Applied Mathematics and Computation 420 (2022) 126799

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Shifted power-GMRES method accelerated by extrapolation for solving PageRank with multiple damping factors

Zhao-Li Shen[a,b,1,*], Meng Su[a,1], Bruno Carpentieri[c,1], Chun Wen[d]

[a] College of Science, Sichuan Agricultural University, Ya'an, Sichuan 625000, PR China
[b] Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, AK Groningen 9700, The Netherlands
[c] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano 39100, Italy
[d] School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, PR China

ARTICLE INFO

Article history:
Received 20 January 2021
Revised 10 August 2021
Accepted 10 November 2021
Available online 2 December 2021

Keywords:
PageRank
Shifted linear systems
Multiple damping factors
Krylov subspace methods
Extrapolation
Power method

ABSTRACT

Starting from the seminal paper published by Brin and Page in 1998, the PageRank model has been extended to many fields far beyond search engine rankings, such as chemistry, biology, bioinformatics, social network analysis, to name a few. Due to the large dimension of PageRank problems, in the past decade or so, considerable research efforts have been devoted to their efficient solution especially for the difficult cases where the damping factors are close to 1. However, there exists few research work concerning about the solution of the case where several PageRank problems with the same network structure and various damping factors need to be solved. In this paper, we generalize the Power method to solving the PageRank problem with multiple damping factors. We demonstrate that the solution has almost the equative cost of solving the most difficult PageRank system of the sequence, and the residual vectors of the PageRank systems after running this method are collinear. Based upon these results, we develop a more efficient method that combines this Power method with the shifted GMRES method. For further accelerating the solving phase, we present a seed system choosing strategy combined with an extrapolation technique, and analyze their effect. Numerical experiments demonstrate the potential of the proposed iterative solver for accelerating realistic PageRank computations with multiple damping factors.

1. Introduction

The PageRank model was proposed by Google in a series of papers [1,4] to evaluate accurately the most important web-pages from the World Wide Web matching a set of keywords entered by a user. Nowadays, the model is routinely adopted for the analysis of many scientific problems far beyond Internet applications, for example in computational chemistry [7], biology [2], bioinformatics [2], social network analysis [5], bibliometrics [3], software debugging [6] and many others [8]. For search engine rankings, the importance of web-pages is computed from the stationary probability vector of the random process of a web surfer who keeps visiting a large set of web-pages connected by hyperlinks. The link structure of the World Wide Web is represented by a directed graph, the so-called web link graph, and its corresponding adjacency matrix

* Corresponding author.
E-mail addresses: szlxiaoyao@163.com (Z.-L. Shen), wchun17@163.com (C. Wen).
[1] These authors have contributed equally to this work.

https://doi.org/10.1016/j.amc.2021.126799
0096-3003/

$G \in \mathbb{N}^{n\times n}$, where n denotes the number of pages and $G(i, j)$ is nonzero (being 1) only if the jth page has a hyperlink pointing to the ith page. The transition probability matrix $P \in \mathbb{R}^{n\times n}$ of the random process has entries

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

To ensure that the random process has a unique stationary distribution and it will not stagnate, the transition matrix P is usually modified to be an irreducible stochastic matrix A (called the Google matrix) as follows

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

In (2), we define $\widetilde{P} = P + vd^T$, where $d \in \mathbb{N}^{n\times 1}$ is a binary vector tracing the indices of dangling web-pages with no hyperlinks, i.e., $d(i) = 1$ if the ith page has no hyperlink, $v \in \mathbb{R}^{n\times 1}$ is a probability vector, $e = [1, 1, \ldots, 1]^T$, and $0 < \alpha < 1$ is the so-called *damping factor* that represents the probability in the model that the surfer transfer by clicking a hyperlink rather than other ways. Mathematically, the PageRank model can be formulated as the problem of finding the positive unit eigenvector x (the so-called PageRank vector) such that

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

or, equivalently, as the solution of the linear system

$$(I - \alpha\widetilde{P})x = (1 - \alpha)v. \tag{4}$$

In the past decade or so, considerable research attention has been devoted to the efficient solution of problems (3)-(4), especially when n is very large. For moderate values of the damping factor, e.g. for $\alpha = 0.85$ as initially suggested by Google for search engine rankings, solution strategies based on the simple Power method have proved to be very effective. However, when α approaches 1, as is required in some applications (see e.g. [9]), the convergence rates of classical stationary iterative methods including the Power method tend to deteriorate sharply, and more robust algorithms need to be used. The development of fast solvers in this area includes extrapolation techniques [10,11], adaptive methods [12], multigrid methods [13,14], and inner-outer strategies [15–19]. Another research direction focuses on the analysis of non-stationary iterative solvers, such as Krylov subspace methods. For instance, some variants of the Arnoldi algorithm are proposed in [20,21]. Experiments with the full orthogonalization method (FOM) [43], the generalized minimal residual method (GMRES) [44] and the biconjugate gradient stabilized method (BiCGSTAB) [46] for PageRank computations are reported in [22–24,37]. According to Gleich et al. [25], Krylov algorithms can be in some cases more than 50% faster than stationary iterative methods when α is close to 1. More recent work is attempting to combine the simplicity of stationary iterative algorithms and the robustness of Krylov subspace solvers to compute difficult PageRank problems, see e.g. [26–29]. In this new framework, a few preliminary iterations of stationary methods are carried out to eliminate the high frequency components of the errors, providing a smooth initial solution for a faster Krylov subspace process. The very good potential of this idea has been demonstrated by solving several realistic PageRank problems very efficiently [26–29], and it has inspired the work presented in this paper.

One area that is largely unexplored in PageRank computations is the efficient solution of problems with the same network structure but multiple damping factors. For example, in the Random Alpha PageRank model used in the design of anti-spam mechanism [9], the rankings corresponding to many different damping factors close to 1 need to be computed simultaneously. This problem can be expressed mathematically as solving a sequence of linear systems

$$(I - \alpha_i\widetilde{P})x^{(i)} = (1 - \alpha_i)v, \quad 0 < \alpha_i < 1, \quad i = 1, 2, \ldots, s. \tag{5}$$

Conventional PageRank algorithms applied to (5) would solve the s linear systems independently. Although these solutions can be performed in parallel, the process would still demand large computational resources for high dimension problems. This consideration motivates the search of novel methods with reduced algorithmic and memory complexity, to afford the solution of larger problems on moderate computing resources.

We can write the PageRank problem with multiple damping factors given at once (5) as a sequence of shifted linear systems of the form:

$$\left(\frac{1}{\alpha_i}I - \widetilde{P}\right)x^{(i)} = \frac{1 - \alpha_i}{\alpha_i}v, \quad i = 1, 2, \ldots, s, \quad 0 < \alpha_i < 1. \tag{6}$$

Shifted variants of Krylov subspace algorithms can solve the whole sequence (6) simultaneously by using the same Krylov basis provided that the initial residual vectors are collinear at the cost of solving the most difficult linear system alone, owing to the shift-invariance property of the Krylov subspace. It is not difficult to see that when the initial solutions are chosen to be 0, the residual vectors of systems (6) are collinear. In practice, however, due to memory constraints the dimension of the search space must be often bounded by a small integer, and the Krylov method may need to be restarted. When this happens, the collinearity of the residual vectors may be lost and it has to be reinforced after each restart to maintain the shift-invariance property of the Krylov subspace[45].

Shifted Krylov methods may still suffer from slow convergence when the damping factor approaches 1, requiring larger search spaces to converge with satisfactory speed, which in turn may lead to unaffordable storage requirements for large-scale engineering applications. As an attempt of a possible remedy in this situation, we present a framework that combines

shifted stationary iterative methods and shifted Krylov subspace methods. In detail, we derive the implementation of the Power method that solves the PageRank problem with multiple damping factors at almost the same computational cost of the standard Power method for solving one single system. Furthermore, we demonstrate that this shifted Power method generates collinear residual vectors. Based on this result, we use the shifted Power iterations to provide smooth initial solutions for running shifted Krylov subspace methods such as GMRES. Besides, we discuss how to apply seed system choosing strategy and extrapolation techniques to further speed up the iterative process.

The paper is structured as follows. In Section 2, we briefly review stationary and Krylov subspace iterative algorithms, and we show how to combine these two approaches for solving practical PageRank problems. Then, we derive the Power-GMRES method [27]. In Section 3, we adapt stationary iterative methods to the solution of problems with multiple damping factors. We propose the new shifted Power method that maintains the collinearity of the residual vectors. In Section 4, we briefly introduce shifted Krylov subspace methods for solving sequences of shifted linear systems, and we propose the new shifted Power-GMRES method for solving PageRank problems with multiple damping factors. We also study the feasibility of extrapolation schemes and seed system choosing strategies to accelerate the shifted Power-GMRES iterations. Numerical experiments are reported in Section 5. Finally, we draw the conclusions arising from this work in Section 6. Note that MATLAB notation is used throughout the article.

2. Overview of iterative methods for solving the classical PageRank problem

The Power method is considered one of the algorithms of choice for solving either the eigenvalue (3) or the linear system (4) formulation of the PageRank problem, as it was originally used by Google. Power iterations write as

$$x_{k+1} = Ax_k = \alpha\widetilde{P}x_k + (1-\alpha)v,$$

and the convergence behaviour is determined mainly by the ratio between the two largest eigenvalues of A. When α gets closer to 1, though, the convergence can slow down significantly as established by the following theorems.

