symmetry

MDPI

Article

A Preconditioned Variant of the Refined Arnoldi Method for Computing PageRank Eigenvectors

Zhao-Li Shen [1,2,*,†], Hao Yang [1,†], Bruno Carpentieri [3,†], Xian-Ming Gu [4,*] and Chun Wen [5,*]

1 College of Science, Sichuan Agricultural University, Ya'an 625000, China; yanghao1@stu.sicau.edu.cn
2 Johann Bernoulli Institute for Mathematics and Computer Science, Faculty of Science and Engineering, University of Groningen, 9700 AK Groningen, The Netherlands
3 Faculty of Computer Science, Free University of Bozen-Bolzano, 39100 Bolzano, Italy; Bruno.Carpentieri@unibz.it
4 School of Economic Mathematics, Southwestern University of Finance and Economics, Chengdu 611130, China
5 School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China
* Correspondence: 14519@sicau.edu.cn (Z.-L.S.); guxm@swufe.edu.cn (X.-M.G.); wchun17@uestc.edu.cn (C.W.)
† These authors contributed equally to this work.

Abstract: The PageRank model computes the stationary distribution of a Markov random walk on the linking structure of a network, and it uses the values within to represent the importance or centrality of each node. This model is first proposed by Google for ranking web pages, then it is widely applied as a centrality measure for networks arising in various fields such as in chemistry, bioinformatics, neuroscience and social networks. For example, it can measure the node centralities of the gene-gene annotation network to evaluate the relevance of each gene with a certain disease. The networks in some fields including bioinformatics are undirected, thus the corresponding adjacency matrices are symmetry. Mathematically, the PageRank model can be stated as finding the unit positive eigenvector corresponding to the largest eigenvalue of a transition matrix built upon the linking structure. With rapid development of science and technology, the networks in real applications become larger and larger, thus the PageRank model always desires numerical algorithms with reduced algorithmic or memory complexity. In this paper, we propose a novel preconditioning approach for solving the PageRank model. This approach transforms the original PageRank eigen-problem into a new one that is more amenable to solve. We then present a preconditioned version of the refined Arnoldi method for solving this model. We demonstrate theoretically that the preconditioned Arnoldi method has higher execution efficiency and parallelism than the refined Arnoldi method. In plenty of numerical experiments, this preconditioned method exhibits noticeably faster convergence speed over its standard counterpart, especially for difficult cases with large damping factors. Besides, this superiority maintains when this technique is applied to other variants of the refined Arnoldi method. Overall, the proposed technique can give the PageRank model a faster solving process, and this will possibly improve the efficiency of researches, engineering projects and services where this model is applied.

Keywords: PageRank; Arnoldi; preconditioning; Krylov subspace methods

Citation: Shen, Z.-L.; Yang, H.; Carpentieri, B.; Gu, X.-M.; Wen, C. A Preconditioned Variant of the Refined Arnoldi Method for Computing PageRank Eigenvectors. *Symmetry* **2021**, *13*, 1327. https://doi.org/10.3390/sym13081327

Academic Editor: Paolo Emilio Ricci

Received: 30 June 2021
Accepted: 19 July 2021
Published: 23 July 2021

1. Introduction

With the rapid development of the Internet, web search engines become very popular for information retrieval. Because a web search engine can usually find an immense set of Web pages matching the search query, it is necessary to rank higher the most important pages to make this tool practical. Fur this purpose, the PageRank model was developed by the Google team to rank the importance of Web pages based on the frequency of page visits recorded by a random user who keeps browsing the World Wide Web with an equal

probability of choosing the hyperlinks on each page. Mathematically speaking, it requires the computation of the stationary distribution of this Markov random walk on the linking structure of pages, the values within the distribution represent the frequency of visits to each Web page. The linking structure of the Web is represented by a large directed graph (called the Web link graph) and by its adjacency matrix $G \in \mathbb{N}^{n\times n}$ (here n is the number of Web pages) such that $G(i,j) \neq 0$ (being 1) only when page j has a hyperlink pointing to page i. The transition probability matrix $P \in \mathbb{R}^{n\times n}$ of this random walk process is defined as

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

However, to avoid that the process stagnates if a dangling page without hyperlinks is visited, in the model P is usually modified as

$$\widetilde{P} = P + vd^T,$$

where $v \in \mathbb{R}^{n\times 1}$ is a probability distribution vector, $d \in \mathbb{N}^{n\times 1}$ is a binary vector, and $d(i) = 1$ $(1 \leq i \leq n)$ if page i has no hyperlinks. According to the Perron-Frobenius theorem, the unique existence of the stationary distribution vector is guaranteed when $\widetilde{P}$ is an irreducible stochastic matrix. To enforce this assumption, $\widetilde{P}$ is further modified into the matrix

$$A = \alpha\widetilde{P} + (1-\alpha)ve^T, \tag{2}$$

where $\alpha \in (0,1)$ is called the damping factor and $e = [1,1,\cdots,1]^T$. Matrix A is often called the Google matrix [1]. Finally, the PageRank model can be stated mathematically as finding the unit positive eigenvector corresponding to the eigenvalue 1 of A, that is the solution of the eigen-problem

$$Ax = x, \quad \|x\|_1 = 1, \quad x > 0. \tag{3}$$

Because of the assumption that A is an irreducible stochastic matrix, given by Equation (2), it follows that Equation (3) is uniquely solvable. The unique stationary distribution vector x is called the PageRank vector. Note that, the PageRank model now is often used as a network centrality measure to identify the most important nodes within large networks arising in several applications such as in chemistry, bioinformatics, neuro-science, bibliometrics, web search engines, social networks, etc. [2]. In some fields such as the bioinformatics, the related networks are usually undirected, thus the corresponding adjacency matrices are symmetry. This symmetry can be used to build efficient storage format and efficient implementation for matrix-vector multiplications, then the difficulty of solving the PageRank model (also called the PageRank problem) can be reduced.

With rapid development of science and technology, the dimension of PageRank problems from various application fields has grown hugely in the last decades and still keeps growing. Accordingly, iterative methods become the only viable option to solve Equation (3) numerically. Stationary iterative methods such as the Power, Jacobi and Gauss-Seidel methods are effective when the damping parameter α is not too close to 1, e.g., for search engine applications Google initially used $\alpha = 0.85$. However, larger values of α (not too close to 1) such as 0.99 may sometimes give a better ranking result [3], and they tend to converge significantly more slowly when α is large. These computational issues make the PageRank problem always require new algorithmic solutions that are more time-saving and memory-saving.

