Research Article

Bing-Yuan Pu*, Chun Wen, and Qian-Ying Hu

A multi-power and multi-splitting inner-outer iteration for PageRank computation

https://doi.org/10.1515/math-2020-0120
received June 4, 2020; accepted November 24, 2020

Abstract: As an effective and possible method for computing PageRank problem, the inner-outer (IO) iteration has attracted wide interest in the past few years since it was first proposed by Gleich et al. (2010). In this paper, we present a variant of the IO iteration, which is based on multi-step power and multi-step splitting and is denoted by MPMIO. The description and convergence are discussed in detail. Numerical examples are given to illustrate the effectiveness of the proposed method.

Keywords: PageRank, power method, inner-outer iteration, multi-splitting, convergence

MSC 2020: 65C40, 65F10

1 Introduction

With the fast development of the Internet, web search engines have become one of the most important Internet tools for information retrieval. How to list the relevant web pages plays a significant role in this filed. Among thousands of search engines, Google is one of the most popular and successful. And this is mainly attributed to its effective algorithm, PageRank.

PageRank, a link-based algorithm formulated by Page et al. [1], gives a rank list of importance of pages related to user's query terms. The hyperlink structure of the web can be viewed as a direct graph and modeled by a Markov chain:

$$Ax = [\alpha P + (1 - \alpha)ve^T]x = x, \tag{1}$$

where $\alpha \in (0, 1)$ is the damping factor, $P \in \mathbb{R}^{n\times n}$ is a column stochastic matrix, e is a column vector of all ones, v is called the personalization or the teleportation vector and set as $v = e/n$. In particular, A is defined as the Google matrix and is both irreducible and aperiodic, which implies that there exists a unique right non-negative eigenvector x for (1). For more details about the PageRank algorithm, we refer the reader to [1–4].

Given the huge size and density of the Google matrices, computing PageRank is faced with the big challenge of computational resources, and only a small set of computational tools can come in handy. The power method was first considered to compute the PageRank for its stable and reliable performances. However, when the largest eigenvalue of matrix A is not separated well from the second one, the power method costs more and works less well. Some accelerated techniques have been proposed to speed up its convergence, such as vector extrapolation [5–8], Arnoldi-type [9,10], aggregation/disaggregation [11], lumping [12,13], adaptive methods [4,14] and inner-outer (IO) iteration methods [15–20].

Recently, researchers have focused on the PageRank's linear system and the corresponding iterative methods have raised concerns [3,15–17,20]. Gleich et al. [15] proposed an IO iteration method, which is

* **Corresponding author: Bing-Yuan Pu,** Department of Basic Science, Chengdu Textile College, Chengdu, 611731, China, e-mail: skypuby@163.com
Chun Wen, Qian-Ying Hu: School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, 611731, China

implemented by solving a linear system with a lower damping factor and similar algebraic structure to the original one, and Gu et al. [17] put forward an improved algorithm, i.e., the power-inner-outer (PIO) iteration. In this paper, we proposed a variant of the IO(PIO) iteration by applying multi-step power and multi-step splitting to combine with the IO iteration to accelerate the computation of PageRank.

The remainder of this paper is structured as follows. In Section 2, we briefly provide the mechanism of the IO iteration for PageRank problem. In Section 3, we introduce the proposed algorithm and investigate the convergence properties in detail. Numerical examples are given to illustrate the effectiveness of the method in Section 4. Finally, some conclusions are drawn in Section 5.

2 The IO(PIO) iteration

In this section, we begin by briefly introducing the derivation of the IO iteration by Gleich et al. [15]. Recalling $e^T x = 1$, the eigenvalue problem $Ax = x$ can be rewritten as

$$(I - \alpha P)x = (1 - \alpha)v. \tag{2}$$

As you know, there is a general agreement that the smaller the damping factor is the easier it is to solve the original PageRank problem. In light of this, Gleich et al. [15] reformulated (2) as

$$(I - \beta P)x = (\alpha - \beta)Px + (1 - \alpha)v, \quad 0 < \beta < \alpha. \tag{3}$$

Then they have the following outer stationary iteration:

$$(I - \beta P)x_{k+1} = (\alpha - \beta)Px_k + (1 - \alpha)v, \quad k = 1, 2, \ldots, \tag{4}$$

where $x_0 = v$ is the initial guess, though other choices are possible too. However, solving this linear system with coefficient matrix $I - \beta P$ is still computationally difficult, even though β is small. Therefore, the inner Richardson iteration is used to compute the approximation of x_{k+1}.

Setting the right-hand side of (4) as

$$f = (\alpha - \beta)Px_k + (1 - \alpha)v, \tag{5}$$

the inner linear system is defined as

$$(I - \beta P)y = f, \tag{6}$$

which is computed by the inner iteration

$$y_{j+1} = \beta P y_j + f, \quad j = 0, 1, 2, \ldots, l - 1, \tag{7}$$

where $y_0 = x_k$ is the initial guess and the computed approximate solution y_l is assigned to new x_{k+1}. The stopping criteria of the outer iteration and the inner iteration are, respectively, given as

$$\|(1 - \alpha)v - (I - \alpha P)x_{k+1}\|_1 < \tau$$

and

$$\|f - (I - \beta P)y_{j+1}\|_1 < \eta,$$

where the parameters τ and η are the outer and inner tolerances, respectively.