Theorem 2.1 ([30]). *Let P be a column-stochastic matrix with eigenvalues $\{1, \lambda_2, \lambda_3, \ldots, \lambda_n\}$. Then the eigenvalues of $A = \alpha P + (1-\alpha)ve^T$, where $0 < \alpha < 1$ and v is a vector with non-negative elements satisfying $e^T v = 1$, are $\{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$.*

Theorem 2.2 ([31]). *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 of $A = \alpha P + (1-\alpha)ve^T$ is given by α.*

Theorem 2.3 ([8]). *Let α, $\widetilde{P}$ and v be the data for a PageRank problem to compute a PageRank vector x. Then the error after k iterations of the update $x_{k+1} = \alpha\widetilde{P}x_k + (1-\alpha)v$ is as follows:*

1. *if $x_0 = v$, then $\|x - x_k\|_1 \le \|x - v\|_1 \alpha^k \le 2\alpha^k$;*
2. *if $x_0 = 0$, then the error vector $x - x_k \ge 0$ for all k and $\|x - x_k\|_1 = e^T(x - x_k) = \alpha^k$.*

The number of iterations required to reduce the initial residual down to a tolerance τ, measured as $\tau = \|Ax_k - x_k\| = \|x_{k+1} - x_k\|$, can be estimated as $\frac{\log_{10}\tau}{\log_{10}\alpha}$ [30]. For example, when $\tau = 10^{-8}$ the Power method requires about 175 steps to converge for $\alpha = 0.9$ but the iteration count rapidly grows to 1833 for $\alpha = 0.99$. Therefore, for values of the damping parameter very close to 1 more robust alternatives to the simple Power algorithm should be used.

The Jacobi method does not improve significantly the convergence behaviour of the Power method because a large proportion of the diagonal elements in $\widetilde{P}$ are 0, due to the fact that there is almost no page pointing to itself. On the other hand, the Gauss-Seidel method is always faster than the Jacobi and the Power methods since $(I - \alpha\widetilde{P})$ is an M-matrix [33]. However, it is not easy to parallelize. Although the successive over-relaxation (SOR) method is often more efficient than Gauss-Seidel for solving general linear systems [34], this may not be the case for PageRank applications, as shown in [35]. Overall, Gauss-Seidel may be considered one of the best stationary methods to use in this context.

Recently, considerable attention has been devoted to inner-outer (IO) iterative schemes of the form

$$x_{k+1} \approx (I - \beta\widetilde{P})^{-1}((\alpha-\beta)\widetilde{P}x_k + (1-\alpha)v), \tag{7}$$

for solving (4), which are derived from the splitting $I - \alpha\widetilde{P} = I - \beta\widetilde{P} - (\alpha-\beta)\widetilde{P}$. Here, "$\approx$" means that $(I - \beta\widetilde{P})^{-1}((\alpha - \beta)\widetilde{P}x_k + (1-\alpha)v)$ is formed by an approximate solution y of the inner system

$$(I - \beta\widetilde{P})y = f, \quad \text{with} \quad f = ((\alpha-\beta)\widetilde{P}x_k + (1-\alpha)v). \tag{8}$$

Note that algorithm (7) reduces to a stationary iterative algorithm if the inner system is solved by a fixed number of steps of a stationary solver. Experiments in [15] show that inner-outer iterations can accelerate both the Power and the Gauss-Seidel methods for large scale PageRank computations. The inner-outer framework (7)-(8) can accommodate any regular splitting of the form $I - \alpha\widetilde{P}$ as in the general inner-outer iteration (GIO in short) scheme proposed by Tian et. al. [18]. For any regular splitting $I - \alpha\widetilde{P} = M - N$, the GIO iteration writes as

$$(M - \phi N)x_{k+1} = (1-\phi)Nx_k + (1-\alpha)v, \quad 0 < \phi < 1. \tag{9}$$

Numerical results presented in [18] demonstrate that GIO iterations can converge faster than the IO iterations. Some variants of this idea utilise a few steps of the Power method as a preconditioner for the inner-outer iteration and can be seen as

forms of multi-step splitting methods. They include the power-inner-outer (PIO) iteration [16], the multi-step power-inner-outer (MPIO) iteration [17] and the relaxed two-step splitting iteration (RTSS) [36] methods.

The above mentioned nested iterative schemes are also applied to accelerate nonstationary Krylov subspace methods. In fact, although Krylov methods are generally significantly more robust than stationary methods, their convergence depends heavily on the dimension m of the search space, and they can still be slow or can even stagnate when α is very close to 1. Recently, it has been demonstrated that the robustness of Krylov methods can be enhanced by carrying out a few preliminary inexpensive stationary iterations to dump the high-frequency components of the error. See e.g. the experiments reported in [27] with the Power-GMRES method, in [26] with the Power-Arnoldi method, in [28] with the Power-GArnoldi method and in [29] the Arnoldi-Inout method. Another class of nonstationary iterative methods includes algebraic multigrid methods (AMGs), see e.g. [13,14]. At each iterative step, AMGs solve a cascade of coarse-level equations obtained by aggregating sets of degrees of freedom according to different criteria. Such methods are beyond the scope of this paper; we point the reader to [13,14] for their application to PageRank problems. Finally, we mention other forms of acceleration, such as extrapolation algorithms [10,11,24,38], blocking or reordering strategies that exploits the structure of the Web link graph [39–42], and ad-hoc preconditioning techniques [24,42] that have been developed in this context.

Note that all the above-mentioned solution approaches are developed for solving the classical PageRank problem. Their application to the case with multiple damping factors is not completely straightforward and requires further analysis.

3. The shifted power method for PageRank computations

In this section we consider extensions of stationary iterative methods for the solution of PageRank problems with multiple damping factors. We look in particular at the Power method, the Gauss-Seidel method, and the GIO iteration scheme. We are concerned with how these methods can be executed with the highest efficiency for solving such problems, especially with the question: for each method, whether there exist an implementation such that the computational cost of solving the PageRank problem with multiple damping factor is comparable to that of solving the ordinary PageRank problem with single damping factor. This topic needs investigation.

3.1. The implementation of the shifted power method

Inspired by the reason why shifted Krylov subspaces can save computational cost, we investigate whether there are duplications in the calculations of multiple linear systems in this problem class by the stationary iterative methods, so that the duplications in the computation can be deleted, or in other words, the associate operations can be computed only once and used for all systems. We first analyze the Power method applied to the sequence of linear systems in (5). It computes at the kth iteration approximate solutions $x_k^{(i)}$ $(1 \le i \le s)$ of the form

$$x_k^{(i)} = \alpha_i \widetilde{P} x_{k-1}^{(i)} + (1-\alpha_i)v \tag{10}$$

$$= \alpha_i \widetilde{P}(\alpha_i \widetilde{P} x_{k-2}^{(i)} + (1-\alpha_i)v) + (1-\alpha_i)v \tag{11}$$

$$= \alpha_i^2 \widetilde{P}^2 x_{k-2}^{(i)} + (1-\alpha_i)(v + \alpha_i \widetilde{P} v) \tag{12}$$

$$= \alpha_i^2 \widetilde{P}^2 (\alpha_i \widetilde{P} x_{k-3}^{(i)} + (1-\alpha_i)v) + (1-\alpha_i)(v + \alpha_i \widetilde{P} v) \tag{13}$$

$$= \alpha_i^3 \widetilde{P}^3 x_{k-3}^{(i)} + (1-\alpha_i)(v + \alpha_i \widetilde{P} v + \alpha_i^2 \widetilde{P}^2) \tag{14}$$

$$= \cdots \tag{15}$$

$$= \alpha_i^k \widetilde{P}^k x_0^{(i)} + (1-\alpha_i) \sum_{j=0}^{k-1} \alpha_i^j \widetilde{P}^j v. \tag{16}$$

Then it is clearly that if the same initial guess $x_0^{(i)} = x_0$ is used for all the linear systems, the latter iteration formulae will be simplified as

$$x_k^{(i)} = \alpha_i^k \widetilde{P}^k x_0 + (1-\alpha_i) \sum_{j=0}^{k-1} \alpha_i^j \widetilde{P}^j v. \tag{17}$$

If the s systems in (5) are solved synchronously, that is all $x_k^{(i)}$'s are computed only after all previous approximations $x_{k-1}^{(j)}$'s $(1 \le j \le s)$ are available, then the computation can be rearranged efficiently as follows:

- at the first iteration,
  - compute and store $\mu_1 = \widetilde{P}x_0$ and $\mu_2 = v$;
  - compute and store $x_1^{(i)} = \alpha_i\mu_1 + (1-\alpha_i)\mu_2 \ (1 \le i \le s)$;
- at any other subsequent iteration $k > 1$,
  - compute and store $x_k^{(i)} := (1-\alpha_i)\sum_{j=0}^{k-2}\alpha_i^j\widetilde{P}^j v = x_{k-1}^{(i)} - \alpha_i^{k-1}\mu_1 \ (1 \le i \le s)$;
  - compute and store $\mu_1 = \widetilde{P}\mu_1$ and $\mu_2 = \widetilde{P}\mu_2$ these two vectors;
  - compute and store $x_k^{(i)} = \alpha_i^k\mu_1 + x_k^{(i)} + (1-\alpha_i)\alpha_i^{k-1}\mu_2 \ (1 \le i \le s)$.

This implementation requires at most 2 matrix-vector products at each step, which is a significant gain compared to the s matrix-vector products required by the standard Power method to compute $x_{k+1}^{(i)}$, especially when $s \gg 2$. This is close to the computational cost, i.e. 1 matrix-vector product per iteration, of using the Power method for computing PageRank with single damping factor. Furthermore, we will show that the computational cost per iteration can be further reduced.

