among all the algorithms the vector extrapolation method is a very popular and efficient one. A detailed review of the vector extrapolation method can be found in the work of Kamvar et al., Smith et al., and Sidi [9, 22, 23]. To our knowledge, there are several kinds of extrapolation methods, such as quadratic extrapolation [9], two polynomial-type methods including the minimal polynomial extrapolation method (MPE) of Cabay and Jackson [24] and the reduced rank extrapolation (RRE) of Eddy [25], and the three epsilon vector extrapolation methods of Wynn [26]. In recent years, many papers have discussed the application of vector extrapolation method to compute the stationary probability vector of Markov chains and web ranking problems, see [6, 8, 9, 22, 23, 27] for details. Numerical experiments in these papers suggest that the polynomial-type methods are in general more economical than the epsilon vector extrapolation methods in the sense of computation time and storage requirements. For this reason, our concern in the context is to consider a polynomial-type vector extrapolation, that is, the generalization of quadratic extrapolation method (GQE) proposed in [23]. Now let us discuss simply the strategy of GQE. As it is known, the starting point of the extrapolation method is to accelerate the convergence of the sequences $\{x_j\}$ generated from a fixed-point iterative method in Markov chain of the form

$$x_{j+1} = F\left(x_j\right), \quad j = 0, 1, \ldots; \; F : \mathbb{R}^n \longrightarrow \mathbb{R}^n, \tag{3}$$

where x_0 is an initial guess. Supposve that we have produced a sequence of iterates $\{x_i\}_{i=1}^{\infty}$, where $x_i \geq 0$. Then, at the kth outer iteration, let

$$X = [x_k, x_{k-1}, \ldots, x_{k-m+1}] \in \mathbb{R}^{n\times m} \tag{4}$$

be a matrix consisting of the last m iterates with x_k being the newest, where m is called the window size as usual. It is evident that X has the following properties:

$$x_i \geq 0, \quad \|x_i\|_1 = 1, \quad i = 1, 2, \ldots. \tag{5}$$

The problem to be solved is transformed into obtaining a vector z satisfying $\sum_{i=1}^{m} z_i = 1$, and, thus we have an updated probability vector

$$\widehat{x}_k = Xz = z_1 x_k + z_2 x_{k-1} + \cdots + z_m x_{k-m+1}, \tag{6}$$

a linear combination of the last m iterates. That is to say, the current iterate can be expressed as a linear combination of some of the first eigenvectors, combined with the Power method, up to the converge of the principal eigenvector. GQE is derived in this light and can be given as follows [23].

Algorithm 1 (the generalization of quadratic extrapolation (GQE)). (1) Input the vectors $x_0, x_1, \ldots, x_{k+1}$.

(2) Compute $u_i = x_{i+1} - x_0$, $i = 0, 1, \ldots, k$, set $U_k = [u_0, u_1, \ldots, u_k]$. Compute the QR-factorization of U_k, namely, $U_k = Q_k R_k$. Obtain $R_{k-1} := R_k(1 : k, 1 : k)$, $Q_{k-1} = Q_k(1 : k, 1 : k)$.

(3) Solve the linear system $R_{k-1}d = -Q_{k-1}^T u_k$, $d = [d_0, d_1, \ldots, d_{k-1}]^T$.

(4) Set $d_k = 1$ and compute $c = [c_0, c_1, \ldots, c_k]^T$ by $c_i = \sum_{j=i}^{k} d_j$, $i = 0, 1, \ldots, k$.

(5) Compute $\widehat{x}_{k+1} = (\sum_{i=0}^{k} c_i)x_0 + Q_k(R_k c)$.

3. Accelerated Aggregation Strategy in PageRank Computation

The core of PageRank computation, as discussed earlier, is to solve the large sparse Markov chain problem (2). Thus, it is evocative of using a especially useful method—multilevel method based on aggregation of Markov states, which has attracted considerable attention [11–14, 28, 29]. Isensee and Horton considered a kind of multilevel methods for the steady state solution of continuous-time and discrete-time Markov chains in [13, 14], respectively. De Sterck et al. proposed a multilevel adaptive aggregation method for calculating the stationary probability vector of Markov matrices in [11], as shown in their context, which is a special case of the adaptive smoothed aggregation [30] and adaptive algebraic multigrid methods [31] for sparse linear systems.

