University of Groningen

Block-accelerated aggregation multigrid for Markov chains with application to PageRank problems

Shen, Zhao-Li; Huang, Ting-Zhu; Carpentieri, Bruno; Wen, Chun; Gu, Xian-Ming

Published in:
Communications in nonlinear science and numerical simulation

DOI:
[10.1016/j.cnsns.2017.11.031](https://doi.org/10.1016/j.cnsns.2017.11.031)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2018

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Shen, Z-L., Huang, T-Z., Carpentieri, B., Wen, C., & Gu, X-M. (2018). Block-accelerated aggregation multigrid for Markov chains with application to PageRank problems. *Communications in nonlinear science and numerical simulation*, 59, 472-487. <https://doi.org/10.1016/j.cnsns.2017.11.031>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Research paper

Block-accelerated aggregation multigrid for Markov chains with application to PageRank problems

Zhao-Li Shen^{a,b}, Ting-Zhu Huang^{a,*}, Bruno Carpentieri^c, Chun Wen^a,
Xian-Ming Gu^{b,d}

^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, Groningen 9700 AK, The Netherlands

^c School of Science and Technology, Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, 39100 Italy

^d School of Economic Mathematics, Southwestern University of Finance and Economics, Chengdu, Sichuan 611130, PR China

ARTICLE INFO

Article history:

Received 8 January 2015

Revised 22 November 2017

Accepted 28 November 2017

Available online 6 December 2017

Keywords:

Aggregation multigrid

Block-Jacobi

Markov chain

PageRank

ABSTRACT

Recently, the adaptive algebraic aggregation multigrid method has been proposed for computing stationary distributions of Markov chains. This method updates aggregates on every iterative cycle to keep high accuracies of coarse-level corrections. Accordingly, its fast convergence rate is well guaranteed, but often a large proportion of time is cost by aggregation processes. In this paper, we show that the aggregates on each level in this method can be utilized to transfer the probability equation of that level into a block linear system. Then we propose a Block-Jacobi relaxation that deals with the block system on each level to smooth error. Some theoretical analysis of this technique is presented, meanwhile it is also adapted to solve PageRank problems. The purpose of this technique is to accelerate the adaptive aggregation multigrid method and its variants for solving Markov chains and PageRank problems. It also attempts to shed some light on new solutions for making aggregation processes more cost-effective for aggregation multigrid methods. Numerical experiments are presented to illustrate the effectiveness of this technique.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Markov chains have been applied in many areas, including performance modelling of communication systems [1], Web page ranking [2,3] and gene ranking [4], blood screening procedures [5], etc. In this study, we are concerned with computing stationary distributions of homogeneous irreducible Markov chains.

For a homogeneous Markov chain with n states, the transition probability that it transfers from each state j to each state i is independent on the time step and can be denoted by $b_{i,j}$. Then the probabilistic behavior of this Markov chain is completely determined by the transition matrix $B = \{b_{i,j}\} \in \mathbb{R}^{n \times n}$ [6]. It is a column-stochastic matrix, i.e.,

$$0 \leq b_{i,j} \leq 1 \text{ for } 1 \leq i, j \leq n$$

and

$$\epsilon^T B = \epsilon^T,$$

* Corresponding author.

E-mail addresses: szlxiaoyao@163.com (Z.-L. Shen), tingzhu Huang@126.com, z.shen@rug.nl (T.-Z. Huang).

where $\epsilon = [1, 1, \dots, 1]^T \in \mathbb{R}^{n \times 1}$. The stationary distribution of this Markov chain is mathematically defined as the vector $x \in \mathbb{R}^{n \times 1}$ that satisfies

$$Bx = x, \quad x \geq 0 \quad \text{and} \quad \|x\|_1 = 1. \quad (1)$$

If B is irreducible, then this Markov chain is said irreducible and has a unique stationary distribution x [7]. Now we focus on the computation of x . More detailed descriptions of Markov chains can be found in [1,8].

Eigenproblem (1) is often reformulated as the singular linear problem:

$$Ax = 0, \quad x \geq 0 \quad \text{and} \quad \|x\|_1 = 1, \quad (2)$$

where $A = I - B$, and I is the identity matrix in $\mathbb{R}^{n \times n}$. The problem matrix A is an irreducible singular M -matrix, which is amenable to most solvers designed for general linear problems [9]. In many applications, due to the large sizes (numbers of states) of the Markov chains, direct methods are impractical for solving system (2). Instead, iterative procedures must be applied. Traditional one-level iterative methods, e.g. the Power and the weighted Jacobi methods, converge very slowly when the subdominant eigenvalue λ_2 of B is close to 1 [1,10]. However, for many Markov chains of interest, λ_2 approaches 1 when the problem size increases [9]. Thus more efficient iterative methods rather than traditional one-level algorithms are needed for such problems.

In 1970s, the iterative aggregation/disaggregation (IAD) method [11] has been proposed for solving Markov chains. At each iterative step, it corrects the error left by a selected one-level method through solving a coarse-level equation on aggregated degrees of freedom. This method is quite effective for nearly completely decomposable Markov chains [7] and has received much attention. Some multilevel extensions proposed by Horton and Leutenegger [12] and Leutenegger and Horton [13] generalize this method to aggregation multigrid methods. These methods perform significantly better than one-level methods and two-level methods in some Markov chain applications, however, their convergence properties are still often unsatisfactory [7]. The reason is probably that the generated aggregates usually cannot support efficient coarse-level corrections. Recently, the adaptive algebraic aggregation multigrid is developed in [14] for solving Markov chains. On each level, this method forms aggregates based on a scaled matrix, in which the columns of the problem matrix are scaled with the current approximate solution on that level [14]. Moreover, it updates the generated aggregates at every iterative step. As a result, the aggregates support high efficiencies of coarse-level corrections throughout the iterative process, and this method shows good potential for solving large Markov chains. Following this idea, several variants of this method have been successfully proposed. These variants seek to improve the efficiency of this method by smoothing the restriction and the interpolation operators [7], generating a better approximate solution from several old iterates on all recursive levels [15] or just on the top level [9,16], and dynamically updating the aggregates [17,18].

It is clear that updating the generated aggregates during the iterative process makes the adaptive algebraic aggregation multigrid powerful. However, there is a fly in the ointment: such updating operations often cost a considerable proportion of the running time of this method [18]. This defect motivates us to further utilize the generated aggregates. Indeed, the performance of any multigrid method relies not only on coarse-level corrections but also on the efficiency of the applied one-level method (called as the relaxation method in a multigrid context) for smoothing error [19]. Hence, we try to utilize the aggregates in the adaptive algebraic aggregation multigrid to develop a relaxation method that is more efficient than the traditional one-level algorithms. We show that the aggregates on each level can be used to transfer the probability equation of that level into a block linear system. In particular, the block matrix of this system will be nearly block diagonally dominant, and thus the additive Schwarz method is expected to get higher efficiency than point-wise methods for solving it [20]. In this study, we choose the Block-Jacobi method (a special case of the additive Schwarz method) as the basis to design the relaxation algorithm that deals with the block system to smooth error. This relaxation algorithm is shown to be well-defined in the adaptive algebraic aggregation multigrid method [14] and its smoothed version [7] for solving Markov chains. The resulting aggregation multigrid methods are also adapted to solve PageRank problems.

The remainder of this paper is organized as follows. In Section 2, we briefly review the adaptive algebraic aggregation multigrid for solving Markov chains [14] and the smoothed version of it [7]. In Section 3, we introduce the motivation and the implementation of building the Block-Jacobi relaxation based on aggregates, and analyse the convergence and the efficiency of this algorithm. Numerical tests are presented in Section 4. Finally, conclusions are outlined in Section 5.

Note on notations. We use MATLAB [21] notations to represent subvectors, submatrices, permutations for matrices and vectors, diagonal matrices constructed from vectors. u denotes an arbitrary column vector with appropriate dimension.

2. Adaptive algebraic aggregation multigrid and its smoothed version for Markov chains

For solving large linear systems from various fields including Markov chains, simple one-level iterative methods like the Power, the weighted Jacobi and the weighted Gauss–Seidel methods often can quickly reduce the oscillatory components of the error, leaving an error that is slow to be resolved [19]. Such error is called the algebraically smooth error. Multigrid schemes have been proved as a promising way to accelerate the elimination of the algebraically smooth error and improve the efficiencies of such one-level methods (referred to as relaxation methods in a multigrid context).