Based on the above discussion, a basic IO iterative algorithm for PageRank problem has been proposed in [15] and its convergence has also been analyzed therein.

From (2) and by the power method, Gu et al. [17] obtained the Richardson iteration

$$x_{k+1} = \alpha P x_k + (1 - \alpha)v, \quad k = 0, 1, 2, \ldots, \tag{8}$$

and then they proposed the PIO iteration as follows:

$$\begin{cases} x_{k+\frac{1}{2}} = \alpha P x_k + (1 - \alpha)v, \\ (I - \beta P)x_{k+1} = (\alpha - \beta)P x_{k+\frac{1}{2}} + (1 - \alpha)v, \end{cases} \quad k = 0, 1, 2, \ldots, \tag{9}$$

where $x_0 = v$ is the initial guess and $\beta \in (0, \alpha)$. The first iteration of (9) is easy to implement. In the second iteration of (9), Gu et al. come up with the IO iteration to get the next approximation x_{k+1}. They then proved the superiority of the PIO over the power method and the IO iteration and some convergence properties of the PIO method have been given as follows.

Theorem 2.1. [17] *The iteration matrix $T(\alpha, \beta)$ of the PIO iteration (9) is given by*

$$T(\alpha, \beta) = \alpha(\alpha - \beta)(I - \beta P)^{-1}P^2, \tag{10}$$

and the modulus of its eigenvalues is bounded by

$$\sigma = \frac{\alpha(\alpha - \beta)}{(1 - \beta)}, \quad \alpha \in (0, 1), 0 < \beta < \alpha. \tag{11}$$

Therefore, the spectral radius satisfies $\rho(T(\alpha, \beta)) \leq \sigma < 1$, i.e., the PIO iteration converges to the unique solution x^* of (2) for any initial vector x_0.

Theorem 2.2. [17] *Suppose the second iteration of the PIO iteration is exact and $0 < \alpha < 1$. Then the PIO iteration converges for any $\beta \in (0, \alpha)$. And it has*

$$\|x_{k+1} - x\|_1 \leq \frac{\alpha(\alpha - \beta)}{(1 - \beta)}\|x_k - x\|_1 \tag{12}$$

and

$$\|x_{k+1} - x_k\|_1 \leq \frac{\alpha(\alpha - \beta)}{(1 - \beta)}\|x_k - x_{k-1}\|_1. \tag{13}$$

3 The multi-power and multi-splitting IO iteration for PageRank

3.1 The MPMIO iteration

It is not hard to find that the first step of PIO iteration (9) is the classical power method. In this way, the PIO iteration can be understood as the combination of one-step power method and IO iteration. It inspires us to expand the IO(PIO) iteration to more efficient methods. First, we consider the multi-step power method with the IO iteration. Then on the second iteration of (9), we consider the multi-step splitting iteration. Hence, the corresponding variant of the IO(PIO) algorithm, MPMIO, can be proposed. Section 4 is about the details, and the convergence properties are described in Section 3.2.

Given an initial guess x_0, we first expand the first step of (9) to the multi-step power as follows:

$$\begin{cases} x_{k+\frac{1}{m+1}} = \alpha P x_k + (1 - \alpha)v, \\ x_{k+\frac{2}{m+1}} = \alpha P x_{k+\frac{1}{m+1}} + (1 - \alpha)v, \\ \vdots \\ x_{k+\frac{m}{m+1}} = \alpha P x_{k+\frac{m-1}{m+1}} + (1 - \alpha)v, \end{cases} \tag{14}$$

where $\alpha \in (0, 1)$, $\beta \in (0, \alpha)$, $v = e/n$ and m (≥ 2) is the step of the power method.

Now, by introducing two parameters β_1 and β_2, $\beta_{1,2} \in (0, \alpha)$ we expand the second step iteration of (9) to the multi-splitting as follows:

$$\begin{cases} (I - \beta_1 P)\omega_{k+1} = (\alpha - \beta_1)P x_{k+\frac{m}{m+1}} + (1 - \alpha)v, \\ (I - \beta_2 P)x_{k+1} = (\alpha - \beta_2)P\omega_{k+1} + (1 - \alpha)v. \end{cases} \tag{15}$$

Taken together, the proposed multi-power and multi-splitting IO iteration can be shown below.