The central idea of multilevel aggregation method is to convert a large linear system to a smaller one by some aggregation strategies, and thus, the stationary state solution can be obtained in an efficient way.

However, aggregation/disaggregation cannot always dissolve the algorithmic scalability due to poor approximation of A in problem (2) by unsmoothed intergrid operators, so a careful choice of aggregation strategy is crucial for the efficiency of the multilevel aggregation hierarchies. Now there are many aggregation methods, such as graph aggregation [32], neighborhood-based aggregation, and (double) pairwise aggregation [4, 13, 14]. In our study, we consider the neighborhood aggregation method as described in [12–14], since it is able to result in well-balanced aggregates of approximately equal size and provide a more regular coarsening throughout the automatic coarsening process [12, 29, 33]. Our aggregation strategies are based on the problem matrix by the current iterate $\overline{A} = A \times \mathrm{diag}(x_i)$, rather than the original coefficient matrix A. Let x_k denote the current iterate; then, we say node i is strongly connected to node j in the graph of $\overline{A}$ if

$$-\overline{a}_{ij} \geq \theta \max_{k \neq i} \left\{-\overline{a}_{ik}\right\} \quad \text{or} \quad -\overline{a}_{ji} \geq \theta \max_{k \neq j} \left\{-\overline{a}_{jk}\right\}, \tag{7}$$

where θ is a strength of connection parameter and $\theta = 0.25$ is used there. Suppose that $\mathcal{N}_i$ is the set of all points which are strongly connected to i in the graph of $\overline{A}$ including node i itself. Then, we have the neighborhood-based algorithm as follows (see [12, 29] for more details).

Algorithm 2 (neighborhood-based aggregation, $\{Q_j\}_{j=1}^{J} \leftarrow \mathrm{NBA}(\overline{A}, \theta)$). (1) Preparing: set $R = \{1, \ldots, n\}$ and $J = 0$.

(2) Assign entire neighborhoods to aggregates: for $i \in \{1, \ldots, n\}$, build strong neighborhoods $\mathcal{N}_i$ based on $(*)$; if $\mathcal{N}_i \subset R$, then $J \leftarrow J + 1$, $\mathrm{Q}_J \leftarrow \mathcal{N}_i$, $\widetilde{\mathrm{Q}}_J \leftarrow \mathcal{N}_i$, $R \leftarrow R \setminus \mathcal{N}_i$.

(3) Put the remaining points in the most connected aggregates: while $R \neq \emptyset$, pick $i \in R$ and set $j = \mathrm{argmax}_{k=1,\ldots,J} \mathrm{card}(N_i \cap \widetilde{Q}_k)$, set $Q_j \leftarrow Q_j \cup \{i\}$ and $R \leftarrow R \setminus \{i\}$.

(4) Obtain the aggregation matrix Q: if $i \in Q_j$, $j = 1, \ldots, J$, then $Q_{ij} = 1$, otherwise $Q_{ij} = 0$.

Note that the aggregation matrix Q in Algorithm 2 has the following form

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & \ldots \\ 1 & 0 & 0 & 0 & \ldots \\ 0 & 1 & 0 & 0 & \ldots \\ 0 & 0 & 1 & 0 & \ldots \\ 0 & 0 & 1 & 0 & \ldots \\ 0 & 0 & 0 & 1 & \ldots \\ 0 & 0 & 0 & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}_{n \times J}. \tag{8}$$

From (8), we find that there exists only one element $Q_{ij} = 1$ in each row, but each column may have several elements $Q_{ij} = 1$, and the sum of the elements in the jth column denotes the number of the nodes combined into the jth aggregation.

The multilevel aggregation method, based on the neighborhood-based aggregation strategy, introduced by De Sterck et al. in [11, 12, 28, 29] can be expressed as follows.

Algorithm 3 (multilevel aggregation method, $x \leftarrow \mathrm{MA}(A, x, \nu_1, \nu_2, \alpha)$). (1) Presmoothing: do ν_1 times $x \leftarrow N(\mathrm{Relax}(A, x))$.

(2) Build Q according to the automatic coarsening process described below. Obtain $R \leftarrow Q^T$ and $P \leftarrow \mathrm{diag}(x)Q$.

(3) Form the coarse-level matrix $A_c \leftarrow \mathrm{RAP}$ and compute $x_c \leftarrow Q^T x$.

