Numerical Algorithms (2022) 89:1845–1863
https://doi.org/10.1007/s11075-021-01175-w

ORIGINAL PAPER

A Hessenberg-type algorithm for computing PageRank Problems

Xian-Ming Gu[1,2] · Siu-Long Lei[3] · Ke Zhang[4] · Zhao-Li Shen[5] · Chun Wen[6] · Bruno Carpentieri[7]

Received: 7 November 2019 / Accepted: 20 July 2021 / Published online: 9 August 2021

Abstract
PageRank is a widespread model for analysing the relative relevance of nodes within large graphs arising in several applications. In the current paper, we present a cost-effective Hessenberg-type method built upon the Hessenberg process for the solution of difficult PageRank problems. The new method is very competitive with other popular algorithms in this field, such as Arnoldi-type methods, especially when the damping factor is close to 1 and the dimension of the search subspace is large. The convergence and the complexity of the proposed algorithm are investigated. Numerical experiments are reported to show the efficiency of the new solver for practical PageRank computations.

Keywords PageRank vector · Hessenberg process · Arnoldi process · Krylov subspace method · Ritz values

Mathematics Subject Classification (2010) 65F15 · 65F10 · 65Y20

1 Introduction

The PageRank model was originally introduced by S. Brin and L. Page in 1999 [1] to develop fast web search engines, and then studied and enhanced in a vast number of research papers (see, e.g., [2–7]). The model provides a powerful network centrality measure to identify the most important nodes within large graphs arising in several applications fields, such as in chemistry, bioinformatics, neuroscience, and bibliometrics [8]. In the original Web problem, the PageRank algorithm determines the ranking of each Web page by computing the stationary probability vector of a random walking process on the Web link graph, which is a directed graph representing

✉ Siu-Long Lei
sllei@um.edu.mo

Extended author information available on the last page of the article.

the linking structure of the Web [7, 9]. The Web link graph is a binary matrix $G \in \mathbb{N}^{n\times n}$, where n denotes the number of pages, such that $G(i, j) = 1$ when page j has a link pointing to page i, and $G(i, j) = 0$ otherwise. From a linear algebra viewpoint, the algorithm finds the vector $\boldsymbol{x}$ that satisfies

$$A\boldsymbol{x} = \boldsymbol{x}, \quad \|\boldsymbol{x}\|_1 = 1, \quad \boldsymbol{x} > 0, \tag{1.1}$$

that is, it finds the principal unit positive eigenvector $\boldsymbol{x}$ [9] of the Google matrix

$$A = \alpha(P + \boldsymbol{v}\boldsymbol{d}^\top) + (1 - \alpha)\boldsymbol{v}\boldsymbol{e}^\top. \tag{1.2}$$

In (1.2), matrix $P \in \mathbb{R}^{n\times n}$ is called the *transition matrix* with respect to the random walking process, and is defined as

$$P(i, j) = \begin{cases} \dfrac{1}{\sum_{k=1}^{n} G(k, j)}, & \text{if } G(i, j) = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{1.3}$$

The *damping factor* $0 < \alpha < 1$ defines the probability that random Web surfers choose a random link from the page they are visiting [25]. The *teleporting vector* $\boldsymbol{v} = [v_1, \cdots, v_n]^\top \in \mathbb{R}^{n\times 1}$ ($\boldsymbol{v} \geq 0$ and $\|\boldsymbol{v}\|_1 = 1$) defines the probability v_i that the Web surfer jumps to an external page i. Finally, $\boldsymbol{d} \in \mathbb{N}^{n\times 1}$ is such that $\boldsymbol{d}(i) = 1$ if page i has no hyperlink and 0 otherwise, and $\boldsymbol{e} = [1, 1, \ldots, 1]^\top \in \mathbb{N}^{n\times 1}$.

The value of the damping factor α plays an important role in the PageRank model. Theoretically, it represents an upper bound $0 < |\lambda_2| \leq \alpha < 1$ for the second largest eigenvalue, λ_2, of A. Further properties of the Google matrix can be found in [2, 4–7, 10]. For low values of α (e.g., $\alpha = 0.85$), λ_2 is well separated from the largest eigenvalue of A, which is $\lambda_1 = 1$, ensuring rapid convergence of the power algorithm applied to problem (1.1). On the other hand, convergence tends to slow down noticeably when α is very close to 1, requiring more robust algorithms than the simple power method. Some computational approaches proposed in the literature include Monte Carlo methods [11], adaptive algorithms [3, 12], extrapolation techniques [2, 7, 13], singular value decompositions [19–21] reordering [14, 15] and inner-outer solution strategies [16].

A significant amount of work has been devoted in the last years to the use of Krylov subspace methods based on the Arnoldi decomposition [17, 18] for large PageRank computations, mainly due to their memory efficiency and attractive inherent parallelism. Golub and Greif extended the refined Arnoldi procedure to PageRank by forcing a relevant shift to be 1, being able to circumvent the drawbacks due to complex arithmetic and showing overall very good algorithmic efficiency [22]. Many techniques have attempted to combine the conventional Arnoldi method and the power algorithm to produce faster solvers, e.g., the Power-Arnoldi [23–25], the Arnoldi-Extrapolation [26], and the Arnoldi-Inout [27] methods. In the technique proposed in [28], the weighted least squares problem is changed adaptively according to the component of the residual. Then, the generalized Arnoldi method is used to compute the approximate PageRank vector. However, when the dimension of the Krylov subspace is large, Arnoldi-based solvers tend to become very expensive in terms of memory and computational costs; on the other hand, if the dimension of the Krylov subspace is low, they sometimes fail to accelerate the basic

power method, especially when the damping factor is high [22–25]. Similarly to the restarted GMRES algorithm [31], they may stagnate in many circumstances [32].

Motivated by costs considerations, other work developed PageRank solvers based on the Bi-Lanczos orthogonalization procedure [18, pp. 139-145] (see, e.g., [29, 30]) instead of Arnoldi. In this paper, we look in particular at the Hessenberg reduction process [33–36] that was introduced by K. Hessenberg in 1940 [33], and revived recently to establish a number of cost-effective Krylov subspace solvers for sparse matrix systems, due to its lower arithmetic and storage requirements. The method has been extended to compute the characteristic polynomial of matrices [33, 34, 37], to solve general nonsymmetric linear systems [17, 35, 36, 38, 39], including systems with multiple right-hand sides [40–44] and multi-shifted coefficient matrices [45–47], other types of matrix equations [42, 48, 49], the action of a matrix function $f(A)\boldsymbol{v}$ [47], and other related problems [51]. Theoretical and numerical studies have investigated the mathematical properties of the Hessenberg process, especially in relation with the more conventional Arnoldi procedure. The Arnoldi method was first introduced in 1951 as a means of reducing a dense matrix A into a Hessenberg form by unitary transformations, whereas the Hessenberg process applies *similarity transformations* [50] and is more suitable for parallel computing. In his paper, Arnoldi hinted that the eigenvalues of the Hessenberg matrix obtained after $k \ll n$ steps, where n is the size of A, could provide accurate approximations of some eigenvalue of A. It was later discovered that this strategy can lead to efficient techniques for approximating eigenvalues of large sparse matrices. In the current work, following a similar development, we modify the Hessenberg process to establish a new family of eigenvalue solvers. We combine the new solvers with the refined and explicitly restarted techniques introduced in [19, 22] to compute realistic PageRank problems. Finally, we analyze their convergence behavior and computational complexity.

The rest of this paper is organized as follows. In Section 2, the Hessenberg process is introduced and a novel family of eigenvalue solvers based on this procedure is described. Moreover, theoretical aspects of such eigenvalue solvers are highlighted in comparison with the classical Arnoldi-like methods. In Section 3, we derive the Hessenberg-type method with explicit restarting and refined techniques for computing PageRank. Both the convergence behavior and the computational cost of the proposed method are discussed. Numerical results in Section 4 show the effectiveness of the proposed algorithm, also against other popular PageRank algorithms. In Section 5, we present some conclusions arising from our study.

