Journal of Computational and Applied Mathematics 346 (2019) 456–470

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Off-diagonal low-rank preconditioner for difficult PageRank problems

Zhao-Li Shen [a,b], Ting-Zhu Huang [a,*], Bruno Carpentieri [c], Chun Wen [a], Xian-Ming Gu [b,d], Xue-Yuan Tan [e]

[a] *School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, PR China*
[b] *Johann Bernoulli Institute for Mathematics and Computer Science, Faculty of Science and Engineering, University of Groningen, 9700 AK Groningen, The Netherlands*
[c] *School of Science and Technology, Faculty of Computer Science, Free University of Bozen-Bolzano, 39100 Bolzano, Italy*
[d] *School of Economic Mathematics, Southwestern University of Finance and Economics, Chengdu, Sichuan 611130, PR China*
[e] *Jiangsu Key Laboratory for NSLSCS, School of Mathematical Science, Nanjing Normal University, Nanjing 210046, PR China*

A R T I C L E I N F O

Article history:
Received 26 February 2017
Received in revised form 16 December 2017

Keywords:
PageRank
Off-diagonal
Low-rank factorization
Matrix partition
Preconditioner

A B S T R A C T

PageRank problem is the cornerstone of Google search engine and is usually stated as solving a huge linear system. Moreover, when the damping factor approaches 1, the spectrum properties of this system deteriorate rapidly and this system becomes difficult to solve. In this paper, we demonstrate that the coefficient matrix of this system can be transferred into a block form by partitioning its rows into special sets. In particular, the off-diagonal part of the block coefficient matrix can be compressed by a simple low-rank factorization, which can be beneficial for solving the PageRank problem. Hence, a matrix partition method is proposed to discover the special sets of rows for supporting the low-rank factorization. Then a preconditioner based on the low-rank factorization is proposed for solving difficult PageRank problems. Numerical experiments are presented to support the discussions and to illustrate the effectiveness of the proposed methods.

1. Introduction

With the rapid development of the Internet, Web search engines become very popular for information retrieval [1]. Because a web search engine can usually find an immense set of Web pages matching the search query, it is necessary to rank higher the most important pages. For this purpose, Google has presented the PageRank model (also called PageRank problem) that employs the link structure of Web pages to quantify the importance of each one.

The detailed mathematical background of the PageRank problem can be found in [2], here we give a brief review. The link structure of the related Web pages is represented by a directed graph named the Web link graph. Denote its adjacency matrix by $G \in \mathbb{N}^{n\times n}$ where n is the number of nodes (pages), then $G(i,j)$ is nonzero (being 1) only when page j has a link pointing to page i. Then the transition matrix $P \in \mathbb{R}^{n\times n}$ with respect to the Web link graph is defined as

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

* Corresponding author.
E-mail addresses: szlxiaoyao@163.com (Z.-L. Shen), tingzhuhuang@126.com (T.-Z. Huang).

https://doi.org/10.1016/j.cam.2018.07.015
0377-0427/© 2018 Elsevier B.V. All rights reserved.

Finally, the PageRank problem can be mathematically stated as solving the linear system:

$$Ax = v, \text{ with } A = (I - \alpha P), \tag{2}$$

where $0 < \alpha < 1$ is the damping factor, $v \in \mathbb{R}^{n\times 1}$ is called the personalization vector, and the solution x is the unknown ranking vector [3].

The coefficient matrix A of the PageRank linear system (2) is usually extremely large and very sparse, thus iterative methods are mandatory to use for solving this system [4,5]. Google initially assigns a moderate value 0.85 to α, and then linear system (2) can be solved efficiently by the simple Power method [2]. However, larger values of α (not too close to 1) such as 0.99 may sometimes give a better ranking result than $\alpha = 0.85$ [6,7]. When α approaches 1, the smallest nonzero eigenvalue of A approaches 0, and the convergence rate of the Power method deteriorates rather quickly [2]. Therefore, the development of more robust and more efficient iterative methods is necessary for PageRank problems [4,7].

In the last decade or so, a series of techniques have been proposed to accelerate the Power method for PageRank computations, such as the adaptive method [8], the inner–outer strategy [9], the extrapolation methods [10,11] and the multi-step splitting iteration method [1,12]. Meanwhile, a considerable amount of work has been devoted to Krylov subspace methods in this context, see e.g. [7,13–17]. In many cases Krylov subspace methods like GMRES, BiCG, and BiCGSTAB methods outperform the Power method and the Jacobi method by 50% in terms of computing time [16]. Moreover, the convergence speed of these methods degrades more gently than that of the Power method when α approaches 1 [17]. Therefore, Krylov subspace methods seem to be a promising way for solving difficult PageRank problems where α is close to 1. Nevertheless, due to the ever-growing size of the World Wide Web, acceleration techniques become more and more attractive for these methods when solving difficult PageRank problems.

Preconditioning is one of the most effective ways to accelerate Krylov subspace methods for solving general linear systems. When solving huge and difficult PageRank problems, however, applying general-purpose preconditioners has received unsatisfactory results. See for example the experiments reported with the polynomial preconditioner in [18,19] and with the Incomplete LU (ILU) factorization preconditioner in [20]. To achieve a fast convergence rate of the iterative process, the polynomial preconditioner is usually a high-order polynomial of the matrix P, which causes a large time cost for computing the matrix–vector multiplications in the preconditioning process, meanwhile the ILU factorization preconditioner often suffers from a large amount of fill-ins and thus high costs in storage and time. From these results, preconditioning the PageRank system as same as a general linear system, its huge dimension and huge amount of nonzero elements may often cause high computational cost of the preconditioning process. This defect motivates us to develop preconditioners from the exclusive characteristics of PageRank problems.

Recently, an elimination strategy is proposed in [21]. This strategy exploits the properties of the PageRank coefficient matrix A for eliminating its nonzero elements, and consequently the PageRank linear system (2) is transferred into a system with higher sparsity. As the experiment results in [21] show, solving the linear system after elimination for seeking the ranking vector generally decreases the computational cost of solving the PageRank problem. This inspires us to pay attention on the properties of the PageRank coefficient matrix A for developing preconditioners.

In this study, we show that partitioning the rows of A into sets that satisfy a condition can derive a block form of A where the rows in each off-diagonal block are identical. Therefore these off-diagonal blocks can be compressed efficiently by a low-rank factorization, which may benefit the efficiency of solving the PageRank problem. Hence we propose a matrix partition method that can efficiently discover the aforementioned sets of rows of A to support the low-rank factorization. Finally a preconditioner based on the low-rank factorization is proposed to accelerate Krylov subspace methods for solving difficult PageRank problems.

The paper is structured as follows. In Section 2, we introduce some notations, some definitions and a property of the coefficient matrix A. In Section 3, we present the low-rank factorization for the PageRank problem. In Section 4, we propose the matrix partition method. The preconditioner is proposed and discussed in Section 5. Numerical experiments are reported in Section 6. Finally, some conclusions arising from this work are given in Section 7.

2. Preliminaries

We use MATLAB [22] notations to represent elements of vectors and matrices, subvectors, submatrices, matrices in the form of columns/rows. Besides, Table 1 lists several notations that frequently appear throughout this paper.

We then give two definitions and present a property of the PageRank coefficient matrix A.

Definition 2.1 (*Off-Diagonal Part*)**.** For any matrix $U \in \mathbb{R}^{n\times n}$, let $R = \{1 : n\}$ and $S = \cup_{i\in\Psi}\{U(i, :)\}$ be a set of rows where $\Psi \subset R$. We define the **off-diagonal part** of S as $U(\Psi, R\backslash\Psi)$.

Definition 2.2 (*Core Hub*)**.** Let S be a set of rows of A. If the rows in the off-diagonal part of S have the same zero-nonzero pattern, S is called as a **core hub** of rows.

Property 2.1. *Let S be any core hub of rows of A. Then the rows in the off-diagonal part of S are identical.*

Proof. See Appendix A.

Table 1
The notations that frequently appear throughout this paper.