Multi-power and multi-splitting IO iteration. Given an initial guess x_0. Compute

$$\begin{cases} x_{k+\frac{1}{m+1}} = \alpha P x_k + (1-\alpha)v, \\ x_{k+\frac{2}{m+1}} = \alpha P x_{k+\frac{1}{m+1}} + (1-\alpha)v, \\ \vdots \\ x_{k+\frac{1}{m+1}} = \alpha P x_{k+\frac{m-1}{m+1}} + (1-\alpha)v, \\ (I-\beta_1 P)\omega_{k+1} = (\alpha-\beta_1)P x_{k+\frac{m}{m+1}} + (1-\alpha)v, \\ (I-\beta_2 P)x_{k+1} = (\alpha-\beta_2)P\omega_{k+1} + (1-\alpha)v \end{cases} \tag{16}$$

until the sequence $\{x_k\}_{k=0}^{\infty}$ converges.

The first m-step power iterations of (16) can be implemented easily and the IO iteration can be used for the second stage. Setting the right-hand side of the splitting iteration of (16) as

$$f_1 = (\alpha-\beta_1)P x_{k+\frac{m}{m+1}} + (1-\alpha)v, \tag{17}$$

we get the first inner iteration

$$y'_{j+1} = \beta_1 y_j + f_1, \quad j = 0, 1, 2, \ldots, l-1. \tag{18}$$

Then we assign $(\alpha-\beta_2)Py'_{j+1} + (1-\alpha)v$ to f_2 and get the second inner iteration

$$y_{j+1} = \beta_2 y'_{j+1} + f_2, \quad j = 0, 1, 2, \ldots, l-1, \tag{19}$$

where we take $y_0 = x_{k+\frac{m}{m+1}}$ as the initial guess and assign the computed approximation y_l to the new x_{k+1}.

Now we switch to the first m-step power method and repeat the procedure until the desired PageRank vector is obtained. For the whole iteration, we stop the outer iteration if

$$\|(1-\alpha)v - (I-\alpha P)x_{k+1}\|_1 < \tau$$

is satisfied, and use

$$\|f_2 - (I-\beta_2 P)y_{j+1}\|_1 < \eta$$

for the inner iteration stopping criterion. Now the main algorithm of this paper is shown in Algorithm 1.

Algorithm 1. (MPMIO)

Input: P, α, β_1, β_2, τ, η, v, m
Output: x
1. $x \leftarrow v$;
2. $y \leftarrow Px$;
3. while $\|\alpha y + (1-\alpha)v - x\|_1 \geq \tau$
4. for $i = 1 : m$, do
5. $x \leftarrow \alpha y + (1-\alpha)v$;
6. $y = Px$;
7. end for
8. $f_1 \leftarrow (\alpha-\beta_1)y + (1-\alpha)v$;
9. $f' \leftarrow f_1 + \beta_1 y$;
10. $f_2 \leftarrow (\alpha-\beta_2)Pf' + (1-\alpha)v$;
11. repeat
12. $x = f_2 + \beta_2 y$;
13. $y \leftarrow Px$;
14. until $\|f_2 + \beta_2 y - x\|_1 < \eta$;
15. end while
16. $x \leftarrow \alpha y + (1-\alpha)v$.

Lines 1 and 2 of Algorithm 1 initialize $x = v$ and $y = Px$. The m-step iterations of (16) are done in lines 4–7. Lines 8–10 are used for the computation of f_2. The inner iteration is implemented in lines 11–14. We use the repeat-until clause to ensure that at least one inner iteration is performed. To terminate the algorithm, line 3 checks the residual of the outer linear (2). And in line 14, the stopping criterion is examined for the inner iteration. At the end of Algorithm 1, a single power method step is used for the possible benefits as given in [21].

3.2 Convergence analysis of the MPMIO iteration

In this subsection, we devote to the convergence properties of our new algorithm and pay particular attention to its superiority over the power method, IO iteration and PIO iteration, respectively.

Return to (16) and substitute the first iteration $x_{k+\frac{1}{m+1}}$ into the second iteration $x_{k+\frac{2}{m+1}}$, we have

$$x_{k+\frac{2}{m+1}} = (\alpha P)^2 x_k + (1-\alpha)v \sum_{i=0}^{1} (\alpha P)^i. \tag{20}$$

Keep going and until the last step of the power method, i.e.,

$$x_{k+\frac{m}{m+1}} = (\alpha P)^m x_k + (1-\alpha)v \sum_{i=0}^{m-1} (\alpha P)^i. \tag{21}$$

At the same time, by introducing two parameters β_1 and β_2 and from the last two steps of (16), we get

$$x_{k+1} = (\alpha-\beta_1)(\alpha-\beta_2)(I-\beta_1 P)^{-1}(I-\beta_2 P)^{-1}P^2 x_{k+\frac{m}{m+1}} + (1-\alpha)v(I-\beta_2 P)^{-1}. \tag{22}$$

Substituting $x_{k+\frac{m}{m+1}}$ into the above formula, we have

$$\begin{aligned} x_{k+1} = {} & \alpha^m(\alpha-\beta_1)(\alpha-\beta_2)(I-\beta_1 P)^{-1}(I-\beta_2 P)^{-1}P^{m+2}x_k \\ & + (1-\alpha)(I-\beta_2 P)^{-1}v\left[I + (\alpha-\beta_1)(\alpha-\beta_2)(I-\beta_1 P)^{-1}P^2 \sum_{i=0}^{m-1}(\alpha P)^i\right]. \end{aligned} \tag{23}$$

Hence, the iteration matrix of (16) is

$$T_m(\alpha,\beta_1,\beta_2) = \alpha^m(\alpha-\beta_1)(\alpha-\beta_2)(I-\beta_1 P)^{-1}(I-\beta_2 P)^{-1}P^{m+2}. \tag{24}$$

Thus, we obtain the following theoretical results for convergence property of the MPMIO iteration (16).