2 The Hessenberg process with applications to eigenvalue computations

In this section, we briefly review the Hessenberg procedure that is at the basis of our development. We recall some fundamental properties of the algorithm and then we describe a Hessenberg-based projection technique for computing eigenvalues of large nonsymmetric matrices. Our theoretical analysis demonstrates the feasibility of the method, showing some computational advantages over the more conventional Arnoldi procedure.

2.1 The Hessenberg process

The Hessenberg process is an an oblique projection technique that reduces a given nonsymmetric matrix $A \in \mathbb{R}^{n\times n}$ to a Hessenberg form [34, pp. 377-381; 45]. Originally, the method was described as a way to compute the characteristic polynomial of a matrix [33]. The basic procedure is presented in Algorithm 1, where a pivoting strategy is included to ensure numerical stability.

Algorithm 1 The Hessenberg procedure with pivoting strategy

1: Initialize the permutation vector $p = [1, 2, \ldots, n]^\top$ and determine i_0 such that $|(v)_{i_0}| = \|v\|_\infty$
2: Compute $\beta = (v)_{i_0}$, then $l_1 = v/\beta$ and $p(1) \leftrightarrow p(i_0)$, where "$\leftrightarrow$" is used to swap contents [35].
3: **for** $j = 1, 2, \ldots, m$, **do**
4: Compute $u = Al_j$
5: **for** $i = 1, 2, \ldots, j$, **do**
6: $h_{i,j} = (u)_{p(i)}$
7: $u = u - h_{i,j} l_i$
8: **end for**
9: **if** ($j < n$ and $u \neq 0$) **then**
10: Determine $i_0 \in \{j+1, ..., n\}$ such that $|(u)_{p(i_0)}| = \|(u)_{p(j+1):p(n)}\|_\infty$;
11: $h_{j+1,j} = (u)_{p(i_0)}$; $l_{j+1} = u/h_{j+1,j}$; $p(j+1) \leftrightarrow p(i_0)$
12: **else**
13: $h_{j+1,j} = 0$; **Stop** ▷ `Happy breakdown`
14: **end if**
15: **end for**

Let $L_k = [\boldsymbol{l}_1, \ldots, \boldsymbol{l}_m]$ denote a matrix, $\bar{H}_m = \left[h_{i,j}\right]$ be an upper Hessenberg matrix and H_m the submatrix obtained from $\bar{H}_m$ by deleting its last row. Finally, we denote by $\mathcal{P}_k^\top = [\boldsymbol{e}_{p_1}, \boldsymbol{e}_{p_2}, \ldots, \boldsymbol{e}_{p_n}]$ where the scalars p_i's (for $i = 1, \ldots, n$) are defined in Algorithm 1. After k steps of Algorithm 1, the following matrix equation can be easily established,

$$\begin{aligned} AL_k &= L_{k+1}\bar{H}_k \\ &= L_k H_k + h_{k+1,k}\boldsymbol{l}_{k+1}\boldsymbol{e}_k^\top, \end{aligned} \tag{2.1}$$

and $\mathcal{P}_k L_k$ is lower trapezoidal [35, 39]. Unlike Arnoldi, the Hessenberg procedure with pivoting is not guaranteed to be backward stable in finite precision arithmetic [50]. However, the backward error is reported to be small for most practical problems [36, 39, 45]. This is also confirmed by our computational experiences. We did not observe noticeable numerical instabilities due to the non-orthogonality of the Krylov basis in our numerical experiments; see also [45] for a discussion of this topic.

2.2 Approximation of eigenpairs based on the Hessenberg process

Methods to approximate eigenpairs of a large nonsymmetric matrix A usually compute them from the Hessenberg decomposition of A given by (2.1). The upper Hessenberg matrix H_m can be seen as the projection of A onto the Krylov subspace

$$\mathcal{K}_m(A, \boldsymbol{v}) = \text{span}\{\boldsymbol{v}, A\boldsymbol{v}, \ldots, A^{m-1}\boldsymbol{v}\}, \tag{2.2}$$

and the columns of the matrix L_m are a basis of $\mathcal{K}_m(A, \boldsymbol{v})$. Under certain conditions, the eigenvalues of H_m converge to the eigenvalues of A [18, 34, 50]. Various eigenvalue solvers are built upon this idea, differing each other mainly on the type of projection technique that is used to derive the decomposition (2.1), for example, the Arnoldi process, the Bi-Lanczos procedure and the Induced Dimension Reduction (IDR) strategy [52, 53]. The approximate eigenpairs of A are retrieved in the form $(\theta_i,\ \boldsymbol{x}^{(i)} = L_m \boldsymbol{y}^{(i)})$, where $\left(\theta_i, \boldsymbol{y}^{(i)}\right)$ are eigenpairs of the small dimensional matrix H_m, such that

$$H_m \boldsymbol{y}^{(i)} = \theta_i \boldsymbol{y}^{(i)} \quad \text{with } \|\boldsymbol{y}^{(i)}\| = 1, \ \ i = 1, 2, \cdots, m. \tag{2.3}$$

A bound on the residual error can be established directly from (2.1), by writing

$$\begin{aligned} A\boldsymbol{x}^{(i)} - \theta_i \boldsymbol{x}^{(i)} &= AL_m \boldsymbol{y}^{(i)} - \theta_i L_m \boldsymbol{y}^{(i)} \\ &= h_{m+1,m} \boldsymbol{l}_{m+1} \boldsymbol{e}_m^\top \boldsymbol{y}^{(i)}. \end{aligned}$$

If we denote as $[\boldsymbol{y}^{(i)}]_m$ the mth component of the vector $\boldsymbol{y}^{(i)}$, then we obtain

$$\|A\boldsymbol{x}^{(i)} - \theta_i \boldsymbol{x}^{(i)}\| \le |h_{m+1,m}| \|\boldsymbol{l}_{m+1,m}\| \left|[\boldsymbol{y}^{(i)}]_m\right| \tag{2.4}$$

or, if we normalize the vector $\boldsymbol{l}_m$,

$$\|A\boldsymbol{x}^{(i)} - \theta_i \boldsymbol{x}^{(i)}\| \le |h_{m+1,m}| \left|[\boldsymbol{y}^{(i)}]_m\right|. \tag{2.5}$$

This analysis is in line with the results described in [19].

For the "`west0479`" problem, a real-valued 479-by-479 sparse matrix that has both real and complex eigenvalues, in Fig. 1, we plot the eigenvalues of A computed by the MATLAB command `eig` and those of the Hessenberg matrix produced by the IDR($s = 4$) projection technique [52], the Sonneveld pencil [52, 53], the Arnoldi, and the Hessenberg procedures. We clearly see that the Hessenberg procedure can estimate the exterior Ritz values very well, in some cases even slightly more accurately than the Arnoldi procedure. The condition number of the Krylov basis matrix L_m is an effective metric to determine the accuracy of the method used. Figure 2 illustrates that for the Hessenberg process, this condition number does not grow significantly when the dimension of the Krylov subspace increases. The numerical error of the Hessenberg decomposition often remains small.[1] This observation is also exemplified by the stochastic analysis presented in [52, Section 3.3] that can produce (numerical) evidence that the Hessenberg process can be as efficient as the Arnoldi process for practical eigenvalue computations.

[1] Here, we give its executable codes in the website: https://github.com/Hsien-Ming-Ku/PageRank- Hessenberg.