Notation	Meaning
u	Arbitrary column vector of appropriate dimension
U	Arbitrary matrix of appropriate dimensions
$\rho(U)$	Spectral radius of matrix U
I	Identity matrix of appropriate dimension
G	Web adjacency matrix
P	Transition matrix of (1)
A	Coefficient matrix of (2)
n	Dimension of matrix A
$\tilde{A}$	Block form (4) of A
Π	Permutation matrix that permutes A into $\tilde{A}$ by $\tilde{A} = \Pi A \Pi^T$
D	Matrix of (8) that is formed by splitting $\tilde{A}$
B_{off}	Matrix of (8) that is formed by splitting $\tilde{A}$
F, H	Low-rank factors (defined in (11) and (10)) of B_{off}
C	Capacitance matrix formed by $I + HD^{-1}F$
m	Row dimension of H & column dimension of F & dimension of C
$rownnz(i)$	Amount of the nonzero elements in $G(i,:)$
θ	Parameter in partition algorithm for controlling the pre-marking process
ω	Parameter in partition algorithm for limiting the length of each inner traversal

For example, let $S = \{A(i,:), A(j,:), A(k,:), A(l,:)\}$ $(1 \le i < j < k < l \le n)$, its off-diagonal part contains the elements (marked by $*$) whose column indexes are from $N = \{1:n\}\backslash\{i,j,k,l\}$, in matrix (3).

$$\begin{bmatrix} A(i,:) \\ A(j,:) \\ A(k,:) \\ A(l,:) \end{bmatrix} = \begin{array}{c|ccccccccccccc|} & 1 & 2 & \cdots & i & \cdots & j & \cdots & k & \cdots & l & \cdots & n \\ i & * & * & \cdots & \diamond & \cdots & \diamond & \cdots & \diamond & \cdots & \diamond & \cdots & * \\ j & * & * & \cdots & \diamond & \cdots & \diamond & \cdots & \diamond & \cdots & \diamond & \cdots & * \\ k & * & * & \cdots & \diamond & \cdots & \diamond & \cdots & \diamond & \cdots & \diamond & \cdots & * \\ l & * & * & \cdots & \diamond & \cdots & \diamond & \cdots & \diamond & \cdots & \diamond & \cdots & * \end{array}. \tag{3}$$

If S is a core hub of rows, then the rows of this off-diagonal part must be identical, i.e.

$$A(i,N) = A(j,N) = A(k,N) = A(l,N).$$

3. Off-diagonal low-rank factorization

If we partition the rows of A into core hubs, permute the rows hub by hub in A, and permute the columns accordingly in A, then we obtain a block matrix $\tilde{A}$ of the form

$$\tilde{A} = \Pi A \Pi^T = \begin{pmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} & \tilde{A}_{1,3} & \cdots & \tilde{A}_{1,l} \\ \tilde{A}_{2,1} & \tilde{A}_{2,2} & \tilde{A}_{2,3} & \cdots & \tilde{A}_{2,l} \\ \tilde{A}_{3,1} & \tilde{A}_{3,2} & \tilde{A}_{3,3} & \cdots & \tilde{A}_{3,l} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{A}_{l,1} & \tilde{A}_{l,2} & \tilde{A}_{l,3} & \cdots & \tilde{A}_{l,l} \end{pmatrix}, \tag{4}$$

where $\Pi \in \mathbb{N}^{n\times n}$ is a permutation matrix, and l is the number of hubs. Accordingly, the PageRank system (2) can be equivalently transferred into the block form:

$$\tilde{A}\tilde{x} = \tilde{v}, \tag{5}$$

where $\tilde{v} = \Pi v$ and $\tilde{x} = \Pi x$. We then focus on solving this block linear system. For each diagonal block $A_{i,i}$ $(1 \le i \le l)$ in (4), its corresponding off-diagonal block row off_i is defined as

$$off_i = [\tilde{A}_{i,1}, \ldots, \tilde{A}_{i,i-1}, 0, \tilde{A}_{i,i+1}, \ldots, \tilde{A}_{i,l}], \tag{6}$$

and the rows in off_i must be identical. Thus each off_i $(1 \le i \le l)$ is a rank-one matrix and can be compressed by a low-rank factorization. We call the combination of all the low-rank factorizations for off-diagonal block rows of $\tilde{A}$ as the off-diagonal low-rank factorization.

For convenience of describing this low-rank factorization, we split $\tilde{A}$ into the form

$$\tilde{A} = D + B_{off}, \tag{7}$$

where B_{off} contains the off-diagonal block rows that will be compressed. It should be noted that the off-diagonal part of a core hub of rows may contain few nonzero elements. As a result, compressing the off-diagonal block row corresponding to such a hub is not that worthwhile, and thus this off-diagonal block row should not be classified into B_{off}. Suppose there are s such hubs of rows, if we permute them to the top of $\tilde{A}$, then D and B_{off} in (7) are given by

$$D = \begin{pmatrix} \tilde{A}_{1,1} & \cdots & \tilde{A}_{1,s} & \tilde{A}_{1,s+1} & \cdots & \tilde{A}_{1,l} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \tilde{A}_{s,1} & \cdots & \tilde{A}_{s,s} & \tilde{A}_{s,s+1} & \cdots & \tilde{A}_{s,l} \\ 0 & \cdots & 0 & \tilde{A}_{s+1,s+1} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \tilde{A}_{l,l} \end{pmatrix}, \quad B_{off} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ off_{s+1} \\ \vdots \\ off_l \end{pmatrix}. \tag{8}$$

At this stage, the off-diagonal low-rank factorization of $\tilde{A}$ is given by

$$\tilde{A} = D + FH \quad \text{with} \tag{9}$$

$$F \in \mathbb{N}^{n\times m} \quad \text{where} \quad F(i,j) = \begin{cases} 1, & \text{if } B_{off}(i,:) \in off_j \ (s+1 \le j \le l), \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

$$H = \begin{pmatrix} off_{s+1}(1,:) \\ off_{s+2}(1,:) \\ \vdots \\ off_l(1,:) \end{pmatrix} \in \mathbb{R}^{m\times n} \quad (m = l - s). \tag{11}$$

That it, each off_i $(s+1 \le i \le l)$ in B_{off} is compressed into one single row in H. If off_i $(i = s+1, 2, \dots, l)$ are fairly dense, then $nnz(H)$ would be much smaller than $nnz(B_{off})$. Meanwhile, according to (10), F contains at most n nonzero elements. Therefore, $nnz(F) + nnz(H)$ would be much smaller than $nnz(B_{off})$. Accordingly, the storage and the time required by any matrix–vector multiplication $\tilde{A}u$ is likely to be reduced if it is computed by $\tilde{A}u = Du + F(Hu)$. As $nnz(F)$ is very small, such reduction in relative terms is generally determined by the quantity

$$\text{compression ratio} = \frac{nnz(B_{off}) - nnz(H)}{nnz(A)}. \tag{12}$$

Besides, a larger compression ratio would also lead to a D with higher sparsity or smaller ranks of F and H. This is helpful to invert $\tilde{A}$ efficiently by using such formulation $\tilde{A} = D + FH$ [23], which is the basis of constructing the preconditioner in the following contents. Motivated by these factors, we propose a matrix partition algorithm to discover appropriate core hubs of rows of A efficiently for maximizing the compression ratio.

4. Matrix partition method for supporting the low-rank factorization

For finding core hubs of rows of $A = I - \alpha P$ according to Definition 2.2, the partition method only needs the information from the off-diagonal zero-nonzero pattern structure of A. This pattern structure is same as that of the adjacency matrix G, because G and P have the same zero-nonzero pattern structure. Hence it is convenient to run the partition method on G that is already a binary matrix to find core hubs of rows of A. The framework of this method can be described as follows:

1 **Denote** n hubs by $hub_i = \{G(i,:)\}$ $(i = 1, 2, \dots, n)$.

2 **Traverse** the unmarked hubs. For each visited hub_i $(1 \le i \le n)$, **traverse** the rows $G(j,:)$ $(j = i+1, i+2, \dots, n)$, **if** a visited row $G(j,:)$ is unmarked and $hub_i \cup G(j,:)$ is a core hub, **then**

   2.1 **Set** $hub_i = hub_i \cup G(j,:)$.

   2.2 **Mark** $G(j,:)$ and hub_j such that they will be skipped during the remaining traversals.

   2.3 **Stop** the current inner traversal and **restart** it for the enlarged hub_i.

3 **Output** the row indexes of each unmarked hub for determining core hubs of rows of A.

Each restart operation for inner traversals occurs after a marking operation that will reduce the number of inner traversals by 1. Accordingly, this method still has n inner traversals and is very costly due to the large dimension of matrix G. Moreover it may form many hubs that contribute little to the compression ratio. Hence we employ some amendments for overcoming these problems. Detailed descriptions (including the motivations and the effects) for these amendments are contained in Appendix B.