Theorem 3.1. *The iteration matrix $T_m(\alpha,\beta_1,\beta_2)$ of the MPMIO iteration* (16) *is given as* (24) *and the modulus of its eigenvalues is bounded by*

$$\sigma_m(\beta_1,\beta_2) = \frac{\alpha^m(\alpha-\beta_1)(\alpha-\beta_2)}{(1-\beta_1)(1-\beta_2)}, \quad \alpha \in (0,1), \quad 0 < \beta_{1,2} < \alpha. \tag{25}$$

Meanwhile, the spectral radius satisfies

$$\rho(T_m(\alpha,\beta_1,\beta_2)) \le \sigma_m(\beta_1,\beta_2) < 1. \tag{26}$$

In other words, the MPMIO iteration converges to the unique solution x^* of the linear system (2) *for any initial vector x_0.*

Proof. The first part of Theorem 3.1 has been proved. Suppose λ_i is an eigenvalue of P, then

$$\mu_i = \frac{\alpha^m(\alpha-\beta_1)(\alpha-\beta_2)\lambda_i^{m+2}}{(1-\beta_1\lambda_i)(1-\beta_2\lambda_i)}$$

is an eigenvalue of the iteration matrix $T_m(\alpha,\beta_1,\beta_2)$. Since $|\lambda_i| \le 1$, we have

$$|\mu_i| = \left| \frac{\alpha^m(\alpha-\beta_1)(\alpha-\beta_2)\lambda_i^{m+2}}{(1-\beta_1\lambda_i)(1-\beta_2\lambda_i)} \right| \le \frac{\alpha^m(\alpha-\beta_1)(\alpha-\beta_2)|\lambda_i|^{m+2}}{(1-\beta_1|\lambda_i|)(1-\beta_2|\lambda_i|)} \le \frac{\alpha^m(\alpha-\beta_1)(\alpha-\beta_2)}{(1-\beta_1)(1-\beta_2)} = \sigma_m(\beta_1,\beta_2), \tag{27}$$

with equality holding for $\lambda_1 = 1$. So (25) is proved and the inequality (26) follows directly from (27). □

Remark 3.1. Back to the right-hand side of (25), and given that $\alpha \in (0,1)$, $\beta_{1,2} \in (0,\alpha)$, we get

$$\sigma_m(\beta_1,\beta_2) \le \min_i \left\{ \frac{\alpha^m(\alpha-\beta_i)}{1-\beta_i} \right\} = \min\{\sigma_m(\beta_1), \sigma_m(\beta_2)\},$$

where $\sigma_m(\beta_1)$ and $\sigma_m(\beta_2)$ are the eigenvalue's upper boundary of the iteration matrix from single-parameter β IO iteration. In this way, it may show the superiority of the multisplitting IO iteration.

Theorem 3.2. *Suppose the last splitting iteration of the MPMIO is solved exactly and $\alpha \in (0,1)$. Then for any $\beta_{1,2} \in (0,\alpha)$, the MPMIO iteration converges. Furthermore,*

$$\|x_{k+1} - x\|_1 \le \frac{\alpha^m(\alpha-\beta_1)(\alpha-\beta_2)}{(1-\beta_1)(1-\beta_2)}\|x_k - x\|_1 \tag{28}$$

and

$$\|x_{k+1} - x_k\|_1 \le \frac{\alpha^m(\alpha-\beta_1)(\alpha-\beta_2)}{(1-\beta_1)(1-\beta_2)}\|x_k - x_{k-1}\|_1. \tag{29}$$

Proof. By definition, we have

$$\begin{aligned} x = {} & \alpha^m(\alpha-\beta_1)(\alpha-\beta_2)(I-\beta_1P)^{-1}(I-\beta_2P)^{-1}P^{m+2}x \\ & + (1-\alpha)(I-\beta_2P)^{-1}v\left[I + (\alpha-\beta_1)(\alpha-\beta_2)(I-\beta_1P)^{-1}P^2\sum_{i=0}^{m-1}(\alpha P)^i \right]. \end{aligned} \tag{30}$$

Subtracting (30) from (23), we have

$$x_{k+1} - x = T_m(\alpha,\beta_1,\beta_2)(x_k - x) = \alpha^m(\alpha-\beta_1)(\alpha-\beta_2)(I-\beta_1P)^{-1}(I-\beta_2P)^{-1}P^{m+2}(x_k - x), \tag{31}$$

where $T_m(\alpha,\beta_1,\beta_2)$ is the iteration matrix as (24), and x_k, x_{k+1} refer to the k-step or $k+1$-step iteration, respectively.