(4) If on the coarsest level, solve $A_c x_c = 0$ by a direct method. Otherwise, apply α iterations of this algorithm $x_c \leftarrow \mathrm{MA}(A_c \mathrm{diag}(Q^T x)^{-1}, x_c, \nu_1, \nu_2, \alpha)$.

(5) Coarse-level correction: $x \leftarrow P(\mathrm{diag}(Q^T x)^{-1})x_c$.

(6) Postsmoothing: do ν_2 times $x \leftarrow N(\mathrm{Relax}(A, x))$.

In Algorithm 3, $A = A_1$ is given in system (2), x is an initial guess vector, A_c denotes the coarse-level matrix, and x_c is the corresponding coarse-level vector, where $c = 1, 2, \ldots, L$ is the number of levels, the finest level is 1, and the coarsest level is L. Q is the aggregation matrix generated by the aggregation method-Algorithm 2, and P_l and R_l, $l = 1, 2, \ldots, L-1$ are the prolongation and restriction operators, respectively, which are created by an automatic coarsening process in step 2, and then, we get the coarse-level matrices by the equation $A_l = R_l \times A_l \times P_l$. Setting the weighted Jacobi method with weight ω, a variant of Jacobi method, as the pre- and postsmoothing approaches and letting ν_1 and ν_2 be their iteration times, respectively, the matrix A in system (2) is splitted into the following form

$$A = D - L - U, \tag{9}$$

where D is the diagonal part of the matrix A with $d_{ii} > 0$ for all i and L and U are the negated strictly lower- and upper-triangular parts of A, respectively. Then, the weighted Jacobi relaxation method can be written as

$$x \longleftarrow N\left((1-\omega)\, x + \omega D^{-1}\,(L+U)\, x\right), \tag{10}$$

with weight $\omega \in (0, 1)$. Here, we use $N(\cdot)$ to denote the normalization operator defined by $N(x) := x/\|x\|_1$ for all

$x \neq 0$. Note that we have carried out the normalization after each relaxation process in Algorithm 3 to ensure that the fine-level iterates x_i can be interpreted as approximations to the stationary probability vector.

It should be noted that the direct method on the coarsest level, used at step 4 of Algorithm 3, is based on the following theorem.

Theorem 4 (see [34, Theorem 4.16]). *If A is an irreducible singular M-matrix, then each of its principal submatrices other than A itself is a nonsingular M-matrix.*

If A_L is the coarsest-level operator, then we use the direct method presented in the coarsest-level algorithm below to solve the coarsest-level equation $A_L x_L = 0$.

Coarsest-Level Algorithm

Step 1. Compute $N_L := \text{size}(A_L, 1)$; % the size of coarsest-level operator A_L.

Step 2. Obtain $A_{Lp} := A_L(1 : N_L - 1, 1 : N_L - 1)$; % the $N_L - 1$ order principal submatrix of A_L.

Step 3. Obtain $b_{Lp} := -A_L(1 : N_L - 1, N_L)$; % the right-hand vector corresponding to A_{Lp}.

Step 4. Compute $x_{Lp} := A_{Lp} \setminus b_{Lp}$; let $x_{Lp}(N_L) = 1$; and obtain the coarsest-level solution $x_L = x_{Lp}/\|x_{Lp}\|_1$.

Now, let us introduce the main idea of this paper and the implementation details of the main algorithm. In fact, our idea is derived from the excellent papers [12, 28, 29]. In the literature, De Sterck et al. studied several strategies to accelerate the convergence of the multilevel aggregation method, including the application of a smoothing technique to the interpolation and restriction operators in [28] and the analysis of a recursively accelerated multilevel aggregation method by computing quadratic programming problems with inequality constraints in [29]. Of particular note, in [12], they introduced a top-level acceleration of adaptive algebraic multilevel method by selecting a linear combination of old iterates to minimize a function over a subset of the set of probability vectors. Their acceleration strategy was involved mainly in cases when the window size $m = 2$, and for larger window size m, such as $m = 3$ or 4, they used the active set method from Matlab's quadprog function [35]. With this in mind, enough interest is aroused to look for a general acceleration method for any case of window size. In view of the effectiveness of vector extrapolation in improving the convergence of the iterate sequence, now consider a new accelerated multilevel aggregation method, combining the multilevel aggregation method and the vector extrapolation method. Giving the finest level iterate sequence $\{x_n\}$ produced by Algorithm 3 in multilevel aggregation, the updated iterate $\widehat{x}_k$ is generated as their linear combination by Algorithm 1. And so we can present our accelerated multilevel aggregation as follows.