The development of more efficient algorithms for solving PageRank problems has been ongoing for the past decade or so. One research direction is to accelerate the convergence speed of stationary iterative methods, related works include adaptive [4] and extrapolation methods [5–7], multigrid solvers [8,9], and inner-outer iterations [10–14]. Meanwhile a significant amount of work has been devoted in particular to the analysis of Krylov subspace methods for computing PageRank. Golub and Greif proposed in [3] a new

variant of the Arnoldi algorithm that is particularly suited for solving problems with a large range of damping values, and can outperform many conventional stationary iterative solvers when α is closer to 1. Yin et al. have used weighted inner-products instead of the standard inner-products in the Arnoldi process reporting faster convergence due to a more favourable distribution of the harmonic Ritz values [15]. The Power-Arnoldi method [16], its weighted version [17] and the Arnoldi-Inout method [18] are hybrid variants of the Arnoldi process that combine periodically stationary and Krylov iterations, and can improve other established PageRank solvers on many examples. All of these techniques could accelerate the Arnoldi method from [3], but there is likely to be a lot of room for further improvement. One obvious thing is that, preconditioning, which is widely used to accelerate Krylov subspace methods for solving linear systems, has not been considered yet for accelerating the Arnoldi-type methods when solving the PageRank eigen-problem. This is an area worth investigating because successful preconditioning usually results in significant acceleration of the solving process, and can be well combined with other acceleration techniques.

In this work we propose a new theoretical formulation of the PageRank eigenvector problem (3) that is characterized by a better eigenvalue separation in the spectrum between the dominant and the second dominant eigenvalues of the coefficient matrix, so that the Arnoldi process may require significantly less iterations to converge, and consequently less BLAS-1 (inner-products and SAXPY) operations in the Gram-Schmidt orthogonalization. Our strategy to transform the original problem into a new one that is more amenable to the iterative solution can be seen as a form of preconditioning. The optimal parameter setting to choose in this preconditioning technique is also discussed. Our experiments confirm the theoretical findings and demonstrate that the Arnoldi-type methods applied to the preconditioned eigen-problem can converge much faster over their standard counterparts. Thus it has potential to solve large-scale PageRank computations more effectively on both sequential and parallel machines. Accordingly, we can expect that the proposed technique can accelerate accomplishing projects from various fields that use the PageRank model. For example, for the GeneRank problem [19] and the ProteinRank [20] problem that use the PageRank model on the gene-gene and protein-protein networks, users can find the genes and proteins that are pathogenic with high probability faster.

The paper is organized as follows. In Section 2, we outline the refined Arnoldi method proposed in [3] that is at the basis of our development, along with its main convergence results. In Section 3, we present a preconditioned variant of the refined Arnoldi method for computing PageRank eigenvectors. Numerical experiments are reported in Section 4 to support our theoretical findings. Finally, some conclusions from this study are presented in Section 5. Note that, MATLAB notations are used throughout this article.

2. The Refined Arnoldi Method for PageRank

The Arnoldi process proposed in 1951 is an algorithm based on the modified Gram-Schmidt orthogonalization that, after m steps, computes an orthonormal basis $\{v_1, v_2, \cdots, v_{m+1}\}$ of the Krylov subspace $K_{m+1}(A, v_0) = span\{v_0, Av_0, \cdots, A^m v_0\}$, where $A \in \mathbb{R}^{n\times n}$ and $v_0 \in \mathbb{R}^{n\times 1}$ are a given matrix and an initial vector, respectively. Here we sketch it in Algorithm 1.

In matrix form, Algorithm 1 yields after m steps the Arnoldi decompositions

$$AV_m = V_{m+1}\bar{H}_m \tag{4}$$

$$= V_m H_m + h_{m+1,m} v_{m+1} {e_m}^T, \tag{5}$$

and

$$V_m^T A V_m = H_m, \tag{6}$$

where $H_m = \{h_{i,j}\} \in \mathbb{R}^{m\times m}$ is an upper Hessenberg matrix, and we denote by $\bar{H}_m = [H_m; h_{m+1,m} e_m^T] \in \mathbb{R}^{(m+1)\times m}$, by $e_m = [0, 0, \ldots, 0, 1]^T$ and by $V_{m+1} = [V_m, v_{m+1}]$. In Table 1 we summarize the computational cost of Algorithm 1 for the case of a general matrix A,

and in Table 2 for the special case of the Google matrix ((2). Note that in the latter case, from the expression $A = \alpha P + v((1-\alpha)e + \alpha d)^T$, it follows that the product Au requires 1 sparse matrix-vector operation αPu (hereafter u represents an arbitrary vector with appropriate dimension), 1 inner product $f^T u$ where $f = ((1-\alpha)e + \alpha d)$ is stored, 1 vector scaling operation and 1 vector addition.

Algorithm 1 The Arnoldi process $[V_m, H_m, v_{m+1}, h_{m+1,m}, \beta] = Arnoldi(A, v_0, m)$

1: Compute $\beta = \|v_0\|_2$, $v_1 = v_0/\beta$.
2: **for** $j = 1 : m$ **do**
3: Compute $w = Av_j$.
4: **for** $i = 1 : j$ **do**
5: Compute $h_{i,j} = v_i^T w$.
6: Compute $w = w - h_{i,j} v_i$.
7: **end for**
8: Compute $h_{j+1,j} = \|w\|_2$.
9: **if** $h_{j+1,j} = 0$ **then**
10: Set $m = j$, $v_{m+1} = 0$, and break.
11: **else**
12: Set $v_{j+1} = w/h_{j+1,j}$.
13: **end if**
14: **end for**
15: Generate $V_m = [v_1, v_2, \cdots, v_m]$.

Table 1. Computational cost of the Arnoldi process (Algorithm 1).

Operation	Line	Times
Matrix-vector multiplication	3	m
Inner-product	5	$\frac{m(m+1)}{2}$
Vector scaling	1, 12	$m+1$
Vector 2-norm computation	1, 8	$m+1$
SAXPY (SAXPY denotes the operation $x + ay$ where a is a scalar, x and y are vectors.)	6	$\frac{m(m+1)}{2}$

Table 2. Computational cost of the Arnoldi process (Algorithm 1) for PageRank.

Operation	Times
Sparse matrix-vector multiplication	m
Inner product	$\frac{m(m+3)}{2}$
Vector scaling	$\frac{m^2+5m+2}{2}$
Vector 2-norm computation	$m+1$
Vector addition	$\frac{m(m+3)}{2}$

The Arnoldi process is the main ingredient of many efficient numerical techniques for solving linear systems and eigen-problems. After m steps, scalars $\lambda_i^{(m)}$ and vectors $\phi_i^{(m)} = V_m y_i^{(m)}$ $(i = 1, 2, \ldots, m)$ satisfying $H_m y_i^{(m)} = \lambda_i^{(m)} y_i^{(m)}$ are called the *Ritz values* and the *Ritz vectors* of A onto the Krylov subspace $K_m(A, v_0)$, respectively. The Ritz values with largest real parts and their corresponding Ritz vectors are often used to approximate the eigenvalues with largest real parts of A and their associated eigenvectors [21]. However, in-depth convergence analysis of the Arnoldi method shows that the Ritz vectors are not guaranteed to converge to the actual eigenvectors, even when the Ritz values do converge.