Note that, the option to choose any the same initial guess x_0 for the s systems is acceptable since the Power method converges for every positive initial probability vector [30]. In particular, it can be found that if $x_0 = v$ then the vector $\widetilde{P}^{k-1}v = \widetilde{P}^{k-1}x_0$ is available from the $(k-1)$th iteration. In this case, only one matrix-vector product $\widetilde{P}^k x_0 = \widetilde{P}\cdot\widetilde{P}^{k-1}x_0$ is needed at the kth iteration. Accordingly, computing all the approximations $x_k^{(i)}$ $(1 \le i \le s)$ has the same computational cost, i.e. 1 matrix-vector product, of solving one linear system in (5), as we wished. The Power iteration formulae (17) then writes as

$$x_i^{(k)} = \alpha_i^k\widetilde{P}^k v + (1-\alpha_i)\sum_{j=0}^{k-1}\alpha_i^j\widetilde{P}^j v \tag{18}$$

$$= \sum_{j=1}^{k}\alpha_i^j\widetilde{P}^j v + v - \sum_{j=1}^{k}\alpha_i^j\widetilde{P}^{j-1}v \tag{19}$$

$$= \sum_{j=1}^{k}\alpha_i^j\widetilde{P}^{j-1}(\widetilde{P}v - v) + v. \tag{20}$$

An efficient implementation can compute and store $\mu = \widetilde{P}v - v$ at the first iteration, then compute and store $\mu = \widetilde{P}^{k-1}(\widetilde{P}v - v) = \widetilde{P}\cdot(\widetilde{P}^{k-2}(\widetilde{P}v - v))$ at each kth $(k > 1)$ iteration, and finally form each approximate solution as $x_k^{(i)} = \alpha_i^k\mu + x_{k-1}^{(i)}$ $(1 \le i \le s)$. The residual vector $r_k^{(i)}$ associated with the approximate solution $x_k^{(i)}$ has the following expression

$$r_k^{(i)} = Ax_k^{(i)} - x_k^{(i)} \tag{21}$$

$$= x_{k+1}^{(i)} - x_k^{(i)} \tag{22}$$

$$= \alpha_i^{k+1}\widetilde{P}^k(\widetilde{P}v - v)). \tag{23}$$

Since in general each of the s linear systems may require a different number of Power iterations to converge, the s residual norms have to be monitored separately to test the convergence. We summarize the efficient implementation of the Power method that we presented in this section for solving problem (5) in Algorithm 1, and we call it the shifted Power method hereafter.

3.2. Whether other stationary methods can perform better than the shifted power method?

Following the same lines of analysis, we can try to derive an efficient implementation of the Gauss-Seidel method for solving PageRank problems with multiple damping factors. The Gauss-Seidel method is based on the splitting $I - \alpha\widetilde{P} = D_\alpha - L_\alpha - U_\alpha$, where D_α, $-L_\alpha$, and $-U_\alpha$ are the diagonal, the lower triangular and the upper triangular parts of $I - \alpha\widetilde{P}$, respectively. The iteration formulae of the method writes as

$$x_k = (D_\alpha - L_\alpha)^{-1}[U_\alpha x_{k-1} + (1-\alpha)v], \tag{24}$$

$$= (D_\alpha - L_\alpha)^{-1}U_\alpha x_{k-1} + (D_\alpha - L_\alpha)^{-1}(1-\alpha)v, \tag{25}$$

$$= T_{G,\alpha}x_{k-1} + b, \quad \text{where} \quad T_{G,\alpha} = (D_\alpha - L_\alpha)^{-1}U_\alpha, \ b = (D_\alpha - L_\alpha)^{-1}(1-\alpha)v, \tag{26}$$

$$= T_{G,\alpha}^k x_0 + \sum_{j=0}^{k-1}T_{G,\alpha}^j b. \tag{27}$$

Algorithm 1 Shifted-Power method for PageRank with multiple damping factors: $[mv, x^i, r^{(i)}(i = 1, \cdots, s)]$=Shifted-Power($\widetilde{P}$, v, τ, max_{mv}, α_i $(1 \le i \le s)$).

Input: ~$\widetilde{P}$, v, τ, max_{mv}, α_i $(1 \le i \le s)$
Output: mv, $x^{(i)}$, $r^{(i)}$ $(1 \le i \le s)$;
Compute $\mu = \widetilde{P}v - v$;
Set $mv = 1$;
for $i = 1 : s$ **do**
  Compute $r^{(i)} = \alpha_i \mu$;
  Compute $Res(i) = \|r^{(i)}\|_2$;
  if $Res(i) \ge \tau$ **then**
    Compute $x^{(i)} = r^{(i)} + v$;
  end if
end for
while $max(res) \ge \tau$ && $mv \le max_{mv}$ **do**
  Compute $\mu = \widetilde{P}\mu$;
  Set $mv = mv + 1$;
  for $i = 1 : s$ **do**
    if $Res(i) \ge \tau$ **then**
      Compute $r^{(i)} = \alpha_i^{k+1}\mu$;
      Compute $Res(i) = \|r^{(i)}\|_2$;
      if $Res(i) \ge \tau$ **then**
        Compute $x^{(i)} = r^{(i)} + x^{(i)}$;
      end if
    end if
  end for
end while

Although these formulaes write similar to the Power method formulae (17), one important difference is that $T^k_{G,\alpha_i}u$ and $T^k_{G,\alpha_j}u$[2] are not collinear when $\alpha_i \neq \alpha_j$. This means that $(D_{\alpha_i} - L_{\alpha_i})^{-1}u$ and $(D_{\alpha_j} - L_{\alpha_j})^{-1}u$ do not share the same computational process, and therefore it is not possible to reduce the computational costs by optimizing the implementation in the case of multiple damping factors, suggesting that the shifted Power method would be a better option than the Gauss-Seidel method for solving the PageRank problem with multiple damping factors, especially when s is large.

Note that, for PageRank problems, the implementation and the performance of the Jacobi method are similar to those of the Power method, and the implementation of the SOR method is similar as that of the Gauss-Seidel method but does not perform any better than it[3]. Thus, these two methods are not better choices than the Power method for developing efficient solvers for the PageRank problem with multiple damping factors.

The last type of stationary methods we consider is the GIO method. Based on the above considerations, the Jacobi, Gauss-Seidel and SOR splittings are not sensible choices to consider as the outer or inner solver in the GIO method. Hence, in this study we focus on the GIO method using the Power method as both the outer and the inner solver, similarly to Gleich et al. [15]. With this choice in mind, the inner-outer scheme with m inner iteration steps, damping factor α and the parameter β writes as

$$x_k = \beta^m \widetilde{P}^m x_{k-1} + \sum_{j=0}^{m-1} \beta^j \widetilde{P}^j((\alpha - \beta)\widetilde{P}x_{k-1} + (1-\alpha)v), \tag{28}$$

$$= (\beta^m \widetilde{P}^m + (\alpha - \beta)T_l\widetilde{P})x_{k-1} + (1-\alpha)T_l v, \quad \text{where} \quad T_l = \sum_{j=0}^{m-1} \beta^j \widetilde{P}^j, \tag{29}$$

$$= (\beta^m \widetilde{P}^m + (\alpha - \beta)T_l\widetilde{P})^k x_0 + (1-\alpha)\sum_{j=0}^{k-1}(\beta^m \widetilde{P}^m + (\alpha - \beta)T_l\widetilde{P})^j T_l v. \tag{30}$$

If the initial guess $x_0 = v$ is used, the above formula writes as

$$x_k = (\beta^m \widetilde{P}^m + (\alpha - \beta)T_l\widetilde{P})^k v + (1-\alpha)\sum_{j=0}^{k-1}(\beta^m \widetilde{P}^m + (\alpha - \beta)T_l\widetilde{P})^j T_l v. \tag{31}$$

[2] Here, u means an arbitrary vector from $\mathbb{R}^{n\times 1}$.

[3] Indeed, the best choice of the parameter ω in the SOR method when solving the PageRank problem is $\omega = 1$, which is just the Gauss-Seidel method [35].

We can see that x_k belongs to the subspace $span\{v, \widetilde{P}v, \widetilde{P}^2v, \cdots, \widetilde{P}^{mk}v\}$, and the damping factors α's form the coefficients of the linear combination $x_k = l_0v + l_1\widetilde{P}v + l_2\widetilde{P}^2v + \cdots + l_{mk}\widetilde{P}^{mk}v$. This suggests that the inner-outer formulae to approximate the solution of the whole sequence of linear systems (5) at the $k+1$th step can be evaluated efficiently as follows:

- compute and store the m vectors $\widetilde{P}^{(mk+1)}v, \widetilde{P}^{(mk+2)}v, \cdots, \widetilde{P}^{(mk+m)}v$;
- use formula (30) to compute the coefficients of the approximate solution $x_{k+1}^{(i)}$ for all the ith systems, then form $x_{k+1}^{(i)}$ $(1 \le i \le s)$.

By this implementation, only m matrix-vector products are needed to construct $x_{k+1}^{(i)}$ $(1 \le i \le s)$ at the $k+1$th step, leading to a considerable gain compared to the sequential solution of using the ordinary inner-outer method for systems (5) that would require ms matrix-vector products. From the memory viewpoint, the procedure above stores the $m(k+1)+1$ vectors $\{v, \widetilde{P}v, \widetilde{P}^2v, \cdots, \widetilde{P}^{m(k+1)}v\}$ of $\mathbb{R}^{n\times 1}$, where $m(k+1)$ is the number of matrix-vector products computed by the first $k+1$ iterations of this inner-outer scheme. That is, the memory cost of this inner-outer scheme increases linearly with the number of iterations, and it is much larger than the constant memory cost of the shifted Power method. Based on the experiments presented in [15], the overall number of matrix-vector products required by the inner-outer method applied to PageRank problems is typically not less than 60% of that required by the Power method. Overall, the procedure above is too demanding memory-wise to be applicable in the case of solving high-dimensional PageRank problems with multiple damping factors.

We conclude that a general stationary iterative method for solving the PageRank problem (4), derived from the general splitting $I - \alpha P = M_\alpha - N_\alpha$ with iteration formulae

$$x_{k+1} = \widetilde{M}_\alpha^{-1}N_\alpha x_k + (1-\alpha)\widetilde{M}_\alpha^{-1}v, \quad \text{where} \quad \widetilde{M}_\alpha^{-1} \approx M_\alpha^{-1},$$

can be extended efficiently to the case of multiple damping factors when the iteration matrices $\widetilde{M}_\alpha^{-1}N_\alpha$ corresponding to different values of α are "collinear", that is when $\widetilde{M}_{\alpha_i}^{-1}N_{\alpha_i} = \theta\widetilde{M}_{\alpha_j}^{-1}N_{\alpha_j}$ $(0 < \alpha_i, \alpha_j < 1,\ \theta \in \mathbb{R})$. The only option satisfying this condition that we found is the Power method, and this will be the basis for our development in the next section.