In [14], the adaptive algebraic aggregation multigrid (referred to as AM) is proposed for computing the stationary distribution x of the Markov chain problem (2). This method starts from an understanding that: in such a computation, the algebraically smooth error of an approximate solution x_i is nearly constant in local neighborhoods, where the corresponding

row entries of the scaled matrix $\bar{A} = \text{Adiag}(x_i)^{-1}$ are relatively large [14]. Based on this understanding, the strengths of the connections between any two degrees of freedom are defined by the relative sizes of the entries in the corresponding rows of $\bar{A}$. Accordingly, the strongly connected degrees of freedom are expected to get similar values in the error of x_i . Hence, the multilevel hierarchies and the interlevel operators of this method are constructed based on aggregation processes that classify degrees of freedom into disjoint aggregates under the principle: strongly connected ones should be in the same aggregate. The results of an aggregation process can be represented by the aggregation matrix $Q = \{q_{i,j}\}$, where $q_{i,j} = 1$ if the degree of freedom i belongs to the j th aggregate, and $q_{i,j} = 0$ otherwise. Note that, as the approximate solution x_i varies after each iteration step, new aggregation processes are implemented in the next iteration step, and the multilevel operators are updated adaptively. With these knowledge, the framework of a two-level iteration step (also called cycle) of this method can be introduced as below.

On the fine-level, some iteration steps of a selected relaxation method are applied to linear system (2) to get the latest approximate solution x_i . This process is called pre-relaxation. Accordingly, linear system (2) can be written as an error equation $\text{Adiag}(x_i)e_i = 0$, where e_i is the multiplicative error vector² whose entries all equal to 1 if convergence is achieved.

Then, an aggregation process is implemented based on the values in $\bar{A} = \text{Adiag}(x_i)$ to generate the aggregation matrix Q . When Q is constructed, the error equation of the coarse-level is given by

$$Q^T \text{Adiag}(x_i) Q e_c = 0, \quad (3)$$

where e_c denotes the coarse-level multiplicative error, and $Q e_c \approx e_i$. Defining the restriction operator R , the interpolation operator P and the coarse-level operator A_c as $R = Q^T$, $P = \text{diag}(x_i)Q$ and $A_c = RAP$ respectively, Eq. (3) can be written as $A_c e_c = 0$. Meanwhile, Rx_i restricts the fine-level approximate solution x_i onto the coarse-level. Thus the coarse-level accurate solution x_c is related to the coarse-level error e_c by $x_c = \text{diag}(Rx_i)e_c$. Instead of solving the coarse-level error Eq. (3), the coarse-level probability equation

$$A_c(\text{diag}(Rx_i))^{-1}x_c = 0 \quad (4)$$

can be solved for seeking x_c . Because Rx_i approximates x_c while e_c has no available approximation, it is more convenient to solve Eq. (4) for x_c than solving Eq. (3) for e_c . If the coarse-level size (the dimension of A_c) is small enough, Eq. (4) can be solved by a direct method, e.g. the algorithm in [16], for getting accurate x_c . Otherwise, it is usually solved by iterative methods for getting an improved approximation of x_c from Rx_i .

Once (approximate) x_c is obtained from Eq. (4), the fine-level error e_i is then approximated by

$$e_i = Q e_c = Q \text{diag}(Rx_i)^{-1} x_c. \quad (5)$$

Based on this formulation, the coarse-level correction

$$x_{i+1} = \text{diag}(x_i)e_i = \text{diag}(x_i)Q \text{diag}(Rx_i)^{-1} x_c = P(\text{diag}(Rx_i))^{-1} x_c \quad (6)$$

is implemented for getting a corrected fine-level approximate solution x_{i+1} . Finally, some iteration steps of a selected relaxation method again are applied to smooth x_{i+1} , which is called post-relaxation.

This two-level cycle can be naturally extended to multiple levels by recursively applying itself to solve the coarse-level probability equation (4) [14]. This recursive coarsening process can be terminated when the coarsest level size is below a user-defined value, and then the probability equation of the coarsest level can be solved by one-level methods. Several types of multilevel cycles are introduced in [14]. In this study we choose the V cycle that solves Eq. (4) once on each intermediate level.

The following theorem supports the soundness of the multilevel framework of the AM method.

Theorem 2.1 (Theorem 3.1 in [7]). *In the AM method, if the fine-level approximate solution x_i is positive, then A_c is an irreducible singular M-matrix, and thus has a unique (up to scaling) right kernel vector with positive components.*

If the initial approximate solution on the fine-level is positive, and the relaxation method can maintain the positivity of any fine-level approximate solution, then $x_i > 0$. In this case, because $R = Q^T \geq 0$, the problem matrix $A_c(\text{diag}(Rx_i))^{-1}$ of the coarse-level probability equation (4) is an irreducible singular M-matrix according to Theorem 2.1. Thus a positive x_c can be obtained by solving Eq. (4). Then, as $P = \text{diag}(x_i)Q \geq 0$, the coarse-level correction (6) outputs a positive x_{i+1} that will still be positive after the post-relaxation. Hereafter, the probability equation on each level is generally represented as $Ax = 0$.

The performance of any multigrid method relies on the complementarity of the relaxation and the coarse-level correction processes [19]. Because the nonzero values in each column of Q all equal to 1, the values of e_i approximated by Eq. (5) will be constant in each aggregate. Therefore, the ability of the aggregation process to group strongly connected degrees of freedom is of primary importance to support the accuracy of the coarse-level correction (6). There exist several aggregation algorithms that can be used to generate aggregates, e.g. the distance-one (two) aggregation, the pairwise aggregation and the Neighbourhood-Based aggregation algorithms. Detailed descriptions of these algorithms can be found in [7,14,15]. Here we employ the Neighbourhood-Based aggregation that often generates well-balanced aggregates and reduces the number of unknowns quickly for the coarsening process [15].

¹ In this paper, $\text{diag}(u)$ means the diagonal matrix constructed from a vector u , i.e. its j th diagonal element equals to $u(j)$ for each appropriate index j .

² The multiplicative error vector e_i is defined as x_i/x_i where $./$ means element-wise division, i.e., $e_i(j) = x_i(j)/x_i(j)$ for each appropriate index j .

For the relaxation process, the ability and the efficiency for smoothing error throughout the multilevel iterative process are important. Such ability is usually guaranteed by the “global” convergence property of the applied relaxation algorithm. In the AM method and some variants of it, the weighted Jacobi method is chosen for smoothing error. It can be stated as

$$x \leftarrow \text{Relax}_{wJ}(A, x, \omega) = (I - \omega W^{-1}A)x, \quad (7)$$

where W is the diagonal matrix of A , and $0 < \omega$ is the weight parameter. When $0 < \omega < 1$, this weighted Jacobi relaxation maintains the positivity of approximate solutions in the AM method, and on each level it can converge to a nontrivial solution of the system $Ax = 0$ from any positive initial guess of x [14]. However, this simple relaxation algorithm may be lack of efficiency, which will be concerned by this study.

Inspired by the smoothed aggregation (SA) multigrid methods for general linear systems, a smoothed version of the AM (hereafter SAM) method is proposed in [7]. In the SAM method, the restriction operator R and the interpolation operator P are smoothed by the weighted Jacobi method with weight $0 < \omega_s$ (typically $\omega_s \leq 1$), forming smoothed operators

$$R_s = R(I - \omega_s A W^{-1}) \quad \text{and} \quad P_s = (I - \omega_s W^{-1}A)P. \quad (8)$$

Accordingly, A_c is replaced by $A_{cs} = R_s P_s$. Then a lumping technique is applied to make A_{cs} maintain the irreducible M -matrix property, so Theorem 2.1 still holds. This smoothing technique can usually improve the convergence rate of the AM method [7]. However, it will also increase the complexities of the operators in the multilevel hierarchy, especially when the problem is relatively dense [18]. Therefore, we employ the framework of SAM when the problem matrix A is with high degree of sparsity, and use that of AM otherwise.

3. Block-accelerated aggregation multigrid methods for Markov chains

3.1. Motivation

In the AM/SAM method, the aggregation process of each level is implemented on every iterative cycle. As a result, the quality of aggregates is well guaranteed throughout the iterative process, while often a large proportion of time is cost by aggregation processes [18]. Hence attentions can be paid on this effective but time consuming scheme to accelerate the AM/SAM method. There arise some strategies, e.g. [17,18], where aggregates are updated only when the updating is “worthwhile”. In this study, we start from another consideration: whether aggregates can be further utilized rather than only using them to support coarse-level corrections.