To overcome this difficulty, Jia proposed in [22] to compute the *refined Ritz vector* $u_i^{(m)}$ associated to the Ritz value $\lambda_i^{(m)}$, defined as

$$\|(A-\lambda_i^{(m)}I)u_i^{(m)}\| = \min_{u\in K_m(A,v_0),\|u\|=1} \|(A-\lambda_i^{(m)}I)u\|. \tag{7}$$

The solution of Equation (7) is given by

$$\begin{aligned} [H_m; h_{m+1,m}e_m^T] - \lambda_i^{(m)}[I;O] &= U\Sigma W^T, \\ u_i^{(m)} &= V_m W(:,m), \end{aligned} \tag{8}$$

where $U\Sigma W^T$ is the singular value decomposition of matrix $[H_m; h_{m+1,m}e_m^T] - \lambda_i^{(m)}[I;O]$, the singular values are stored in the diagonal part of Σ in decreasing order and the columns of W are the right singular vectors. Besides, the 2-norm of the residual vector $\|(A-\lambda_i^{(m)}I)u_i^{(m)}\|_2$ is equal to the smallest singular value $\Sigma(m,m)$.

Although in exact arithmetic the refined Ritz vector $u_i^{(m)}$ is guaranteed to converge to the eigenvector associated to λ_i when $\lambda_i^{(m)} \to \lambda_i$, some potential numerical difficulties may still arise in practice. Firstly, the largest Ritz value may be complex, and the use of complex arithmetic may be a memory burden for large-scale computations. Secondly, when α is close to 1, slow or irregular convergence can still happen due to a weak separation of the eigenvalues of A. Golub and Greif propose to use the largest eigenvalue of the Google matrix, which is known and equal to 1, as a shift in (8) instead of the largest Ritz value $\lambda_1^{(m)}$ as an attempt to overcome problems related to slow solving process [3]. We present the Golub and Greif variant of the refined Arnoldi method for PageRank in Algorithm 2, and refer to it shortly as the GG-Arnoldi method hereafter.

Algorithm 2 The Golub and Greif variant of the restarted refined Arnoldi method (shortly referred to as GG-Arnoldi)

Input: The PageRank coefficient matrix A, initial guess x_0, parameters m and tol.
1: Run Algorithm 1 as $[V_m, H_m, v_{m+1}, h_{m+1,m}, \beta] = Arnoldi(A, x_0, m)$.
2: Compute the SVD factorization $[H_m; h_{m+1,m}e_m^T] - [I;O] = U\Sigma W^T$.
3: Compute the updated approximated solution $x = V_m W(:,m)$.
4: Compute the residual 2-norm by the smallest singular value $r = \Sigma(m,m)$.
5: **if** $r/\|x\|_1 \le tol$ **then**
6: Outputs $x = x/\|x\|_1$ and ends.
7: **else**
8: Set $x_0 = x$ and go to step 1.
9: **end if**
10: **return** x.

The overall computational cost of Algorithm 2 is summarized in Table 3. Note that the singular value decomposition computed at line 3 is omitted because this cost is negligible when m is very small. Analogously, the vector scaling $x = x/\|x\|_1$ at line 6 is not reported in the table as it is computed only at the last cycle. Finally, the operation $x = V_m W(:,m)$ is equivalent to m vector scaling operations and $m-1$ vector additions.

The convergence analysis of the GG-Arnoldi method is presented in [20]. Here we recall the main result. We denote as $X_\perp$ the standard orthogonal basis of the orthogonal complement of the space $span\{x\}$, so that $[x, X_\perp]$ is orthonormal. Then, we have

$$\begin{bmatrix} x^T \\ X_\perp^T \end{bmatrix} A[x, X_\perp] = \begin{bmatrix} \|x\|_2^2 & x^T A X_\perp \\ O & A_2 \end{bmatrix},$$

where $A_2 = X_\perp^T A X_\perp$. According to the Perron-Frobenius theorem, the spectral radius of A is equal to 1, A has only one eigenvalue 1 on the unit circle, and the eigenspace of this

eigenvalue is 1-dimensional. As 1 is a simple eigenvalue of A, we introduce a separation function named "*sep*" and defined as

$$sep(1, A_2) = \|(1 - A_2)^{-1}\|_2^{-1} = \sigma_{min}(I - A_2). \tag{9}$$

Table 3. The computational cost per cycle of the GG-Arnoldi method for PageRank.

Operation	Times
Sparse matrix-vector multiplication	m
Inner-product	$\frac{m(m+3)}{2}$
Vector-scaling	$\frac{m^2+7m+4}{2}$
Vector 2-norm computation	$m+1$
Vector-addition	$\frac{m^2+5m-2}{2}$
Vector 1-norm computation	1

Since $\sigma_{min}(I - A_2) \leq |1 - \lambda_2|$, where λ_2 is the second largest eigenvalue of A [20], we obtain

$$sep(1, A_2) \leq |1 - \lambda_2|.$$

Under these assumptions, the theorem below suggests that the larger the second dominant eigenvalue λ_2, the slower the convergence rate of the GG-Arnoldi method applied to the eigen-problem (3).

Theorem 1 (Theorem 2.2 in [20]). *Let P_m be the orthogonal projector onto the subspace $K_m(A, v_0)$, $\epsilon_m = \|(I - P_m)x\|_2$ be the distance between the PageRank vector x and the subspace $K_m(A, v_0)$, and x_{AT} be the solution generated by the GG-Arnoldi method. Then it holds*

$$sin\angle(x, x_{AT}) \leq \frac{\epsilon_m}{\sqrt{1 - \epsilon_m^2}} \cdot \frac{\|A - I\|_2}{sep(1, A_2)}. \tag{10}$$

We recall below another property of the eigenvalues of the Google matrix A that is very relevant to our analysis.

Theorem 2 ([23]). *If a column-stochastic matrix $\widetilde{P}$ has at least two irreducible closed subsets (which is the case for the web hyperlink matrix), then the second eigenvalue λ_2 of $A = \alpha\widetilde{P} + (1 - \alpha)ve^T$, where $0 < \alpha < 1$ and v is a vector with non-negative elements satisfying $\|v\|_1 = 1$, is given by α.*

Therefore, slow convergence of the GG-Arnoldi method for PageRank problems may be expected in particular when α approaches 1. As this situation arises in several applications, some acceleration techniques have been developed in the past years to enhance the robustness of the GG-Arnoldi method. They can be classified in two types: (1) using weighted inner products instead of the standard inner products in Algorithm 2 to ensure a more favourable distribution of Ritz values [15,17]; (2) combining the GG-Arnoldi method with a few cycles of stationary iterative solvers, like in the Power-Arnoldi method [16], the Inout-Arnoldi method [18], the Arnoldi-PET method [17] and others. Both approaches attempt to provide better initial guesses for the Arnoldi process. To the best of our knowledge, no research work has investigated efficient ways to precondition the PageRank eigen-problem (3), in order to make it easier to be solved by the Arnoldi-type method. This is the main objective of our study.