4. Shifted power-GMRES method for computing PageRank problems with multiple dampling factors

In this section, we combine the simplicity of the shifted Power method with the fast convergent shifted GMRES method to produce a robust hybrid solution scheme for solving difficult PageRank problems with multiple damping factors.

4.1. The shifted power-GMRES method with a seed system choosing strategy

First, we briefly review the restarted GMRES method (hereafter referred to as GMRES in short), which is a nonsymmetric Krylov subspace solver based on the Arnoldi decomposition procedure [43] sketched in Algorithm 2. Given a matrix $A \in \mathbb{R}^{n\times n}$

Algorithm 2 $[V_m, H_m, v_{m+1}, h_{m+1,m}, \beta, j]$=*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**

and an initial vector $v_0 \in \mathbb{R}^{n\times 1}$, after m steps Algorithm 2 constructs an orthonormal basis $\{v_1, v_2, \ldots, v_m\}$ of the Krylov subspace $\mathcal{K}_m(A, v_0) \equiv span\{v_0, Av_0, A^2v_0, \ldots, A^{m-1}v_0\}$. The Arnoldi decomposition can be written as

$$AV_m = V_{m+1}\bar{H}_m \tag{32}$$

$$= V_mH_m + h_{m+1,m}v_{m+1}e_m^T \tag{33}$$

with

$$V_m^TAV_m = H_m, \tag{34}$$

where we denote by $V_m = [v_1, v_2, \ldots, v_m] \in \mathbb{R}^{n\times m}$ the matrix with orthonormal columns v_i, by $V_{m+1} = [V_m, v_{m+1}]$, while $H_m = \{h_{i,j}\} \in \mathbb{R}^{m\times m}$ is an upper Hessenberg matrix, $\bar{H}_m = [H_m; h_{m+1,m}e_m^T] \in \mathbb{R}^{(m+1)\times m}$, and $e_m = [0, 0, \ldots, 1]^T \in \mathbb{R}^{m\times 1}$. Starting from $v_0 = b - Ax_0$ with an initial guess x_0, after running m steps of Algorithm 2 the GMRES method computes the approximate solution $\widetilde{x}$ of the linear system $Ax = b$ that minimizes the residual norm $\|b - A\widetilde{x}\|_2$ in the space $x_0 + \mathcal{K}_m(A, v_0)$. By writing $\widetilde{x} = x_0 + V_m y$, in the GMRES method the following equation has to be solved for the vector y:

$$\min_{\widetilde{x}\in x_0+\mathcal{K}_m(A,v_0)} \|b - A\widetilde{x}\|_2 = \min_{y\in\mathbb{R}^{m\times 1}} \|r_0 - AV_m y\|_2 = \min_{y\in\mathbb{R}^{m\times 1}} \|\beta V_{m+1}e_1 - V_{m+1}\bar{H}_m y\|_2,$$

where $e_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^{(m+1)\times 1}$. Because V_{m+1} is an orthonormal transformation, the last equation can be simplified as

$$\min_{\widetilde{x}\in x_0+\mathcal{K}_m(A,v_0)} \|b - A\widetilde{x}\|_2 = \min_{y\in\mathbb{R}^{m\times 1}} \|\beta e_1 - \bar{H}_m y\|_2.$$

Therefore the computation of y, and hence of $\widetilde{x}$, reduces to solving the small size least squares problem $\min_{y\in\mathbb{R}^{m\times 1}} \|\beta e_1 - \bar{H}_m y\|_2$.

The accuracy of the approximate solution $\widetilde{x}$ of GMRES depends heavily on the dimension m of the search space. If the linear system dimension n is very large and the value of m has to be by necessity a small integer because of memory constraints, the accuracy of $\widetilde{x}$ may be unsatisfactory. In such case, the GMRES method needs to be restarted using $\widetilde{x}$ as the starting vector for a new cycle of m iterations. The process is repeated until the required accuracy is achieved. The GMRES method with restart applied to the solution of the PageRank problem is outlined in Algorithm 3.

Algorithm 3 Restarted GMRES method for solving the PageRank linear system $(I - \alpha\widetilde{P})x = (1 - \alpha)v$.

Input: $\sim\widetilde{P}$, α, v, m, τ, $maxit$, initial guess $x \in \mathbb{R}^{n\times 1}$ with $\|x\|_1 = 1$;
Output: x, Res, mv, $iter$;

```
Set mv = 0;
for iter = 1 : maxit do
    Compute v_0 = (1 − α)v − (I − αP̃)x;
    Run Algorithm 2 by [V_m, H_m, v_{m+1}, h_{m+1,m}, β, j]=Arnoldi((I − αP̃), v_0, m);
    mv = mv + j;
    Solve the least square problem y = min_y ‖βe_1 − H̄_m y‖_2;
    Set Res = ‖βe_1 − H̄_m y‖_2;
    Form the approximate solution: x = x + V_m y;
    if Res < τ then
        break;
    end if
end for
```

The shifted GMRES method proposed by Frommer and Glassner is an extension of the standard GMRES algorithm that can use the same Krylov subspace to solve the whole sequence of shifted linear systems $(A + \sigma_i I)x_i = b_i$ $(\sigma_i \in \mathbb{R}, b_i \in \mathbb{R}^{n\times 1}, i = 1, \ldots, s)$ simultaneously [45]. The core idea of the method is to run standard GMRES on a seed system and to approximate the other solutions as by products. The theoretical basis is the shift-invariance property of the Krylov subspace, namely $\mathcal{K}_m(kI + lA, cv) \equiv \mathcal{K}_m(A, v)$ for any $k, l, c \in \mathbb{R}$, that enables us to use only one Krylov subspace to solve the whole sequence at once provided that the initial residual vectors of the shifted linear systems are collinear one another. However, after m steps of Algorithm 2 the residuals vectors r_m^i $(i = 2, 3, \ldots, s)$ of the additional systems and the residual vector r_m^1 of the seed system will in general lose the collinearity, and consequently the shift-invariance property of the search space will not be maintained, that is $\mathcal{K}_m(A + \sigma_i I, r_m^i) \neq \mathcal{K}_m(A + \sigma_1 I, r_m^1)$, for $i = 2, 3, \ldots, s$. This means: upon restarting the GMRES algorithm, the mutual collinearity of all the residual vectors needs to be enforced explicitly. More precisely, let us assume that the initial residual vectors are collinear, i.e. $r_0^i = \gamma_0^i r_0^1$ $(\gamma_0^i \in \mathbb{R}, i = 2, 3, \ldots, s)$ where r_0^i is the initial residual vector corresponding to the ith system. Note the following two facts:

- the initial residual vectors of the shifted systems are collinear with $(\gamma_0^i = 1)$ when the initial guesses are $x_0^i = 0$ $(i = 1, 2, \ldots, s)$;
- if $(A + \sigma_1 I)V_m = V_{m+1}\bar{H}_m^1$ is the Arnoldi decomposition of the base system, then the decomposition of the ith shifted system writes as $(A + \sigma_i I)V_m = V_{m+1}\bar{H}_m^i$ with an upper Hessenberg matrix

$$\bar{H}_m^i = \bar{H}_m^1 + (\sigma_i - \sigma_1)\begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ 0 & \cdots & 0 \end{pmatrix}.$$

Denoting by $x_m^i = x_0^i + V_m y_m^i$ the approximate solution of the ith system after an m-steps cycle, we can derive the collinearity condition to impose on the residual vectors as follows

$$r_0^i - (A + \sigma_i I)V_m y_m^i = \gamma_m^i [r_0^1 - (A + \sigma_1 I)V_m y_m^1] \tag{35}$$

$$\iff \quad \gamma_0^i r_0^1 - (A+\sigma_i I)V_m y_m^i = \gamma_m^i [r_0^1 - (A+\sigma_1 I)V_m y_m^1] \tag{36}$$

$$\iff \quad \gamma_0^i \beta V_m e_1 - (A+\sigma_i I)V_m y_m^i = \gamma_m^i [r_0^1 - (A+\sigma_1 I)V_m y_m^1] \tag{37}$$

$$\iff \quad \gamma_0^i \beta V_{m+1} e_1 - V_{m+1}\bar{H}_m^i y_m^i = \gamma_m^i [\beta V_{m+1} e_1 - V_{m+1}\bar{H}_m^1 y_m^1] \tag{38}$$

$$\gamma_0^i \beta e_1 = \gamma_m^i [\beta e_1 - \bar{H}_m^1 y_m^1] + \bar{H}_m^i y_m^i. \tag{39}$$

Equation (39) can be converted in matrix form as

$$\begin{bmatrix} \bar{H}_m^i & z \end{bmatrix} \begin{bmatrix} y_m^i \\ \gamma_m^i \end{bmatrix} = \gamma_0^i \beta e_1, \tag{40}$$

where $\beta = \|r_0^1\|_2$, $z = \beta e_1 - \bar{H}_m^1 y_m^1$. By solving this small size system, we can obtain vector y_m^i and scalar γ_m^i that ensure the collinearity of the shifted residuals. This is the approach of generating approximate solutions for additional systems and maintaining the collinearity of residual vectors in the restarted shifted GMRES method. For more details about this procedure, see [45].

One computational aspect to consider for the shifted GMRES method is that if some of the additional systems converge much more slowly than the seed system, the latter one may need to be solved by far more iterations than it is actually required. The use of a seed switching technique can remedy the problem, and this is still an open research topic. For this special problem, we apply a seed choosing strategy that is enough and is introduced as follows.