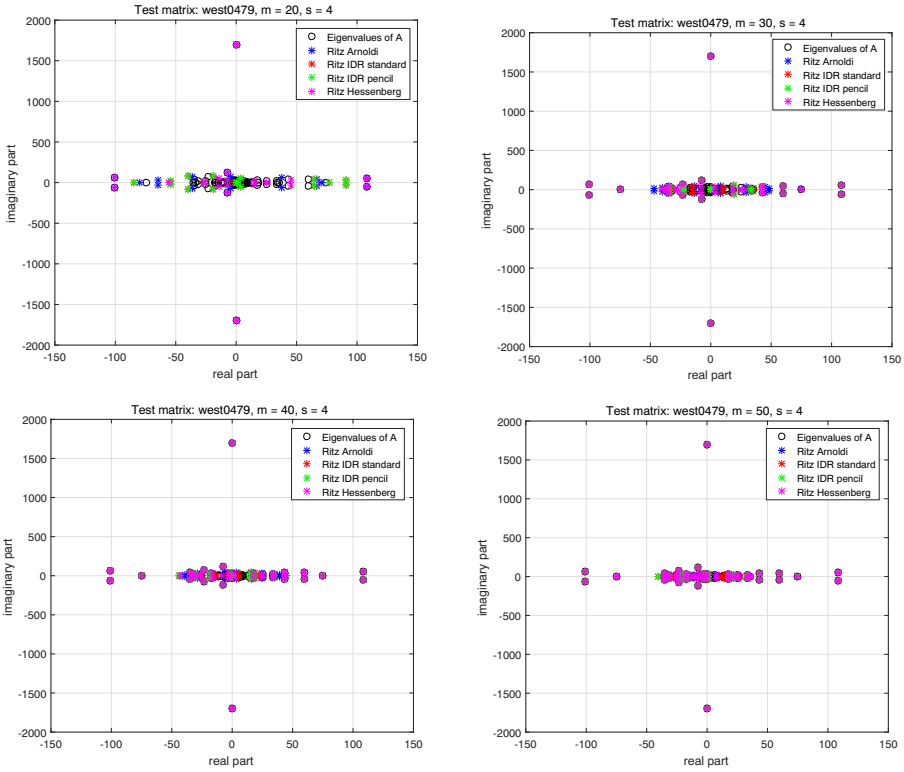

Fig. 1 Plots of the Ritz values generated by the IDR($s = 4$) factorization, the Sonneveld pencil, the Arnoldi and the Hessenberg procedures

2.3 Relation between the Arnoldi and Hessenberg decompositions

In this subsection, we provide more theoretical background supporting the use of the Hessenberg process to approximate effectively eigenpairs of a given nonsymmetric matrix A. The starting point of our analysis is a comparison between the Hessenberg decompositions computed by the Arnoldi and by the Hessenberg procedures. After m steps of the Arnoldi method applied to A, starting with an initial vector $\boldsymbol{v}_0$ and assuming no breakdown, the following Hessenberg decomposition is derived:

$$AV_m = V_m H_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^\top = V_{m+1}\bar{H}_m. \tag{2.6}$$

On the other hand, after m steps of the Hessenberg procedure applied to A with same initial vector $\boldsymbol{l}_0$, the resulting matrix factorization writes as

$$AL_m = L_m H_m^{(h)} + h_{m+1,m}^{(h)}\boldsymbol{l}_{m+1}\boldsymbol{e}_m^\top = L_{m+1}\hat{H}_m^{(h)}. \tag{2.7}$$

Differently from Arnoldi, however, the columns of L_m in (2.7) are not mutually orthogonal. By computing the reduced QR factorization of L_{m+1},

$$L_{m+1} = Q_{m+1}R_{m+1}, \tag{2.8}$$

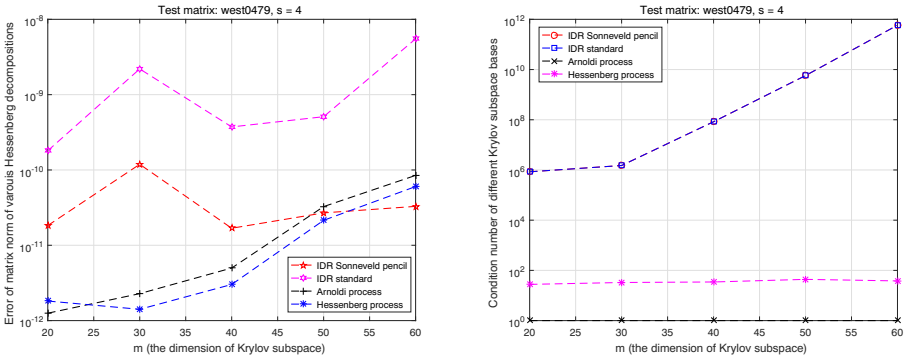

Fig. 2 The quality of different basis matrices; Left: the error of different Hessenberg decompositions; Right: the condition number of basis matrices generated by different Hessenberg decompositions

we can establish the following relation between (2.7) and (2.8):

$$AQ_m = Q_{m+1}R_{m+1}\hat{H}_m^{(h)}R_m^{-1}. \tag{2.9}$$

Due to the uniqueness of the Arnoldi decomposition (see Section 3.3 in [52]), by comparing (2.6) and (2.9) we conclude that $Q_{m+1} = V_{m+1}$ and

$$\bar{H}_m = R_{m+1}\hat{H}_m^{(h)}R_m^{-1}. \tag{2.10}$$

The above result is summarized in the following proposition:

Proposition 2.1 *It follows that*

$$H_m = R_m H_m^{(h)} R_m^{-1} + \frac{h_{m+1,m}^{(h)}}{r_{m,m}}\tilde{\boldsymbol{r}}\boldsymbol{e}_m^\top, \tag{2.11}$$

$$\frac{h_{m+1,m}^{(h)}}{r_{m,m}} = \frac{h_{m+1,m}}{r_{m+1,m+1}}, \tag{2.12}$$

where $\tilde{\boldsymbol{r}} = [r_{i,m+1}]_{i=1}^m$ *is the vector containing the first* m *components of the* $(m+1)$*th column of* R_{m+1}.

In fact, (2.11)–(2.12) can be also found in [38, 52]. According to (2.12), in exact arithmetic both procedures produce upper Hessenberg matrices with the same eigenvalues. If the Arnoldi process terminates successfully (i.e., a happy breakdown situation $h_{m+1,m} = 0$ occurs), so does the Hessenberg procedure ($h_{m+1,m}^{(h)} = 0$). On the other hand, a direct consequence of Proposition 2.1 is that the Ritz values produced by the Arnoldi and by the Hessenberg processes are not the same. The condition number of the Krylov basis matrix L_m, which is the same as the condition number of matrix R_m, gives a clear indication of the accuracy of the eigenvalues of $H_m^{(h)}$ compared to those resulting from the Arnoldi process.

In conclusion, it can be expected that the Hessenberg process can produce feasible approximations of eigenpairs of large nonsymmetric matrices matrix.

3 A Hessenberg-type algorithm for computing PageRank

In this section, we will propose a Hessenberg-based algorithm to compute the PageRank vector, that is the positive unit eigenvector corresponding to the largest eigenvalue of the Google matrix. Golub and Greif suggested that the explicitly restarted Arnoldi process for computing eigenvalues and eigenvectors should be implemented in complex arithmetic, thus it is not suitable (it needs to be refined) to compute the PageRank vector [22]. Moreover, since the Hessenberg process is similar in nature to Arnoldi, except that it produces a non-orthogonal basis of the Krylov subspace, we follow a similar development to the *refined Arnoldi method* for PageRank problems. In other words, the solver described in this section may be called the *refined Hessenberg method* for PageRank. However, we do not approximate the eigenvectors of A from those of H_m (the so-called Ritz-like vectors). Instead, we compute the refined Ritz-like vectors, i.e., the singular vectors associated with the smallest singular values of $A - \theta_i I$ [19, 22], where $\{\theta_i\}_{i=1}^{m}$ are named the Ritz-like values; cf. (2.3). The Hessenberg-based method enjoys similar numerical properties to the Arnoldi-based variant: an effective separation of the eigenvectors is ensured, complex arithmetic is avoided by using a shift equal to 1 (due to the fact that the largest eigenvalue of the Google matrix is known), the smallest singular value converges more smoothly to zero than the largest Ritz value to 1 [22, Section 3]. The Hessenberg-type method for computing the PageRank vector is presented in Algorithm 2. The following convergence result can be established after each cycle of m iterations of Algorithm 2.