3. Preconditioning the Refined Arnoldi Method

Preconditioning is an established technique used to accelerate Krylov subspace methods for solving linear systems. A nonsingular linear system $Ay = c$ can be transformed into an equivalent one of either the form $MAy = Mc$ (*left preconditioned* system) or the form $AMz = c$ with $y = Mz$ (*right preconditioned* system), where $M \approx A^{-1}$ is called the

preconditioner matrix or simply the *preconditioner*. Goal of preconditioning is to improve the spectral distribution of the coefficient matrix A so that a Krylov subspace algorithm can solve the transformed preconditioned system much faster than the original one. The development of efficient preconditioners is a very important topic in numerical linear algebra because it can enable rapid solution of problems that may initially appear numerically intractable. However, preconditioning is seldom used for solving eigen-problems $Ay = \lambda y$ because the left preconditioned ($MAy = \lambda My$) and right preconditioned ($AMz = \lambda y$, $y = Mz$) eigenproblems are no longer equivalent to the original one.

Here we propose a new approach to precondition the PageRank eigen-problem (3) for the computation of the dominant eigenvector x corresponding to the largest eigenvalue $\lambda_1 = 1$ of the Google matrix A. According to the analysis presented in Section 2, the convergence speed of the GG-Arnoldi method is mainly determined by the separation between the largest and second largest eigenvalues of A, namely by the quantity $|\lambda_1 - \lambda_2| = 1 - \alpha$. Therefore, our strategy for preconditioning Equation (3) is to transform it into an equivalent eigen-problem that has a better separation between its two dominant eigenvalues. The main theoretical result underlying our method is presented in the theorem below.

Theorem 3. *For Google matrix A in (2) and its eigenvalues $1 > |\lambda_2| > |\lambda_3| > \cdots > |\lambda_s|$, if a polynomial $\mathcal{P}$ satisfies $\mathcal{P}(\lambda_i) \neq \mathcal{P}(1)$ $(2 \leq i \leq s)$, then the PageRank eigen-problem (3) is equivalent to the eigen-problem:*

$$\mathcal{P}(A)x = \mathcal{P}(1)x,\ x > 0 \text{ and } \|x\|_1 = 1. \tag{11}$$

Proof. It is clear that any solution of $Ax = x$ is also the solution of $\mathcal{P}(A)x = \mathcal{P}(1)x$. Suppose

$$A = T^{-1}\begin{bmatrix} 1 & & & & \\ & J_2 & & & \\ & & J_3 & & \\ & & & \ddots & \\ & & & & J_s \end{bmatrix} T,$$

where each J_i $(2 \leq i \leq s)$ is a Jordan matrix whose diagonal elements equal to λ_i. Then,

$$\mathcal{P}(A) = T^{-1}\begin{bmatrix} \mathcal{P}(1) & & & & \\ & \mathcal{P}(J_2) & & & \\ & & \mathcal{P}(J_3) & & \\ & & & \ddots & \\ & & & & \mathcal{P}(J_s) \end{bmatrix} T.$$

Clearly, each $\mathcal{P}(J_i)$ $(2 \leq i \leq s)$ is still a lower triangle matrix, and its diagonal elements equal to $\mathcal{P}(\lambda_i)$. Because $\mathcal{P}(\lambda_i) \neq \mathcal{P}(1)$ $(2 \leq i \leq s)$, therefore the algebraic multiplicity of the eigenvalue $\mathcal{P}(1)$ of $\mathcal{P}(A)$ equals to 1, as same as that of the eigenvalue 1 of A. Therefore $\mathcal{P}(A)x = \mathcal{P}(1)x$ has the same solution space as $Ax = x$, i.e., problem (3) is equivalent to problem (11). □

The obvious question to address is whether such a polynomial $\mathcal{P}$ satisfying the condition $\mathcal{P}(\lambda_i) \neq \mathcal{P}(1)$ $(2 \leq i \leq s)$ on the eigenvalues $1 > |\lambda_2| > |\lambda_3| > \cdots > |\lambda_s|$ of A exists, and then how to make $\mathcal{P}(A)x = \mathcal{P}(1)x$ easier to solve than $Ax = x$. The simple polynomial $\mathcal{P}(x) = x^k$ ($k \in \mathbb{N}^+$) clearly satisfies $\mathcal{P}(\lambda_i) \neq \mathcal{P}(1) = 1$ $(2 \leq i \leq s)$. According to Theorem 3, the two eigen-problems

$$A^k x = x,\ x > 0 \text{ and } \|x\|_1 = 1$$

and

$$Ax = x,\ x > 0 \text{ and } \|x\|_1 = 1$$

are equivalent. Besides, in A^k the distance between the largest two eigenvalues equals to $|\mathcal{P}(1) - \mathcal{P}(\alpha)| = 1 - \alpha^k >> 1 - \alpha$. As a result, we can expect that the GG-Arnoldi method will require less cycles and operations to converge when it is applied to $A^k x = x$ than to the problem $Ax = x$. We sketch the complete preconditioned version of the Golub and Greif variant of the refined Arnoldi method with the polynomial choice $\mathcal{P}(x) = x^k$, hereafter shortly referred to as the PGG-Arnoldi method, in Algorithm 3.

Algorithm 3 The preconditioned version of the Golub and Greif variant of the refined Arnoldi method, with the polynomial choice $\mathcal{P}(x) = x^k$ (shortly, PGG-Arnoldi) for PageRank.

Input: the PageRank coefficient matrix A, initial guess x_0, parameters m, k and tol.
1: Run Algorithm 1 as $[V_m, H_m, v_{m+1}, h_{m+1,m}, \beta] = Arnoldi(A^k, x_0, m)$.
2: Compute the SVD factorization $[H_m; h_{m+1,m} e_m^T] - [I; O] = U \Sigma W^T$;
3: Compute the updated approximated solution $x = V_m W(:, m)$.
4: Compute the residual 2-norm by the smallest singular value $r = min(diag(\Sigma))$.
5: **if** $r\|x\|_1 \leq tol$ **then**
6: Outputs $x = x/\|x\|_1$ and ends.
7: **else**
8: Set $x_0 = x$ and go to step 1.
9: **end if**
10: **return** x

Note that the matrix A^k in Algorithm 3 is never formed explicitly. The operations associated with A^k in this algorithm are all matrix-vector multiplications, such operation $A^k u$ is computed by carrying out k sparse matrix-vector products without assembling the Google matrix A shown as follows in Algorithm 4.

Algorithm 4 Implementation of the matrix-vector product $y \leftarrow A^k u$

1: **for** $i = 1 : k$ **do**
2: Compute $u \leftarrow \alpha P u + (f^T u) v$.
3: **end for**
4: Set $y \leftarrow u$.
5: **return** y.