Recall that, aggregates on each level form a disjoint partition of the degrees of freedom. Thus, we can permute and partition the degrees of freedom of each level according to the aggregates. Then any square matrix on each level will be transformed into a block version where the degrees of freedom in each diagonal block are in the same aggregate. On each level, the information required for implementing such permutation and partition can be obtained from the aggregation matrix Q . The row indexes of the nonzero elements in any j_{th} column of Q represent the degrees of freedom contained in the j_{th} aggregate. Thus, picking out such indexes in each column forms the partition information, and connecting them together forms the permutation information. This permutation information is represented by a row vector $perm$. For example, if

$$Q = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

then we can get a permutation vector $perm = [3, 4, 1, 2, 5, 6]$ and the partition information as $\{\{3, 4\}, \{1, 2\}, \{5, 6\}\}$ that represents aggregates. With these information, any square matrix (generally represented by U) on the same level with Q can be permuted as $U_{reorder} = U(perm, perm)^3$ and then partitioned into a block version according to the aggregates. In this example, $U_{reorder}$ has the following form:

$$U_{reorder} = \begin{array}{c} \begin{array}{cc|cc|cc} 3 & 4 & 1 & 2 & 5 & 6 \\ \hline * & * & * & * & * & * \\ * & * & * & * & * & * \\ \hline * & * & * & * & * & * \\ * & * & * & * & * & * \\ \hline * & * & * & * & * & * \\ * & * & * & * & * & * \end{array} \end{array}.$$

³ $U(perm, perm)$ means permuting matrix U symmetrically according to the permutation vector $perm$.

On each level, the aggregates are generated by grouping the degrees of freedom that are strongly connected in the scaled matrix $\tilde{A} = \text{Adiag}(x_i)$ [14]. Therefore, the diagonal blocks in the block matrix $\tilde{A}_{\text{reorder}} = \tilde{A}(\text{perm}, \text{perm})$ should contain a large proportion of the nonzero elements of this matrix. Moreover, the nonzero elements contained in these diagonal blocks would generally be larger than the remaining ones. Accordingly, $\tilde{A}_{\text{reorder}}$ should be nearly block diagonally dominant.

It is known that the additive Schwarz method is highly efficient and usually outperforms traditional point-wise methods for solving linear systems with block diagonally dominant matrices [20]. Besides, the suitability for parallelization is an attractive advantage of this method over point-wise methods. Therefore, regarding solving linear systems with $\tilde{A}_{\text{reorder}}$ in the AM/SAM method, the additive Schwarz method would be a better choice than traditional point-wise methods. Moreover, there are some application fields of Markov chains, where the problem matrices are typically block-structured. For example, PageRank problems are usually with nested block-structured matrices [22]. For such problems, the nested block structures can be effectively captured by aggregation processes in the AM/SAM framework as the connections between the degrees of freedom in the same block will be stronger than the connections between those from different blocks. As a result, the block diagonally dominance of the matrix $\tilde{A}_{\text{reorder}}$ on each level will be more remarkable. Thus, the additive Schwarz method may get even higher efficiency for solving linear systems with $\tilde{A}_{\text{reorder}}$ in PageRank problems than in general Markov chains. For these reasons, we consider the possibility of applying the additive Schwarz method on the block matrix $\tilde{A}_{\text{reorder}}$ to smooth error in the AM/SAM framework. In this study, we choose the Block-Jacobi method (a special case of the additive Schwarz method) as the basis to construct the relaxation algorithm.

3.2. Implementation of the Block-Jacobi relaxation

On each level of the AM/SAM method, the Block-Jacobi method is expected to have high efficiency on solving linear systems with $\tilde{A}_{\text{reorder}} = \tilde{A}(\text{perm}, \text{perm})$ that is nearly block diagonally dominant. Therefore, unlike the point-wise Jacobi relaxation (7), which deals with $Ax = 0$, Block-Jacobi relaxation should deal with the equation

$$\tilde{A}_{\text{reorder}} y = 0 \text{ with } y = \text{diag}(x_i(\text{perm}))^{-1} x(\text{perm}) \quad (9)$$

to get an improved approximation of x . The Block-Jacobi iteration for the equation $\tilde{A}_{\text{reorder}} y = 0$ can be given by

$$y_{i+1} = (I - \omega \tilde{D}^{-1} \tilde{A}_{\text{reorder}}) y_i, \quad (10)$$

where $0 < \omega$ is the weight factor, and $\tilde{D}$ is the block diagonal matrix formed by the diagonal blocks of $\tilde{A}_{\text{reorder}}$. Denote x_{in} as the approximation of x that is input to the Block-Jacobi relaxation. Accordingly, $y_0 = \text{diag}(x_i(\text{perm}))^{-1} x_{\text{in}}(\text{perm})$ is input to the iteration (10) as the initial guess of y . After $k > 0$ steps of the iteration (10), we get an improved guess y_k of y . Then, an improved approximation x_{out} of x can be obtained by the process: compute $x_{\text{out}} = \text{diag}(x_i(\text{perm})) y_k$, and then permute x_{out} back to the original order, i.e. $x_{\text{out}} = x_{\text{out}}(\text{iperm})$ where iperm is a row vector with the same dimension (denoted by m) of perm , and $\text{perm}(\text{iperm}) = [1, 2, \dots, m]$.

The implementability of this relaxation process depends on the invertibility of $\tilde{D}$. To prove this invertibility, we introduce a theorem first.

Theorem 3.1 (Theorem 4.16 in [23]). *If A is an irreducible singular M -matrix, then each of its principal submatrices other than A itself is a nonsingular M -matrix.*

Then the invertibility of $\tilde{D}$ is demonstrated in the following theorem.

Theorem 3.2. *If $0 < \omega < 1$, the matrix $\tilde{D}$ in the iteration (10) is invertible on each level of the AM/SAM method throughout the iterative process when solving the Markov chain problem (2) from any positive initial guess.*

Proof. Because the symmetric permutation $\tilde{A}_{\text{reorder}} = \tilde{A}(\text{perm}, \text{perm})$ of $\tilde{A}$ does not affect the proof process, we ignore it here. Thus $\tilde{A}_{\text{reorder}}$ in (10) equals to $\tilde{A}$ in this proof.

On each level, if the approximate solution x_i keeps positive throughout the iterative process, then the problem matrix A keeps being an irreducible singular M -matrix according to Theorem 2.1, and $\tilde{A} = \text{Adiag}(x_i)$ keeps this property too. Then, based on Theorem 3.1, every diagonal block in the matrix $\tilde{D}$ on each level keeps being a nonsingular M -matrix, so $\tilde{D}$ keeps being a nonsingular M -matrix, and thus $\tilde{D}$ keeps invertible with $\tilde{D}^{-1} \geq 0$.

On each level with a positive x_i , $\tilde{A}$ is a M -matrix, so we can get a matrix $\tilde{N} = \tilde{D} - \tilde{A} \geq 0$, and then the iteration (10) can be written as $y_{i+1} = ((1 - \omega)I + \omega \tilde{D}^{-1} \tilde{N}) y_i$ with $(1 - \omega)I + \omega \tilde{D}^{-1} \tilde{N} > 0$. In this case, if an approximate solution $x_{\text{in}} > 0$ is input to the Block-Jacobi relaxation, which leads the initial guess $y_0 > 0$ for the iteration (10), then we can get $y_k > 0$ after any k steps of this iteration, and thus the Block-Jacobi relaxation outputs a positive approximate solution x_{out} .

In conclusion, the positivity of the approximate solution on each level makes $\tilde{D}$ of that level invertible. On the other hand, the Block-Jacobi relaxation maintains the positivity of approximate solutions in the AM/SAM method. Thus, this theorem is proved. $\square$

Finally, we summarize in Algorithm 1 the implementation of the Block-Jacobi relaxation.

Algorithm 1 Block-Jacobi relaxation, $x_{out} \leftarrow \text{Relax}_{wBj}(A, x_i, x_{in}, Q, s, \omega)$.

Input:

A : problem matrix,
 x_i : approximate solution for scaling A ,
 x_{in} : approximate solution to be improved,
 Q : aggregation matrix,
 s : number of iteration steps,
 ω : weight factor.

Output:

x_{out} : improved approximate solution.

- 1: $y \leftarrow \text{diag}(x_i)^{-1} x_{in}$.
 - 2: $\bar{A}_{reorder} \leftarrow$ permute $A \text{diag}(x_i)$ according to Q .
 - 3: $y_{reorder} \leftarrow$ permute y according to Q .
 - 4: Build the block diagonal matrix $\bar{D}$ from $\bar{A}_{reorder}$ according to Q .
 - 5: Iterate $y_{reorder} \leftarrow y_{reorder} - \omega \bar{D}^{-1} (\bar{A}_{reorder} y_{reorder})$ by s times.
 - 6: $y \leftarrow$ permute $y_{reorder}$ back to the original order.
 - 7: $x_{out} \leftarrow \text{diag}(x_i) y$.
 - 8: **return** x_{out} ;
-

3.3. Convergence of the Block-Jacobi relaxation

With the implementability of the Block-Jacobi relaxation proved, the remaining issue that concerns the feasibility of this relaxation algorithm is whether it can always smooth error on each level of the AM/SAM method throughout the iterative process. Thus it is necessary to discuss the convergence property of this relaxation algorithm for solving the block equation (9) on each level in the AM/SAM method.