Algorithm 2 The Hessenberg-type algorithm for computing PageRank

Input: the Google matrix $A \in \mathbb{R}^{n\times n}$, initial guess $\boldsymbol{q}_0$ and m
Output: the approximate PageRank vector $\boldsymbol{q}$

Choose q_0 and determine i_0 such that $|(\boldsymbol{q}_0)_{i_0}| = \|\boldsymbol{q}_0\|_\infty$, m is the dimension of search subspace
Compute $\beta = (\boldsymbol{q}_0)_{i_0}$, then $\boldsymbol{q}_0 = \boldsymbol{q}_0/\beta$
for $\ell = 1, 2, \ldots,$ until convergence **do**
    Compute L_m and $H_{m+1,m}$ via running Algorithm 1with the inputs $A,\ m,\ \boldsymbol{q}_0$
    Compute the singular value decomposition: $H_{m+1,m} - [I_m; \mathbf{0}] = U\Gamma V^\top$
    Compute $\boldsymbol{q} = L_m \boldsymbol{v}_m$
    Compute $\boldsymbol{r} = \sigma_m L_{m+1} \boldsymbol{u}_m$
    if $\|\boldsymbol{r}\|_1/\|\boldsymbol{q}\|_1 < tol$ **then**
        Stop and exit
    else
        Set $\boldsymbol{q}_0 = \boldsymbol{q}$
end if
end for

Theorem 3.1 *Let $Q_m = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_m]$ be the matrix obtained from running m-steps of either the Arnoldi or the Hessenberg procedures applied to A starting from*

an initial vector $\boldsymbol{q}_0$, then the Hessenberg matrix decomposition can be uniformly written as

$$\begin{aligned} AQ_m &= Q_m H_m + h_{m+1,m}\boldsymbol{q}_{m+1}\boldsymbol{e}_m^\top \\ &= Q_{m+1}\bar{H}_m. \end{aligned} \tag{3.1}$$

Denote as σ_m the smallest singular value of $H_{m+1,m} - [I_m; 0]$, then $\boldsymbol{v}_m$ at Line 6 of Algorithm 2 is the corresponding right singular vector, and $Q_m\boldsymbol{v}_m$ is the approximate PageRank vector. The residual vector at each restarting cycle can be computed as $\boldsymbol{r} = \sigma_m Q_{m+1}\boldsymbol{u}_m$.

Proof According to (3.1) and Algorithm 2, it follows that

$$\begin{aligned} \boldsymbol{r} &= A\boldsymbol{q}_m - \boldsymbol{q}_m = AQ_m\boldsymbol{v}_m - Q_m\boldsymbol{v}_m \\ &= Q_{m+1}H_{m+1,m}\boldsymbol{v}_m - Q_m\boldsymbol{v}_m \\ &= Q_{m+1}\left[H_{m+1,m} - \begin{pmatrix} I_m \\ \boldsymbol{0} \end{pmatrix}\right]\boldsymbol{v}_m \\ &= \sigma_m Q_{m+1}\boldsymbol{u}_m, \end{aligned} \tag{3.2}$$

where $\boldsymbol{q}_m$ is an approximate PageRank vector. Thus, the assertion is verified. □

Below, we give the norms of the residual vectors computed by the Arnoldi and by the Hessenberg procedures, respectively:

$$\|\boldsymbol{r}\|_1 = \sigma_m \|Q_{m+1}\boldsymbol{u}_m\|_1, \quad \text{where} \quad Q_m^\top Q_m : \begin{cases} = I_m & \text{(Arnoldi process)}, \\ \neq I_m & \text{(Hessenberg process)}, \end{cases} \tag{3.3}$$

and

$$\|\boldsymbol{r}\|_2 = \begin{cases} \sigma_m, & Q_m^\top Q_m = I_m \quad \text{(Arnoldi process)}, \\ \sigma_m \|Q_{m+1}\boldsymbol{u}_m\|_2, & Q_m^\top Q_m \neq I_m \quad \text{(Hessenberg process)}. \end{cases} \tag{3.4}$$

Although the 2-norm of the Arnoldi residual vectors is much cheaper than the 2-norm of the Hessenberg residual vectors, it should be noted that for PageRank computations it is generally recommended to use the 1-norm; refer, e.g., to [2, 25, 26]. Therefore, the computational complexity of the stopping criterion (at line 8 of Algorithm 2) is almost the same for both methods [24, 28, 30].

Before we end this section, we provide estimates on the storage requirement and the computational complexity of the new algorithm, also compared against other popular methods.

Table 1 shows the memory required in addition to A for running k iterations of the power method (referred to as `Power` in the table), the power method with quadratic extrapolation (called as `QE-power`), the Arnoldi-type method (abbreviated as `Arnoldi`), the adaptively accelerated Arnoldi method (called as `A-Arnoldi`), and the Hessenberg-type method (abbreviated as `Hessenberg`). Here, $\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{u}$, and $\boldsymbol{r}$ are intermediate working vectors used at the kth step, and Q_k denotes the k orthonormal vectors in the modified Gram-Schmidt process. Analogously, L_k denotes the $n \times k$ non-orthonormal matrix in the variant of the LU-like factorization process.

Table 2 shows the computational workloads required to execute one cycle of each different iterative algorithm. Here, N_z represents the number of nonzero entries of

Table 1 Memory requirement for running k iterations of different PageRank algorithms

Algorithm	dim(n)	dim(k)	total
`Power`	$\boldsymbol{x}_k,\ \boldsymbol{x}_{k-1}$	–	$2n$
`QE-Power`	$\boldsymbol{x}_k,\ \boldsymbol{x}_{k-1},\ \boldsymbol{x}_{k-2},\ \boldsymbol{x}_{k-3}$	–	$4n$
`Inner-Outer`	$\boldsymbol{x},\ \boldsymbol{y},\ \boldsymbol{f}$	–	$3n$
`Arnoldi`	$Q_k,\ \boldsymbol{w}$	H_k [1]	$(k+1)n+k^2/2+2k$
`Hessenberg`	$L_k,\ \boldsymbol{u}$	H_k	$(k+1)n+k^2/2+2k$
`A-Arnoldi`	$Q_k,\ \boldsymbol{w},\ \boldsymbol{r}$	H_k	$(k+2)n+k^2/2+2k$

[1]To minimize the memory requirements, both L_k and $\bar{H}_k$ can be written into the same array as A. Hence, the Hessenberg process requires slightly less storage than the Arnoldi's procedure; refer to [35, 39] for discussions on this issue

matrix A. In fact, both `Arnoldi` and `A-Arnoldi` for computing PageRank enjoy the similar pseudo-code of Algorithm 2, the only difference is to choose the Hessenberg, Arnoldi, or generalized Arnoldi process at Line 4 of Algorithm 2. It implies that we need to compare the cost of the Hessenberg process with both the Arnoldi and generalized Arnoldi procedures. We can see that one cycle of the Hessenberg-type method is cheaper than for Arnoldi and for the generalized Arnoldi methods. Thus its use can be computationally attractive for large PageRank computation. The convergence performance of the Hessenberg method is also superior to Arnoldi algorithms (i.e., `Arnoldi` and `A-Arnoldi`), as proved numerically in the next section. Besides matrix-vector multiplications, also the computation of vector norms and `SAXPY`[2] (which stands for "Single-Precision A·X Plus Y" and is a combination of the scalar multiplication and vector addition) operations determines the total computational cost of these three algorithms. Overall, when m increases, the cost of each cycle increases too but the total number of iterations decreases. The optimal value of the restart parameter that minimizes the total solution time remains problem dependent, and this issue will be examined in our numerical experiments section.