The overall algorithmic complexity of a cycle of the PGG-Arnoldi method (Algorithm 3) is summarized in Table 4.

Table 4. The computational cost per cycle of the PGG-Arnoldi method for PageRank.

Operation	Times
Sparse matrix-vector multiplication	km
Inner-product	$\frac{m(m+2k+1)}{2}$
Vector-scaling	$\frac{m^2+2km+5m+4}{2}$
Vector 2-norm computation	$m+1$
Vector-addition	$\frac{m^2+2km+3m-2}{2}$
Vector 1-norm computation	1

It is clear that one cycle of Algorithm 3 is computationally more expensive than that of the GG-Arnoldi method (Algorithm 2) for the same value of m, because the former requires to compute the matrix-vector product $A^k u$. In detail, the PGG-Arnoldi algorithm needs additional $(k-1)m$ sparse matrix-vector products, $(k-1)m$ inner products, $(k-1)m$ vector scalings and $(k-1)m$ vector additions compared with the GG-Arnoldi algorithm per cycle. The computation of an inner product requires n floating-point multiplications and $n-1$ floating-point additions while scaling a vector only needs n floating-point multiplications. Thus, we will assume that a vector scaling operation has half the cost of an inner product

with vectors of the same dimension. For the same reason, the cost of each vector addition, 2-norm and 1-norm of a vector can be counted as 0.5, 1 and 0.5 inner-product, respectively. Finally, the algorithmic complexity estimates of one cycle of GG-Arnoldi and PGG-Arnoldi in terms of sparse matrix-vector multiplications and vector inner-products are presented in Table 5.

Table 5. Estimated algorithmic complexity of the GG-Arnoldi (Algorithm 2) and of its preconditioned version PGG-Arnoldi (Algorithm 3).

Method	GG-Arnoldi	PGG-Arnoldi
Sparse matrix-vector multiplications	m	km
Inner-products	$\frac{2m^2+11m+4}{2}$	$\frac{2m^2+4km+7m+4}{2}$

We observe from Table 5 that one cycle of the PGG-Arnoldi costs less than k times the cost of GG-Arnoldi. This means that the operations in addition to the sparse matrix-vector product in PGG-Arnoldi are less than those required in GG-Arnoldi. Therefore, if the number of cycles required by the PGG-Arnoldi is less than $\frac{1}{k}$ times of the GG-Arnoldi, the former must be faster. Indeed, it may be faster even when the number of cycles required by the PGG-Arnoldi is larger than $\frac{1}{k}$ times of the GG-Arnoldi, which will be shown by numerical experiments. Besides, in real applications, the computation of a matrix-vector product is more efficient than that of vector operations in either serial or parallel environments, as the former belongs to BLAS-2 classification while the latter belongs to BLAS-1. Note that, it is very desirable to improve the efficiency of parallel computing for solving large problems such as PageRank. Given a good estimate of the convergence rate comparison between these two methods and the density of matrix G, the results presented in Table 5 can guide the choice between the PGG-Arnoldi or the GG-Arnoldi methods for solving the problem at hand. However, few things (in particular, quantitative conclusions) are known about the convergence rate of the GG-Arnoldi method besides that it increases with the gap $|1 - \lambda_2|$, therefore we rely on numerical experiments to analyze the performance of the PGG-Arnoldi method.

Finally, we remark that:

- acceleration techniques designed to work with the GG-Arnoldi, such as extrapolation methods [6,7], formulations based on weighted inner products [3] and hybrid solution schemes [16–18] that combine stationary iterative iterations with the Arnoldi method, can still be applied to the PGG-Arnoldi;
- the performance of the PGG-Arnoldi can be further improved by using pre-processing algorithms such as the elimination strategy in [24] and the low-rank factorization in [25] that reduce the time cost and the memory cost of computing the sparse matrix-vector product αPu, because the sparse matrix-vector products in the PGG-Arnoldi account for a larger proportion of computation compared to the GG-Arnoldi.

4. Numerical Results

In this section, we present results of numerical experiments on a suite of Web matrix problems obtained from the University of Florida matrix repository [26] and from the Laboratory for Web Algorithmics [27–29]. The characteristics of our test problems are presented in Table 6. For each Web adjacency matrix G, we build the Google matrix using Equation (2) with personalization vector $v = [1, 1, \ldots, 1]^T/n$. We use the value $\alpha = 0.99$ for the damping parameter, so that the resulting PageRank problems are rather difficult to solve for iterative methods. All the runs with iterative solvers are started from the initial guess $x_0 = [1/n, 1/n, \cdots, 1/n]^T$ and are stopped when either the approximate solution x_i satisfies $\frac{\|(I-A)x_i\|_2}{\|x_i\|_1} < 10^{-8}$, or the number of matrix-vector products computed exceeds 20,000. All the runs are carried out in MATLAB R2016b on a 64-bit Windows-10 computer equipped with an Intel core i7-8750H processor and 16 GB RAM memory.

Table 6. Characteristics of the Web adjacency matrices G tested in our experiments. The symbol n is the dimension of the matrix, and nnz is the number of nonzero elements (listed in increasing matrix size).

Name	n	nnz	nnz/n^2
`web-Stanford`	281,903	2,312,497	2.9e-5
`cnr-2000`	325,557	3,216,152	3.0e-5
`web-BerkStan`	685,230	7,600,595	1.6e-5
`eu-2005`	862,664	19,235,140	2.6e-5
`in-2004`	1,382,908	16,917,053	8.8e-6
`indochina-2004`	7,414,866	194,109,311	3.5e-6
`wb-edu`	9,845,725	57,156,537	5.9e-7

4.1. Performance Analysis of the PGG-Arnoldi Method

We first assess the performance behaviour of the PGG-Arnoldi method with the polynomial choice $\mathcal{P}(x) = x^k$, using different values of degree k and increasing dimensions m of the Krylov search space. Because of the very large size of PageRank problems in real applications, the dimension of the Krylov subspace should be kept as small as possible. In our experiments on the `web_NotreDame` and `in-2004` matrices, we use $m = [3, 4, \cdots, 10]$ and $k = [1, 2, \cdots, 10]$. In Tables 7 and 8 we report on the results of our experiments in terms of number of iterations and CPU time (in seconds) for each run.

Table 7. Performances of the PGG-Arnoldi method on the `web_NotreDame` problem.