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

$$\|x_{k+1} - x\|_1 \le \alpha^m(\alpha-\beta_1)(\alpha-\beta_2)\|(I-\beta_1P)^{-1}\|_1\|(I-\beta_2P)^{-1}\|_1\|P\|_1^{m+2}\|x_k - x\|_1. \tag{32}$$

Since the matrix P is column stochastic, it holds that $\|P\|_1 = 1$. Meanwhile, since $I-\beta_1P$ and $I-\beta_2P$ are diagonally dominant M-matrix, their inverses are nonnegative and it follows that

$$\|(I-\beta_1P)^{-1}\|_1 = \frac{1}{1-\beta_1} \tag{33}$$

and

$$\|(I-\beta_2P)^{-1}\|_1 = \frac{1}{1-\beta_2}. \tag{34}$$

The inequality of (28) follows from (32), (33) and (34).

Setting (23) for step k, subtracting it from (23) and taking norms in the same way like (32), we can easily derive (29). □

Remark 3.2. Denote the contraction factor by

$$g_\alpha(\beta_1, \beta_2) = \frac{\alpha^m(\alpha - \beta_1)(\alpha - \beta_2)}{(1 - \beta_1)(1 - \beta_2)},$$

it is easy to see that

$$g_\alpha(\beta_1, \beta_2) \le \min_i g_\alpha(\beta_i) = \min\{g_\alpha(\beta_1), g_\alpha(\beta_2)\},$$

where $g_\alpha(\beta_i)$, $i = 1, 2$, is the contraction factor of single-parameter IO iteration.

Thus, we can deduce that the MPMIO iteration may converge faster than that of the single parameter β_1 or β_2.

Remark 3.3. It is easy to prove that $g_\alpha(\beta_1, \beta_2)$ is a monotonically decreasing function related to m and thus show the superiority of our MPMIO iteration over the single-power IO iteration, like PIO.

4 Numerical experiments

In this section, we carry out some numerical examples to test the effectiveness of the MPMIO iteration and compare it with the power method, the IO iteration and the PIO iteration. All the numerical results are obtained by using MATLAB9.7.0(R2019b) on a PC with 3.97 GHz Inter(R)Core(TM)i7 processor with 8GB RAM.

For the sake of justice, we take the same initial guess $x_0 = v = e/n$ for all algorithms, where $e = [1, 1, \ldots, 1]^T$. All the stopping criteria are set as $\|\alpha P x_k + (1 - \alpha)v - x_k\|_1 < \tau$, where τ is specified in experiment description. We choose $\alpha = 0.85, 0.90, 0.95$ and 0.99, and set $m = 2, 3, 5, 10, 20$ and 50. Referring to the analysis and empirical choices by Gleich et al. [15] and Gu et al. [17], we use the choices $\eta = 0.01$, $\beta = 0.5$, and assign different values to β_1 and β_2 around β.

The Web matrices are listed in Table 1, where "nnz" denotes the number of nonzero elements and "avg nnz per row" denotes the average number of nonzero elements per row. For convenience, in all the tables to follow we have abbreviated the power method, the inner-outer iteration method, the PIO iteration method and our MPMIO iteration method as Power, IO, PIO, MPMIO, respectively. We denote by "ite" the iteration counts, "mv" the number of matrix-vector products and by "CPU" the CPU time used in seconds.

Example 1. This example aims at discussing some of the options for parameters β_1 and β_2. The test matrix is amazon0505 Web matrix. With $m = 5$, $\tau = 10^{-8}$, we run the different methods for PageRank problem with different pairs of values for (β_1, β_2). Numerical results are presented in Table 2.

It is easy to see that MPMIO performs the best both in terms of iteration numbers and CPU time. As for the matrix-vector products, they are approximately equal and with the increase in α, MPMIO gradually reflects its advantages. At the same time, we find that there are still different performances with different values of parameters β_1 and β_2, and it is currently very hard to get the best choices of β_1 and β_2. Thus, we choose empirically the parameters as $\beta_1 = 0.6$ and $\beta_2 = 0.5$ in the following experiments.

Example 2. In this example, we examine the performance of the four methods for PageRank with various values of m. The test matrix is the Stanford-Berkeley Web matrix. Numerical results are listed in Table 3.