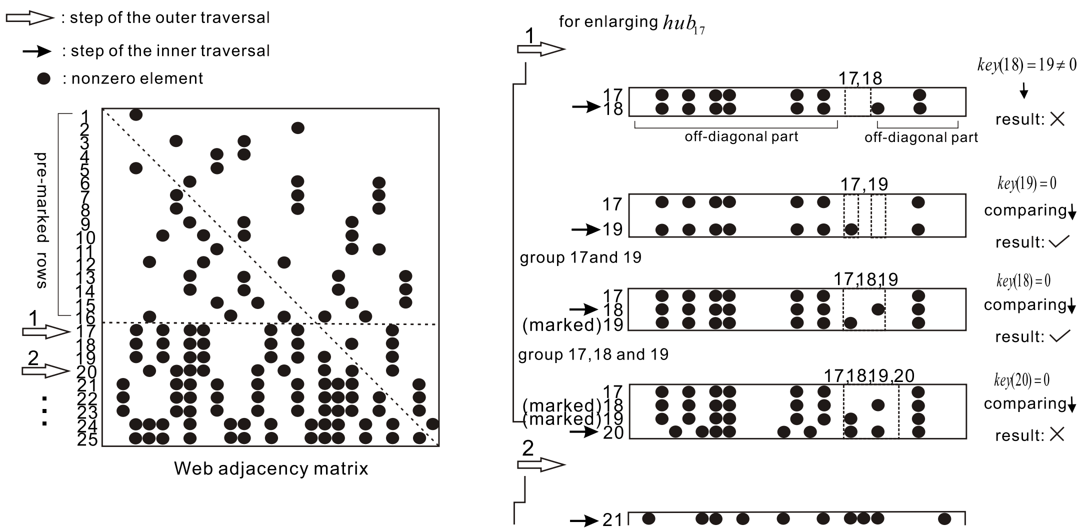

Fig. 1. A period of the partition process of Algorithm 1 for a small artificial Web adjacency matrix.

Amendment 1. We denote the amount of the nonzero entries of each row $G(k, :)$ $(1 \le k \le n)$ by $rownnz(k)$. After the initial setting (step 1), we mark the rows of G and their hubs in a $rownnz$ increasing order until the amount of nonzeros in the marked rows reaches $\theta nnz(G)$, where $0 < \theta < 1$. We call this step as pre-marking step.

Amendment 2. We only traverse at most $\omega = 100$ rows in each inner traversal.

Amendment 3. Let $R = 1 : n$ and denote the set of the row indexes of hub_i $(1 \le i \le n)$ by N_i. For each inner traversal related with hub_i $(1 \le i \le n)$, we compute a value $key(j)$ for each visited unmarked row $G(j, :)$ $(i + 1 < j < min(n, i + \omega))$ by

$$key(j) = \sum_{G(j,k)=1} k - \sum_{G(i,k)=1} k \text{ with } k \in \{R\backslash\{N_i \cup j\}\}. \tag{13}$$

If $key(j) \neq 0$, then $G(j, :)$ should not be added to hub_i. Otherwise, we implement an element-wise comparison for the nonzero patterns of the rows of $hub_i \cup G(j, :)$ to judge whether it is a core hub.

With these amendments, the complete version of the matrix partition method is presented in Algorithm 1, and a small example is shown in Fig. 1 to illustrate the aspects of this method.

5. Off-diagonal low-rank preconditioner

Preconditioning a general linear system $Uy = b$ can be represented as

$$M^{-1}Uy = M^{-1}b \qquad \text{called left preconditioning,}$$
$$UM^{-1}z = b,\ y = M^{-1}z \qquad \text{called right preconditioning,}$$

where $M^{-1} \approx U^{-1}$ is the applied preconditioner to make the spectrum properties of the linear system more favorable for iterative methods. Generally speaking, the acceleration effect of the preconditioner M^{-1} on iterative methods depends on its efficiency for approximating U^{-1}. The pre-described techniques transfer the PageRank problem as solving the block linear system (5) with the coefficient matrix $\tilde{A} = D + FH$. This formulation may be helpful to compute $\tilde{A}^{-1}$ efficiently, according to the analysis in [23]. Hence, we propose a preconditioner based on this formulation for solving difficult PageRank problems where the damping factor α is close to 1.

5.1. Formulation and implementation

The theoretical basis of this preconditioner is the Woodbury formula recalled in Lemma 5.1.

Lemma 5.1 (*Woodbury Formula Stated in [24]*). *Let $W \in \mathbb{R}^{n\times n}$ be a nonsingular matrix, $X \in \mathbb{R}^{n\times m}$ and $Y \in \mathbb{R}^{m\times n}$ $(n \ge m)$. If $I + YW^{-1}X$ is nonsingular, then $W + XY$ is nonsingular and its inverse can be given by*

$$(W + XY)^{-1} = W^{-1} - W^{-1}X(I + YW^{-1}X)^{-1}YW^{-1}.$$

Algorithm 1 Partition method for finding core hubs of rows of A

```
Input:  $G, \theta, \omega$
1:  Compute the $rownnz$ value for each row of $G$.
2:  Set the hub of each row as itself, i.e. $hub_i = \{G(i,:)\}$ for $i = 1 : n$.
3:  Mark the rows of $G$ and their hubs in a $rownnz$ increasing order until the amount of nonzero elements in the marked
    rows reaches $\theta nnz(G)$.
4:  for $i = 1 : n$ do
5:    if $hub_i$ is unmarked then
6:      for $j = i + 1 : min(n, i + \omega)$ do
7:        if $G(j,:)$ is unmarked then
8:          Compute $key(j)$ according to (22).
9:          if $key(j) = 0$ then
10:           if $\{G(j,:) \cup hub_i\}$ is a core hub then
11:             $hub_i \leftarrow \{hub_i \cup G(j,:)\}$; $hub_j \leftarrow \varnothing$.
12:             mark $G(j,:)$ and $hub_j$.
13:             back to step 6.
14:           end if
15:         end if
16:       end if
17:     end for
18:   end if
19: end for
20: return the information of the row indexes of the non-empty hubs.
```

The matrix $I + YW^{-1}X$ is often called the *capacitance matrix*. For the formulation $\tilde{A} = D + FH$, the capacitance matrix is $I + HD^{-1}F$. We denote it by C. Then the following theorem gives a formulation of $\tilde{A}^{-1}$.

Theorem 5.1. *D is a nonsingular M-matrix and $C = I + HD^{-1}F$ is nonsingular. Therefore,*

$$\tilde{A}^{-1} = D^{-1}(I - FC^{-1}HD^{-1}). \tag{14}$$

Proof. According to (1), $P \geq 0$ and $\rho(P) \leq \|P\|_1 \leq 1$. Because $0 < \alpha < 1$, $A = I - \alpha P$ and $\tilde{A} = \Pi A \Pi^T$ must be nonsingular M-matrices. According to (8), D is formed by changing some off-diagonal nonzero elements of $\tilde{A}$ to 0, thus D is also a nonsingular M-matrix.

To prove the invertibility of C, we construct a matrix T as

$$T = \begin{pmatrix} D & -F \\ H & I \end{pmatrix}.$$

Then we can write

$$\begin{pmatrix} I & 0 \\ -HD^{-1} & I \end{pmatrix} T = \begin{pmatrix} D & -F \\ 0 & C \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} I & F \\ 0 & I \end{pmatrix} T = \begin{pmatrix} \tilde{A} & 0 \\ H & I \end{pmatrix}.$$

Taking the determinants of the matrices in these two equations indicates the relation:

$$det(T) = det(D)det(C) = det(\tilde{A}).$$

As $\tilde{A}$ and D are nonsingular, C must be nonsingular. Finally, based on Lemma 5.1, $\tilde{A}^{-1}$ can be written as (14). □

With this theorem, formulation (14) can be used to construct preconditioners that approximate $\tilde{A}^{-1}$. In a practical PageRank problem, the related Web link graph is usually very large, thus forming a preconditioner $M^{-1} \approx \tilde{A}^{-1}$ explicitly may need an inordinate amount of storage and time. Hence, we employ the implicit preconditioning, i.e. M^{-1} is not formed, instead, the preconditioner is given by an algorithm (In Algorithm 2) for computing $M^{-1}u$. We call this preconditioner as off-diagonal low-rank (in short, ODLR) preconditioner.

In Algorithm 2, the linear systems with D at lines 1 and 5 may be solved approximately. We use $\tilde{D}^{-1}$ to represent the approximation of D^{-1} in this preconditioner, i.e. the result $\tilde{y}$ of solving such linear system $Dy = u$ satisfies $\tilde{y} \approx \tilde{D}^{-1}u$. At line 3, $C = I+HD^{-1}F$ is then approximated as $I+H\tilde{D}^{-1}F$ (denoted by $\bar{C}$), and the system with $\bar{C}$ may also be solved approximately. We use $\tilde{C}^{-1}$ to represent the overall approximation for C^{-1}. Finally the ODLR preconditioner M^{-1} writes as

$$M^{-1} = \tilde{D}^{-1} - \tilde{D}^{-1}F\tilde{C}^{-1}H\tilde{D}^{-1}. \tag{15}$$

Algorithm 2 Matrix–vector multiplication for the ODLR preconditioner: $x \leftarrow M^{-1}u$

```
1: Solve $Dy = u$.
2: Compute $z \leftarrow Hy$.
3: Solve $Cx = z$.
4: Compute $z \leftarrow Fx$.
5: Solve $Dx = z$.
6: Compute $x \leftarrow y + x$.
7: return $x$;
```

To summarize, using this preconditioner for solving a PageRank problem follows the steps:

1. run Algorithm 1 to discover core hubs of rows of A of the linear system (2);
2. permute the linear system (2) into the block form $\tilde{A}\tilde{x} = \tilde{v}$ of (5), and then get the formulation $\tilde{A} = D + FH$ by the off-diagonal low-rank factorization (9)–(11);
3. apply Algorithm 2 at each step of an iterative method to solve $\tilde{A}M^{-1}\tilde{y} = \tilde{v}$;
4. obtain the PageRank vector x by $x = \Pi^T M^{-1}\tilde{y}$.