4 Numerical experiments

In this section, numerical experiments are reported to illustrate the efficiency of the Hessenberg-based PageRank algorithm presented in this paper also against other popular PageRank algorithms that are the conventional power method including its variants with quadratic extrapolation [2] and with linear extrapolation [13], the Arnoldi-based PageRank method introduced in [22], and the adaptively accelerated Arnoldi method [28][3]. The performance of these methods were assessed in terms of

[2]See the details from https://developer.nvidia.com/blog/six-ways-saxpy/.

[3]Numerical results with the IDR(s)-based PageRank method are omitted due to its unsatisfactory performance for large values of s and m. However, the MATLAB code of the IDR(s)-based PageRank method is still included in our GitHub repository: https://github.com/Hsien-Ming-Ku/PageRank-Hessenberg for testing purposes.

Table 2 Computational cost of one cycle of different algorithms for computing PageRank

operation	`Arnoldi`	`Hessenberg`	`A-Arnoldi`
matrix-vector product (see Line 4 of Algorithm 1)	$2mN_z$	$2mN_z$	$2mN_z$
inner product (e.g., see Line 6 of `Arnoldi`)	$m(m+1)n$	0	$3m(m+1)n/2$
$(\boldsymbol{u})_{\boldsymbol{p}(i)}$ (see Line 6 of Algorithm 1)	0	mn	0
`SAXPY`: $\boldsymbol{x}+\alpha\boldsymbol{y}$ (see Line 7 of Algorithm 1)	$m(m+1)n$	$m(m+1)n$	$m(m+1)n$
$\|\boldsymbol{u}\|_2$ or $\|\boldsymbol{u}\|_\infty$ (e.g., see Line 10 of Algorithm 1)	$2mn$	mn	$3mn$
vector scaling (see Line 11 of Algorithm 1)	mn	mn	mn

number of matrix-vector products (or, equivalently, number of iteration steps for the first three algorithms) and elapsed CPU time (in seconds) required to achieve convergence to a prescribed accuracy. Unless otherwise stated, the stopping criterion used in our runs was

$$\frac{\|A\boldsymbol{q}-\boldsymbol{q}\|_1}{\|\boldsymbol{q}\|_1} < tol = 10^{-8},$$

and all the algorithms were started from the initial vector $\boldsymbol{q}_0 = \boldsymbol{e}/\|\boldsymbol{e}\|_1$, where $\boldsymbol{e} = [1,\ldots,1]^T$. According to Theorem 3.1, the cost of implementing the above stopping criterion can be alleviated for both Arnoldi- and Hessenberg-type methods, since $A\boldsymbol{q}-\boldsymbol{q} = \sigma_m Q_{m+1}\boldsymbol{u}_m$ and the computation of $\sigma_m Q_{m+1}\boldsymbol{u}_m$ is actually cheaper than that of $A\boldsymbol{q}-\boldsymbol{q}$, when m is not large. In our experiments with the method denoted as `QE-power`, the quadratic extrapolation technique was applied every five iterations, following the observations made in [2]. The experiments were run in MATLAB R2017b (64 bit) on a computer equipped with Intel Core i5-8250U processor (CPU 1.60∼1.80 GHz), 8 GB of RAM using double precision floating point arithmetic with machine epsilon set equal to 10^{-16}.

The matrix problems used in our runs are obtained from the *SuiteSparse Matrix Collection*, which is available online at https://sparse.tamu.edu/. In Table 3, we describe the characteristics of our test matrices, including number of rows (n), number of nonzeros (N_z), number of zero columns ($zcol$), average nonzeros of every row (aN_z), and density (den) which is defined as

$$den = \frac{N_z}{n\times n}\times 100.$$

Table 3 The characteristic of test matrices

Matrix ID	Matrix name	Size	Nonzeros	$zcol$	aN_z	den
I	`soc-Slashdot0902`	82,168	948,464	3,727	11.543	1.405×10^{-2}
II	`amazon0312`	400,727	3,200,440	12,353	7.987	1.993×10^{-3}
III	`amazon-2008`	735,323	5,158,388	88,557	7.015	9.540×10^{-4}
IV	`wiki-Talk`	2,394,385	5,021,410	2,246,783	2.097	8.759×10^{-5}
V	`ljournal-2008`	5,363,260	79,023,142	545,626	14.734	2.747×10^{-4}

Table 4 Number of iterations required by the Arnoldi-, GArnoldi- and Hessenberg-based algorithms with different restart numbers. (problem `soc-Slashdot0902` and $tol = 10^{-7}$)

α	$m = 6$			$m = 7$			$m = 8$			$m = 9$			$m = 10$		
	A-P	GA-P	H-P	A-P	GA-P	H-P	A-P	GA-P	H-P	A-P	GA-P	H-P	A-P	GA-P	H-P
0.85	5	5	5	4	4	4	3	3	4	3	3	3	3	3	3
0.90	6	6	7	5	5	5	4	4	4	4	4	4	3	3	4
0.95	9	8	9	7	7	8	5	5	6	5	5	6	4	4	4
0.99	16	15	21	15	13	15	11	10	11	9	8	10	7	7	8

Here, the number of zero columns corresponds to the number of dangling nodes. The largest problem in our set has size 5,363,260 and 79,023,142 nonzeros.

4.1 Choice of the restart value *m*

First, we investigate the effect of the restart parameter m on the convergence of the `Arnoldi (A-P)`, `A-Arnoldi (GA-P)` and `Hessenberg (H-P)` methods in terms of number of iterations and elapsed CPU time, since this parameter may noticeably affect the performance of Krylov subspace-based methods. The results are presented numerically in Tables 4, 5 and 6. In Figs. 3 and 4, for the test matrix '`soc-Slashdot0902`' we plot the curves showing the total CPU time versus m for different damping factors and tol's values.

According to the results reported in Tables 4–6, the number of iterations required to converge by these three algorithms tends to decrease for higher restart numbers m, especially for larger damping factors. This behaviour is expected because larger search spaces may provide better approximations. On the other hand, the total solution time of the three methods is not significantly reduced. As mentioned in Section 3, and explained in Tables 1–2, the storage requirement and the computational cost of one Arnoldi and Hessenberg cycles increase with m. However, it should be noted that `Hessenberg` is more cost effective than both `Arnoldi` and `A-Arnoldi` for larger m. In our numerical experiments, we choose the restart numbers equal to

Table 5 Number of iterations required by the Arnoldi-, GArnoldi- and Hessenberg-type algorithms with different restart numbers (problem `soc-Slashdot0902` and $tol = 10^{-8}$).

α	$m = 6$			$m = 7$			$m = 8$			$m = 9$			$m = 10$		
	A-P	GA-P	H-P	A-P	GA-P	H-P	A-P	GA-P	H-P	A-P	GA-P	H-P	A-P	GA-P	H-P
0.85	6	5	6	5	4	5	4	4	4	4	4	4	3	3	3
0.90	7	7	7	6	5	6	5	5	5	4	4	5	4	4	4
0.95	11	9	10	8	7	9	6	6	7	6	5	6	5	5	5
0.99	18	19	24	17	15	16	14	11	12	11	10	12	9	8	9