Algorithm 5 (extrapolation-accelerated multilevel aggregation methods, $x \leftarrow \text{EAMA}(A, \widehat{x}_0, m, \epsilon)$). (1) Set $k = 1$, if no initial guess is given, choose $\widehat{x}_0$.

(2) Run the multilevel aggregation method, $x_k \leftarrow \text{MA}(A, \widehat{x}_{k-1}, \nu_1, \nu_2, \alpha)$.

(3) Set $m \leftarrow \min\{M, k\}$. % set the window size.

(4) Set $X \leftarrow [x_k, x_{k-1}, \ldots, x_{k-m-1}]$. % the last $m + 2$ iterates.

(5) Apply GQE algorithms to obtain $\widehat{x}_k$ from X, respectively.

(6) If $\|A\widehat{x}_k\|_1 > \|Ax_k\|_1$, then $\widehat{x}_k \leftarrow x_k$.

(7) Check convergence, if $\|A\widehat{x}_k\|_1/\|A\widehat{x}_0\|_1 < \epsilon$, then $\widehat{x}_k \leftarrow \widehat{x}_k/\|\widehat{x}_k\|_1$, otherwise set $k \leftarrow k + 1$ and go to step 2.

Note that m in Algorithm 5 is the window size and ϵ is a prescribed tolerance. In particular, there exists a difference in constructing the matrix X between our Algorithm 5 and the algorithm of [36]. From Algorithm 1, the main reason is that the GQE needs $k + 2$ vectors $x_0, x_1, \ldots, x_{k+1}$ as their input. That is to say, when the window size is $m = 2$, four approximate probability vectors given in the matrix $X = [x_0, x_1, x_2, x_3] \in \mathbb{R}^{n\times 4}$ are required as its input of GQE algorithms. In particular, the window size $m = 2$ corresponds to the quadratic vector extrapolation method presented in [9]. Hence, the matrix X is given as the form of step 4 in Algorithm 5. The efficiency of our accelerated multilevel aggregation algorithms will be shown in the next section, where the relative advantage of our method over Matlab's quadprog function in the Markov chain and over other methods in PageRank computation will be demonstrated.

4. Numerical Results and Analysis

In this section, we report some numerical experiments to show the numerical behavior of extrapolation-accelerated multilevel aggregation methods. All the numerical results are obtained with a Matlab 7.0.1 implementation on a 2.93 GHz processor with 2 GB main memory.

For the sake of justice, the initial guess vector is selected as $x_0 = e/\|e\|_1$ for all the algorithms. All the iterations are terminated when the residual 1-norm

$$\frac{\|Ax_k\|_1}{\|Ax_0\|_1} \leq \text{tol}, \tag{11}$$

where x_k is the current approximate solution and tol is a user described tolerance. For convenience, in all the previous tables we have abbreviated the Power method, the generation of quadratic extrapolation method, and the extrapolation-accelerated multilevel aggregation method as Power, GQE, and EAMA, respectively. We use "res" to denote the 1-norm residual, "lev" to the number of levels in the multilevel aggregation, "ite" to the number of iterations, and "CPU" to the CPU time used in seconds.

Example 1. As described in Section 3, the active set method from Matlab's quadprog function [12, 35] is usually used for the acceleration strategy in Markov chains and other problems. And in this paper, we have suggested the extrapolation-acceleration strategy instead of the Matlab's one. This example aims to compare the two methods and to test the efficiency of our new method. The test example is a birth-death chain

Table 1: Numerical comparison results for EAMA and Q-matlab for Example 1.