In [45], Frommer and Glassner prove the following result:

Theorem 4.1 ([45]). *Let A be a positive real matrix, i.e. $Re(< Ax, x >) > 0$ for all $x \neq 0$, and $\alpha > 0$, $Ax = b$ be the seed system, and $(A+\alpha I)x = b$ be the additional system. Then the restarted shifted GMRES method converges for the seed and the additional system for every restart value $m \in \mathbb{N}^+$. In particular, the iterative solutions for the additional system always exist. Moreover, it holds:*

$$\|\hat{r}^{jm}\|_2 \leq \|r^{jm}\|_2,$$

where r^{jm} and $\hat{r}^{jm}$ are the residual vectors of the seed system and the additional system after any jth restart, respectively.

Theorem 4.1 can be applied to the solution of the PageRank shifted linear system (6) since the coefficient matrices $\frac{1}{\alpha_i} I - \widetilde{P}$ $(i = 1, \cdots, s)$ are positive real. Suppose that $\alpha_1 < \alpha_2 < \cdots < \alpha_s$ in (6). If we choose the linear system associated with the damping factor α_s as our seed system, the coefficient matrix of any other ith system can be rewritten as $\left(\frac{1}{\alpha_i} - \frac{1}{\alpha_s}\right) I + \left(\frac{1}{\alpha_s} I - \widetilde{P}\right)$. Then, according to Theorem 4.1, the whole sequence converges and the seed system is the most slowly convergent of the sequence. Finally, we outline the shifted-GMRES method for solving the PageRank problem with multiple damping factors in Algorithm 4.

Although the shifted GMRES method can solve the whole sequence (6) at roughly the cost of solving one of its systems, it will still suffer from slow convergence for value of the damping factor that is most close to 1, especially if the dimension m of the search space is set equal to a small integer due to memory constraints. Hence, it is attractive to develop some strategies to accelerate this method for solving this problem class.

Motivated by the hybrid framework proposed in [26,27,29], since according to equation (23) the residual vectors of the shifted Power algorithm remain collinear after every cycle, we propose to carry out a few cheap iterations of the shifted Power method to reduce the high frequency components of the error associated to the small eigenvalues of the coefficient matrix, followed by some steps of the shifted GMRES method to reduce the low frequency components of the errors. By this strategy, we aim at improving the efficiency of the shifted GMRES method for solving the PageRank problem with multiple damping factors including some close to 1. Hereafter, the resulting method will be referred to as the shifted Power-GMRES method, and its computational steps are outlined in Algorithm 5.

4.2. An Anderson-type extrapolation technique

Motivated by the extrapolation techniques proposed in [10,11,24,47], we consider whether the convergence of the shifted Power-GMERS method can be further accelerated by applying extrapolation techniques. The above-mentioned extrapolation techniques form a new approximate solution $\bar{x}_i$ as a linear combination $\bar{x}_i = l_i x_i + l_{i-1} x_{i-1} + \cdots + l_{i-k+1} x_{i-k+1}$ of k solutions x_j $(i-k+1 \leq j \leq i)$ computed at previous steps, and the coefficients $l_i, \cdots, l_{i-k+1} \in \mathbb{R}$ are specially chosen such that a more rapid convergence can be achieved. Different extrapolation techniques correspond to different choices of $0 \leq l_i, \cdots, l_{i-k+1} \leq 1$. In this study, we follow the idea of the Anderson acceleration [47] to carry out the investigation.

Algorithm 4 shifted-GMRES method for PageRank with multiple damping factors: $[x^i,\ res_i\ (1 \le i \le s),\ mv]$=shifted-GMRES($\widetilde{P}$, v, m, α_i, τ, $maxit$, $x_0^i\ (1 \le i \le s)$).

Input: ~$\widetilde{P}$, v, m, α_i, τ, $maxit$, $x_0^i\ (1 \le i \le s)$;
Output: x^i, $res_i\ (1 \le i \le s)$, mv;
1: Set $r_0^i = \frac{1-\alpha_i}{\alpha_i} v - (\frac{1}{\alpha_i} I - \widetilde{P}) x_0^i$, $e_1 = [1, 0, \ldots, 0]$, $iter = 1$;
2: Set $res_i = \alpha_i \|r_0^i\|_2\ (1 \le i \le s)$;
3: Set $mv = 0$;
4: **while** $max(res_i) \ge \tau$ && $iter \le maxit$ **do**
5: Find k that satisfies $res_k = max(res_i)$;
6: Compute $\gamma^i = \frac{res_i \alpha_k}{res_k \alpha_i}$ for all $(i \ne k)$;
7: Run Algorithm 2 by $[V_m, \bar{H}_m^k, v_{m+1}, \bar{h}_{m+1,m}, \beta, j]$=*Arnoldi*($(\frac{1}{\alpha_k} I - \widetilde{P})$, r_0^k, m);
8: Set $mv = mv + j$;
9: Compute y^k, the minimizer of $\|\beta e_1 - \bar{H}_m^k y^k\|_2$;
10: Compute $x^k = x_0^k + V_m y^k$, $res_k = \alpha_k \|\beta e_1 - V_m y^k\|_2$;
11: **for** $i = 1, 2, \cdots, k-1, k+1, \cdots, s$ **do**
12: **if** $res_i \ge \tau$ **then**
13: Set $\bar{H}_m^i = \bar{H}_m^k + (\frac{1-\alpha_i}{\alpha_i} - \frac{1-\alpha_k}{\alpha_k}) I_m$;
14: Solve y^i and γ^i from $\begin{bmatrix} \bar{H}_m^i & z \end{bmatrix} \begin{bmatrix} y^i \\ \gamma^i \end{bmatrix} = \gamma^i \beta e_1$;
15: set $x^i = x_0^i + V_m y^i$;
16: set $res_i = \frac{\alpha_i}{\alpha_k} \gamma_k^i res_k$;
17: **end if**
18: **end for**
19: Set $x_0^i = x^i$, $iter = iter + 1$;
20: **end while**

Algorithm 5 shifted-Power-GMRES method for PageRank with multiple damping factors: $[x^i,\ res_i\ (1 \le i \le s),\ iter]$=shifted-Power-GMRES($\widetilde{P}$, v, $\alpha_i\ (i = 1, \cdots, s)$, τ, m, $maxit_p$ $maxit_g$).

Input: ~$\widetilde{P}$, v, $\alpha_i\ (i = 1, \cdots, s)$, τ, m, $maxit_p$ $maxit_g$;
Output: x^i, $res_i\ (1 \le i \le s)$, mv;
1: Run Algorithm 1 by $[mv_p, x^i, r_i\ (i = 1, \cdots, s)]$=Shifted-Power($\widetilde{P}$, v, τ, $maxit_p$, $\alpha_i\ (1 \le i \le s)$);
2: Run Algorithm 4 by $[x^i, res_i\ (1 \le i \le s), mv_g]$=shifted-GMRES($\widetilde{P}$, v, m, α_i, τ, $maxit_g$, $x^i\ (1 \le i \le s)$)
3: Set $mv = mv_p + mv_g$;

We choose the coefficients $l_i, \cdots, l_{i-k+1}$ in a way that $\bar{x}_i = l_i x_i + l_{i-1} x_{i-1} + \cdots + l_{i-k+1} x_{i-k+1}$ is guaranteed to have the smallest residual norm $\|b - A\bar{x}_i\|_2$ among all the choices. For the PageRank problem, the approximate solution $\bar{x}_i$ must satisfy $\|\bar{x}_i\|_1 = 1$ and be non-negative. Therefore, there must be $0 \le l_i, \cdots, l_{i-k+1} \le 1$ and $\sum_{j=0}^{k-1} l_{i-j} = 1$. If s linear systems have to be solved simultaneously as in the case of multiple damping factors, the extrapolation method needs to be adapted to ensure the collinearity of all the residual vectors. Supposing that $r_i^{(j)} = \beta_i^{(j)} r_i^{(1)}$, $1 \le j \le s$ where $r_i^{(j)}$ denotes the residual vector of the jth system after the ith iteration, meanwhile the Anderson acceleration applied to the first system yields $\bar{x}_i^{(1)} = l_i^{(1)} x_i^{(1)} + l_{i-1}^{(1)} x_{i-1}^{(1)} + \cdots + l_{i-k+1}^{(1)} x_{i-k+1}^{(1)}$, and the corresponding residual vector writes

$$\bar{r}_i^{(1)} = (1 - \alpha_1) v - (I - \alpha_1 P) \bar{x}_i^{(1)} \tag{41}$$

$$= l_i^{(1)} ((1 - \alpha_1) v - (I - \alpha_1 P) x_i^{(1)}) + \cdots + l_{i-k+1}^{(1)} ((1 - \alpha_1) v - (I - \alpha_1 P) x_{i-k+1}^{(1)}) \tag{42}$$

$$= l_i^{(1)} r_i^{(1)} + l_{i-1}^{(1)} r_{i-1}^{(1)} + \cdots + l_{i-k+1}^{(1)} r_{i-k+1}^{(1)}. \tag{43}$$

Similarly, for the jth system, we have

$$\bar{r}_i^{(j)} = l_i^{(j)} r_i^{(j)} + l_{i-1}^{(j)} r_{i-1}^{(j)} + \cdots + l_{i-k+1}^{(j)} r_{i-k+1}^{(j)} \tag{44}$$

$$= l_i^{(j)} \beta_i^{(j)} r_i^{(1)} + l_{i-1}^{(j)} \beta_{i-1}^{(j)} r_{i-1}^{(1)} + \cdots + l_{i-k+1}^{(j)} \beta_{i-k+1}^{(j)} r_{i-k+1}^{(1)}. \tag{45}$$

Table 1
Information of the selected Web matrices.