Table 1: Web matrices for PageRank problems

Name	Size	nnz	Avg nnz per row
Amazon0505	410,236	3,356,824	19.9
Stanford-Berkeley	683,446	7,583,376	11.1
Web-Google	916,428	5,105,039	5.6
Wikipedia-20051105	1,634,989	19,753,078	12.1

Table 2: Numerical results for Example 1 with $m = 5$, $\beta = 0.5$, $\tau = 10^{-8}$ and various value pairs of (β_1, β_2)

Method	alpha = 0.85		alpha = 0.90		alpha = 0.95		alpha = 0.99	
	ite (mv)	CPU	ite (mv)	CPU	ite (mv)	CPU	ite (mv)	CPU
Power	81 (81)	0.514029	122 (122)	0.730376	238 (238)	1.430304	1,043 (1,043)	6.436183
IO	77 (84)	0.577852	118 (126)	0.888501	233 (244)	1.753881	1,032 (1,047)	7.334066
PIO	39 (82)	0.492393	60 (125)	0.733401	118 (243)	1.421175	517 (1,044)	5.876933
MPMIO (0.5,0.4)	12 (85)	0.429520	18 (127)	0.630644	34 (240)	1.192104	149 (1,045)	5.381957
MPMIO (0.5,0.5)	12 (85)	0.439055	18 (128)	0.662717	34 (240)	1.221052	148 (1,040)	5.294960
MPMIO (0.6,0.4)	12 (85)	0.433683	18 (127)	0.634136	34 (240)	1.183174	149 (1,045)	5.342428
MPMIO (0.6,0.5)	12 (85)	0.436202	18 (128)	0.635568	34 (240)	1.189355	148 (1,040)	5.312114
MPMIO (0.7,0.4)	12 (85)	0.430336	18 (127)	0.642486	34 (240)	1.188378	149 (1,045)	5.301542
MPMIO (0.7,0.5)	12 (85)	0.440045	18 (128)	0.657614	34 (240)	1.239790	148 (1,040)	5.336923
MPMIO (0.7,0.6)	12 (85)	0.434616	18 (128)	0.658883	34 (241)	1.233188	148 (1,040)	5.310513
MPMIO (0.8,0.5)	12 (85)	0.429087	18 (128)	0.652059	34 (240)	1.215909	148 (1,040)	5.319561
MPMIO (0.8,0.6)	12 (85)	0.430386	18 (128)	0.644962	34 (241)	1.222724	148 (1,040)	5.267550
MPMIO (0.8,0.7)	12 (86)	0.428920	17 (122)	0.620944	34 (241)	1.226266	148 (1,041)	5.345917

Table 3: Numerical results for Example 2 with $\beta = 0.5$, $\beta_1 = 0.6$, $\beta_2 = 0.5$, $\tau = 10^{-8}$ and various values of m

Method	alpha = 0.85		alpha = 0.90		alpha = 0.95		alpha = 0.99	
	ite (mv)	CPU	ite (mv)	CPU	ite (mv)	CPU	ite (mv)	CPU
Power	91 (91)	1.025699	138 (138)	1.506846	277 (277)	3.041634	1,341 (1,341)	14.958891
IO	76 (85)	1.116774	115 (126)	1.622780	225 (238)	3.112812	1,024 (1,041)	14.065285
PIO	39 (83)	0.913716	58 (123)	1.331178	114 (236)	2.566159	513 (1,038)	11.338970
MPMIO ($m = 2$)	22 (91)	0.902490	30 (125)	1.243692	58 (237)	2.350283	257 (1,035)	10.291378
MPMIO ($m = 3$)	18 (93)	0.890123	24 (124)	1.200560	46 (235)	2.289176	206 (1,037)	10.301479
MPMIO ($m = 5$)	12 (86)	0.816488	19 (136)	1.291546	33 (235)	2.245291	148 (1,041)	10.003695
MPMIO ($m = 10$)	8 (97)	0.881731	10 (122)	1.112733	22 (267)	2.454582	86 (1,036)	9.521987
MPMIO ($m = 20$)	5 (111)	0.979710	7 (155)	1.382064	13 (287)	2.558793	58 (1,279)	11.093444
MPMIO ($m = 50$)	2 (105)	0.900378	3 (157)	1.359985	6 (313)	2.740545	26 (1,353)	11.957493

From Table 3, we find that among the four methods, MPMIO outperforms the other three methods, i.e., Power, IO, PIO, especially in iteration counts and CPU time. Meanwhile, we can find that the performance of MPMIO is sensitive to the choice of m. For example, when m takes a small value MPMIO is outstanding but when m takes a big value, like 20 or 50, the performance of MPMIO is barely satisfactory. Then, based on other similar observations of test matrices, we tend to choose a modest value of m, i.e., $m = 5$.

Example 3. In this example, we give a comprehensive investigation into the performance of MPMIO. We choose empirically the parameters as $\alpha = 0.99$, $\beta = 0.5$, $\beta_1 = 0.6$, $\beta_2 = 0.5$, $m = 5$, $\eta = 0.01$. We test all the matrices in Table 1 to compare the number of matrix-vector products and CPU time required for convergence to three different outer tolerances τ. To state a speedup of an algorithm a over another one b, we use speedup formula as

$$\text{Speedup} = \frac{\text{CPU}_b}{\text{CPU}_a}.$$

Numerical results are given in Table 4.