Table 6 Matrix-vector products and CPU time in seconds versus damping factors

ID	α	Power		Power-Tan		QE-Power		Arnoldi $m=8$		Arnoldi $m=10$		A-Arnoldi $m=8$		A-Arnoldi $m=10$		Hessenberg $m=8$		Hessenberg $m=10$	
		Mvp	CPU	Mvp	CPU	Mvp	CPU	Mvp	CPU	Mvp	CPU	Mvp	CPU	Mvp	CPU	Mvp	CPU	Mvp	CPU
I	0.85	82	0.216	83	0.228	52	0.168	32	0.152	30	0.136	32	0.122	30	0.132	32	0.122	30	**0.118**
	0.90	125	0.324	126	0.333	52	0.159	40	0.186	40	0.187	40	0.159	40	0.163	40	**0.144**	40	0.146
	0.95	241	0.628	243	0.634	98	0.298	48	0.192	50	0.206	48	0.195	50	0.202	56	0.187	50	**0.176**
	0.99	824	2.089	832	2.105	169	0.504	112	0.419	90	0.350	88	0.327	80	0.308	96	0.319	90	**0.303**
II	0.85	80	1.032	80	1.035	66	1.014	40	0.951	40	0.943	40	0.996	40	1.089	48	0.854	40	**0.726**
	0.90	120	1.553	121	1.562	98	1.522	56	1.253	40	1.187	48	1.166	40	1.292	56	0.993	40	**0.918**
	0.95	234	3.033	235	3.041	152	2.422	80	1.779	56	1.681	72	1.732	70	1.802	88	1.554	80	**1.393**
	0.99	1071	13.80	1076	14.16	391	6.196	224	4.917	190	4.404	176	4.212	160	4.095	216	**3.798**	250	4.387
III	0.85	79	1.782	80	1.821	64	1.779	40	1.633	40	1.704	40	1.761	40	1.896	40	**1.315**	40	1.321
	0.90	119	2.782	120	2.839	95	2.659	48	1.981	50	2.304	48	2.130	50	2.376	56	1.828	50	**1.627**
	0.95	235	5.248	236	5.289	152	4.215	72	2.889	70	3.034	72	3.152	70	3.283	72	**2.345**	80	2.632
	0.99	1082	24.85	1084	25.02	411	11.79	208	8.428	180	7.779	152	6.817	150	7.107	176	**5.843**	190	6.358
IV	0.85	67	4.798	67	4.819	52	4.727	32	4.321	30	4.403	32	4.683	30	4.808	32	**3.327**	40	4.264
	0.90	104	7.344	104	7.464	52	4.770	40	5.352	40	5.838	40	5.948	40	6.266	40	**4.217**	40	4.299
	0.95	214	15.11	214	15.27	84	7.327	56	7.413	60	8.736	48	7.108	60	9.532	64	6.768	50	**5.335**
	0.99	1066	20.21	1068	75.96	175	**15.57**	156	20.21	150	21.47	112	16.24	110	17.08	160	16.69	150	15.93
V	0.85	80	36.59	79	36.47	61	30.61	48	28.53	50	30.88	48	29.79	50	32.38	56	29.44	50	**26.87**
	0.90	122	54.77	122	55.12	92	45.32	64	38.26	60	36.66	64	39.21	60	38.65	72	37.62	60	**31.82**
	0.95	247	110.9	244	109.5	148	71.61	104	61.85	90	55.92	96	59.05	90	57.84	104	54.59	90	**47.97**
	0.99	1153	794.7	1061	529.1	630	322.6	352	214.7	240	145.8	216	132.3	190	121.4	248	127.7	220	**115.9**

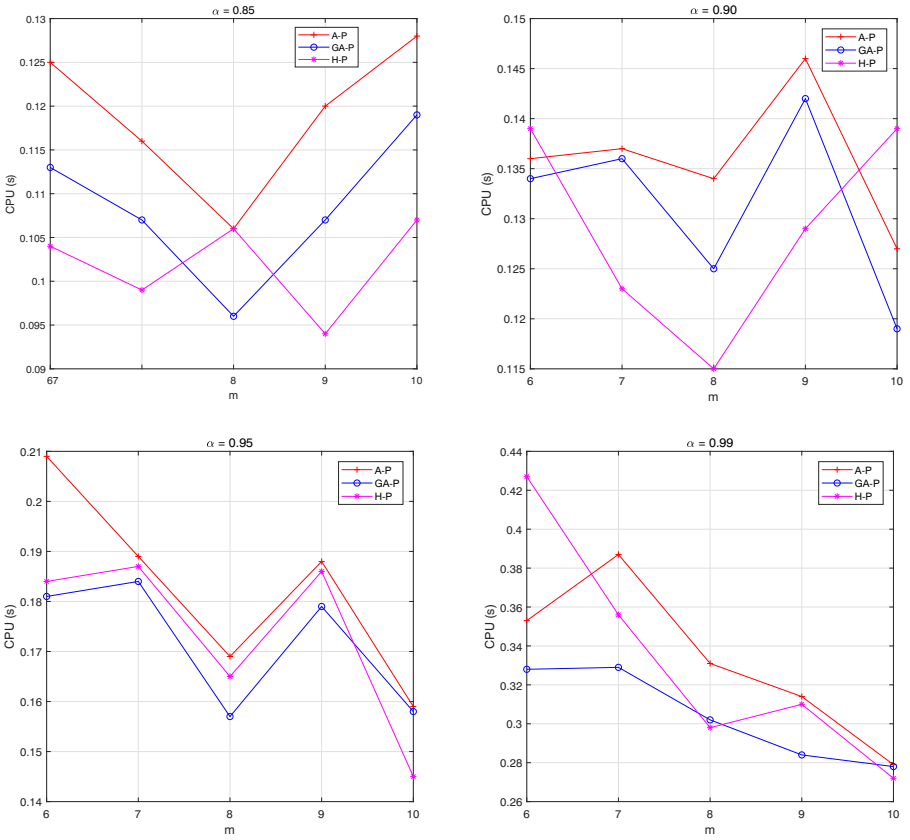

Fig. 3 Plot of the elapsed CPU time in seconds versus the restart number (i.e., m) for the test problem '`soc-Slashdot0902`' using $tol = 10^{-7}$

$m = 8, 10$ due to memory constraints[4], since the number of iterations and the total elapsed CPU time are still acceptable. It may be worth investigating techniques that can effectively reduce the dimension of the Krylov subspace for the `Hessenberg` method, e.g., by optimizing the choice of the starting vector [52] and by utilizing vector extrapolations [13], but this analysis is beyond the scope of this study.

4.2 Effect of damping factors on the CPU time and the number of iterations

For the five matrix problems listed in Table 3, we report on the number of matrix-vector products (`Mvp` in short) and the elapsed CPU time of the power method, the power methods with quadratic extrapolation and with linear extrapolation, the Arnoldi-type method, the adaptively accelerated Arnoldi method and the Hessenberg-type method for various values of the damping factor α ranging from 0.85 to 0.99.

[4]In fact, the restart number is often chosen in the PageRank literature as $m \leq 10$ [27, 30, 31].

We can see from the results of Table 6 that the power method accelerated by quadratic extrapolation outperforms the conventional power method and its linearly extrapolated variant, while in most cases our Hessenberg-based solver is the fastest method in terms of elapsed CPU time, with the only exception for matrix 'IV' using $\alpha = 0.99$. It can be observed that `Arnoldi` is more cost-effective than `A-Arnoldi` at equal number of `Mvp`, especially for large problems. This behaviour is in agreement with the cost analysis presented in Table 2. Except these few cases, the `A-Arnoldi` method is still attractive to consider. On the other hand, one observes from Table 6 that the numerical behaviors of the `Arnoldi`, `A-Arnoldi` and `Hessenberg` algorithms relies on the choice of m and α. For example, when m is small, say $m = 8$, these three algorithms are only slightly better than the `Power`, `Power-Tan` and `QE-Power` methods. However, as m and α increase, their improvements become gradually more significant. Unlike the `Arnoldi`, `A-Arnoldi`, and `Hessenberg`, the `Power`, `Power-Tan`, and `QE-Power` methods are simple and their main computational cost is the evaluation