In [24], the additive Schwarz method, which includes the Block-Jacobi method as a special case, is proved to be convergent under some conditions when solving the Markov chain problem (2). To present a theoretical result of [24], we first introduce some necessary contents as follows.

A domain decomposition for the Markov chain problem (2) can be regarded as a partition that divides the state space into several subspaces [24]. Denote the state space of the Markov chain by a set of indexes $S = \{1, 2, \dots, n\}$, the amount of subspaces by $1 < p < n$, each subspace by

$$S_i = \{s_1, s_2, \dots, s_{n_i}\}, \quad 1 \leq i \leq p, \quad 1 \leq s_1 < s_2 < \dots < s_{n_i} \leq n,$$

then we get each $S_i \subset S$ ($1 \leq i \leq p$) and $\cup_{i=1}^p S_i = S$. Note that, these subspaces need not to be mutually disjoint, and the case that they have non-empty intersections is called the overlap [24].

Then a series of operators $R_i \in \mathbb{R}^{n_i \times n}$ ($i = 1, 2, \dots, p$) are defined. Each R_i ($1 \leq i \leq p$) is defined as⁴

$$R_i = [I_i, O] \pi_i,$$

where I_i is the identity matrix on $\mathbb{R}^{n_i \times n_i}$, $\pi_i \in \mathbb{R}^{n \times n}$ is a permutation matrix, and $\pi_i u$ permutes any vector $u \in \mathbb{R}^{n \times 1}$ such that its elements with indexes in S_i are at top locations in $\pi_i u$. Accordingly, $E_i = R_i^T R_i$ ($1 \leq i \leq p$) is a $n \times n$ diagonal matrix where $E_i(j, j) = 1$ if $j \in S_i$, and $E_i(j, j) = 0$ otherwise. With these definitions, a series of principal submatrices $A_i \in \mathbb{R}^{n_i \times n_i}$ ($i = 1, 2, \dots, p$) of the problem matrix A are given by

$$A_i = R_i A R_i^T. \quad (11)$$

For each value of i from $1, 2, \dots, p$, the set of row/column indexes of the submatrix A_i in A equals to S_i , meanwhile a principal submatrix $A_{-i} \in \mathbb{R}^{(n-n_i) \times (n-n_i)}$ of A corresponding to the indexes outside S_i can be obtained by $A_{-i} = [O, I_{-i}] \pi_i A \pi_i^T [O, I_{-i}]^T$, where I_{-i} is the identity matrix on $\mathbb{R}^{(n-n_i) \times (n-n_i)}$. Then a matrix M_i for each value of i is defined as

$$M_i = \pi_i^T \begin{bmatrix} A_i & \\ & W_{-i} \end{bmatrix} \pi_i, \quad (12)$$

where $W_{-i} \in \mathbb{R}^{(n-n_i) \times (n-n_i)}$ is a diagonal matrix, and $W_{-i}(j, j) = A_{-i}(j, j)$ ($1 \leq j \leq n - n_i$).

Finally, the additive Schwarz iteration based on the above-defined domain decomposition for solving the Markov chain problem (2) can be given by

$$x_{i+1} = T_\omega x_i \quad \text{with} \quad T_\omega = I - \omega \sum_{i=1}^p E_i M_i^{-1} A, \quad (13)$$

⁴ In this paper, O means a zero matrix with appropriate row and column dimensions

where $0 < \omega \leq 1$ is the weight factor [24].

If no overlap is present, then $\sum_{i=1}^p E_i = I$. Meanwhile, if $\pi_i = I$ ($i = 1, 2, \dots, p$), which happens when the state space S is partitioned consecutively, i.e. the indexes in the subspaces satisfy $\max(S_j) < \min(S_{j+1})$ ($j = 1, 2, \dots, p-1$), then A_i ($i = 1, 2, \dots, p$) in (11) are diagonal blocks of A , and $\sum_{i=1}^p E_i M_i^{-1} = D^{-1}$ where

$$D = \begin{pmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_p \end{pmatrix}.$$

In this case, the additive Schwarz iteration (13) becomes the Block-Jacobi iteration that deals with the Markov chain problem (2).

Define an index q that equals to the smallest integer satisfying $\sum_{i=1}^p E_i \leq qI$. Then a convergence condition of the additive Schwarz iteration for Markov chain problems is demonstrated by the following theorem.

Theorem 3.3 (Theorem 3.3 in [24]). *Let $A = I - B$, where B is an irreducible column-stochastic matrix. For the additive Schwarz iteration (13), let $N_i = M_i - A$ ($i = 1, 2, \dots, p$). If $M_i^{-1} N_i \geq 0$ ($i = 1, 2, \dots, p$) and $0 < \omega < 1/q$, then the iteration matrix T_ω in (13) is convergent and $T_\omega \geq 0$.*

It is mentioned in [24] that: Theorem 3.3 is valid for any singular M -matrix A that has a one-dimensional null space; when T_ω is convergent, i.e. $\lim_{k \rightarrow \infty} T_\omega^k$ exists, this limit is a projection onto the null space of A . Then, the following theorem describes the convergence of the Block-Jacobi iteration (10).

Theorem 3.4. *If $0 < \omega < 1$, the Block-Jacobi iteration given by (10) is convergent on each level in any AM/SAM cycle, when solving the Markov chain problem (2).*

Proof. Similar to proving Theorem 3.2, in this proof we ignore the permutation $\bar{A}_{reorder} = \bar{A}(\text{perm}, \text{perm})$ of $\bar{A}$. Thus $\bar{A}_{reorder}$ equals to $\bar{A}$ on each level.

On each level, the Block-Jacobi iteration (10) is equivalent to the additive Schwarz iteration that is applied on $\bar{A}$ and based on a domain decomposition where $\pi_i = I$ ($i = 1, 2, \dots, p$) and $q = 1$. Meanwhile, each matrix M_i ($1 \leq i \leq p$) in (12) is formed by a diagonal block and the remaining diagonal elements of $\bar{A}$, and $\sum_{i=1}^p E_i M_i^{-1} = \bar{D}^{-1}$.

According to the proof of Theorem 3.2, $\bar{A}$ on each level keeps being an irreducible singular M -matrix. Thus on each level of any cycle, $\bar{A}$ always has a one-dimensional null space, meanwhile any diagonal block of it is a nonsingular M -matrix that has a nonnegative inverse. Accordingly, for each value of i from $1, 2, \dots, p$, $M_i^{-1} \geq 0$, and $N_i = M_i - \bar{A} \geq 0$, so $M_i^{-1} N_i \geq 0$. Then, based on Theorem 3.3, the iteration matrix $(I - \omega \bar{D}^{-1} \bar{A})$ of the Block-Jacobi iteration (10) is convergent when $0 < \omega < 1/q = 1$. $\square$

Therefore, on each level in any AM/SAM cycle, the Block-Jacobi relaxation (Algorithm 1) can always reduce (smooth) the error of the approximate solution by several steps of the iteration (10).

When applying the Block-Jacobi relaxation to the AM/SAM method, it may be implemented several times in each cycle, and the matrix $\bar{D}$ may well be large on some levels that are close to the top. Therefore it is better to compute inexactly the product $\bar{D}^{-1} u^5$ that may be required many times during the implementations of the Block-Jacobi relaxation in each cycle. Computing the product $z = \bar{D}^{-1} u$ is equivalent to solving the system $\bar{D} z = u$. Recall that the matrix $\bar{D}$ is a nonsingular M -matrix on each level. Incomplete LU (ILU) factorization methods are inexact solvers and stable for linear systems with nonsingular M -matrices [25,26]. Meanwhile it is demonstrated in [27] that the Block-Jacobi method is still convergent for solving the Markov chain problem (2) when the diagonal blocks are solved by ILU factorizations with sufficient accuracies. Therefore, ILU factorization methods are suitable for computing the product $\bar{D}^{-1} u$ required by the Block-Jacobi relaxation. As the purpose of the Block-Jacobi relaxation is just to smooth error, the required accuracy for computing the product $\bar{D}^{-1} u$ can be relaxed in practical computations. Finally, we use ILU(0) provided by Matlab [21] as the inexact solver for computing the product $\bar{D}^{-1} u$ in the Block-Jacobi relaxation.

3.4. A scheme to execute the Block-Jacobi relaxation in the AM/SAM framework

For solving a linear system, the cost per iteration step and the convergence rate of the Block-Jacobi method generally increase with the dimensions of the diagonal blocks [20]. On each level of any AM/SAM cycle, the dimensions of the diagonal blocks of $\bar{D}$ are determined by the sizes of the aggregates. We execute the Block-Jacobi relaxation on a level only when the average size of the aggregates is larger than a lower bound $\zeta \in \mathbb{R}$ and smaller than an upper bound $\sigma \in \mathbb{R}$. The purpose of this scheme is to balance the cost and the performance of this relaxation for smoothing error in the AM/SAM method. Then, the AM method with the Block-Jacobi relaxation is referred to as the BAM method, and a cycle of it is presented in Algorithm 2. The SAM method with this relaxation is called the BSAM method and can be presented similarly to Algorithm 2.