This example shows that our proposed algorithm MPMIO can reduce the number of matrix-vector products obviously and is proved to be efficient. Based on the CPU time, MPMIO iteration performs the best for each test PageRank problem. When $\tau = 10^{-8}$, MPMIO achieves a speedup from 1.13× to 1.48× over IO

Table 4: Numerical results for Example 3 with $\alpha = 0.99$, $m = 5$, $\beta = 0.5$, $\beta_1 = 0.6$, $\beta_2 = 0.5$, $\eta = 0.01$. *Speedup* represents the relative speedup in CPU time of MPMIO over IO and PIO (in brackets)

Tol	Web matrix	mv				CPU				
		Power	IO	PIO	MPMIO	Power	IO	PIO	MPMIO	Speedup
10^{-4}	Amazon0505	283	291	288	243	1.906843	1.979214	1.940777	1.559590	1.27 (1.24)
	Stanford-Berkeley	473	260	258	220	5.106515	3.266863	2.748681	2.371718	1.38 (1.16)
	Web-Google	489	304	303	261	10.132581	6.862155	5.917642	5.195980	1.32 (1.14)
	Wikipedia20051105	42	44	43	37	2.890107	3.146707	2.877156	2.810747	1.12 (1.02)
10^{-6}	Amazon0505	652	659	656	561	4.127204	4.772426	4.057435	3.375080	1.41 (1.20)
	Stanford-Berkeley	902	631	628	538	10.429152	8.750668	7.169710	6.206567	1.41 (1.16)
	Web-Google	942	636	635	549	18.705083	14.338529	12.547242	11.190281	1.28 (1.12)
	Wikipedia20051105	390	342	341	295	26.986150	25.628100	23.665184	22.695913	1.13 (1.04)
10^{-8}	Amazon0505	1,043	1,047	1,044	891	6.608148	7.811836	6.310431	5.269707	1.48 (1.20)
	Stanford-Berkeley	1,341	1,041	1,038	892	14.471853	13.509255	10.938215	9.651750	1.40 (1.13)
	Web-Google	1,399	991	993	867	27.265529	22.284097	19.162797	17.304339	1.29 (1.11)
	Wikipedia20051105	847	799	799	691	59.130899	59.895641	55.289397	53.219597	1.13 (1.04)

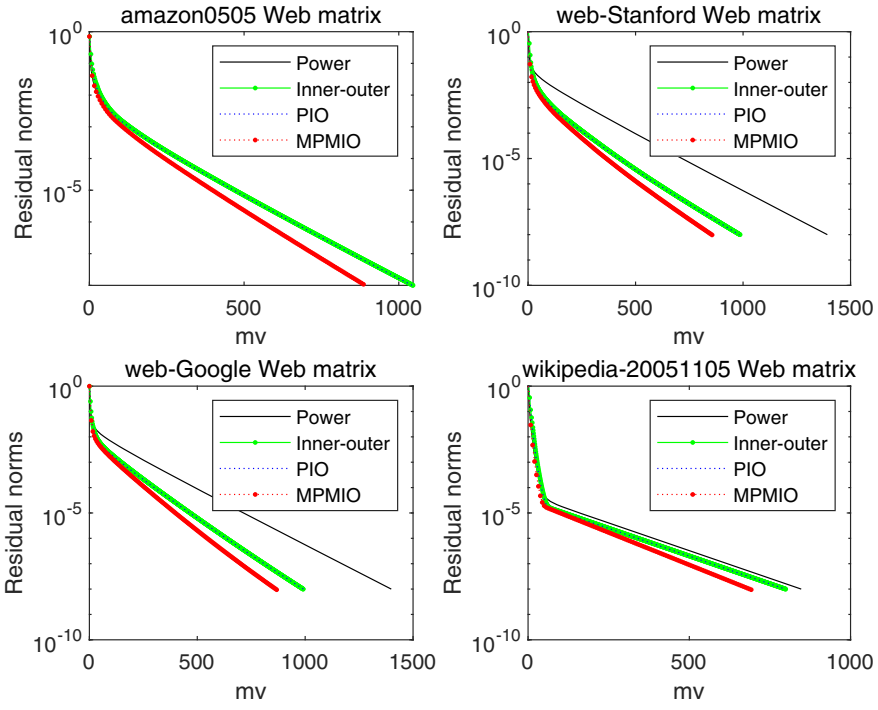

Figure 1: Convergence of the computation for the four Web matrix. $\alpha = 0.99$, $\tau = 10^{-8}$.

and from 1.04× to 1.20× over PIO. Taken together, our proposed MPMIO iteration converges faster than the other three methods, and this has indeed been shown by the convergent curves in Figure 1.

5 Conclusions

In this paper, we have improved the IO iteration for accelerating PageRank computation by introducing multi-step power and multi-step splitting. Our algorithm can be implemented easily, and theoretical results show its efficiency. Numerical experiments on several PageRank problems have indicated that the new

algorithm is superior to the power method and the IO iteration methods, IO and PIO. At the same time, we have also realized that the new algorithm is parameter-dependent and appropriate choice of parameters can be made in our experiments. It is meaningful to explore how to determine the optimal parameters for our algorithm and may be included in the future work.