5.2. Performance analysis

Because the linear systems with D and the linear system with C in Algorithm 2 may be solved approximately, it is necessary to study the relationship between the performance of the ODLR preconditioner M^{-1} of (15) and solving such systems (generally called inner systems). We start by studying the value of $\|M^{-1} - \tilde{A}^{-1}\|$ ($\|\cdot\|$ may be any type of matrix norm) and the spectral properties of $\tilde{A}M^{-1}$.

In the ODLR preconditioner, the accuracies of solving inner systems with D and those with C can be represented by $\|E_D = \tilde{D}^{-1} - D^{-1}\|$ and $\|E_C = \tilde{C}^{-1} - C^{-1}\|$, respectively. By subtracting $\tilde{A}^{-1}$ of (14) from M^{-1} of (15), we can get

$$\begin{aligned}
M^{-1} - \tilde{A}^{-1} &= (\tilde{D}^{-1} - D^{-1}) + (D^{-1}FC^{-1}HD^{-1} - \tilde{D}^{-1}F\tilde{C}^{-1}H\tilde{D}^{-1}) \\
&= (\tilde{D}^{-1} - D^{-1}) + (D^{-1} - \tilde{D}^{-1})FC^{-1}HD^{-1} + \tilde{D}^{-1}F(C^{-1}HD^{-1} - \tilde{C}^{-1}H\tilde{D}^{-1}) \\
&= E_D(I - FC^{-1}HD^{-1}) + \tilde{D}^{-1}F(C^{-1} - \tilde{C}^{-1})HD^{-1} + \tilde{D}^{-1}F\tilde{C}^{-1}H(D^{-1} - \tilde{D}^{-1}) \\
&= E_DD(D^{-1} - D^{-1}FC^{-1}HD^{-1}) - (D^{-1} + E_D)FE_CHD^{-1} \\
&\quad - (D^{-1} + E_D)F(C^{-1} + E_C)HE_D \\
&= E_DD\tilde{A}^{-1} - D^{-1}FE_CHD^{-1} - E_DFE_CHD^{-1} - D^{-1}FC^{-1}HE_D \\
&\quad - D^{-1}FE_CHE_D - E_DFC^{-1}HE_D - E_DFE_CHE_D
\end{aligned} \tag{16}$$

Formula (16) expresses $M^{-1} - \tilde{A}^{-1}$ in terms of E_D and E_C with coefficients in matrix forms. We try to find upper-bounds for the norm values of these coefficients to reveal some relationships between $\|M^{-1} - \tilde{A}^{-1}\|$, $\|E_D\|$ and $\|E_C\|$. For this purpose, we first introduce the following lemma.

Lemma 5.2 (*Corollary 2 in [25]*). *If a matrix $B \in \mathbb{R}^{n\times n}$ is strictly diagonally dominant by columns (i.e., $|B(k,k)| > \sum_{i\neq k}|B(i,k)|$ for any $1 \le k \le n$), and*

$$\beta = \min_k(|B(k,k)| - \sum_{i\neq k}|B(i,k)|),$$

then $\|B^{-1}\|_1 \le 1/\beta$.

Based on this lemma, we give the following upper bounds in matrix 1-norm.

Lemma 5.3. *For the formulation $\tilde{A} = D + FH$, the following inequalities hold.*

$$\begin{aligned}
&\|D^{-1}\|_1 \le \frac{1}{1-\alpha}, \quad \|H\|_1 \le \alpha \\
&\|D\tilde{A}^{-1}\|_1 \le \frac{1}{1-\alpha}, \quad \|F\|_1 \le \omega, \\
&\|FC^{-1}H\|_1 \le \frac{\alpha}{1-\alpha}, \quad \|D^{-1}FC^{-1}H\|_1 \le \frac{\alpha}{1-\alpha}.
\end{aligned}$$

Proof. Recall that $A = I - \alpha P$ where $P \geq 0$, $\|P\|_1 \leq 1$ and $0 < \alpha < 1$. Thus A is strictly diagonally dominant by columns and $\min_k(|A(k,k)| - \sum_{i\neq k}|A(i,k)|) \geq 1-\alpha$. According to Lemma 5.2, $\|A^{-1}\|_1, \|\tilde{A}^{-1}\|_1 \leq 1/(1-\alpha)$. Because D of (8) is formed by changing some off-diagonal nonzero elements of $\tilde{A}$ to 0, D is also strictly diagonally dominant by columns and $\min_k(|D(k,k)| - \sum_{i\neq k}|D(i,k)|) \geq 1-\alpha$. Thus $\|D^{-1}\|_1 \leq 1/(1-\alpha)$.

H of (11) can be regarded as a sub-matrix of B_{off} that only contains some off-diagonal elements of $\tilde{A} = I - \alpha \Pi P \Pi^T$. Thus $\|H\|_1 \leq \|B_{off}\|_1 \leq \|\alpha \Pi P \Pi^T\|_1 \leq \alpha$. As $\tilde{A} = D + B_{off}$,

$$\|D\tilde{A}^{-1}\|_1 = \|(\tilde{A} - B_{off})\tilde{A}^{-1}\|_1 \leq \|I\|_1 + \|B_{off}\|_1\|\tilde{A}^{-1}\|_1 = \frac{1}{1-\alpha}.$$

Note that any hub formed by Algorithm 1 can contain at most ω rows. Then according to the definition (10) of F, we get $\|F\|_1 \leq \omega$.

For $C^{-1} = (I + HD^{-1}F)^{-1}$, we apply the Woodbury formula (Lemma 5.1) again and get

$$\begin{aligned} C^{-1} &= I - HD^{-1}(I + FHD^{-1})^{-1}F \\ &= I - H(D + FH)^{-1}F = I - H\tilde{A}^{-1}F. \end{aligned}$$

Then

$$FC^{-1}H = FH - FH\tilde{A}^{-1}FH = B_{off} - B_{off}\tilde{A}^{-1}B_{off},$$

and thus

$$\|FC^{-1}H\|_1 \leq \|B_{off}\|_1 + \|B_{off}\|_1\|\tilde{A}^{-1}\|_1\|B_{off}\|_1 = \alpha + \frac{\alpha^2}{1-\alpha} = \frac{\alpha}{1-\alpha}.$$

Finally, because $\tilde{A}^{-1} = (I - D^{-1}FC^{-1}H)D^{-1}$,

$$\|D^{-1}FC^{-1}H\|_1 = \|I - \tilde{A}^{-1}D\|_1 = \|\tilde{A}^{-1}B_{off}\|_1 \leq \|\tilde{A}^{-1}\|_1\|B_{off}\|_1 = \frac{\alpha}{1-\alpha}. \quad \square$$

Then the following theorem is obtained directly by combining Lemma 5.3 and formula (16).

Theorem 5.2. *The approximation error $\|M^{-1} - \tilde{A}^{-1}\|_1$ of the ODLR preconditioner M^{-1} to $\tilde{A}^{-1}$, and the errors $\|E_D\|_1$ and $\|E_C\|_1$ of solving inner systems when computing $M^{-1}u$ satisfy*

$$\|M^{-1} - \tilde{A}^{-1}\|_1 \leq \sigma_1\|E_D\|_1 + \sigma_2\|E_C\|_1 + \sigma_3\|E_D\|_1\|E_C\|_1 + \sigma_4\|E_D\|_1^2 + \sigma_5\|E_D\|_1^2\|E_C\|_1, \tag{17}$$

where

$$\sigma_1 = \frac{1+\alpha}{1-\alpha}, \quad \sigma_2 = \frac{\alpha\omega}{(1-\alpha)^2}, \quad \sigma_3 = \frac{2\alpha\omega}{1-\alpha}, \quad \sigma_4 = \frac{\alpha}{1-\alpha}, \quad \sigma_5 = \alpha\omega.$$

Based on this theorem, the spectral properties of $\tilde{A}M^{-1}$ can be described as below.

Corollary 5.1. *Let λ be an eigenvalue of $\tilde{A}M^{-1}$, then*

$$|\lambda - 1| \leq \eta_1\|E_D\|_1 + \eta_2\|E_C\|_1 + \eta_3\|E_D\|_1\|E_C\|_1 + \eta_4\|E_D\|_1^2 + \eta_5\|E_D\|_1^2\|E_C\|_1, \tag{18}$$

where $\eta_i = (1+\alpha)\sigma_i$, $i = 1, 2, 3, 4, 5$.