Number of iterations										
$m \downarrow$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3	351	226	117	110	73	72	53	55	39	44
4	233	147	84	71	50	47	35	34	25	27
5	173	109	61	53	37	35	27	25	22	19
6	139	87	49	42	29	27	22	20	17	15
7	112	70	35	34	23	22	17	16	13	13
8	90	58	34	27	19	18	15	13	10	11
9	78	47	28	23	16	16	12	12	10	8
10	62	39	24	18	14	12	10	8	8	7
CPU time (in seconds)										
$m \downarrow$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3	8.0	6.7	4.6	5.1	4.0	4.5	3.8	4.3	3.5	4.4
4	6.4	5.8	4.3	4.4	3.7	4.0	3.4	3.7	3.0	3.6
5	5.9	5.4	3.9	4.2	3.5	3.7	3.2	3.4	3.3	3.1
6	5.9	5.2	3.7	4.0	3.4	3.4	3.2	3.3	3.0	2.9
7	5.5	4.8	3.1	3.7	2.9	3.3	2.9	3.0	2.7	3.0
8	5.0	4.6	3.5	3.3	2.8	3.1	2.9	2.8	2.4	2.9
9	5.1	4.3	3.2	3.2	2.7	3.1	2.7	2.9	2.7	2.4
10	4.4	3.9	3.1	2.9	2.6	2.6	2.4	2.2	2.4	2.3

We observe that, in general, the number of iterations decreases when higher degree polynomials and Krylov subspaces of higher dimensions are used. By a simple multiple linear regression of number of iterations ($Iter$), $\frac{1}{k}$ and $\frac{1}{m}$ for Table 7, we obtained $Iter = -66.4 + 394.2\frac{1}{m} + 157.1\frac{1}{k}$ with the R^2 statistic being 0.8, the probability of the F-statistic is less than 0.05, and all the confidence intervals not including the origin. Therefore, the linear relationship can be considered significant. Similar results are obtained for Table 8. The CPU time cost generally decreases when m increases, while it oscillates with the increase of k. For the purpose of saving computing time, it is suggested to set the dimension of the Krylov subspace to the maximum value allowed by the available memory. For a given m, the number of iteration tends to increase inversely proportional with k, or

approximately proportional to $\frac{1}{k}$. Then, if the ratio between the time costs of computing the matrix-vector product αPu and an inner product between vectors of the same dimension is known, the complexity estimates given in Table 5 can guide the user in the choice of the almost optimal k in the range 1:10. Here we set $m = 10$ and $k = 5$ based on our experiments.

Table 8. Performances of the PGG-Arnoldi method on the `in-2004` problem.

Number of iterations										
$m \downarrow$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
3	355	272	121	137	75	88	56	67	42	52
4	235	163	82	82	52	57	35	41	27	30
5	175	114	61	58	39	37	25	28	20	18
6	110	74	44	38	28	21	20	18	15	12
7	95	44	37	25	23	17	16	12	12	10
8	62	42	26	20	17	14	12	10	10	8
9	57	30	21	17	12	10	10	9	8	6
10	46	24	17	18	10	10	8	7	7	6
CPU time (in seconds)										
$m \downarrow$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
3	41.5	46.3	27.3	38.6	25.4	34.9	25.3	34.0	23.7	32.2
4	35.9	37.2	24.9	30.9	23.9	30.0	20.9	27.5	20.1	24.6
5	33.2	32.2	22.9	27.3	22.3	24.2	18.9	23.7	18.8	21.4
6	27.7	25.5	21.6	22.7	20.2	20.4	18.2	18.2	16.8	14.8
7	25.7	17.6	19.8	16.5	18.3	15.7	17.6	14.7	16.3	14.7
8	19.8	22.0	15.9	15.2	15.5	14.8	14.5	13.6	15.0	13.3
9	20.7	15.8	14.6	14.6	12.3	12.0	14.1	13.8	13.8	11.2
10	18.6	14.3	13.2	17.3	11.5	13.3	12.1	11.9	13.2	12.5

4.2. Preconditioning Combined with Weighted Inner-Product

We study the effect of using weighted inner products instead of standard inner products on the performance of the PGG-Arnoldi method. The experiments are carried out on the same problems as in our previous experiments. The results are presented in Tables 9 and 10.

The general trend is still the same, that is the number of iterations generally decreases with the increase of the polynomial degree k and of the Krylov subspace dimension m, but some exceptions to this trend can be observed. Here we also carry out a simple multiple linear regression of the data: the number of iterations ($Iter$), $\frac{1}{k}$ and $\frac{1}{m}$. The result for Table 9 is $Iter = -72.4 + 426.3\frac{1}{m} + 146.0\frac{1}{k}$, with the R^2 statistic being 0.75 that is smaller than 0.8 of the results of Table 7. This phenomenon may be explained by the fact that the adaptively weighting technique [15] makes the Arnoldi method more irregular. Besides, we can see that the preconditioning strategy proposed in this paper can also accelerate the weighted Arnoldi remarkably when solving these difficult PageRank problems. For nearly all the value of m, the CPU time cost corresponding to the optimal value of k is less than 50% of the time cost corresponding to $k = 1$ that is the case with no preconditioning.

Table 9. Performances of the weighted PGG-Arnoldi method on the `web_NotreDame` problem.

Number of iterations										
$m \downarrow$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3	355	226	114	112	71	71	52	54	39	43
4	231	145	82	69	48	47	35	33	25	27
5	199	108	62	54	32	35	25	24	19	20
6	128	81	48	41	25	26	19	19	14	15
7	86	60	30	31	20	22	14	15	11	13
8	59	33	27	25	14	17	12	12	10	11
9	60	30	19	17	11	12	11	9	7	7
10	42	24	13	16	11	9	8	9	7	7
CPU time (in seconds)										
$m \downarrow$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3	10.3	8.3	5.2	6.0	4.4	5.0	4.1	4.7	3.7	4.4
4	8.6	7.0	4.9	4.9	3.9	4.3	3.6	3.8	3.5	4.0
5	9.6	6.8	4.7	4.7	3.2	4.0	3.3	3.5	2.9	3.4
6	7.1	5.8	4.4	4.3	3.0	3.6	2.9	3.2	2.6	3.0
7	5.6	5.0	3.1	3.8	2.8	3.5	2.5	3.0	2.4	3.1
8	4.4	3.2	3.2	3.5	2.3	3.1	2.5	2.7	2.5	3.0
9	5.1	3.3	2.5	2.7	2.0	2.5	2.5	2.3	2.0	2.1
10	4.0	2.9	1.9	2.8	2.2	2.1	2.1	2.6	2.2	2.4

Table 10. Performances of the weighted PGG-Arnoldi method on the `in-2004` problem.