⁵ u represents an arbitrary column vector with an appropriate dimension in this paper.

Algorithm 2 A multilevel cycle of the BAM method for Markov chain, $x \leftarrow \text{BAM}(A, x, v_1, v_2, \omega_1, \omega_2, \zeta, \sigma)$.

Input:

A : problem matrix,
 x : approximate solution,
 v_1, v_2 : iteration steps for pre- and post- relaxation processes,
 ω_1, ω_2 : weight factors for Jacobi and Block-Jacobi relaxations,
 ζ : lower bound for the average size of aggregates,
 σ : upper bound for the average size of aggregates.

Output:

x : approximate solution.

```

1: if not on the coarsest level then
2:    $x \leftarrow \text{Relax}_{\text{wJ}}(A, x, \omega_1) \quad v_1 \text{ times.}$ 
3:   Construct  $Q$ .
4:    $R \leftarrow Q^T$  and  $P \leftarrow \text{diag}(x)Q$ .
5:    $A_c \leftarrow RAP$ ,  $x_c \leftarrow Rx$ .
6:    $x_c = \text{BAM}(A_c \text{diag}(x_c)^{-1}, x_c, v_1, v_2, \omega_1, \omega_2, \zeta, \sigma)$ .
7:    $x_{\text{in}} \leftarrow P(\text{diag}(Rx))^{-1}x_c$ .
8:   if  $\zeta < \text{average size of the aggregates} < \sigma$  then
9:      $x \leftarrow \text{Relax}_{\text{wBJ}}(A, x, x_{\text{in}}, Q, v_2, \omega_2)$ .
10:  else
11:     $x \leftarrow \text{Relax}_{\text{wJ}}(A, x_{\text{in}}, \omega_1) \quad v_2 \text{ times.}$ 
12:  end if
13: else
14:    $x \leftarrow \text{solve } Ax = 0, \|x\|_1 = 1, x > 0$  by a direct solver.
15: end if
16: return  $x$ ;

```

In a BAM/BSAM cycle, the probability equation on the coarsest level and the product $\tilde{D}^{-1}u$ are all solved by direct solvers. Thus σ can be set based on the upper bound of the size of the coarsest level to limit the cost per step of the Block-Jacobi iteration. In this study, because we use the inexact solver ILU(0) to compute the product $\tilde{D}^{-1}u$ while coarsest level equations are solved by an exact solver from [16], σ is simply set as twice as the upper bound of the coarsest level size. Meanwhile, ζ is empirically set as 3 to guarantee that the Block-Jacobi relaxation smooths error more effectively than the point-wise version per iteration step.

4. Numerical results

In this section, we report some numerical results that are obtained by a MATLAB R2015b implementation on a 64-bit Windows-7 computer equipped with an Intel core i5-6500 processor and 8GB RAM memory. The test problems consist of three types of structured Markov chains, a type of unstructured Markov chains, and several PageRank problems constructed based on some Web graphs from [28].

We compare the BAM/BSAM method with the AM/SAM method to test the effect of the Block-Jacobi relaxation. We also put the extrapolation-accelerated smoothed/unsmoothed aggregation multigrid method from [3,16] and the preconditioned/unpreconditioned extrapolation-accelerated GMRES method from [2] into comparisons.

Note that the preconditioning technique in [2] is proposed for PageRank problems when they are stated as solving non-singular linear systems, and it is based on a polynomial expansion of the inverse of the coefficient matrix in each system. Thus this technique is inapplicable for solving the singular linear systems from general Markov chain problems. On the other hand, if PageRank problems are formulated as Markov chain problems, the coefficient matrices of the involved singular systems will be dense [29]. When using aggregation multigrid methods to solve such systems, smoothing R and P by (8) will dramatically increase their densities and thus cause extremely large computational cost per multilevel cycle. Due to this fact, it is inadvisable to use the SAM and the BSAM methods to solve PageRank problems. More details of PageRank problems and applying aggregation multigrid methods to solve this type of problems will be introduced in Section 4.2.

Based on the above factors, we run the SAM and the BSAM methods, the extrapolation-accelerated SAM (hereafter ESAM) method, the ESAM method with the Block-Jacobi relaxation (hereafter BESAM) and the extrapolation-accelerated GMRES (hereafter EGMRES) method to solve general Markov chains. For PageRank problems, we run the AM and the BAM methods, the extrapolation-accelerated AM (hereafter EAM) method, the EAM method with the Block-Jacobi relaxation (hereafter BEAM) and the preconditioned EGMRES (hereafter PEGMRES) method.

In the selected multigrid methods, the numbers of iteration steps for the pre-relaxation and the post-relaxation processes are set as $v_1 = v_2 = 1$, the weight parameter ω is set as 0.7 and 1.1 for the point-wise Jacobi relaxation and the Block-Jacobi relaxation, respectively, each coarsening process is terminated when the size of the coarsest level is below 50, and then the

Fig. 1. Graph for a 2D lattice with uniform weights.

Table 1

Numerical results for Markov chains on 2D lattices with uniform weights.

n	C_{op}	SAM		BSAM		ESAM		BESAM		EGMRES	
		IT	CPU_t	IT	CPU_t	IT	CPU_t	IT	CPU_t	IT	CPU_t
4096	1.34	34	0.99 s	24	0.75 s	18	0.55 s	16	0.51 s	710	0.18 s
16,384	1.34	36	1.76 s	26	1.41 s	20	1.04 s	17	0.97 s	740	0.59 s
65,536	1.34	39	6.94 s	30	5.99 s	20	3.75 s	18	3.75 s	5150	13.20 s
262,184	1.34	42	32.79 s	35	30.78 s	23	19.11 s	21	19.49 s	6510	97.80 s
1,048,576	1.34	47	153.3 s	36	132.1 s	25	86.56 s	22	85.25 s	> 10000	> 2060s

probability equation on the coarsest level is solved by the direct solver from [16]. Meanwhile, the ‘window size’ parameter for the extrapolation technique from [3,16] is set as 3, and the dimensions of Krylov subspaces in restarted GMRES are limited by 10.

Each run of any selected method is started from an initial guess x_0 , whose elements are randomly sampled from the interval (0, 1) and normalized by $x_0 \leftarrow x_0 / \|x_0\|_1$, and is terminated when $\|Ax_k\|_1 / \|Ax_0\|_1 \leq 10^{-126}$ or the number of iteration steps reaches 10000.

The following statistics are reported in tables to show numerical results. The *problem size* (the number of states of the tested Markov chain) is denoted by “ n ”. The *number of iteration steps* is denoted by “IT”. The *total computing time* (in seconds) is denoted by “ CPU_t ”. The *operator complexity* for each tested multigrid method is denoted by “ C_{op} ”. It is defined in [14] as the sum of the number of nonzero entries in problem matrices A on all levels divided by the number of nonzero entries in the matrix A on the finest level. For each tested multigrid method, C_{op} acts as an estimate for the amount of work required by the relaxation processes on all levels per cycle [9].

4.1. Experiments for general Markov chains

4.1.1. Structured Markov chains

(Uniform 2D lattice). We first test three examples of structured Markov chains. In the first example, we consider a type of Markov chains from [9]. Each Markov chain of this type is based on a 2D lattice with uniform weights (illustrated in Fig. 1), and the transition probability from each node i to each node j is defined as the weight of the directed edge from node i to node j , divided by the sum of the weights of all outgoing edges from node i [7]. This type of Markov chains are typically structured, and the problem matrix A corresponding to each Markov chain is essentially a scaled graph-laplacian on a 2D uniform quadrilateral lattice with 5-point stencil [9]. Here we test five uniform 2D lattices where the numbers of nodes are set as 4096, 16384, 65536, 262184 and 1048576, respectively. Numerical results for this type of Markov chains are presented in Table 1.

The first observation from Table 1 is that the operator complexities C_{op} of all the multigrid methods tested here are constant when n increases, and they have a same value that is not much larger than 1. This is because the Block-Jacobi relaxation and the extrapolation technique have no effect on forming the problem matrix of any level, and the Neighbourhood-Based aggregation reduces the number of unknowns fast for the coarsening process [15]. As a result, the computational cost per cycle of each tested multigrid method will be relatively moderate. The same observation holds for other Markov chain problems, and thus will not be described again.

⁶ x_k represents the latest approximate solution.

Fig. 2. Graph for the 2D lattice from a tandem queuing network.