Method		Window size								
EAMA		2			3			4		
n	lev	C_{op}	res	it	C_{op}	res	it	C_{op}	res	it
27	2	2.0127	$7.4095e-6$	10	2.0127	$1.9435e-6$	10	2.0127	$3.9440e-6$	9
81	3	3.0166	$7.0117e-6$	14	3.0166	$6.1956e-6$	15	3.0166	$9.0883e-6$	13
243	4	4.5433	$5.4196e-6$	22	4.0481	$7.4238e-6$	20	4.0481	$7.3145e-6$	16
1024	5	4.9831	$1.2408e-6$	48	5.5179	$6.2320e-6$	40	4.9831	$8.4259e-6$	31
Q-matlab		2			3			4		
n	lev	C_{op}	res	it	C_{op}	res	it	C_{op}	res	it
27	2	2.0127	$5.3065e-6$	10	2.0127	$3.8436e-6$	8	2.0127	$6.3351e-6$	7
81	3	3.0166	$7.4583e-6$	15	3.0166	$8.0007e-6$	14	3.0166	$6.4303e-6$	13
243	4	4.0481	$7.5300e-6$	22	4.0481	$9.1973e-6$	22	4.0481	$8.9225e-6$	18
1024	5	5.8769	$3.9990e-6$	77	4.9831	$9.4107e-6$	50	4.9987	$8.6577e-6$	48

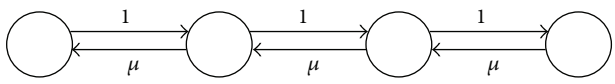

Figure 1: Graph of an M/PH/1 queue.

with invariant birth and death rates as shown in Figure 1 [37–39], which is usually used in queuing theory, demographics, performance engineering, or in biology [32, 40].

Setting $\mu = 0.96$, relaxation parameter $\omega = 0.7$, and the strength parameter of connection $\theta = 0.25$ in our experiment, we use Q-matlab to denote the Matlab's quadprog function acceleration method that improves the W-cycles on the finest level and "n" to denote the length of Markov chains. "C_{op}" denotes the operator complexity of the last cycle, which is defined as the sum of the number of nonzero entries in all operators on all levels $A_l, l = 1, \ldots, L$, divided by the number of nonzero entries in the finest-level operator A_1. That is, $C_{op} = \sum_{l=1}^{L} \mathrm{nnz}(A_l)/\mathrm{nnz}(A_1)$ where $\mathrm{nnz}(A)$ is the number of nonzero entries in the matrix A. Numerical results for EAMA and Q-matlab methods have been reported in Table 1.

From Table 1, it is evident that our accelerated multilevel aggregation method EAMA has better performance than Matlab's function Q-matlab for the testing problem from both an operator complexity and the iteration count perspective, and moreover, our method is more efficient with the length of Markov chain increasing. For instance, when $n = 1024$ and the window size m: $m = 2, 3, 4$, the EAMA method cuts the number of iterations by up to 37.7%, 20%, and 35.4%, respectively, compared to the Q-matlab method.

Example 2. In this example, we consider the performance of EAMA in the PageRank computation. We take a typical website of "Hollions" as our numerical example, which has been listed in Chapter 14 of [41], which contains 6012 web pages and 23875 hyperlinks. We compare the Power method, the generalization of quadratic extrapolation method, and the extrapolation-accelerated multilevel aggregation method for the PageRank problem.

In Algorithm 5, we consider that accelerators, W-cycles ($r = 2$), and V-cycles can be treated in a similar way.

Table 2: Comparisons of the residual vectors for the Power, GQE, and EAMA methods when given the number of iterations.

Method	ite = 5	ite = 10	ite = 20	ite = 30
Power	0.148848	0.02414	0.002329	$3.6687e-004$
GQE	0.029646	0.00210	$4.0479e-005$	$1.2882e-007$
EAMA	0.003400	$1.4285e-004$	$2.0306e-007$	$4.7062e-010$

Some specific sets of parameters are used. We use the weighted Jacobi as the pre- and postsmoothing approaches in Algorithm 2, with relaxation parameter $\omega = 0.7$. Direct coarsest-level solvers are performed by the coarsest-level algorithm, and the strength parameter of connection is chosen as $\theta = 0.25$. There are some numerical results reported in Tables 2 and 3 in the following parts.

The residual analysis in Table 2 indicates that EAMA does much better in PageRank computation than the other two methods, namely, the classic Power method and GQE, and EAMA has a more obvious advantage over the other two methds, with the iteration count increasing.

From Table 3, when an error tolerance is provided and no matter what value the window size is, the accelerated multilevel aggregation method outperforms the Power and GQE, in terms of both iteration count and CPU time. For instance, when the window size is 3, the number of iterations by EAMA is about half of the one by GQE and less than one third of the one by Power.