Number of iterations										
$m \downarrow$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3	354	269	118	133	73	87	53	65	40	53
4	231	156	81	82	49	50	33	38	26	17
5	161	59	53	33	33	28	23	20	17	13
6	88	35	32	21	21	15	16	13	12	9
7	49	25	24	15	16	10	13	8	9	9
8	38	24	15	14	10	9	10	8	8	6
9	36	23	14	10	11	8	8	7	7	5
10	34	17	12	10	8	8	6	6	6	5
CPU time (in seconds)										
$m \downarrow$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3	53.5	54.2	30.4	41.7	27.3	37.2	25.7	35.2	24.2	34.7
4	44.2	41.6	27.6	34.6	24.0	28.4	21.1	27.1	20.5	14.6
5	38.1	19.5	22.5	17.1	20.2	20.3	18.8	17.8	16.7	14.0
6	25.0	13.9	16.3	13.0	15.6	12.7	15.2	13.9	14.1	11.6
7	16.3	11.6	14.2	10.8	13.7	9.9	14.5	10.0	12.4	13.5
8	14.5	12.7	10.2	11.6	9.8	10.1	12.8	11.4	12.6	10.3
9	15.7	13.9	10.8	9.4	12.2	10.2	11.6	11.3	12.9	9.9
10	16.8	11.5	10.4	10.7	9.9	11.4	9.7	10.8	11.9	10.9

4.3. Comparisons with Other Methods

In this section, we compare the performance of the PGG-Arnoldi algorithm against other algorithms including the FOM method in [30] and its weighted version (referred to as W-FOM), the Golub and Greif variant of the refined Arnoldi method for PageRank (referred to as GG-Arnoldi) and its weighted version in [15] (referred to as W-Arnoldi), the Power method (referred to as Power), the extrapolation accelerated Power-Arnoldi method in [7,17] (referred to as EXT-Arnoldi) and the multi-step Power-inner-outer method in [12] (referred to as MPIO). Note that, we also test the performances of the preconditioned weighted Arnoldi method (referred to as PW-Arnoldi) and the preconditioned extrapolation accelerated Power-Arnoldi method (referred to as EXT-PArnoldi) where the GG-Arnoldi

method is replaced by our preconditioned Arnoldi method. All the matrices listed in Table 6 are tested. For all the tested methods, the Krylov subspace dimension m is set as $m = 10$, and the polynomial degree k of the preconditioner is set as $k = 5$. For the MPIO method, the number of steps of the Power method is set as 7, the parameters β and η for controlling the inner iterations are set as 0.5 and 0.1 respectively. For the EXT-Arnoldi method and the EXT-PArnoldi method, the extrapolation technique is applied every 40 Power iterations, and the residual tolerance value ϵ_1 of the Power method is set as 10^{-6}. Note that, the parameter settings of the MPIO and the EXT-Arnoldi methods are almost the best settings given in the literatures [7,12]. The numerical results are presented in Table 11, where the smallest CPU time cost is typeset for clarity in bold font.

Table 11. Comparative analysis of different methods for solving PageRank problems with $\alpha = 0.99$. Notation: *MV* denotes the *number of matrix-vector multiplications* that have been computed for achieving convergence, CPU_t is the *CPU time cost* (in seconds).

Method	***MV***	$\boldsymbol{CPU_t}$	***MV***	$\boldsymbol{CPU_t}$	***MV***	$\boldsymbol{CPU_t}$
	`web-Stanford`		`web-NotreDame`		`in-2004`	
FOM	271	2.10	721	4.87	491	18.70
W-FOM	291	3.16	411	4.12	281	13.77
GG-Arnoldi	421	3.40	621	4.35	461	17.89
PGG-Arnoldi	401	2.12	701	2.58	501	11.38
PW-Arnoldi	401	2.30	551	2.29	401	9.87
Power	1141	5.75	910	3.04	1094	23.78
EXT-Arnoldi	375	**2.02**	501	2.33	430	11.21
EXT-PArnoldi	415	2.20	451	**1.69**	400	**9.33**
MPIO	800	3.99	803	2.65	809	17.20
	`wiki-Talk`		`indochina`		`web-edu`	
FOM	61	2.99	641	187.39	621	144.03
W-FOM	61	4.21	551	184.88	601	183.14
GG-Arnoldi	101	5.22	611	175.33	591	138.89
PGG-Arnoldi	101	**2.28**	651	129.38	701	81.56
PW-Arnoldi	101	2.56	501	104.23	551	70.53
Power	688	15.16	933	179.49	942	104.95
EXT-Arnoldi	252	6.10	453	105.44	459	78.58
EXT-PArnoldi	292	6.71	513	**101.77**	529	**62.76**
MPIO	784	16.53	715	135.01	650	70.08

We can see from Table 11 the clear potential of the proposed preconditioning strategy for accelerating the GG-Arnoldi method remarkably. The time costs of PGG-Arnoldi and PW-Arnoldi are significantly smaller than their unpreconditioned versions. Moreover, the time cost can be often further reduced when PGG-Arnoldi is combined with Power iterations and extrapolation techniques. The resulting EXT-PArnoldi outperforms all the other methods on four out of six problems, PGG-Arnoldi acts as the fastest method on one problem and is close to the best on the remaining problem. Note that, the GG-Arnoldi method is always slower than W-FOM or MPIO, while PGG-Arnoldi outperforms them in most cases. We can conclude that the proposed preconditioning strategy can improve the efficiency of the Arnoldi-type methods, and makes them faster than some other state-of-the-art methods, when solving difficult PageRank problems.

Because the Arnoldi-type methods are very competitive for computing PageRank, it can be expected that the developed preconditioning technique can possibly accelerate the computation of PageRank problems arising from various fields. Besides, this technique can improve the efficiency of parallel computing of the PageRank problem. Accordingly,

any project using the PageRank model can have a faster solving process, this will possibly improve the efficiencies of scientific researches, engineering and services. As pre-described, for the GeneRank problem [19] and the ProteinRank [20] problem, users can find the genes and proteins that are pathogenic with high probability faster.

5. Conclusions

In this paper, we show that if a polynomial $\mathcal{P}$ satisfies $\mathcal{P}(1) \neq \mathcal{P}(\lambda_i)$ for any $|\lambda_i| < 1$, then the PageRank problem $Ax = x$ is equivalent to the new eigen-problem $\mathcal{P}(A)x = \mathcal{P}(1)x$. Moreover, by a suitable choice of the polynomial $\mathcal{P}$ such as $\mathcal{P} = x^k$ chosen in this paper, the new eigen-problem can exhibit a much better separation between the two largest eigenvalues, thus the Arnoldi-type methods can solve this problem by less iterations. Accordingly, the number of vector-vector operations with low parallelism in the solving process can be reduced. Based on this result, we introduce a preconditioned version of the Golub and Greif variant of the refined Arnoldi method for computing PageRank. Numerical experiments demonstrate that this method can solve PageRank problem much faster than the refined Arnoldi in a wide range of parameter settings, meanwhile the weighted-Arnoldi and the extrapolated-Arnoldi methods can also be accelerated by this preconditioning strategy. Finally, preconditioned Arnoldi-type methods show superiority over some other state-of-the-art methods for solving this problem class.