Table 2

Numerical results for Markov chains from tandem queuing networks.

n	C_{op}	SAM		BSAM		ESAM		BESAM		EGMRES	
		IT	CPU_t	IT	CPU_t	IT	CPU_t	IT	CPU_t	IT	CPU_t
4096	1.24	34	0.93 s	23	0.68 s	24	0.68 s	18	0.53 s	2190	0.60 s
16,384	1.24	33	1.43 s	22	1.10 s	25	1.11 s	18	0.91 s	4410	3.47 s
65,536	1.24	42	5.81 s	25	4.03 s	27	3.94 s	19	3.19 s	> 10000	> 24.03s
262,184	1.24	36	21.25 s	23	16.00 s	27	17.17 s	19	14.11 s	> 10000	> 143.8s
1,048,576	1.24	36	79.62 s	24	63.10 s	27	65.46 s	20	56.55 s	> 10000	> 1403s

The other comparison results in Table 1 show that the BSAM method needs fewer iteration steps and less computing time than the SAM method for solving these highly structured Markov chains. In detail, 16.66% – 29.41% iteration steps and 6.13% – 24.24% computing time are reduced by applying the Block-Jacobi relaxation to the SAM method. The reduction in computing time is not as remarkable as that in the number of iteration steps. This is due to the higher cost per run of the Block-Jacobi relaxation (Algorithm 1) compared with the point-wise Jacobi relaxation on the same level. Meanwhile, the ESAM method converges in fewer iteration steps and smaller time than the BSAM method, which indicates the higher effectiveness of the extrapolation technique from [3,16] than the Block-Jacobi relaxation on accelerating the SAM method for these problems. When applying the Block-Jacobi relaxation to the ESAM method, the resulting BESAM method converges faster than the ESAM method here, but such improvement is slight. In short, the Block-Jacobi relaxation can improve the performances⁷ of the SAM and the ESAM methods for solving these highly structured Markov chains, but the effect is not significant. Besides, EGMRES converges in smaller time than other tested methods when the problem size is no larger than 16384. However, when the problem size becomes larger, the time cost of EGMRES increases significantly, leading EGMRES uncompetitive with the multigrid methods here.

(Tandem queuing network). In the next example, we consider several Markov chains that come from tandem queuing networks [1]. Tandem queuing networks have been applied in wireless networks [30] and blood screening procedures [5]. For each tandem queuing network, there are two finite queues with single-servers, customers arrive according to a Poisson distribution with rate μ , and the service time distributions at the two single-server stations are Poisson with rates μ_1 and μ_2 , respectively [7]. Then each tandem queuing system can be regarded as a Markov chain on a 2D lattice (illustrated in Fig. 2) that is like the previous example, except that the link structure and the weights here are more complicated. We test five such 2D lattices. In each lattice, we set $\mu = 10$, $\mu_1 = 11$ and $\mu_2 = 10$ according to the settings in [7]. Numerical results are presented in Table 2.

Seen from Table 2, the BSAM method performs much better than the SAM method for solving these Markov chains. In detail, using the BSAM method instead of the SAM method, 32.35% – 40.48% iteration steps and 20.75% – 30.64% computing time are saved. Meanwhile, the performance of the BSAM method becomes similar as that of the ESAM method in this example. These results indicate that the effect of the Block-Jacobi relaxation on accelerating the SAM method in this example

⁷ In the following contents, if no more description is given, the performance and the efficiency of any method both mean the general performance of that method in terms of IT and CPU_t .

Fig. 3. Graph for a H2/E3/1 queueing system.

Table 3

Numerical results for Markov chains from H2/E3/1 queues.

n	C_{op}	SAM		BSAM		ESAM		BESAM		EGMRES	
		IT	CPU_t	IT	CPU_t	IT	CPU_t	IT	CPU_t	IT	CPU_t
6578	1.92	53	1.87 s	33	1.35 s	36	1.37 s	26	1.08 s	> 10000	> 3.99s
24,578	1.92	51	3.93 s	32	2.82 s	47	3.82 s	29	2.58s	> 10000	> 10.44s
120,002	1.92	57	15.86 s	43	13.79 s	45	13.25 s	37	12.39 s	> 10000	> 56.70s

is more remarkable than that in the previous example. The reason is probably that: the tested Markov chains become less regular in this example, which increases the differences between the nonzero values in the problem matrices, and thus enhances the block diagonally dominance of the matrix $\tilde{A}_{reorder}$ in the Block-Jacobi relaxation (Algorithm 1). Due to the same reason, when applying the Block-Jacobi relaxation to the ESAM method, the resulting BESAM method converges faster than the ESAM method, and this speedup is more remarkable than that in the previous example. Accordingly, the BESAM method shows the best performance among all the multigrid methods tested here. Besides, the performance of EGMRES is still uncompetitive here as it deteriorates rapidly when the problem size increases.

(H2/E3/1 queueing system). The final tested structured Markov chains come from H2/E3/1 queueing systems. Each H2/E3/1 queueing system can be shown graphically in Fig. 3, where the arrival process is a two-phase hyperexponential distribution, and the service process is an Erlang-3 distribution [1]. Here we set $\alpha_1 = 0.4$, $\alpha_2 = 0.6$, $\lambda_1 = 1$, $\lambda_2 = 2$, $\mu_1 = 2$, $\mu_2 = 8$, $\mu_3 = 8$, and tune the limitation for the maximum number of the customers in the queue to generate Markov chains with different numbers of states. As an exhaustive introduction for this type of systems and the corresponding Markov chains needs a long section, we refer the reader to [1] for more complete descriptions. Numerical results are presented in Table 3.

Seen from Table 3, the BSAM method needs 24.56% – 37.74% fewer iteration steps and 13.05% – 28.24% less time than the SAM method for solving these Markov chains. The BSAM and the ESAM methods show very similar performances for solving the Markov chains whose sizes are 6578 and 120002, but the former performs much better than the latter when the problem size equals to 24578. Applying the Block-Jacobi relaxation to the ESAM method, the resulting BESAM method shows the highest efficiency among all the multigrid methods tested here. In general, the comparison results in this example are similar as those in the previous example, except that EGMRES has never converged in 10,000 iteration steps here.

We summarize the results obtained from the above three examples as follows. The proposed Block-Jacobi relaxation shows effectiveness on improving the performance of the SAM method for solving structured Markov chains. The improvement becomes more remarkable when the structured Markov chains become less regular, and in such cases the BSAM method shows similar or even better performance than the ESAM method. Moreover, the Block-Jacobi relaxation also shows effectiveness on accelerating the ESAM method, especially for the less regular Markov chains. This indicates the potential of the Block-Jacobi relaxation to accelerate variants of the SAM method.

4.1.2. Unstructured Markov chains

(Random walk on unstructured planar graph.) Then we test a type of unstructured Markov chains from [7]. They are derived from random walk processes on unstructured planar graphs. The nodes of each graph are randomly distributed in a unit square $(0, 1)^2$, and then are connected by using Delaunay triangulation [7]. We consider both the symmetric and the nonsymmetric cases. In the symmetric cases, the edges of each graph are bidirectional. In the nonsymmetric cases, the triangles in each graph are partly marked with “+” in such a way that there is no edge shared by two marked triangles, and each unmarked triangle is next to a marked triangle, then one of the three bidirectional edges of each marked triangle is made uni-directional. Small examples of the symmetric and the nonsymmetric unstructured planar graphs are illustrated in Fig. 4. The random walk process on each graph is indeed a Markov chain on that graph, where the transition probability from each node i to each node j is nonzero only when node i has a link pointing to node j , and then is given by the

Fig. 4. Unstructured planar graphs, symmetric (left) and **nonsymmetric** (right). For the figure on the right, the triangles marked by “+” are the selected ones where one edge of each triangle is made as a uni-directional link.

Table 4

Numerical results for Markov chains from random walks on unstructured planar graphs (symmetric).

n	C_{op}	SAM		BSAM		ESAM		BESAM		EGMRES	
		IT	CPU _t	IT	CPU _t	IT	CPU _t	IT	CPU _t	IT	CPU _t
4096	1.24	69	1.79 s	42	1.27 s	31	0.86 s	22	0.68 s	460	0.19 s
16,384	1.24	71	5.13 s	44	3.70 s	30	2.17 s	23	1.89 s	460	0.51 s
65,536	1.24	83	23.98 s	53	18.16 s	34	9.95 s	25	8.40 s	1800	7.55s
262,184	1.24	89	145.7 s	56	102.6 s	37	62.44 s	25	47.69 s	5600	115.1s

Table 5

Numerical results for Markov chains from random walks on unstructured planar graphs (nonsymmetric).

n	C_{op}	SAM		BSAM		ESAM		BESAM		EGMRES	
		IT	CPU _t	IT	CPU _t	IT	CPU _t	IT	CPU _t	IT	CPU _t
4096	1.23	62	1.63 s	39	1.18 s	28	0.85 s	22	0.65 s	490	0.19 s
16,384	1.24	72	5.02 s	44	3.65 s	30	2.19 s	24	2.06 s	550	0.58 s
65,536	1.25	86	24.62 s	56	18.59 s	34	10.02 s	26	8.86 s	2390	9.67s

reciprocal of the number of directional links from node i [7]. Numerical results for such Markov chains on the symmetric and the nonsymmetric unstructured planar graphs are presented in Tables 4 and 5, respectively.