Proof. The eigenvalue problem $\tilde{A}M^{-1}u = \lambda u$ can be rewritten as $\tilde{A}(M^{-1} - \tilde{A}^{-1})u = (\lambda - 1)u$. Thus,

$$|\lambda - 1| \leq \rho(\tilde{A}(M^{-1} - \tilde{A}^{-1})) \leq \|\tilde{A}\|_1\|M^{-1} - \tilde{A}^{-1}\|_1.$$

Then based on Theorem 5.2 and $\|\tilde{A}\|_1 = \|A\|_1 = \|I + \alpha P\|_1 \leq 1 + \alpha$, this corollary is proved. $\square$

Theorem 5.2 and Corollary 5.1 give the following indications. By increasing the accuracies of solving the inner systems, the approximation error $\|M^{-1} - \tilde{A}^{-1}\|_1$ can be decreased to any degree, and the spectral properties of the preconditioned matrix $\tilde{A}M^{-1}$ can be improved to be finally comparable with those of I. Thus, the ODLR preconditioner M^{-1} theoretically can give any required acceleration effect on iterative methods and can be said well-defined. As the coefficients σ_i, η_i $(i = 1, 2, 3, 4, 5)$ of (17) and (18) increase with the damping factor α, the inner systems should be solved more accurately for larger α to maintain the same degree of the acceleration effect. In (17), the first-order terms $\sigma_1\|E_D\|_1$ and $\sigma_2\|E_C\|_1$ contribute mostly to $\|M^{-1} - \tilde{A}^{-1}\|_1$, and $\sigma_2 = \alpha\omega/(1-\alpha)^2$ is much larger and increases faster with α than $\sigma_1 = (1+\alpha)/(1-\alpha)$. Thus the inner systems with C should be solved more accurately than those with D, especially when α approaches 1.

Note that $C = I + HD^{-1}F$ is usually much denser than A, thus it is important for the efficiency of the ODLR preconditioner M^{-1} to make C relatively small. If C is sufficiently small such that the inner systems with $\bar{C} = I + H\tilde{D}^{-1}F$ can be solved very accurately or exactly by a small computational cost, then we can say $\tilde{C}^{-1} = \bar{C}^{-1}$. In this case, M^{-1} writes as

$$M^{-1} = \tilde{D}^{-1} - \tilde{D}^{-1}F(I + H\tilde{D}^{-1}F)^{-1}H\tilde{D}^{-1},$$

Table 2
Characteristics of the Web adjacency matrices G tested in our experiments. The symbol n is the dimension of the matrix, and nnz is the number of nonzero elements (listed in increasing matrix size).

Name	n	nnz	nnz/n^2
uk-2007-100000	100,000	3,050,615	3.1e−4
web-Stanford	281,903	2,312,497	2.9e−5
cnr-2000	325,557	3,216,152	3.0e−5
web-BerkStan	685,230	7,600,595	1.6e−5
in-2004	1,382,908	16,917,053	8.8e−6
indochina-2004	7,414,866	194,109,311	3.5e−6
wb-edu	9,845,725	57,156,537	5.9e−7
uk-2002-13m	13,000,000	193,986,531	1.1e−6

and we obtain the expression $M = (M^{-1})^{-1} = \tilde{D} + FH$. Accordingly, the eigenvalue problem $\tilde{A}M^{-1}u = \lambda u$ considered in Corollary 5.1 can be formulated as

$$(\tilde{A}M^{-1} - I)u = (\tilde{A} - M)M^{-1}u = (D - \tilde{D})M^{-1}u = (\lambda - 1)u.$$

Thus

$$|\lambda - 1| \leq \|(D - \tilde{D})M^{-1}\|_1 \leq \|(D - \tilde{D})\|_1 \|M^{-1}\|_1. \tag{19}$$

When a general-purpose preconditioner (denoted by ${M_A}^{-1}$) is applied to the PageRank system (2), the preconditioned system can be given by $A{M_A}^{-1}y = v$ with $x = M_A^{-1}y$. According to the derivation above, any eigenvalue $\lambda_{AM_A^{-1}}$ of AM_A^{-1} satisfies

$$|\lambda_{AM_A^{-1}} - 1| \leq \|(A - M_A)\|_1 \|M_A^{-1}\|_1. \tag{20}$$

For this preconditioner, such computation $\tilde{y} = {M_A}^{-1}u$ is implemented by (or regarded as) solving $Ay = u$ to get an approximation $\tilde{y}$ of y. Comparing (19) and (20) indicate that: if the computational costs of $M^{-1}u$ and ${M_A}^{-1}u$ are roughly the same, but the inner systems with D required by computing $M^{-1}u$ are solved more accurately than the linear systems with A required by computing ${M_A}^{-1}u$, then the ODLR preconditioner M^{-1} is likely to be more efficient than the general-purpose preconditioner. Recall that two inner systems with D need to be solved for computing $M^{-1}u$ while only one linear system with A needs to be solved for computing ${M_A}^{-1}u$. Therefore, the computational properties, e.g. sparsity and spectral properties, of D should be much more favorable for solvers than those of A to meet the assumption above.

According to (8), D is more sparse than $\tilde{A}$, and the degree of its sparsity will be increased if the splitting (7) drops more off-diagonal nonzero elements of $\tilde{A} = I - \Pi P \Pi^T$ into B_{off}. Denote this dropping operation by $drop()$, and then $D = I - drop(\Pi P \Pi^T)$. If operation $drop_1$ drops more nonzero elements than another operation $drop_2()$, then

$$\|drop_1(\Pi P \Pi^T)\|_1 \leq \|drop_2(\Pi P \Pi^T)\|_1 \leq 1.$$

Note that $\rho(U) \leq \|U\|_1$ for any matrix U. Thus if more off-diagonal nonzero elements of $\tilde{A}$ are dropped into B_{off}, the smallest eigenvalue of D whose modulus equals $1 - \rho(drop(\Pi P \Pi^T))$ may get better separated from 0. Besides, after such dropping process for $\tilde{A}$, the center of each Gershgorin disc [26] of $\tilde{A}$ keeps unchanged while the radius of some discs becomes smaller. Consequently, the eigenvalues of D are likely to be more clustered than those of A, especially when the dropping process drops more nonzero elements.

In conclusion, to achieve a high efficiency of the ODLR preconditioner or to let it outperform general-purpose preconditioners, probably a large proportion of the off-diagonal nonzero elements of $\tilde{A}$ should be dropped into B_{off}. Meanwhile, B_{off} should be compressed into a small number of rows in H, as this number equals the dimension of C. Thus, for the same purpose, a large compression ratio of the PageRank problem would be required. Besides, assigning a suitable value to the parameter θ for the partition method (Algorithm 1) is important. The reason is that: the size m of C is bounded above by the number of unmarked rows after the pre-marking step in Algorithm 1; $nnz(D)/nnz(A)$ would be roughly equal to θ when θ is not close to 0, as D of (8) contains all the elements of the pre-marked rows but only a small fraction of the elements of the remaining rows. Finally, it is also important to choose suitable solvers for efficiently solving the inner systems with D and those with C.

6. Numerical experiments

In this section, we report some numerical experiments that were carried out by a MATLAB R2015b implementation on a 64-bit Ubuntu computer equipped with an Intel core i5-6500 processor and 16 GB RAM memory. The test Web adjacency matrices G are extracted from the matrix repository of [27] and the Laboratory for Web Algorithmics [28–30]. Table 2 summarizes the characteristics of each G. For the PageRank linear system (2) corresponding to each G, we set the personalization vector $v = [1, 1, \ldots, 1]^T/n$, and set the damping factor $\alpha = 0.99$ that makes this system difficult to solve.

We first test the effect of the parameter θ on the performance of the matrix partition method (Algorithm 1). Then we discuss choices of solvers to solve the inner systems in the ODLR preconditioner. Finally, we present some comparison results between the ODLR preconditioner and several other methods for solving PageRank problems.

Table 3
Results of running the partition method. *Pmark* denotes the *proportion of the rows in G that are pre-marked*, T_p is the *time cost* (in seconds) of the partition method, and $C.R.$ represents the *compression ratio* defined in (12).