Acknowledgments: This research was supported by the Key Fund Project of Sichuan Provincial Department of Education (17za0003).

References

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Technical Report, Stanford InfoLab, 1999.
[2] A. N. Langville and C. D. Meyer, *Deeper inside PageRank*, Internet Math. **1** (2004), no. 3, 335–380, DOI: https://doi.org/10.1080/15427951.2004.10129091.
[3] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, Princeton, NJ, 2012.
[4] P. Berkhin, *A survey on PageRank computing*, Internet Math. **2** (2005), no. 1, 73–120, DOI: https://doi.org/10.1080/15427951.2005.10129098.
[5] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, *Extrapolation methods for accelerating PageRank computations*, in: WWW '03 – Proceedings of the 12th International Conference on World Wide Web, Association for Computing Machinery, New York, NY, USA, 2003, pp. 261–270.
[6] A. Sidi, *Vector extrapolation methods with applications to solution of large systems of equations and to PageRank computations*, Comput. Math. Appl. **56** (2008), no. 1, 1–24, DOI: https://doi.org/10.1016/j.camwa.2007.11.027.
[7] B. Y. Pu, T. Z. Huang and C. Wen, *A preconditioned and extrapolation-accelerated GMRES method for PageRank*, Appl. Math. Lett. **37** (2014), 95–100, DOI: https://doi.org/10.1016/j.aml.2014.05.017.
[8] H. Migallon, V. Migallon, J. A. Palomino, and J. Penades, *A heuristic relaxed extrapolated algorithm for accelerating PageRank*, Adv. Eng. Softw. **120** (2018), 88–95, DOI: https://doi.org/10.1016/j.advengsoft.2016.01.024.
[9] X. Tan, *A new extrapolation method for PageRank computations*, J. Comput. Appl. Math. **313** (2017), 383–392, DOI: https://doi.org/10.1016/j.cam.2016.08.034.
[10] G. Wu and Y. Wei, *An Arnoldi-extrapolation algorithm for computing PageRank*, J. Comput. Appl. Math. **234** (2010), no. 11, 3196–3212, DOI: https://doi.org/10.1016/j.cam.2010.02.009.
[11] H. De Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, and J. Ruge, *Multilevel adaptive aggregation for Markov chains, with application to web ranking*, SIAM J. Sci. Comput. **30** (2008), no. 5, 2235–2262, DOI: https://doi.org/10.1137/070685142.
[12] Y. Lin, X. Shi, and Y. Wei, *On computing PageRank via lumping the Google matrix*, J. Comput. Appl. Math. **224** (2009), no. 2, 702–708, DOI: https://doi.org/10.1016/j.cam.2008.06.003.
[13] I. R. Mendes and P. B. Vasconcelos, *PageRank computation with MAAOR and lumping methods*, Math. Comput. Sci. **12** (2018), no. 2, 129–141, DOI: https://doi.org/10.1007/s11786-018-0335-7.
[14] S. Kamvar, T. Haveliwala, and G. Golub, *Adaptive methods for the computation of PageRank*, Linear Algebra Appl. **386** (2004), 51–65, DOI: https://doi.org/10.1016/j.laa.2003.12.008.
[15] D. Gleich, A. Gray, C. Greif, and T. Lau, *An inner-outer iteration for computing PageRank*, SIAM J. Sci. Comput. **32** (2010), 349–371, DOI: https://doi.org/10.1137/080727397.
[16] Z. Z. Bai, *On convergence of the inner-outer iteration method for computing PageRank*, Numer. Algebra Control Optim. **2** (2012), no. 4, 855–862, DOI: https://doi.org/10.3934/naco.2012.2.855.
[17] C. Gu, F. Xie, and K. Zhang, *A two-step matrix splitting iteration for computing PageRank*, J. Comput. Appl. Math. **278** (2015), 19–28, DOI: https://doi.org/10.1016/j.cam.2014.09.022.
[18] C. Gu and W. Wang, *An Arnoldi-Inout algorithm for computing PageRank problems*, J. Comput. Appl. Math. **309** (2017), 219–229, DOI: https://doi.org/10.1016/j.cam.2016.05.026.
[19] Y. J. Xie and C. F. Ma, *A relaxed two-step splitting iteration method for computing PageRank*, Comp. Appl. Math. **37** (2018), 221–233, DOI: https://doi.org/10.1007/S40314-016-0338-4.
[20] Z. Tian, Y. Liu, Y. Zhang, Z. Liu, and M. Tian, *The general inner-outer iteration method based on regular splittings for the PageRank problem*, Appl. Math. Comput. **356** (2019), 479–501, DOI: https://doi.org/10.1016/j.amc.2019.02.066.
[21] R. S. Wills and I. C. F. Ipsen, *Ordinal ranking for Google's PageRank*, SIAM J. Matrix Anal. Appl. **30** (2009), no. 4, 1677–1696, DOI: https://doi.org/10.1137/070698129.