Example 3. The test matrix is the widely used CS-Stanford Web matrix, which is available from http://www.cise.ufl.edu/research/sparse/matrices/index.html. It contains 9914 nodes and 35,555 links. In this example, we run the Power method, the generalized quadratic extrapolation method, and our extrapolation-accelerated multilevel aggregation method. The convergence tolerance is tol $= 10^{-5}$, and the window size is selected as $m = 3$. Table 4 lists the results.

It is seen from Table 4 that the numerical behavior of the three algorithms strongly relies on the choice of the damping factor α in PageRank computation. When α is high, say 0.95 and 0.99, the computing time and iteration count are sizable. However, when α is moderate, say 0.85, the computing time

Table 3: Comparisons of the iteration counts and CPU for the Power, GQE, and EAMA methods when the residual vector is 10^{-5}.

	Method						
	Power	GQE			EAMA		
Window size		2	3	4	2	3	4
ite	50	26	24	26	18	14	15
CPU	34.25	28.16	28.67	27.57	23.13	22.63	22.75

Table 4: Numerical results of the three algorithms (with various α) on the CS—Stanford Web matrix.

	Method		
	Power	GQE	EAMA
$\alpha = 0.85$			
ite	273	147	98
CPU	39.78	36.61	25.31
res	$9.9972e-006$	$6.1911e-006$	$3.2462e-006$
$\alpha = 0.90$			
ite	379	214	126
CPU	54.13	45.31	38.42
res	$9.9718e-006$	$6.3426e-006$	$6.5672e-006$
$\alpha = 0.95$			
ite	629	425	253
CPU	90.08	68.33	52.41
res	$9.9834e-006$	$8.32671e-006$	$7.4938e-006$
$\alpha = 0.99$			
ite	1920	734	527
CPU	263.54	198.32	139.42
res	$9.9940e-006$	$7.1028e-006$	$7.3542e-006$

Table 5: Numerical comparison results of the three algorithms (with various convergence tolerances) for Example 4.

Method	Power		GQE		EAMA	
tol	ite	CPU	ite	CPU	ite	CPU
10^{-5}	284	68.76	256	57.33	227	48.32
10^{-6}	316	97.55	283	86.42	248	80.21
10^{-8}	392	145.72	334	121.75	291	106.43
10^{-10}	514	234.61	438	178.92	387	152.49

and iteration count are greatly reduced, like, about one-seventh of the case $\alpha = 0.99$. This just confirms Page and Brin's point [1]: $\alpha = 0.85$ is the most common choice.

Besides, just as expected, it is obvious to see that the GQE method is superior to the Power method, while the new multilevel aggregation method (EAMA) performs the best, in both computing time and iteration count terms. For instance, when α is 0.85, comparing with the results by Power and GQE, the CPU time for EAMA has been reduced by 36.4% and 30.9%, respectively, and the iteration count has been reduced by 64.1% and 33.3%, respectively.

Example 4. This example aims to examine the influence of the choice of the convergence tolerance on the algorithms. The test matrix is the Stanford-Berkeley Web matrix (available from http://www.stanford.edu/~sdkamvar/research.html). It contains 683,446 pages and 7.6 million links. We run the Power method, the generalized quadratic extrapolation method, and our extrapolation-accelerated multilevel aggregation method on this problem and choose tol to be 10^{-5}, 10^{-6}, 10^{-8}, and 10^{-10}, respectively. The window size for the extrapolation procedure is selected as $m = 3$, and α in PageRank is set as 0.85. Table 5 reports the results obtained.

It is easy to see from Table 5 that our new algorithm, EAMA, performs the best in most cases, regardless of the convergence tolerance and both in terms of computing time and iteration numbers. For instance, when tol varies from 10^{-6} to 10^{-8}, the iteration count needed for the three algorithms (power, GQE, and EAMA) increases 24.1%, 18.0%, and 17.3%, respectively, but nonetheless, the count for EAMA is still less than the corresponding one for GQE and far less than that for the Power, only about 74.2% of the latter. It can be seen that about the same goes for CPU time.

5. Conclusions

This paper illustrates the accelerated multilevel aggregation method for calculating the stationary probability vector of an irreducible stochastic matrix, Google matrix, in PageRank computation. It is conducted to combine the vector extrapolation method and the multilevel aggregation method, where the neighborhood-based aggregation strategy [11, 12, 28, 29] is used, and the vector extrapolation acceleration is carried out on the finest level. Our approach is inspired by De Sterck et al. in [12] where the active set method from Matlab's quadprog function was used therein for window size m, which is greater than two. Thus, it is natural to seek for a general acceleration method for any case of window size, and thus, our EAMA is produced.