Problem & θ	*Pmark*	T_p	$C.R.$	$\frac{nnz(D)}{nnz(A)}$	$\frac{m}{n}$
`uk-2007-100000`					
0	0	3.34	36.1%	6.6%	45.8%
0.1	83.5%	1.08	33.7%	13.6%	12.5%
0.2	94.0%	0.77	32.6%	22.6%	3.7%
0.3	97.7%	0.64	29.8%	30.7%	1.5%
0.4	99.0%	0.61	27.7%	40.5%	0.6%
`cnr-2000`					
0	0	9.28	42.9%	14.2%	44.2%
0.1	53.1%	4.73	39.2%	19.2%	30.6%
0.2	81.7%	2.31	37.3%	26.3%	11.7%
0.3	93.5%	1.33	35.2%	34.5%	2.9%
0.4	97.4%	0.94	31.5%	41.7%	1.4%
`web-BerkStan`					
0	0	19.41	32.5%	13.3%	61.4%
0.1	55.8%	9.41	30.6%	18.7%	34.3%
0.2	80.6%	5.18	29.8%	26.3%	12.5%
0.3	92.2%	3.14	27.5%	34.7%	4.1%
0.4	97.3%	2.17	24.5%	43.0%	1.2%
`in-2004`					
0	0	41.00	45.5%	15.9%	46.8%
0.1	58.5%	19.10	42.1%	21.3%	27.7%
0.2	83.7%	9.88	40.6%	28.8%	8.7%
0.3	94.6%	5.73	38.0%	36.3%	2.3%
0.4	98.2%	4.18	34.4%	44.3%	0.6%

6.1. The effect of the parameter θ on the partition method

We test four PageRank problems constructed from the Web adjacency matrices: `uk-2007-100000`, `cnr-2000`, `web-BerkStan` and `in-2004`. For each problem, we run the matrix partition method (Algorithm 1) with $\theta = 0, 0.1, 0.2, 0.3, 0.4$. The results are reported in Table 3.

The following observations are drawn from Table 3. When $\theta = 0$, no row of G is pre-marked, thus the partition method deals with all the rows of G and achieves a very large compression ratio (refer to the values of $C.R.$), but meanwhile it becomes time-consuming (See T_p). Besides, $\theta = 0$ leads to the most sparse D (refer to $nnz(D)/nnz(A)$) but also causes the largest capacitance matrix C ($m/n > 44\%$). Thus $\theta = 0$ is not an advisable choice for either controlling the time cost of the partition method or achieving a high efficiency of the ODLR preconditioner. When θ increases from 0 to 0.1, more than 50% of the rows of G are pre-marked. As a result, the time cost of the partition method and the dimension of C decrease significantly while the compression ratio $C.R.$ and the sparsity of D decrease marginally. Such trend continues when θ increases from 0.1 to 0.2, from 0.2 to 0.3 and from 0.3 to 0.4. Besides, the results in Table 3 also verify the previous deduction that the dimension m of C is bounded from above by the amount of the unmarked rows after the pre-marking step, i.e. $m/n \leq 1 - Pmark$, and $nnz(D)/nnz(A)$ roughly equals θ when θ is not close to 0. Based on these results, setting $0.2 \leq \theta \leq 0.3$ is reliable for the partition method to make the compression ratio close to the largest value that can be achieved, to let D have a good level of sparsity and the dimension of C be relatively small, by a moderate computational cost. We set $\theta = 0.3$ in the following experiments.

6.2. Discussion of solvers to solve inner systems with D and those with C

As previously demonstrated, the partition method with a suitable value of θ may well lead to a very sparse D and a very small C, relative to A. For the ODLR preconditioner in this situation, we first focus on the choices of solvers for solving the inner systems with D, since $C = I + HD^{-1}F$ is very small and the construction of it depends on the selected solver for D^{-1} or said for the inner systems with D. Unlike solving the PageRank system (2) with A, direct solvers are applicable for solving the inner systems with D, because D is much more sparse than A and these systems are usually not required to be solved very accurately. Iterative solvers are clearly another type of applicable choices. Recall that D is a M-matrix and strictly diagonally dominant by columns, and its spectral properties would be more favorable for iterative methods, compared to A. Thus, Krylov subspace methods such as the Generalized Minimum Residual Method (GMRES) [31] may be a good choice for solving the inner systems with D iteratively, as their convergence rates depend heavily on the spectral properties of linear systems. Meanwhile, Incomplete LU (ILU) factorization algorithms are generally robust and efficient for directly solving the linear systems with diagonally dominant M-matrices that are very sparse [32,33]. Thus, such algorithms

Table 4
ILUPACK (in short, ILU) and GMRES (20) (in short, GME) for solving $Dy = b$ with $b = [1, \dots, 1]^T$ until obtaining a solution $\tilde{y}$ that satisfies: $\|D\tilde{y}-b\|_2/\|b\|_2 \leq 10^{-1}, 10^{-2}, 10^{-4}$. LU_D denotes the *ILU factors* of D, *Iter* is the *number of iterations*, T_f and T_s are the *time costs* (in seconds) required by the ILU factorization and the solving phase, respectively, and $T_{total} = T_f + T_s$ is the *total time*.

$\|D\tilde{y} - b\|_2/\|b\|_2$	$\frac{nnz(LU_D)}{nnz(A)}$	*Iter*	T_f	T_s	T_s	T_{total}	T_{total}
uk-2007-100000	ILU	GME	ILU	ILU	GME	ILU	GME
10^{-1}	0.37	19	0.70	0.02	0.18	0.72	0.18
10^{-2}	0.46	39	0.77	0.02	0.40	0.79	0.40
10^{-4}	0.64	77	0.88	0.02	0.92	0.90	0.92
cnr-2000							
10^{-1}	0.44	35	0.92	0.02	1.47	0.94	1.47
10^{-2}	0.53	82	0.93	0.03	3.63	0.96	3.63
10^{-4}	0.57	163	0.96	0.03	7.12	0.99	7.12
wb-berk							
10^{-1}	0.48	75	2.22	0.06	8.39	2.28	8.39
10^{-2}	0.56	230	2.34	0.07	25.51	2.41	25.51
10^{-4}	0.59	571	2.33	0.07	63.88	2.40	63.88
in-2004							
10^{-1}	0.42	36	5.15	0.12	7.86	5.27	7.86
10^{-2}	0.52	95	5.20	0.14	20.93	5.34	20.93
10^{-4}	0.55	202	5.38	0.15	45.19	5.43	45.19

may be another good choice. We test both options in MATLAB, using the built-in implementation of GMRES (20),[1] and the ILUPACK implementation proposed by Bollhöfer and Saad [34] for ILU factorizations.

We test the same set of Web adjacency matrices G as the previous subsection. For the PageRank problem corresponding to each G, we implement the partition method and the off-diagonal low-rank factorization to get the matrix D. Then for each D, we solve the system $Dy = b$ where $b = [1, \dots, 1]^T$ until obtaining an approximate solution $\tilde{y}$ that satisfies: $\|D\tilde{y} - b\|_2/\|b\|_2 \leq 10^{-1}, 10^{-2}, 10^{-4}$. The numerical results are reported in Table 4.

Seen from Table 4, for the first also the smallest problem, the total time cost (T_{total}) of GMRES (20) is less than that of ILUPACK for reducing $\|D\tilde{y} - b\|_2/\|b\|_2$ to 10^{-1}, but this advantage decays rapidly when higher accuracies 10^{-2} and 10^{-4} are required. For the last three problems, ILUPACK requires significantly less total time than GMRES (20) in each case, especially in the cases where higher accuracies of $\tilde{y}$ are required. Moreover, the solving time (T_s) of ILUPACK is always much smaller than that of GMRES (20). Because a number of inner systems with D will be solved during solving the preconditioned system, the smaller solving time of ILUPACK becomes a large advantage over GMRES (20) here. Besides, the values of $nnz(LU_D)/nnz(A)$ in Table 4 show that: solving the systems with D causes just moderate memory costs even when highly accurate ILU factorizations are implemented. Overall, in these experiments, ILUPACK performs as a better choice than GMRES (20) for solving linear systems with matrices D.

The implemented ILU factorization of D can be reused to compute the approximate capacitance matrix $\bar{C} = I + H\tilde{D}^{-1}F$. The comparison results of applying ILUPACK and GMRES (20) for solving linear systems with $\bar{C}$ are very similar as the previous results of solving systems with D. Thus we omit these results here. In conclusion, direct methods such as the ILU factorization algorithms may well be better choices than iterative methods such as GMRES for solving the inner systems. Hereafter, we use ILUPACK for solving the inner systems in the ODLR preconditioner.

6.3. ODLR preconditioner against some other methods for solving PageRank problems

We first compare the performances of the ODLR preconditioner, ILUPACK as a preconditioner, and the elimination strategy [21] combined with the ILUPACK preconditioner. The iterative solver is GMRES (20). Because ILUPACK is also applied to solve the inner systems in the ODLR preconditioner, comparing the ILUPACK and the ODLR preconditioners can intuitively demonstrate some behaviors of the ODLR preconditioner (as a preconditioning strategy). The elimination strategy also exploits properties of PageRank problems and inspires the development of the ODLR preconditioner, thus we add it into the comparison here. Note that because the elimination strategy implements operations of subtracting some rows from some other rows in A, the resulting matrix loses the property of 2.1. Thus an effective combination of this strategy and the ODLR preconditioner cannot be achieved by simple ways and needs further study.

The memory cost (denoted by *mem*) is recorded as

$$mem = \begin{cases} \dfrac{nnz(LU_D) + nnz(LU_C) + nnz(H) + nnz(F)}{nnz(A)} & \text{for the ODLR preconditioner,} \\ \dfrac{nnz(LU_A)}{nnz(A)} & \text{for the ILUPACK preconditioner,} \\ \dfrac{nnz(LU_{A_E})}{nnz(A)} & \text{for the elimination strategy,} \end{cases} \tag{21}$$

[1] The conventional GMRES algorithm restarted every 20 iterations.