Name	n	nnz	den	$annz_r$
web-Stanford	281,903	2,312,497	2.91×10^{-5}	8.20
cnr-2000	325,557	3,216,152	3.03×10^{-5}	9.88
web-NotreDame	325,729	1,497,134	1.41×10^{-5}	4.60
Stanford-Berkeley	683,446	7,583,376	1.62×10^{-5}	11.10
web-BerkStan	685,230	7,600,595	1.62×10^{-5}	11.09
eu-2005	862,664	19,235,140	2.58×10^{-5}	22.30
in-2004	1,382,908	16,917,053	8.84×10^{-6}	12.23
wiki-talk	2,394,385	5,021,410	8.76×10^{-7}	2.10
wikipedia-20060925	2,983,494	37,269,096	4.19×10^{-6}	12.49
wikipedia-20070206	3,566,907	45,030,389	3.54×10^{-6}	12.62
web-edu	9,845,725	57,156,537	5.90×10^{-7}	5.81

Clearly, the choices $l_i^{(j)} = \frac{l_i^{(1)}}{\beta_i^{(j)}}$ $(2 \le j \le s)$ ensure the collinearity condition $\bar{r}_i^{(j)} = \bar{r}_i^{(1)}$. However, such choice may result in $\sum_{m=0}^{k-1} l_{i-m}^{(j)} \neq 1$. Hence, we normalize the coefficients as $\bar{l}_i^{(j)} = \frac{l_i^{(j)}}{\sum_{m=0}^{k-1} l_{i-m}^{(j)}}$ $(2 \le j \le s)$, and we set $\bar{x}_i^{(j)} = \bar{l}_i^{(j)} x_i^{(j)} + \bar{l}_{i-1}^{(j)} x_{i-1}^{(j)} + \cdots + \bar{l}_{i-k+1}^{(j)} x_{i-k+1}^{(j)}$, leading to the residual vectors $\bar{r}_i^{(j)} = \frac{\bar{r}_i^{(1)}}{\sum_{m=0}^{k-1} l_{i-m}^{(j)}}$. If the residual norms of the additional systems are smaller than that of the seed system, that is $0 < \beta_i^{(j)} < 1$ $(2 \le j \le s)$, then we get $l_i^{(j)} = \frac{l_i^{(1)}}{\beta_i^{(j)}} > l_i^{(1)}$. Thus $\sum_{m=0}^{k-1} l_{i-m}^{(j)} > 1$, and $\bar{r}_i^{(j)} < \bar{r}_i^{(1)}$.

According to Eq. (23), the system with the largest value of α (suppose it to be α_1) will have the largest residual norm after running the shifted Power method, and thus it will be chosen as the seed system for running the shifted GMRES method, so there must be $0 < \beta_0^{(j)} < 1$, $2 \le j \le s$. Meanwhile, according to Theorem 4.1, the linear systems corresponding to smaller damping factors α_j will converge faster than the seed system. Therefore, we have $0 < \beta_i^{(j)} < 1$, $2 \le j \le s$ for any ith cycle of the shifted-GMRES procedure, and the convergence rate of the seed system determines the computational cost of implementing this method. As a result, we can conclude that: by applying the Anderson extrapolation to the seed system, and defining $\bar{x}_i^{(j)} = \bar{l}_i^{(j)} x_i^{(j)} + \bar{l}_{i-1}^{(j)} x_{i-1}^{(j)} + \cdots + \bar{l}_{i-k+1}^{(j)} x_{i-k+1}^{(j)}$ for the additional systems, the efficiency of the shifted Power-GMRES method is expected to be improved.

5. Numerical results

In this section, we test the performance of the proposed shifted Power and shifted Power-GMRES methods, with and without the extrapolation acceleration, for solving some selected PageRank problems available at the University of Florida matrix repository [48] and at the Laboratory for Web Algorithmics [49–51]. The characteristics of the Web graph matrices considered in our experiments are reported in Table 1, where for each matrix n indicates the dimension, nnz is the number of nonzero elements, $den = nnz/n^2$ represents the density, and $annz_r$ is the average number of nonzero elements per row. We set the personalization vector v equal to $v = [1, 1, \cdots, 1]^T/n$. All our experiments are run in MATLAB R2016a on a 64-bit Windows 10 computer equipped with a core i7-8550u processor and with 16GB of RAM memory.

In the PageRank formulation with multiple damping factors, the iterative solution of each ith $(i = 1, 2, \cdots, s)$ linear system is started from the initial guess $x_0{}^{(i)} = v$ and it is stopped when either the approximate solution $x_k{}^{(i)}$ satisfies

$$\frac{\|(1-\alpha_i)v - (I - \alpha_i \widetilde{P} x_k{}^{(i)})\|_2}{\|x_k{}^{(i)}\|_1} < 10^{-8},$$

or the number of matrix-vector products exceeds 10000.

Example 5.1

In this experiment, we test the performance of the **shifted Power** method against the conventional **Power** method for solving PageRank problems with multiple damping factors, namely $\{\alpha_1 = 0.85, \alpha_2 = 0.86, \cdots, \alpha_{15} = 0.99\}$, on the web-NotreDame, cnr-2000, web-BerkStan and eu-2005 graph matrices. For each method, in Table 2, we report on the numbers of matrix-vector products (denoted by the acronym *MV*), the CPU time in seconds (denoted by the acronym *CPU*) and the speedup factors of the **shifted-Power** method versus the **Power** method measured as

$$Speedup_{MV} = \frac{MV_{power} - MV_{shifted\ power}}{MV_{power}}$$

and

$$Speedup_{CPU} = \frac{CPU_{power} - CPU_{shifted\ power}}{CPU_{power}}.$$

Table 2
Comparison results between the **Power** method and the **shifted Power** method.

Methods:	Power		shifted Power		Power for latest		*Speedup_*	
Problems:	*MV*	*CPU*	*MV*	*CPU*	*MV*	*CPU*	*MV*	*CPU*
web-Stanford	3739	17.90	1141	8.54	1141	5.36	69.48%	52.29%
cnr-2000	3491	21.68	1058	10.35	1058	6.72	69.69%	52.26%
web-NotreDame	3148	14.12	910	7.03	910	4.01	71.09%	50.21%
Stanford-Berkeley	3571	49.96	1071	23.76	1071	14.91	70.01%	52.44%
web-BerkStan	3861	54.20	1216	26.56	1216	17.05	68.51%	51.00%
eu-2005	3089	88.32	892	34.97	892	25.11	71.12%	60.41%
in-2004	3596	111.05	1094	52.04	1094	33.59	69.58%	54.94%
wiki-talk	2286	69.97	688	42.18	688	21.20	69.90%	39.72%
wikipedia-20060925	3364	607.43	912	198.68	912	163.09	72.89%	67.29%
wikipedia-20070206	2274	522.74	640	175.58	640	145.52	71.86%	66.41%
web-edu	3159	492.33	942	259.99	942	143.24	70.18%	47.19%

Seen from Table 2, the number of matrix-vector products and the time cost required by the shifted Power method are remarkably smaller than those required by the ordinary method for solving each PageRank problem with multiple damping factors. The reduction in *MV* is around 70%, and the reduction in *CPU* is almost larger than 50%. Note that, this advantage of the shifted Power method over the ordinary one will be enlarged when computations with more damping factors and larger damping factors are needed. Besides, the number of matrix-vector products required by the shifted Power method for solving all the linear systems in each problem equals to that of using the ordinary Power method to solve the single system with largest α. This is consistent with the theoretical analysis. However, the time cost of the shifted Power method for all systems is larger than that of the Power method for the system with the largest α. This is due to the fact that the shifted Power method needs implementing additional vector operations such as vector-scalings, vector-addition operations and 2-norm computations of vectors.

Example 5.2

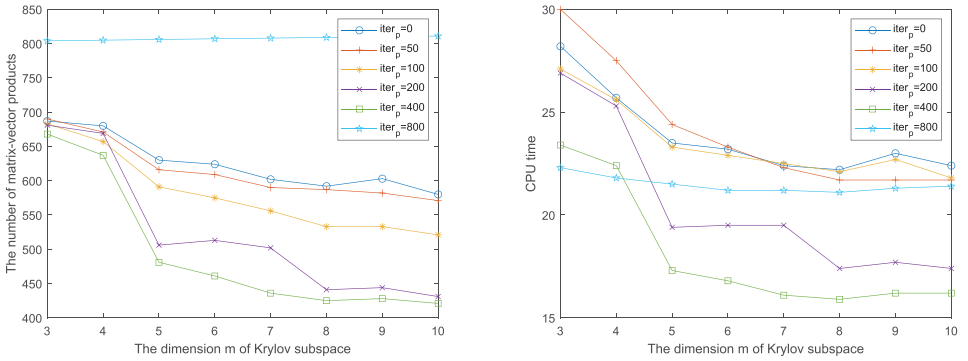
In this experiment, we compare the numerical behaviours of the **shifted Power-GMRES**, the **shifted Power** and the **shifted GMRES** methods, and we analyze the effect of the choice of the restart value m on the **shifted Power-GMRES** method for different numbers of **shifted Power** iterations $iter_p$. The damping factors are still set as $\{\alpha_1 = 0.85, \alpha_2 = 0.86, \cdots, \alpha_{15} = 0.99\}$. Because of the typically huge sizes of PageRank problems, m should be upper bounded by a small value due to memory constraints. Therefore, in our runs we vary m from 3 to 10. For $iter_p$, we test 5 values: $iter_p = 50, 100, 200, 400, 800$. In Fig. 1, we illustrate the convergence histories of the **shifted Power-GMRES** method for various combinations of parameters $(m, iter_p)$.