Author Contributions: Methodology, Z.-L.S., B.C. and H.Y.; software, Z.-L.S. and H.Y.; writing—original draft preparation, Z.-L.S. and H.Y.; writing—review and editing, B.C., X.-M.G. and C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Two-Way Support Programs of Sichuan Agricultural University (Grant No.1921993077). The third author is a member of the *Gruppo Nazionale per il Calcolo Scientifico* (GNCS) of the Istituto Nazionale di Alta Matematica (INdAM) and this work was partially supported by INdAM-GNCS under Progetti di Ricerca 2020.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Langville, A.N.; Meyer, C.D. *Google's Pagerank and Beyond: The Science of Search Engine Rankings*, 3rd ed.; Princeton University Press: Princeton, NJ, USA, 2006.
2. Gleich, D.F. PageRank beyond the web. *SIAM Rev.* **2015**, *57*, 321–363. [CrossRef]
3. Golub, G.H.; Greif, C. An Arnoldi-type algorithm for computing pagerank. *BIT* **2006**, *46*, 759–771. [CrossRef]
4. Kamvar, S.D.; Haveliwala, T.H.; Golub, G.H. Adaptive methods for the computation of the PageRank. *Linear Algebra Appl.* **2004**, *386*, 51–65. [CrossRef]
5. Kamvar, S.D.; Haveliwala, T.H.; Manning, C.D.; Golub, G.H. Extrapolation methods for accelerating PageRank computation. In Proceedings of the 12th International World Wide Web Conference, Budapest, Hungary, 20–24 May 2003; pp. 261–270.
6. Wu, G.; Wei, Y. An Arnoldi-Extrapolation algorithm for computing PageRank. *J. Comput. Appl. Math.* **2010**, *234*, 3196–3212. [CrossRef]
7. Tan, X. A new extrapolation method for PageRank computations. *J. Comput. Appl. Math.* **2017**, *313*, 383–392. [CrossRef]
8. Sterck, H.D.; Manteuffel, S.F.; Nguyen, Q.; Ruge, J. Multilevel adaptive aggregation for Markov chains, with application to web ranking. *SIAM J. Sci. Comput.* **2008**, *30*, 2235–2262. [CrossRef]
9. Shen, Z.-L.; Huang, T.-Z.; Carpentieri, B.; Wen, C.; Gun, X.-M. Block-accelerated aggregation multigrid for Markov chains with application to PageRank problems. *Commun. Nonlinear Sci.* **2018**, *59*, 472–487. [CrossRef]
10. Gleich, D.F.; Gray, A.P.; Greif, C.; Wen, C.; Lau, T. An Inner-Outer Iteration for Computing PageRank. *SIAM J. Sci. Comput.* **2010**, *32*, 349–371. [CrossRef]
11. Gu, C.-Q.; Xie, F.; Greif, C.; Zhang, K. A two-step matrix splitting iteration for computing PageRank. *J. Comput. Appl. Math.* **2015**, *278*, 19–28. [CrossRef]
12. Wen, C.; Huang, T.-Z.; Shen, Z.-L. A note on the two-step matrix splitting iteration for computing PageRank. *J. Comput. Appl. Math.* **2017**, *315*, 87–97. [CrossRef]

13. Tian, Z.-L.; Liu, L.; Zhang, Y.; Liu, Z.-Y.; Tian, M.-Y. The general inner-outer iteration method based on regular splittings for the PageRank problem. *Appl. Math. Comput.* **2019**, *356*, 479–501. [CrossRef]
14. Tian, M.-Y.; Liu, L.; Zhang, Y.; Wang, Y.-D. A general multi-splitting iteration method for computing PageRank. *Comput. Appl. Math.* **2019**, *38*, 1–29. [CrossRef]
15. Yin, J.-F.; Yin, G.-J.; Ng, M. On adaptively accelerated Arnoldi method for computing PageRank. *Numer. Linear Algebra Appl.* **2012**, *19*, 73–85. [CrossRef]
16. Wu, G.; Wei, Y. A power-Arnoldi algorithm for computing pagerank. *Numer. Linear Algebra Appl.* **2007**, *14*, 521–546. [CrossRef]
17. Hu, Q.-Y.; Wei, C.; Huang, T.-Z.; Shen, Z.-L.; Gu, X.-M. A variant of the Power-Arnoldi algorithm for computing PageRank. *J. Comput. Appl. Math.* **2021**, *381*, 113034. [CrossRef]
18. Gu, C.-Q.; Wang, W.-W. An Arnoldi-Inout algorithm for computing PageRank problems. *J. Comput. Appl. Math.* **2017**, *309*, 219–229. [CrossRef]
19. Morrison, J.L.; Breitling, R.; Higham, D.J.; Gilbert, D.R. GeneRank: Using search engine technology for the analysis of microarray experiments. *BMC Bioinform.* **2005**, *6*, 233. [CrossRef] [PubMed]
20. Wu, G.; Zhang, Y.; Wei, Y. Accelerating the Arnoldi-type algorithm for the PageRank problem and the ProteinRank problem. *J. Sci. Comput.* **2013**, *57*, 74–104. [CrossRef]
21. Saad, Y. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Numer. Linear Algebra Appl.* **1980**, *34*, 269–295. [CrossRef]
22. Jia, Z. Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems. *Numer. Linear Algebra Appl.* **1997**, *259*, 1–23. [CrossRef]
23. Haveliwala, T.; Kamvar, S. *The Second Eigenvalue of the Google Matrix*; Stanford University Technical Report; Stanford InfoLab: Stanford, CA, USA, 2004
24. Shen, Z.-L.; Huang, T.-Z.; Carpentieri, B.; Gun, X.-M.; Wen, C. An efficient elimination strategy for solving PageRank problems. *Appl. Math. Comput.* **2017**, *298*, 111–122. [CrossRef]
25. Shen, Z.-L.; Huang, T.-Z.; Carpentieri, B.; Wen, C.; Gun, X.-M.; Tan, X.-Y. Off-diagonal low-rank preconditioner for difficult PageRank problems. *J. Comput. Appl. Math.* **2019**, *346*, 456–470. [CrossRef]
26. Davis, T.A.; Hu, Y.; Wei, Y. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.* **2011**, *38*, 1:1–1:25. [CrossRef]
27. Boldi, P.; Vigna, S. The WebGraph Framework I: Compression Techniques. In Proceedings of the 13th International Conference on World Wide Web, New York, NY, USA, 17–20 May 2004; pp. 595–602.
28. Boldi, P.; Rosa, M.; Santini, M.; Vigna, S. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In Proceedings of the 20th International Conference on World Wide Web, Hyderabad, India, 28 March–1 April 2011; pp. 587–596.
29. Boldi, P.; Codenotti, B.; Santini, M.; Vigna, S. Ubicrawler: A scalable fully distributed Web crawler. *Softw. Pract. Exp.* **2004**, *34*, 711–726. [CrossRef]
30. Zhang, H.-F.; Huang, T.-Z.; Shen, Z.-L. FOM accelerated by an extrapolation method for solving PageRank problems. *J. Comput. Appl. Math.* **2016**, *296*, 397–409. [CrossRef]