Table 5
Numerical results of solving PageRank problems with $\alpha = 0.99$. $C.R.$ represents the *compression ratio* defined in (12), *mem* denotes the *memory cost* recorded as (21), *Iter* is the *number of iterations*, T_p, T_f and T_s are the *time costs* (in seconds) required by the partition method (or the elimination process), the ILU factorization process and the solving process respectively, T_{total} is the *total time cost*, '#' means *convergence is not reached*, and '–' means *no corresponding index* in the given run.

Problem & Method	$C.R.$	*Mem*	*Iter*	T_p	T_f	T_s	T_{total}
`uk-2007-100000`							
ODLR	**29.83%**	**1.02**	**7**	**0.63**	**1.73**	**0.22**	**2.58**
ILUPACK	–	1.04	90	–	28.28	1.77	30.05
Elimination	–	0.20	47	0.32	6.53	0.73	7.58
`web-Stanford`							
ODLR	**14.04%**	**1.55**	**6**	**1.48**	**1.80**	**0.28**	**3.56**
ILUPACK	–	1.56	10	–	6.33	0.42	6.75
Elimination	–	0.85	16	1.31	3.38	0.61	5.30
`cnr-2000`							
ODLR	**35.20%**	**1.00**	**7**	**1.03**	**1.27**	**0.32**	**2.62**
ILUPACK	–	1.04	66	–	7.12	2.26	9.38
Elimination	–	0.38	23	0.74	2.19	0.74	3.67
`web-BerkStan`							
ODLR	**27.50%**	**1.15**	**6**	**2.32**	**6.31**	**0.68**	**9.31**
ILUPACK	–	1.13	74	–	48.09	6.29	54.38
Elimination	–	0.42	19	1.85	9.01	1.48	12.34
`in-2004`							
ODLR	**38.00%**	**0.88**	**6**	**4.17**	**6.23**	**1.24**	**11.64**
ILUPACK	–	0.89	148	–	29.75	22.96	52.71
Elimination	–	0.14	19	3.19	11.01	2.61	16.81
`indochina-2004`							
ODLR	**48.10%**	**0.71**	**41**	**33.55**	**64.88**	**66.43**	**164.86**
ILUPACK	–	0.71	162	–	695.57	217.87	913.43
Elimination	–	0.20	24	88.36	258.08	30.05	376.49
`wb-edu`							
ODLR	**18.20%**	**1.62**	**7**	**48.64**	**44.42**	**9.15**	**102.21**
ILUPACK	–	1.58	38	–	90.72	39.99	130.71
Elimination	–	1.16	11	41.15	79.43	10.74	131.32
`uk-2002-13m`							
ODLR	**38.70%**	**0.85**	**15**	**41.48**	**76.64**	**51.19**	**169.31**
ILUPACK	–	0.85	232	–	287.53	525.88	813.41
Elimination	–	0.34	9	116.27	160.39	62.90	339.56

where LU_D, LU_C, LU_A and LU_{A_E} are the ILU factors of D, C, A, and the coefficient matrix A_E of the PageRank system eliminated by the strategy [21], respectively. We compare the ODLR and the ILUPACK preconditioners in the situation that their memory costs are roughly the same. Because the eliminated PageRank system is often much more sparse than the original system (2), the memory cost of the elimination strategy is often much smaller than those of the other two preconditioners, even if the eliminated system is preconditioned by very accurate ILU factorizations.

Then we also put into comparisons the Power method and some recently proposed methods: the algorithm proposed by [11] that combines the Arnoldi method and an extrapolation technique (hereafter, Ext-Arnoldi), the multi-step splitting iteration (hereafter, M-Iter) method from [12] and the preconditioned and extrapolation-accelerated GMRES (hereafter, PEGMRES) from [18].

Each iterative process is started from the initial guess $x_0 = [0, 0, \dots, 0]^T$ and is stopped when either $\|Ax_i - v\|_2/\|v\|_2 \leq 10^{-8}$, where x_i is the latest approximate solution, or the number of iterations reaches 5000. The numerical results are presented in Tables 5 and 6.

Seen from Table 5, a large compression ratio ($C.R.$) is achieved by the ODLR preconditioner for most tested problems. Accordingly, the matrix D would be much more sparse than A, and the matrix C would be very small. As a result, when the ODLR and the ILUPACK preconditioners cause similar memory costs, the ILU factorizations of D and C in the ODLR preconditioner would be much more accurate than those of A in the ILUPACK preconditioner. Therefore, compared with the ILUPACK preconditioner, the ODLR preconditioner would let the spectral properties of the preconditioned system more favorable for iterative methods, according to the analysis in Section 5.2. Thus, as Table 5 shows, the ODLR preconditioner converges in much fewer iterations (*Iter*) and less time (T_s) than the ILUPACK preconditioner. Besides, the time cost (T_f) required by the ILU factorizations in the ODLR preconditioner is also remarkably smaller than that in the ILUPACK preconditioner. As the time cost (T_p) for the partition process is moderate, the ODLR preconditioner finally shows a much better overall performance (T_{total}) than the ILUPACK preconditioner. The overall performance (T_{total}) of the elimination strategy is better than that of the ILUPACK preconditioner, thanks to the high sparsity of the eliminated PageRank systems. However, this strategy is not competitive with the ODLR preconditioner in time costs T_f, T_s and T_{total}. In Table 6, the comparison results of all the tested methods show that: the total time (T_{total}) required by the ODLR preconditioner is generally

Table 6
Numerical results of solving PageRank problems with $\alpha = 0.99$. *MV* denotes the *number of matrix–vector multiplications* that have been computed for achieving convergence, T_{total} is the *total time cost* (in seconds)

Method	*MV*	T_{total}	*MV*	T_{total}	*MV*	T_{total}	*MV*	T_{total}
	`uk-2007-100000`		`web-Stanford`		`cnr-2000`		`web-BerkStan`	
ODLR	–	**2.58**	–	**3.56**	–	**2.62**	–	**9.31**
Power	1976	4.70	2150	7.74	2142	7.83	2116	19.93
M-Iter	1824	4.25	1879	5.91	1904	7.09	1886	17.05
PEGMRES	864	3.12	1078	4.74	941	4.94	1293	13.11
Ext-Arnoldi	650	2.03	1100	4.28	700	3.44	1300	13.00
ILUPACK	–	30.05	–	6.75	–	9.38	–	54.38
	`in-2004`		`indochina-2004`		`wb-edu`		`uk-2002-13m`	
ODLR	–	**11.64**	–	**164.86**	–	**102.21**	–	**169.31**
Power	2120	39.91	2070	380.67	1999	190.00	2096	466.03
M-Iter	1993	38.29	1903	355.73	1860	191.02	2047	469.40
PEGMRES	1089	24.79	1324	288.39	1523	162.81	1450	348.47
Ext-Arnoldi	840	17.21	1430	293.91	1560	150.59	1440	336.23
ILUPACK	–	52.71	–	913.43	–	130.71	–	813.41

much smaller than those required by other tested methods, when solving each of these difficult PageRank problems. We consider that the results in Tables 5 and 6 have demonstrated the potential of the ODLR preconditioner for accelerating the computations of difficult PageRank problems.

7. Conclusions

In this paper, we have defined a special set of rows (named core hub of rows) of the PageRank coefficient matrix A. We demonstrated that A can be transferred into a block form $\tilde{A}$ by partitioning its rows into core hubs, and the rows of each off-diagonal block in $\tilde{A}$ are identical. Then we present a low-rank factorization for compressing the off-diagonal blocks that are relatively dense, and thus $\tilde{A}$ is formulated as $\tilde{A} = D + FH$. This formulation is beneficial for inverting $\tilde{A}$ if the low-rank factorization compresses a large proportion of the nonzero elements of A such that D is much more sparse and the ranks of F and H are much smaller, relative to A. Hence, we propose a matrix partition method to discover core hubs of rows of A for supporting the low-rank factorization. Then a preconditioner based on the formulation $\tilde{A} = D + FH$ is proposed. Numerical experiments demonstrate that: the matrix partition method can generally discover quality core hubs of rows of A efficiently such that a large compression ratio is achieved, and meanwhile D is with high sparsity, and F and H have very small ranks; direct solvers such as ILU factorization methods are suitable choices for solving the inner systems in the proposed preconditioner; finally the proposed preconditioner generally shows better performance than some other preconditioners and methods for solving difficult PageRank problems.

Acknowledgments

The authors would like to express sincere gratitude to the anonymous referees and editor Prof. Michael Ng for their insightful comments and suggestions that improved this paper. This research is supported by NSFC (61772003, 11501085).

Appendix A. The proof of Property 2.1

We first introduce the following lemma.