We see that, for a fixed choice of $iter_p$, the number of matrix-vector products and the CPU time generally decrease with m. For small restart values, such as $m = 3$ and $m = 4$, the number of matrix-vector products changes very slightly with $iter_p$ in the range 50,100,200,400. For $iter_p = 800$, only one cycle of **shifted GMRES** is needed because convergence is almost achieved after only the **shifted Power** iterations are implemented. Although the smoothing process that dumps the high frequency components of the error can be performed more effectively by the **shifted GMRES** method, especially when m is relatively large, it should be considered that one iteration of the **shifted GMRES** is more costly than one step of the **shifted Power** method due to the extra vector operations. Consequently, the overall CPU time can be reduced remarkably when $m \leq 4$ and $800 \geq iter_p \geq 400$. On the other hand, when m increases, the **shifted GMRES** algorithm becomes more efficient, and the setting $400 \geq iter_p \geq 200$ seems to be an overall good choice. By taking into account the memory costs, the experiments shown in Fig. 1 suggest the choice $m = 8, iter = 400$ as the best parameter-setting to reduce the CPU time of the **shifted Power-GMRES** method. Besides, we point out that in a few cases the **shifted Power-GMRES** and the **shifted GMRES** methods cost unusual amounts of time. Also, we observe cases where the **shifted GMRES** does not converge, while the **shifted Power-GMRES** converges. Overall, we can conclude that the **shifted Power-GMRES** method is more efficient and robust than the **shifted GMRES** method.

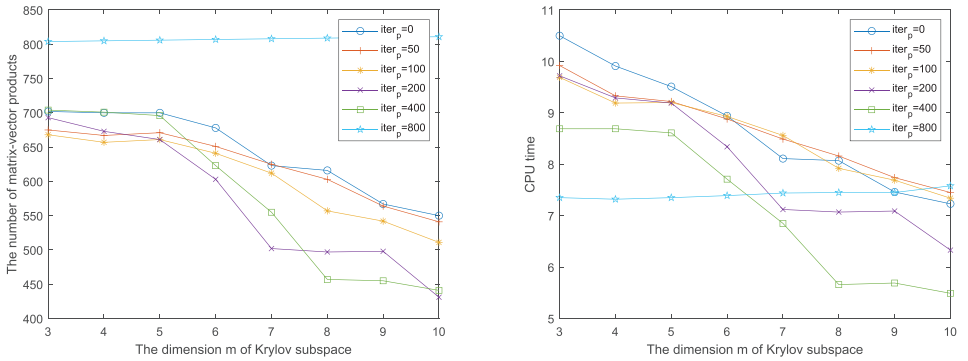
Example 5.3

In this experiment, we analyze the acceleration effect produced by the extrapolation technique on the **shifted Power-GMRES** method. For the parameter setting, we set the number of restart steps as $m = 8$, and we vary the number of iterations of the **shifted Power** method as $iter_p = [50, 100, 200, 400]$. Note that, when $iter_p = 800$, the extrapolation technique is not used since convergence is achieved after one GMRES cycle. Finally, we vary the window size (the number of previous iterative solutions being used) of the extrapolation technique in the range $l = [3, 4, 5]$.

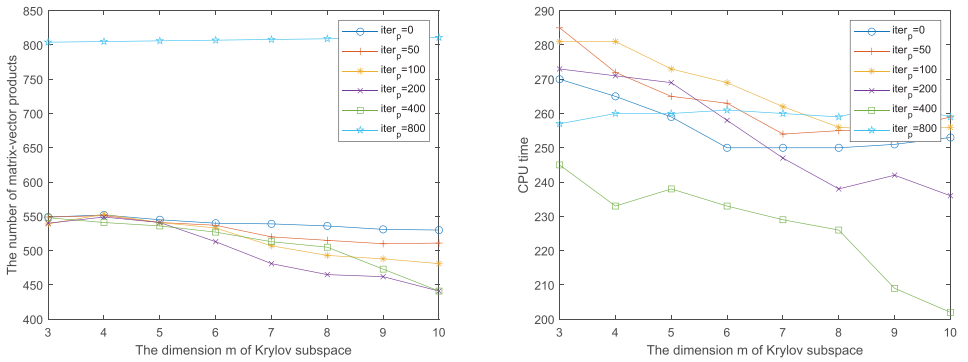
Figure 2 illustrates the number of matrix-vector products and the CPU time required by the **extrapolated shifted Power-GMRES** method for each selected pair of parameters $(iter_p, l)$.



The number of matrix-vector products (left) and the CPU time cost (right) of `web-BerkStan`



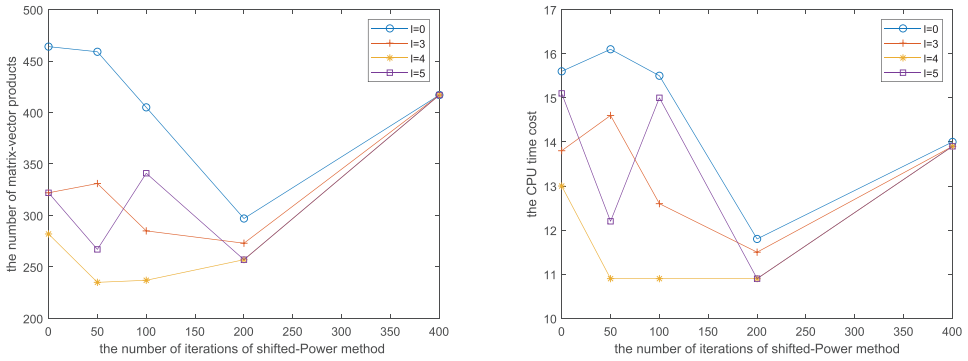
The number of matrix-vector products (left) and the CPU time cost (right) of `web-NotreDame`



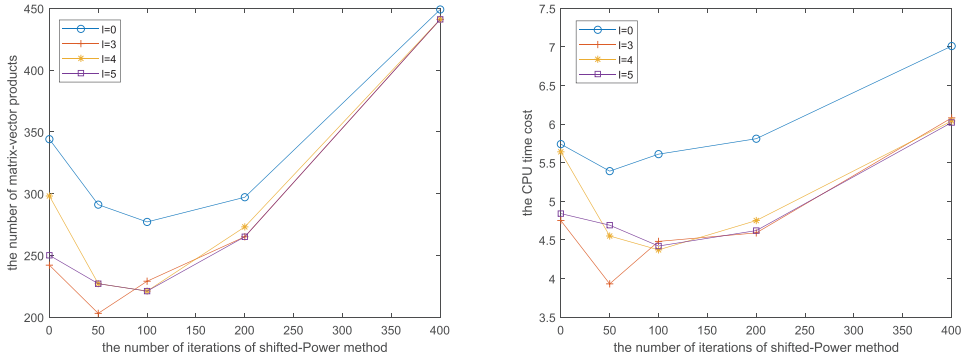
The number of matrix-vector products (left) and the CPU time cost (right) of `web-edu`

Fig. 1. Performances of **shifted-Power-GMRES** versus various couples of parameter settings $(m, iter_p)$.

We observe that in our experiments the extrapolation technique is effective for reducing both the number of matrix-vector products and the CPU time of the **shifted-Power-GMRES** method, to a different extent depending on the choice of parameters $(iter_p, l)$. For smaller values of $iter_p$, the number of cycles required by **shifted GMRES** increases, and clearly the acceleration (reduction in *MV* and *CPU*) due to the extrapolation is more evident. The acceleration effect of the extrapolation depends upon the selected window size l, but the dependence is not clear. For example, on the `Stanford-Berkeley` matrix, the acceleration due to the extrapolation strategy is more remarkable for $l = 5$ than for $l = 3$ when $iter_p = 50$, but the opposite result is observed when $iter_p = 100$. Besides, acceleration effect is more remarkable with $l = 4$ than with $l = 3$ for the `Stanford-Berkeley` matrix, while the opposite behaviour is observed for the `cnr-2000` matrix. From our experiments, we can conclude that the extrapolation technique is computationally more attractive to use when $iter_p$ is not so large. Although the best choice of l seems to be problem-dependent and its optimal setting requires further investigation, Fig. 2 clearly demonstrates the potential of this technique on accelerating the convergence of the **shifted Power-GMRES** method for realistic PageRank computations.



The number of matrix-vector products (left) and the CPU time cost (right) of `Stanford-Berkeley`



The number of matrix-vector products (left) and the CPU time cost (right) of `cnr-2000`

Fig. 2. Performances of **shifted-Power-GMRES** versus various couples of parameter settings $(iter_p, l)$.

Table 3
Comparison results between with & without the extrapolation technique.

method:	shifted-Power-GMRES							
cases:	shortest CPU time				largest speedup factor			
technique:	original		extrapolation		original		extrapolation	
Problems:	*MV*	*CPU*	*MV*	*CPU*	*MV*	*CPU*	*MV*	*CPU*
`web-Stanford`	257	3.79	249	3.74	417	5.06	417	4.84
`cnr-2000`	297	5.39	203	3.93	297	5.39	203	3.03
`web-NotreDame`	457	5.66	277	5.04	557	7.92	277	5.04
`Stanford-Berkeley`	297	11.8	235	10.9	459	16.1	235	10.9
`web-BerkStan`	425	15.9	425	15.2	441	17.4	353	15.7
`eu-2005`	163	10.9	131	9.52	157	12.3	141	10.4
`in-2004`	395	32.9	235	23.3	395	32.9	235	23.3
`wiki-talk`	72	7.87	50	7.53	72	7.87	50	7.53
`wikipedia-20060925`	75	27.7	67	25.5	224	62.3	122	39.6
`wikipedia-20070206`	125	53.6	91	40.0	171	63.5	91	40.0

Finally, in Table 3 we present comparative results between the **shifted Power-GMRES** method with and without using the extrapolation strategy. For each test problem, we present comparative figures of results in two scenarios: 1) the fastest run (that is, the shortest CPU time); 2) the largest speedup factor produced by the extrapolation technique.