The following observations are drawn from Tables 4 and 5. Firstly note that each tested multigrid method shows similar performances for solving each pair of Markov chains with the same size in Tables 4 and 5. Thus these two tables give similar comparison results of the tested multigrid methods. The BSAM method converges in significantly fewer iteration steps and smaller time than the SAM method, which indicates the high effectiveness of the Block-Jacobi relaxation on accelerating the SAM method for solving this type of Markov chains. The ESAM method performs better than the BSAM method in this example. On the other hand, the Block-Jacobi relaxation can effectively accelerate the ESAM method, and the resulting BESAM method shows the best performance among the tested multigrid methods. EGMRES outperforms the tested multigrid methods when the problem size is relatively small, but this advantage becomes smaller and then disappears as the problem size increases. Based on these results, applying the Block-Jacobi relaxation to accelerate the SAM method and its variants is expected to be worthwhile when solving unstructured Markov chains.

4.2. Application to the PageRank Problem

4.2.1. The PageRank problem

Google designed a model named PageRank to determine importances of Web pages. This model considers a random walk process on the Web graph that represents the concerned Web pages by nodes and hyperlinks by directed edges [31]. It assumes that: the pages, which are visited more frequently by this random walk, are more important [31]. Accordingly, the importances of the concerned pages should be determined by the stationary distribution of the Markov chain corresponding to this random walk. On this stage, the PageRank problem is mathematically the same as the random walk problems considered in the previous example. However, in the Web graph there may exist some dangling nodes that represent the pages having no hyperlink, and the random walk will stagnate when it reaches such a node. Besides, the irreducibility of

the Web graph is not guaranteed. Therefore, some modifications are applied to the random walk on the Web graph to avoid stagnation and to guarantee that the corresponding Markov chain has a unique stationary distribution.

Suppose the Web graph has n pages numbered from 1 to n . Denote the transition matrix of the random walk on the Web graph by $B = \{b_{i,j}\} \in \mathbb{R}^{n \times n}$, where $b_{i,j}$ is the transition probability from node (page) j to node (page) i , and is given by the same way as the random walk problems in the previous example. Then it is clear that the dangling nodes in the Web graph will result in zero columns of B . To avoid stagnation of this random walk, Google assumes that whenever this random walk reaches a dangling node, it can possibly transfer to each node i ($1 \leq i \leq n$) by a probability $v(i)$, where $v \in \mathbb{R}^{n \times 1}$ is a probability distribution vector. Accordingly, a modified version $\tilde{B}$ of B is given by

$$\tilde{B} = B + vd^T, \quad (14)$$

where $d \in \mathbb{N}^{n \times 1}$, and $d(i)$ ($1 \leq i \leq n$) is nonzero (being 1) only when node i is a dangling node. Now $\tilde{B}$ is a column-stochastic matrix. To guarantee the irreducibility of this matrix, there arises another assumption: at every step of this random walk, a transition that follows an edge in the directed graph of $\tilde{B}$ happens with probability $0 < \alpha < 1$, a transition without following any edge happens with probability $1 - \alpha$, and $v(i)$ ($1 \leq i \leq n$) is the probability that this transition reaches node i . Consequently, $\tilde{B}$ is modified into the final version $\hat{B}$ by

$$\hat{B} = \alpha \tilde{B} + (1 - \alpha)v\epsilon^T, \quad (15)$$

where $\epsilon = [1, 1, \dots, 1]^T \in \mathbb{N}^{n \times 1}$. Finally, the PageRank problem can be mathematically stated as solving the probability equation:

$$Ax = 0 \quad \text{with} \quad A = (I - \hat{B}). \quad (16)$$

Note that the PageRank problem can also be stated as solving a nonsingular linear system other than the singular system (16) [31]. In this case, the computation of this problem is mathematically different from solving the general Markov chain problem (2) and beyond the scope of this study.

4.2.2. Block-accelerated aggregation multigrid methods for the PageRank problem

For the PageRank problem, although the Web graph is usually very sparse [32], the problem matrix $A = I - \hat{B}$ of (16) is dense since $v\epsilon^T$ in (15) is typically dense [34]. Thus, the operations in aggregation multigrid methods when solving this problem will cost huge storage and time if they directly deal with the whole matrix A . To address this issue, an adaption strategy is proposed in [14], and we give a brief description as follows.

The matrix A in (16) is rewritten as

$$A = A_{\text{sparse}} + hf^T, \quad (17)$$

where $A_{\text{sparse}} = I - \alpha B$ is the sparse part of A because matrix B is as sparse as the Web graph, $h = -v$ and $f = \alpha d + (1 - \alpha)\epsilon$ are two vectors that form the rank-one part of A . Then A_{sparse} , h and f are stored instead of A to save storage. Meanwhile, any dense matrix-vector product Au required by the relaxation processes on the finest level can be computed by $A_{\text{sparse}}u + h(f^Tu)$ that almost only costs a sparse matrix-vector multiplication. The problem matrix of the coarse-level probability equation (4) can be written as

$$A_c(\text{diag}(Rx_i))^{-1} = A_{\text{sparse},c} + h_c f_c^T,$$

where

$$A_{\text{sparse},c} = RA_{\text{sparse}}P\text{diag}(Rx_i)^{-1}, \quad h_c = Rh \quad \text{and} \quad f_c = (f^TP\text{diag}(Rx_i)^{-1})^T.$$

Accordingly, $A_{\text{sparse},c}$ acts as the sparse part of the coarse-level problem matrix, h_c and f_c are two vectors forming the rank-one part. They are counterparts of A , h and f .

In the AM method, because $R = Q^T$ and $P = Q\text{diag}(x_i)$ are very sparse, $A_{\text{sparse},c}$ is sparse as well. Thus the storage and the time required by dealing with the coarse-level probability equation can be reduced through using the formulation $A_c(\text{diag}(Rx_i))^{-1} = A_{\text{sparse},c} + h_c f_c^T$. In the SAM method, however, because the problem matrix A is dense, smoothing the operators R and P increases significantly their densities. Therefore, the restriction and the interpolation operations in a SAM cycle are much more costly than those in a AM cycle for this problem. Besides, the “sparse” part $A_{\text{sparse},c} = R_s A_{\text{sparse}} P_s \text{diag}(R_s x_i)^{-1}$ of the coarse-level problem matrix in a SAM cycle will actually be dense, which makes the coarse-level probability equation (4) difficult to solve. Hence we use the AM framework instead of the SAM framework here to solve the PageRank problem. Then we represent the problem matrix on each level of any AM cycle generally as $A = A_{\text{sparse}} + hf^T$, where A_{sparse} is always sparse.

To apply the Block-Jacobi relaxation (Algorithm 1) to the AM method when solving this problem, we also employ an adaption strategy that exploits the formulation $A = A_{\text{sparse}} + hf^T$ to avoid operations with dense matrices. The scaled matrix $\bar{A} = A\text{diag}(x_i)$ can be written as

$$\bar{A} = \bar{A}_{\text{sparse}} + h\bar{f}^T,$$

where $\bar{A}_{\text{sparse}} = A_{\text{sparse}}\text{diag}(x_i)$ and $\bar{f} = \text{diag}(x_i)f$. Accordingly, the block matrix $\bar{A}_{\text{reorder}}$ considered in the Block-Jacobi relaxation can be written as

$$\bar{A}_{\text{reorder}} = \bar{A}_{\text{sparse}}(\text{perm}, \text{perm}) + h(\text{perm})\bar{f}(\text{perm})^T.$$

Table 6
Information of Web graphs.

name	n	nnz	$\frac{nnz}{n^2}$
wb-cs-stanford	9914	36,854	3.75e–4
ca-CondMat	23,133	186,936	3.49e–4
email-enron	36,692	367,662	2.73e–4
web-NotreDame	325,729	1,497,134	1.41e–5

Therefore, any matrix-vector product $\bar{A}_{reorder}u$ required by the Block-Jacobi relaxation can be computed economically through the previously described way. We denote $r = \bar{A}_{reorder}u$. Then the only remaining issue for implementing the Block-Jacobi relaxation is how to compute $\bar{D}^{-1}r$ efficiently. We first present the Sherman–Morrison formula below, which supports the efficient computation of $\bar{D}^{-1}r$.