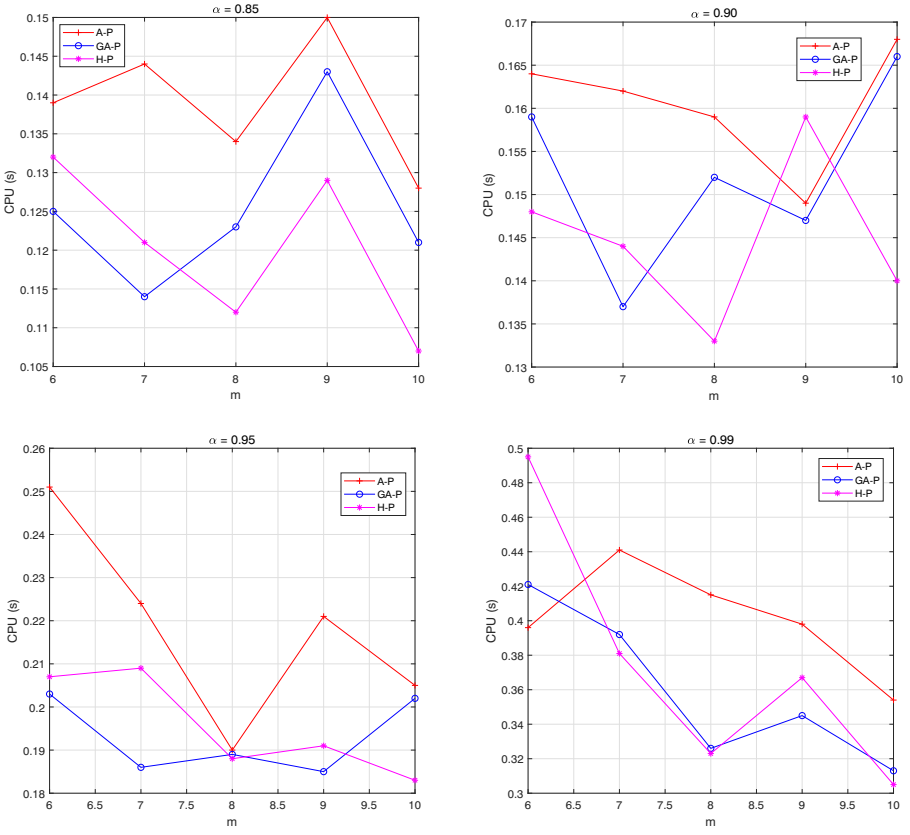

Fig. 4 Plot of the elapsed CPU time in seconds versus the restart number m for the test problem '`soc-Slashdot0902`' using $tol = 10^{-8}$

of matrix-vector products. These characteristics often make them still feasible for computing PageRank when the damping factor α is not large.

In addition, it is interesting to mention that the Hessenberg-type method often needs more `Mvp`'s for convergence than both `Arnoldi` and `A-Arnoldi`, whereas the total CPU time of `Hessenberg` is still less. This is because the `Hessenberg` uses the cheap similarity transformations to reduce the large matrix into the Hessenberg form, whereas the latter two methods use expensive (weighted) unitary transformations.

5 Conclusions

In this paper, we proposed a novel approach for computing the PageRank problem. The proposed method has lower computational cost than both `Arnoldi` and `A-Arnoldi` to find the approximate PageRank vector; thus, it can afford to use higher dimensional Krylov subspaces. Extensive numerical experiments are reported to illustrate the efficiency of the proposed method also compared to other state-of-the-art matrix solvers for this problem class, especially when the damping factor is large. Hence, we conclude that the `Hessenberg` method as well as the `Arnoldi` and `A-Arnoldi` methods can be useful computational tools for practical large-scale PageRank computations.

Future research will focus on the theory of the Hessenberg process and the convergence of the Hessenberg-type algorithm is still required to be further analyzed. In addition, it is interesting to study how to optimize the restart number m and improve the convergence speed of our methods. Moreover, the proposed method can be extended to compute the more general Markov chains [8, 29], e.g., in ProteinRank and CiteRank.

Acknowledgements The authors would like to thank Prof. Zhongxiao Jia for his comments about the strategy used in the refined Arnoldi algorithm. Meanwhile, the authors are grateful to Dr. Reinaldo Astudillo (ASML Holding N.V.) for his kind suggestions about executing the IDR-based Hessenberg decompositions used in Section 2.2.

Funding This research is supported by NSFC (11601323 and 11801463), the Applied Basic Research Program of Sichuan Province (2020YJ0007), and the research grants MYRG2018-00025-FST, MYRG2020-00208-FST from University of Macau. The last author is member of the Gruppo Nazionale per il Calcolo Scientifico (GNCS) of the Istituto Nazionale di Alta Matematica (INdAM) and his work was partially supported by INdAM-GNCS under Progetti di Ricerca 2020.

References

1. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web, Technical Report No. 1999-66, Stanford InfoLab., Jan. 29, 1999, 17 pages. Available online at: http://ilpubs.stanford.edu:8090/422/
2. Kamvar, S.D., Haveliwala, T.H., Manning, C.D., Golub, G.H.: Extrapolation methods for accelerating PageRank computations, in: WWW '03 Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary, May 20-24, 2003, ACM New York, NY (2003): 261–270. https://doi.org/10.1145/775152.775190

3. Kamvar, S., Haveliwala, T., Golub, G.: Adaptive methods for the computation of PageRank, Linear Algebra Appl., 386 (2004): 51–65
4. Langville, A.N., Meyer, C.D.: Deeper inside PageRank, Internet Math., 1(3) (2005): 335–380
5. Langville, A.N., Meyer, C.D.: A survey of eigenvector methods of web information retrieval, SIAM Rev., 47(1) (2005): 135–161
6. Berkhin, P.: A survey on PageRank computing, Internet Math., 2(1) (2005): 73–120
7. Langville, A.N., Meyer, C.D.: Google's PageRank and beyond: the Science of Search Engine Rankings, Princeton University Press, Princeton, NJ (2006)
8. Gleich, D.F.: PageRank beyond the web, SIAM Rev., 57(3) (2015): 321–363
9. Bryan, K., Leise, T.: The 25,000,000,000 eigenvector: the linear algebra behind Google, SIAM Rev., 48(3) (2006): 569–581
10. Cicone, A., Serra-Capizzano, S.: Google PageRanking problem: the model and the analysis, J. Comput. Appl. Math., 234(11) (2010): 3140–3169
11. Avrachenkov, K., Litvak, N., Nemirovsky, D., Osipova, N.: Monte Carlo methods in PageRank computation: when one iteration is sufficient, SIAM J. Numer. Anal., 45(2) (2007): 890–904
12. Liu, W., Li, G., Cheng, J.: Fast PageRank approximation by adaptive sampling, Knowl. Inf. Syst., 42(1) (2015): 127–146
13. Tan, X.: A new extrapolation method for PageRank computations, J. Comput. Appl. Math., 313 (2017): 383–392
14. Langville, A.N., Meyer, C.D.: A reordering for the PageRank problem, SIAM J. Sci. Comput., 27(6) (2006): 2112–2120
15. Lin, Y., Shi, X., Wei, Y.: On computing PageRank via lumping the Google matrix, J. Comput. Appl. Math., 224(2) (2009): 702–708
16. Gleich, D.F., Gray, A.P., Greif, C., Lau, T.: An inner-outer iteration for computing PageRank, SIAM J. Sci. Comput., 32(1) (2010): 349–371
17. Heyouni, M., Sadok, H.: On a variable smoothing procedure for Krylov subspace methods, Linear Algebra Appl., 268 (1998): 131–149
18. Saad, Y.: Numerical methods for large eigenvalue problems (Revised Ed.), SIAM, Philadelphia, PA (2011)
19. Jia, Z.: Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems, Linear Algebra Appl., 259 (1997): 1–23
20. Jia, Z.: Polynomial characterizations of the approximate eigenvectors by the refined Arnoldi method and an implicitly restarted refined Arnoldi algorithm, Linear Algebra Appl., 287(1-3) (1999): 191–214
21. Jia, Z.: A refined subspace iteration algorithm for large sparse eigenproblems, Appl. Numer. Math., 32(1) (2000): 35–52
22. Golub, G.H., Greif, C.: An Arnoldi-type algorithm for computing page rank, BIT, 46(4) (2006): 759–771
23. Wu, G., Wei, Y.: A Power-Arnoldi algorithm for computing PageRank, Numer. Linear Algebra Appl., 14(7) (2007): 521–546
24. Yin, G.-J., Yin, J.-F.: On Arnoldi method accelerating PageRank cmputations, in: Web Information Systems and Mining. WISM 2010 (F.-L. Wang, Z. Gong, X. Luo, J. Lei, eds.), Lecture Notes in Computer Science, vol 6318, Springer, Berlin, Heidelberg (2010): 378–385. https://doi.org/10.1007/978-3-642-16515-3_47
25. Wu, G., Zhang, Y., Wei, Y.: Accelerating the Arnoldi-type algorithm for the PageRank problem and the ProteinRank problem, J. Sci. Comput., 57(1) (2013): 74–104
26. Wu, G., Wei, Y.: An Arnoldi-extrapolation algorithm for computing PageRank, J. Comput. Appl. Math., 234(11) (2010): 3196–3212
27. Gu, C., Wang, W.: An Arnoldi-Inout algorithm for computing PageRank problems, J. Comput. Appl. Math., 309 (2017): 219–229
28. Yin, J.-F., Yin, G.-J., Ng, M.: On adaptively accelerated Arnoldi method for computing PageRank, Numer. Linear Algebra Appl., 19(1) (2012): 73–85
29. Freund, R.W., Hochbruck, M.: On the use of two QMR algorithms for solving singular systems and applications in Markov chain modeling, Numer. Linear Algebra Appl., 1(4) (1994): 403–420
30. Teramoto, K., Nodera, T.: A note on Lanczos algorithm for computing PageRank, in: Forging Connections between Computational Mathematics and Computational Geometry (K. Chen, A. Ravindran, eds.), Springer Proceedings in Mathematics & Statistics, Vol. 124, Springer, Cham, Switzerland (2016): 25–33. https://doi.org/10.5176/2251-1911_CMCGS14.15_3