We can see that the fastest run of **shifted Power-GMRES** is always further accelerated by using the extrapolation technique, with a time cost reduction factor up to 29.18% (appears for the `in-2004` problem). The reduction in terms of *MV* is generally more significant than the *CPU* time saving. For example, on the same problem `in-2004`, the *MV* number is reduced by 40.51%. This difference can be explained by the additional cost due to the extra vector operations and QR factorizations performed by the extrapolation. Considered that our implementation is not totally optimized, we believe that larger reductions in terms of CPU time can be produced by the extrapolation technique. Besides, in the best case scenario (experiments with largest CPU time reductions **largest speedup factor**), more remarkable reductions brought by the extrapolation in terms of both *MV* and *CPU* are observed. Overall, these results show the good potential of the proposed extrapolation technique for accelerating **shifted Power-GMRES** for solving this problem class.

6. Conclusions

In this paper, we propose a variant of the Power method, called the shifted Power method, that can solve the shifted linear systems of PageRank problems with multiple damping factors simultaneously, and costs much smaller time than the ordinary Power method applied to this problem directly. Different from other stationary iterative methods, the proposed algorithm generates collinear residual vectors. For the solution of difficult PageRank problems with damping factors very close to 1, this method can be efficiently combined with powerful non-stationary methods such as GMRES. By choosing a suitable seed system, the resulting shifted Power-GMRES method has almost the same complexity of the Power-GMRES method [27] applied to the seed system alone. Finally, we have derived an Anderson-type extrapolation technique to further accelerate the proposed shifted Power-GMRES iterations. Numerical experiments demonstrate the potential of the proposed iterative solver and the acceleration techniques to accelerate realistic PageRank computations with multiple damping factors effectively.

Acknowledgements

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.

References

[1] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, 1998.
[2] B. Liu, S. Jiang, Q. Zou, HITS-PR-HHblits: protein remote homology detection by combining PageRank and hyperlink-induced topic search, Brief. Bioinf. 21 (1) (2020) 298–308.
[3] F.A. Massucci, D. Docampo, Measuring the academic reputation through citation networks via pagerank, J. Inf. 13 (1) (2019) 185–201.
[4] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web, Technical report, Stanford InfoLab, 1999.
[5] M. Rafiei, A.A. Kardan, A novel method for expert finding in online communities based on concept map and PageRank, Hum.-Centric Comput. Inf. Sci. 5 (1) (2015) 10.
[6] M. Zhang, X. Li, L. Zhang, S. Khurshid, Boosting spectrum-based fault localization using PageRank, in: Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2017, pp. 261–272.
[7] T. Zhou, E. Martinez-Baez, G. Schenter, A.E. Clark, PageRank as a collective variable to study complex chemical transformations and their energy landscapes, J. Chem. Phys. 150 (13) (2019) 134102.
[8] D.F. Gleich, Pagerank beyond the web, SIAM Rev. 57 (2015) 321–363.
[9] P.G. Constantine, D.F. Gleich, Random alpha PageRank, Internet Math. 6 (2009) 189–236.
[10] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, G.H. Golub, Extrapolation methods for accelerating PageRank computation, in: Proc. of the 12th International World Wide Web Conference, 2003, pp. 261–270.
[11] X. Tan, A new extrapolation method for PageRank computations, J. Comput. Appl. Math. 313 (2017) 383–392.
[12] S.D. Kamvar, T.H. Haveliwala, G.H. Golub, Adaptive methods for the computation of the PageRank, Linear Algebra Appl. 386 (2004) 51–65.
[13] H.D. Sterck, T.A. Manteuffel, S.F. McCormick, Q. Nguyen, J. Ruge, Multilevel adaptive aggregation for Markov chains, with application to web ranking, SIAM J. Sci. Comput. 30 (2008) 2235–2262.
[14] Z.-L. Shen, T.-Z. Huang, B. Carpentieri, C. Wen, X.M. Gu, Block-accelerated aggregation multigrid for Markov chains with application to PageRank problems, Commun. Nonlinear. Sci. 59 (2018) 472–487.
[15] D.F. Gleich, A.P. Gray, C. Greif, T. Lau, An inner-outer iteration for computing PageRank, SIAM J. Sci. Comput. 32 (2010) 349–371.
[16] C.-Q. Gu, F. Xie, K. Zhang, A two-step matrix splitting iteration for computing PageRank, J. Comput. Appl. Math. 278 (2015) 19–28.
[17] C. Wen, T.-Z. Huang, Z.L. Shen, A note on the two-step matrix splitting iteration for computing PageRank, J. Comput. Appl. Math. 315 (2017) 87–97.
[18] Z.-L. Tian, Y. Liu, Y. Zhang, Z.-Y. Liu, M.Y. Tian, The general inner-outer iteration method based on regular splittings for the PageRank problem, Appl. Math. Comput. 356 (2019) 479–501.
[19] M.-Y. Tian, Y. Zhang, Y.D. Wang, A general multi-splitting iteration method for computing PageRank, Comput. Appl. Math. 38 (2019) 1–29.
[20] G.H. Golub, C. Greif, An Arnoldi-type algorithm for computing PageRank, BIT 46 (2006) 759–771.
[21] J.-F. Yin, G.-J. Yin, M. Ng, On adaptively accelerated Arnoldi method for computing PageRank, Numer. Linear Algebra Appl. 19 (2012) 73–85.
[22] H.-F. Zhang, T.-Z. Huang, C. Wen, Z.L. Shen, FOM accelerated by an extrapolation method for solving PageRank problems, J. Comput. Appl. Math. 296 (2016) 397–409.
[23] G. Wu, Y.M. We, Arnoldi versus GMRES for computing PageRank: a theoretical contribution to google's PageRank problem, ACM Trans. Inf. Syst. 28 (2010) 1–28.
[24] B.-Y. Pu, T.-Z. Huang, C. Wen, A preconditioned and extrapolation-accelerated GMRES method for PageRank, Appl. Math. Lett. 37 (2014) 95–100.
[25] D.F. Gleich, L. Zhukov, P. Berkhin, Fast Parallel PageRank: A Linear System Approach, Yahoo! 489 Tech. Rep., 2005.
[26] G. Wu, Y. Wei, A power-Arnoldi algorithm for computing PageRank, Numer. Linear Algebra Appl. 14 (2007) 521–546.
[27] C.-Q. Gu, X.-L. Jiang, C.-C. Shao, Z.B. Chen, A GMRES-power algorithm for computing PageRank problems, J. Comput. Appl. Math. 343 (2018) 113–123.
[28] Q.-Y. Hu, C. Wen, T.-Z. Huang, Z.-L. Shen, X.M. Gu, A variant of the power-Arnoldi algorithm for computing PageRank, J. Comput. Appl. Math. 381 (2021) 113034.
[29] C.-Q. Gu, W.W. Wang, An Arnoldi-inout algorithm for computing PageRank problems, J. Comput. Appl. Math. 309 (2017) 219–229.
[30] A.N. Langville, C.D. Meyer, Deeper inside PageRank, Internet Math. 1 (2004) 335–380.
[31] T. Haveliwala, S. Kamvar, The Second Eigenvalue of the Google Matrix, Stanford University Technical Report, 2003.
[33] G.M.D. Corso, A. Gulli, F. Romani, Fast PageRank computation via a sparse linear system, Internet Math. 2 (2005) 251–273.
[34] D.M. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.
[35] C. Greif, D. Kurokawa, A note on the convergence of SOR for the PageRank problem, SIAM J. Sci. Comput. 33 (2011) 3201–3209.
[36] Y.-J. Xie, C.F. Ma, A relaxed two-step splitting iteration method for computing PageRank, Comput. Appl. Math. 37 (2018) 221–233.
[37] G.M.D. Corso, A. Gulli, F. Romani, Comparison of Krylov subspace methods on the PageRank problem, J. Comput. Appl. Math. 210 (2007) 159–166.
[38] G. Wu, Y. Wei, An Arnoldi-extrapolation algorithm for computing PageRank, J. Comput. Appl. Math. 234 (2010) 3196–3212.
[39] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, G.H. Goloub, Exploiting the Block Structure of the Web for Computing PageRank, Technical Report, Stanford University, 2003.
[40] Q. Yu, Z. Miao, G. Wu, Y.M. Wei, Lumping algorithms for computing google's PageRank and its derivative, with attention to unreferenced nodes, Inf. Retr. 15 (2012) 503–526.

[41] Z.-L. Shen, T.-Z. Huang, B. Carpentieri, X.-M. Gu, C. Wen, An efficient elimination strategy for solving PageRank problems, Appl. Math. Comput. 298 (2017) 111–122.
[42] Z.-L. Shen, T.-Z. Huang, B. Carpentieri, C. Wen, X.-M. Gu, X.Y. Tan, Off-diagonal low-rank preconditioner for difficult PageRank problems, J. Comput. Appl. Math. 346 (2019) 456–470.
[43] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, 2003.
[44] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[45] A. Frommer, U. Glassner, Restarted GMRES for shifted linear systems, SIAM J. Sci. Comput. 19 (1998) 15–26.
[46] H.A. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 13 (1992) 631–644.
[47] F.H. Walker, P. Ni, Anderson acceleration for fixed-point iterations, SIAM J. Numer. Anal. 49 (2011) 1715–1735.
[48] T.A. Davis, Y. Hu, The university of florida sparse matrix collection, ACM Trans. Math. Softw. 38 (1) (2011) 1:1–1:25. Available at the URL: http://www.cise.ufl.edu/research/sparse/matrices
[49] P. Boldi, S. Vigna, The WebGraph framework I: compression techniques, in: Proc. of the 13th international conference on World Wide Web, 2004, pp. 595–602.
[50] P. Boldi, M. Rosa, M. Santini, S. Vigna, Layered label propagation: a multiresolution coordinate-free ordering for compressing social networks, in: Proc. of the 20th International Conference on World Wide Web, 2011, pp. 587–596.
[51] P. Boldi, B. Codenotti, M. Santini, S. Vigna, UbiCrawler: a scalable fully distributed web crawler, Softw. Pract. Experience 34 (2004) 711–726.