Lemma A.1 (*Property 2.1 in [21]*). *In P of (1), nonzero entries of each column are positive and have the same value.*

Suppose the set of row indexes of the rows in S is N_s. Let $R = \{1 : n\}$, then the off-diagonal part of S can be written as $A(N_s, R\backslash N_s)$. Because $A = I - \alpha P$, it is clear that $A(N_s, R\backslash N_s)$ is a sub-matrix of αP.

For any two rows row_1 and row_2 of $A(N_s, R\backslash N_s)$, their zero-nonzero patterns must be the same, because S is a core hub satisfying Definition 2.2. Therefore the nonzero elements of row_1 are distributed at the same set of columns as the nonzero elements of row_2. Meanwhile, based on Lemma A.1, any nonzero element of row_1 and any nonzero element of row_2 must have the same value if they are in the same column. Thus $row_1 = row_2$. That is, the rows in the off-diagonal part of S are all identical.

Appendix B. Motivations and effects of the amendments for the partition method

Amendment 1. Suppose a core hub hub_i $(1 \leq i \leq n)$ has n_i rows, and each row of its off-diagonal part has d nonzero entries. If the off-diagonal block row corresponding to this hub is compressed by the low-rank factorization (10)–(11), actually this hub increases the compression ratio (12) by $(n_i - 1)d/nnz(A)$ since the off-diagonal block row is compressed into one single row. We denote the amount of the nonzero entries of each row $G(j,:)$ $(1 \le j \le n)$ by $rownnz(j)$. Because $d \le min_{G(j,:)\in hub_i} nnz(G(j,:))$, rows with small $rownnz$ values will contribute little to the compression ratio. Hence, after the initial setting (step 1), we mark the rows of G and their hubs in a $rownnz$ increasing order until the amount of nonzeros in the marked rows reaches $\theta nnz(G)$, where $0 < \theta < 1$. We call this step as pre-marking step. For example, if θ is set as 25%, the partition method indeed only deals with the relatively dense rows that contain about 75% of the nonzero entries of G and contribute most to the compression ratio. Because the pre-marked rows are more sparse than the remaining rows, a large proportion of rows can be pre-marked and be skipped during the remaining process of the partition method, even with a tiny θ. Therefore, applying this pre-making step would reduce the time cost of the partition method, without decreasing much the compression ratio. Meanwhile, the size m of F and H will be bounded from above by the amount of the unmarked rows after this pre-marking step. Suppose s rows of G are pre-marked. The corresponding rows of A can be permuted to the top and be regarded as the prior s block rows of $\tilde{A}$ that should be contained in D (8) and not be compressed.

Amendment 2. In the Web link graph, few pages can contain more than 50 hyperlinks. Thus few columns of G can get more than 50 nonzero elements. As a result, each core hub of rows of G is unlikely to contain more than 50 rows. Moreover, after the pre-marking step, only the unmarked rows, which are relatively dense, will be traversed by the inner traversals to enlarge hubs. Thus, indeed, each set of rows that can be classified into a core hub is listed closer with each other in the inner traversals. Based on these factors, we only traverse at most $\omega = 100$ unmarked rows in each inner traversal. This amendment makes the length of each inner traversal be smaller than 100, and thus will decrease the time cost of the partition method. Moreover it will not worsen much the quality of the core hubs that will be discovered.

Amendment 3. When we judge whether an unmarked row $G(j,:)$ $(1 \le j \le n)$ could be added to a hub hub_i $(1 \le i \le n)$, we have to check whether the rows in the off-diagonal part of $hub_i \cup G(j,:)$ have the same zero-nonzero patterns. This check can be done by an element-wise comparison for the rows of $hub_i \cup G(j,:)$, however, which is costly. Hence we employ a strategy to deal with this problem. Let $R = 1 : n$ and denote the set of the row indexes of hub_i $(1 \le i \le n)$ by N_i. For each inner traversal related with hub_i $(1 \le i \le n)$, we compute a value $key(j)$ for each visited unmarked row $G(j,:)$ $(i+1 < j < min(n, i+\omega))$ by

$$key(j) = \sum_{G(j,k)=1} k - \sum_{G(i,k)=1} k \text{ with } k \in \{R \backslash \{N_i \cup j\}\}. \tag{22}$$

If $key(j) \ne 0$, then $G(j,:)$ should not be added to hub_i. Otherwise, we implement an element-wise comparison for the nonzero patterns of the rows of $hub_i \cup G(j,:)$ to judge whether it is a core hub. This amendment generally can decrease the times of implementing element-wise comparison of the nonzero patterns of different rows without affecting the information of the core hubs that will be discovered.

References

[1] C.-Q. Gu, F. Xie, K. Zhang, A two-step matrix splitting iteration for computing PageRank, J. Comput. Appl. Math. 278 (2015) 19–28.
[2] D.F. Gleich, PageRank beyond the web, SIAM Rev. 57 (2015) 321–363.
[3] A.N. Langville, C.D. Meyer, A reordering for the PageRank problem, SIAM J. Sci. Comput. 27 (2006) 2112–2120.
[4] A.N. Langville, C.D. Meyer, Google's Pagerank and Beyond: The Science of Search Engine Rankings, Princeton University Press, Princeton, NJ, USA, 2006.
[5] W. Zhang, W. Tong, Z. Chen, R. Glowinski, Current Trends in High Performance Computing and Its Applications, Springer Press, Berlin, Germany, 2005.
[6] P. Boldi, M. Santini, A deeper investigation of PageRank as a function of the damping factor: In Dagstuhl Seminar Proceedings of Web Information Retrieval and Linear Algebra Algorithms, 2007.
[7] G.H. Golub, C. Greif, An Arnoldi-type algorithm for computing PageRank, BIT Numer. Math. 46 (2006) 759–771.
[8] S.D. Kamvar, T.H. Haveliwala, G.H. Golub, Adaptive methods for the computation of the PageRank, Linear Algebra Appl. 386 (2004) 51–65.
[9] 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.
[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] 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.
[13] 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.
[14] G. Wu, Y.-M. Wei, An Arnoldi-Extrapolation algorithm for computing PageRank, J. Comput. Appl. Math. 234 (2010) 3196–3212.
[15] G. Wu, Y.-M. Wei, Arnoldi versus GMRES for computing PageRank: A theoretical contribution to Google's PageRank problem, ACM Trans. Inf. Syst. 28 (3) (2010), Article 11.
[16] D.F. Gleich, L. Zhukov, P. Berkhin, Fast Parallel Pagerank: A Linear System Approach, Yahoo! Tech. Rep., 2005.
[17] P. Berkhin, A survey on PageRank computing, Internet Math. 2 (2005) 73–120.
[18] B.-Y. Pu, T.-Z. Huang, C. Wen, A preconditioned and extrapolation-accelerated GMRES method for PageRank, Appl. Math. Lett. 37 (2014) 95–100.
[19] G. Wu, Y.-C. Wang, X.-Q. Jin, A preconditioned and shifted GMRES algorithm for the PageRank problem with multiple damping factors, SIAM J. Sci. Comput. 34 (2012) 2558–2575.
[20] G.M. Del Corso, A. Gullí, F. Romani, Comparison of Krylov subspace methods on the PageRank problem, J. Comput. Appl. Math. 210 (2007) 159–166.
[21] 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.
[22] The MathWorks, Inc., MATLAB Reference Guide, Natick, USA, 1992.
[23] W.W. Hager, Updating the inverse of a matrix, SIAM Rev. 31 (1989) 221–239.
[24] C. Wang, H. Li, D. Zhao, An explicit formula for the inverse of a pentadiagonal Toeplitz matrix, J. Comput. Appl. Math. 278 (2015) 12–18.
[25] J.M. Varah, A lower bound for the smallest singular value, Linear Algebra Appl. 11 (1975) 3–5.
[26] S. Gerschgorin, Über die Abgrenzung der Eigenwerte einer Matrix, Izv. Akad. Nauk. USSR Otd. Fiz.-Mat. Nauk 6 (1931) 749–754.

[27] T.A. Davis, Y. Hu, The University of Florida sparse matrix collection, ACM Trans. Math. Software 38 (1) (2011) 1:1–1:25. Available at the URL: http://www.cise.ufl.edu/research/sparse/matrices.
[28] 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.
[29] 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.
[30] P. Boldi, B. Codenotti, M. Santini, S. Vigna, Ubicrawler: A scalable fully distributed Web crawler, Softw. - Pract. Exp. 34 (2004) 711–726.
[31] 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.
[32] M. Benzi, Preconditioning techniques for large linear systems: A survey, J. Comput. Phys. (182) (2002) 418–477.
[33] J. Diaz, Mathematics for Large Scale Computing, CRC Press, 1989.
[34] M. Bollhöefer, Y. Saad, O. Schenk, ILUPACK — preconditioning software package, 2010. Available online at http://ilupack.tu-bs.de/. Release V2.3, December 2010.