31. Wu, G., Wang, Y.-C., Jin, X.-Q.: A preconditioned and shifted GMRES algorithm for the PageRank problem with multiple damping factors, SIAM J. Sci. Comput., 34(5) (2012): A2558–A2575
32. Wu, G., Wei, Y.: Arnoldi versus GMRES for computing pageRank: A theoretical contribution to google's pageRank problem, ACM Trans Inf. Syst., 28(3) (2010): 11. https://doi.org/10.1145/1777432. 1777434
33. Hessenberg, K.: Behandlung Linearer Eigenwertaufgaben Mit Hilfe Der Hamilton-Cayleyschen Gleichung, Numerische Verfahren, Bericht 1, Institut Für Praktische Mathematik (IPM), Technische Hochschule Darmstadt. The scanned report and a biographical sketch of Karl Hessenberg's life are available at. http://www.hessenberg.de/karl1.html (1940)
34. Wilkinson, J.H.: The algebraic eigenvalue problem, Clarendon Press, Oxford, UK (1965)
35. Sadok, H.: CMRH: a new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm. Numer. Algorithms **20**(4), 303–321 (1999)
36. Stephens, D.: ELMRES: An oblique projection method to solve sparse non-symmetric linear systems (Ph.D Dissertation), Florida Institute of Technology, Melbourne USA. http://ncsu.edu/hpc/Documents/Publications/gary_howell/stephens.pdf (1999)
37. Householder, A.S., Bauer, F.L.: On certain methods for expanding the characteristic polynomial, Numer. Math., 1(1) (1959): 29–37
38. Sadok, H., Szyld, D.B.: A new look at CMRH and its relation to GMRES, BIT, 52(2) (2012): 485–501
39. Heyouni, M., Sadok, H.: A new implementation of the CMRH method for solving dense linear systems, J. Comput. Appl. Math., 213(2) (2008): 387–399
40. Zhang, K., Gu, C.: Flexible global generalized Hessenberg methods for linear systems with multiple right-hand sides, J. Comput. Appl. Math., 263 (2014): 312–325
41. Heyouni, M.: The global Hessenberg and CMRH methods for linear systems with multiple right-hand sides, Numer. Algorithms, 26(4) (2001): 317–332
42. Heyouni, M., Essai, A.: Matrix Krylov subspace methods for linear systems with multiple right-hand sides, Numer. Algorithms, 40(2) (2005): 137–156
43. Amini, S., Toutounian, F., Gachpazan, M.: The block CMRH method for solving nonsymmetric linear systems with multiple right-hand sides, J. Comput. Appl. Math., 337 (2018): 166–174
44. Amini, S., Toutounian, F.: Weighted and flexible versions of block CMRH method for solving nonsymmetric linear systems with multiple right-hand sides. Comput. Math. Appl. **76**(8), 2011–2021 (2018)
45. Gu, X.-M., Huang, T.-Z., Yin, G., Carpentieri, B., Wen, C., Du, L.: Restarted Hessenberg method for solving shifted nonsymmetric linear systems, J. Comput. Appl. Math., 331 (2018): 166–177
46. Gu, X.-M., Huang, T.-Z., Carpentieri, B., Imakura, A., Zhang, K., Du, L.: Efficient variants of the CMRH method for solving a sequence of multi-shifted non-Hermitian linear systems simultaneously, J. Comput. Appl. Math., 375 (2020): 112788. https://doi.org/10.1016/j.cam.2020.112788
47. Ramezani, Z., Toutounian, F.: Extended and rational Hessenberg methods for the evaluation of matrix functions, BIT, 59(2) (2019): 523–545
48. Addam, M., Heyouni, M., Sadok, H.: The block Hessenberg process for matrix equations, Electron. Trans. Numer. Anal., 46 (2017): 460–473
49. Heyouni, M., Saberi-Movahed, F., Tajaddini, A.: On global Hessenberg based methods for solving Sylvester matrix equations, Comput. Math. Appl., 77(1) (2019): 77–92
50. Businger, P.A.: Reducing a matrix to Hessenberg form, Math. Comp., 23(108) (1969): 819–821
51. Heyouni, M.: Newton Generalized Hessenberg method for solving nonlinear systems of equations, Numer. Algorithms, 21(1-4) (1999): 225–246
52. Astudillo, R., van Gijzen, M.B.: A restarted induced dimension reduction method to approximate eigenpairs of large unsymmetric matrices, J. Comput. Appl. Math., 296 (2016): 24–35
53. Gutknecht, M.H., Zemke, J.-P.M.: Eigenvalue computations based on IDR, SIAM J. Matrix Anal. Appl., 34(2) (2013): 283–311

Affiliations

Xian-Ming Gu[1,2] · Siu-Long Lei[3] · Ke Zhang[4] · Zhao-Li Shen[5] · Chun Wen[6] · Bruno Carpentieri[7]

Xian-Ming Gu
guxianming@live.cn; guxm@swufe.edu.cn

Ke Zhang
xznuzk123@126.com

Zhao-Li Shen
szlxiaoyao@163.com

Chun Wen
wchun17@163.com

Bruno Carpentieri
Bruno.Carpentieri@unibz.it

1 School of Economic Mathematics/Institute of Mathematics, Southwestern University of Finance and Economics, Chengdu, Sichuan 611130, People's Republic of China

2 Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, Nijenborgh 9, P.O. Box 407, 9700 AK Groningen, The Netherlands

3 Department of Mathematics, Faculty of Science and Technology, University of Macau, Avenida da Universidade, Macao, People's Republic of China

4 College of Arts and Sciences, Shanghai Maritime University, Shanghai 201306, People's Republic of China

5 Department of Applied Mathematics, College of Science, Sichuan Agricultural University, Yaan, Sichuan 625014, People's Republic of China

6 School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, People's Republic of China

7 Facoltà di Scienze e Tecnologie Informatiche, Libera Università di Bolzano, Dominikanerplatz 3 - piazza Domenicani, 3 Italy - 39100, Bozen-Bolzano, Italy