Although our method is effective, however, there are still many places worth exploring. The Google matrix, for example, can be reordered according to dangling nodes and nondangling nodes of the matrix [16, 42–44], which reduces the computation of PageRank to that of solving a much smaller problem. And for the special linear system (2) in Markov chains, there maybe other even better options than the neighborhood-based aggregation strategy in multilevel method. With some improvement, our algorithm will be worth watching and will be a part of future work.

Acknowledgments

This research is supported by the Chinese Universities Specialized Research Fund for the Doctoral Program (20110185110020), Scientific Research Fund of Sichuan Provincial Education Department (T11008), and the Science and Technology Planning Project of Hunan Province (2010JT4042).

References

[1] L. Page and S. Brin, "The PageRank citation ranking: bringing order to the web," Technical Report, Computer Science Department, Stanford University, 1998.

[2] R. Baeza-Yates, F. Saunt-Jean, and C. Castillo, "Web structure, dynamics and page quality," in *Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE '02)*, pp. 117–130, 2002.

[3] I. C. F. Ipsen and S. Kirkland, "Convergence analysis of a PageRank updating algorithm by Langville and Meyer," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 952–967, 2006.

[4] S. D. Kamvar, T. H. Haveliwala, and G. H. Golub, "Adaptive methods for the computation of PageRank," *Linear Algebra and Its Applications*, vol. 386, pp. 51–65, 2004.

[5] A. N. Langville and C. D. Meyer, "Updating PageRank with iterative aggregation," in *Proceedings of the 13th World Wide Web Conference (WWW '04)*, pp. 392–393, ACM Press, New York, NY, USA, May 2004.

[6] C. Brezinski, M. Redivo-Zaglia, and S. Serra-Capizzano, "Extrapolation methods for PageRank computations," *Comptes Rendus Mathematique*, vol. 340, no. 5, pp. 393–397, 2005.

[7] C. Brezinski and M. Redivo-Zaglia, "Rational extrapolation for the PageRank vector," *Mathematics of Computation*, vol. 77, no. 263, pp. 1585–1598, 2008.

[8] T. H. Haveliwala, S. D. Kamvar, D. Klein, C. Manming, and G. H. Golub, "Computing PageRank using power extrapolation," Standford University Technical Report, 2003.

[9] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Extrapolation methods for accelerating PageRank computations," in *Proceedings of the 12th International World Wide Web Conference*, 2003.

[10] G. Wu and Y. Wei, "An Arnoldi-extrapolation algorithm for computing PageRank," *Journal of Computational and Applied Mathematics*, vol. 234, no. 11, pp. 3196–3212, 2010.

[11] H. de Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, and J. Ruge, "Multilevel adaptive aggregation for Markov chains, with application to web ranking," *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2235–2262, 2008.

[12] H. de Sterck, K. Miller, T. A. Manteuffel, and G. Sanders, "Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains," *Advances in Computational Mathematics*, vol. 35, no. 2–4, pp. 375–403, 2011.

[13] C. Isensee and G. Horton, "A multi-level method for the steady state solution of Markov chains," in *Simulation and Visualization*, Magdeburg, Germany, 2004.

[14] C. Isensee and G. Horton, "A multi-level method for the steady state solution of discretetime Markov chains," in *Proceedings of the 2nd Balkan Conference in Informatics*, pp. 413–420, Ohrid, Macedonia, November 2005.

[15] C. P. Lee, G. H. Golub, and S. A. Zenios, "A fast two-stage algorithm for computing PageRank and its extensions," Stanford University Technical Report SCCM-03-15, 2003.

[16] Y. Lin, X. Shi, and Y. Wei, "On computing PageRank via lumping the Google matrix," *Journal of Computational and Applied Mathematics*, vol. 224, no. 2, pp. 702–708, 2009.

[17] G. H. Golub and C. Greif, "An Arnoldi-type algorithm for computing PageRank," *BIT Numerical Mathematics*, vol. 46, no. 4, pp. 759–771, 2006.

[18] P. Berkhin, "A survey on PageRank computing," *Internet Mathematics*, vol. 2, no. 1, pp. 73–120, 2005.

[19] Oursearch: Google Technology, http://www.google.com/technology/index.html.

[20] G. Grimmett and D. Stirzaker, *Probability and Random Processes*, Oxford University Press, 1989.

[21] T. H. Haveliwala and S. D. Kamvar, "The second eigenvalue of the Google matrix," Stanford University Technical Report, 2003.

[22] D. A. Smith, W. F. Ford, and A. Sidi, "Extrapolation methods for vector sequences," *SIAM Review*, vol. 29, no. 2, pp. 199–233, 1987.

[23] A. Sidi, "Vector extrapolation methods with applications to solution of large systems of equations and to PageRank computations," *Computers & Mathematics with Applications*, vol. 56, no. 1, pp. 1–24, 2008.

[24] S. Cabay and L. W. Jackson, "A polynomial extrapolation method for finding limits and antilimits of vector sequences," *SIAM Journal on Numerical Analysis*, vol. 13, no. 5, pp. 734–752, 1976.

[25] R. P. Eddy, "Extrapolating to the limit of a vector sequence," in *Information Linkage Between Applied Mathematics and Industry*, P. C. C. Wang, Ed., pp. 387–396, Academic Press, New York, NY, USA, 1979.

[26] P. Wynn, "Acceleration techniques for iterated vector and matrix problems," *Mathematics of Computation*, vol. 16, pp. 301–322, 1962.

[27] C. Brezinski and M. Redivo-Zaglia, "The PageRank vector: properties, computation, approximation, and acceleration," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 2, pp. 551–575, 2006.

[28] H. de Sterck, T. A. Manteuffel, S. F. McCormick et al., "Smoothed aggregation multigrid for Markov chains," *SIAM Journal on Scientific Computing*, vol. 32, no. 1, pp. 40–61, 2010.

[29] H. de Sterck, K. Miller, G. Sanders, and M. Winlaw, "Recursively accelerated multilevel aggregation for Markov chains," *SIAM Journal on Scientific Computing*, vol. 32, no. 3, pp. 1652–1671, 2010.

[30] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. Mccormick, and J. Ruge, "Adaptive smoothed aggregation (αSA) multigrid," *SIAM Review*, vol. 47, no. 2, pp. 317–346, 2005.

[31] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. McCormick, and J. Ruge, "Adaptive algebraic multigrid," *SIAM Journal on Scientific Computing*, vol. 27, no. 4, pp. 1261–1286, 2006.

[32] D. A. Bini, G. Latouche, and B. Meini, *Numerical Methods for Structured Markov Chains*, Oxford University Press, London, UK, 2005.

[33] P. Vaněk, J. Mandel, and M. Brezina, "Algebraic multigrid on unstructured meshes," Technical Report X, Center for Computational Mathematics, Mathematics Department, 1994.

[34] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematics Science*, SIAM, Philadelphia, Pa, USA, 1987.

[35] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, UK, 1981.

[36] M. Benzi and D. B. Szyld, "Existence and uniqueness of splittings for stationary iterative methods with applications to alternating methods," *Numerische Mathematik*, vol. 76, no. 3, pp. 309–321, 1997.

[37] I. Marek and P. Mayer, "Convergence theory of some classes of iterative aggregation/disaggregation methods for computing stationary probability vectors of stochastic matrices," *Linear Algebra and Its Applications*, vol. 363, pp. 177–200, 2003.

[38] E. Virnik, "An algebraic multigrid preconditioner for a class of singular M-matrices," *SIAM Journal on Scientific Computing*, vol. 29, no. 5, pp. 1982–1991, 2007.

[39] W. O. Yuen, W. K. Ching, and M. K. Ng, "A hybrid algorithm for queueing systems," *Calcolo*, vol. 41, no. 3, pp. 139–151, 2004.

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

[41] A. N. Langville and C. D. Meyer, *Google's PageRank and beyond: the science of search engine rankings*, Princeton University Press, Princeton, NJ, USA, 2006.

[42] A. N. Langville and C. D. Meyer, "Deeper inside PageRank," *Internet Mathematics*, vol. 1, no. 3, pp. 335–380, 2004.

[43] A. N. Langville and C. D. Meyer, "A reordering for the PageRank problem," *SIAM Journal on Scientific Computing*, vol. 27, no. 6, pp. 2112–2120, 2006.

[44] I. C. F. Ipsen and T. M. Selee, "PageRank computation, with special attention to dangling nodes," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 4, pp. 1281–1296, 2007.