Theorem 4.1 (Theorem 1 in [33]). Let $W \in \mathbb{R}^{N \times N}$, $g, z \in \mathbb{R}^{N \times 1}$, where N can be any positive integer. If both W and $W + gz^T$ are invertible, then the following formula holds:

$$(W + gz^T)^{-1} = W^{-1} - \frac{W^{-1}gz^TW^{-1}}{1 + z^TW^{-1}g}. \quad (18)$$

Now we ignore the symmetric permutation $\bar{A}_{reorder} = \bar{A}(\text{perm}, \text{perm})$, and thus $\bar{A}_{reorder} = \bar{A}$. Suppose

$$\bar{D} = \begin{bmatrix} \bar{D}_1 & & & \\ & \bar{D}_2 & & \\ & & \ddots & \\ & & & \bar{D}_p \end{bmatrix}, \quad (19)$$

where each $\bar{D}_i$ ($1 \leq i \leq p$) is a diagonal block of $\bar{A}$ and corresponds to a non-empty set S_i of indexes. Then each $\bar{D}_i$ can be given by

$$\bar{D}_i = \bar{D}_{i,sparse} + h_i \bar{f}_i^T, \quad (20)$$

where $\bar{D}_{i,sparse} = \bar{A}_{sparse}(S_i, S_i)$ is a diagonal block of $\bar{A}_{sparse}$ and is sparse, $h_i = h(S_i)$ and $\bar{f}_i = \bar{f}(S_i)$ are subvectors of h and $\bar{f}$ respectively. According to the proof of Theorem 3.2, each $\bar{D}_i$ must be invertible. Because $0 < \alpha < 1$, $B \geq 0$ and $\|B\|_1 \leq 1$, $A_{sparse} = I - \alpha B$ is a nonsingular M -matrix. Accordingly, $\bar{A}_{sparse} = A_{sparse} \text{diag}(x_i)$ is a nonsingular M -matrix as well, and thus each $\bar{D}_{i,sparse}$ is invertible. Based on Theorem 4.1, each $\bar{D}_i^{-1}r_i$, where $r_i = r(S_i)$ is a subvector of r , can be computed by

$$\bar{D}_i^{-1}r_i = (I - \frac{(\bar{D}_{i,sparse}^{-1}h_i)\bar{f}_i^T}{1 + \bar{f}_i^T(\bar{D}_{i,sparse}^{-1}h_i)})\bar{D}_{i,sparse}^{-1}r_i. \quad (21)$$

Then the result of $\bar{D}^{-1}r$ can be formed by $[\bar{D}_1^{-1}r_1; \bar{D}_2^{-1}r_2; \dots; \bar{D}_p^{-1}r_p]$. Note that, the result of $\bar{D}_{i,sparse}^{-1}h_i$ ($1 \leq i \leq p$) can be used for computing $\bar{D}_i^{-1}r_i$ with any value of r_i that may appear during multiple iteration steps in the Block-Jacobi relaxation. We store the result of each $\bar{D}_{i,sparse}^{-1}h_i$ after the first iteration step. Then the cost of computing each $\bar{D}_i^{-1}r_i$ by (21) in the remaining iteration steps almost only comes from computing $\bar{D}_{i,sparse}^{-1}r_i$.

4.2.3. Numerical results for PageRank problems

In this example, we test several PageRank problems based on some Web graphs from [28]. The characteristics of each Web graph are summarized in Table 6, where “ n ” denotes the number of pages, “ nnz ” denotes the number of hyperlinks and “ nnz/n^2 ” represents the sparsity. For each PageRank problem, we set $\alpha = 0.9$ and the vector v as a uniform probability distribution, i.e. $v = \epsilon/n$. As previously explained, smoothing the restriction and the interpolation operators in aggregation multigrid methods is not suitable when solving PageRank problems. Therefore we run the AM, the BAM, the EAM and the BEAM methods here. Remark that, the implementation of the selected Neighbourhood-Based aggregation has not been well adapted for PageRank problems, in other words, the formulation $A = A_{sparse} + hf^T$ is exploited insufficiently. As a result, the computational cost of this aggregation algorithm is very large when it deals with a scaled PageRank problem matrix. Hence, for each tested multigrid method, we implement a series of aggregation processes in advance to generate aggregation matrices for all levels, and we use them without updating throughout the iterative process. The PEGMRES method acts as a competitor for the tested multigrid methods here.

Numerical results are presented in Table 7. Note that, here C_{op} is defined as $\frac{\sum nnz(A_{sparse,l})}{nnz(A_{sparse,1})}$, where $A_{sparse,l}$ is the sparse part of the problem matrix on level l .

Seen from Table 7, C_{op} is smaller than 1.4 for each problem. This indicates that the sparse part A_{sparse} of the problem matrix maintains a high degree of sparsity during the coarsening processes in each tested multigrid method when solving these problems. As a result, the time cost per cycle is moderate in the AM and the EAM methods, thanks to the exploitation of

Table 7
Numerical results for PageRank problems.

n	C_{op}	AM		BAM		EAM		BEAM		EPGMRES	
		IT	CPU_t	IT	CPU_t	IT	CPU_t	IT	CPU_t	IT	CPU_t
wb-cs-stanford	1.07	81	1.33 s	32	0.66 s	43	0.78 s	23	0.47 s	40	0.15 s
ca-CondMat	1.21	65	1.75 s	32	1.24 s	36	1.07 s	23	0.95 s	40	3.61 s
email-enron	1.32	72	2.86 s	32	1.90 s	36	1.61 s	22	1.41 s	30	2.81 s
web-NotreDame	1.06	89	13.13 s	42	9.70 s	50	10.27 s	25	7.72 s	60	11.04 s

the formulation $A = A_{sparse} + hf^T$. The BAM method shows much better performance than the AM method for solving these PageRank problems. In detail, it needs 50.77% – 60.49% fewer iteration steps and 26.12% – 50.38% less time than the AM method. This result demonstrates the high effectiveness of the Block-Jacobi relaxation combined with the adaption strategy (for PageRank computations) on accelerating the AM method when solving these PageRank problems. Note that, such accelerating effect is more remarkable than that in the previous examples for general Markov chains. This is due to the fact that Web graphs are often nested block-structured [22], and such block structures are then captured by aggregation processes to enhance the block diagonally dominance of $A_{reorder}$ in Block-Jacobi relaxations. Due to the same reason, the BAM method shows a faster convergence rate than the EAM method for solving these problems, and the EAM method can also be accelerated significantly by applying the Block-Jacobi relaxation. The resulting BEAM method performs better than other multigrid methods here. The EPGMRES method is competitive with the tested multigrid methods in convergence rate but shown to be more time-consuming than the BAM, the EAM and the BEAM methods. We consider that these numerical results have demonstrated the potential of the proposed Block-Jacobi relaxation technique for accelerating PageRank computations.

5. Conclusions

In this paper, we briefly reviewed the adaptive algebraic aggregation multigrid (in short, AM) method and its smoothed version (in short, SAM) for solving Markov chains. We showed that the aggregates on each level of the AM/SAM method can be utilized to transfer the probability equation of that level into a block linear system. In particular, this system would be nearly block diagonally dominant. Then a Block-Jacobi relaxation algorithm is proposed. On each level of the AM/SAM method, it deals with the block linear system, and in theory it can converge to a nontrivial solution of the probability equation. Thus it is feasible to be applied in the AM and the SAM methods for smoothing error. An adaption strategy is proposed for this Block-Jacobi relaxation to avoid operating dense matrices when solving PageRank problems. This relaxation is shown to be more efficient than the point-wise Jacobi relaxation for the AM and the SAM methods when solving Markov chains and PageRank problems. This technique also shows effectiveness on accelerating some other variants of the AM method. These results may support new solutions that deeply exploit the aggregates in aggregation multigrid methods for improving the efficiencies of these methods.

Acknowledgments

This research is supported by 973 Program (2013CB329404), NSFC(61370147, 11501085).

References

- [1] Stewart WJ. Introduction to the numerical solutions of Markov chains. Princeton, USA: Princeton Univ. Press; 1994.
- [2] Pu BY, Huang TZ, Wen C. A preconditioned and extrapolation-accelerated GMRES method for pagerank. Appl Math Lett 2014;37:95–100.
- [3] Pu BY, Huang TZ, Wen C, Lin YQ. The extrapolation-accelerated multilevel aggregation method in pagerank computation. Math Probl Eng 2013;2013.
- [4] Benzi M, Kuhlmann V. Chebyshev acceleration of the generank algorithm. Electron Trans Numer Anal 2013;40:311–20.
- [5] Bar-Lev SK, Blanc H, Boxma O, Janssen G, Perry D. Tandem queues with impatient customers for blood screening procedures. Methodol Comput Appl Probab 2013;15(2):423–51.
- [6] Freund RW, Hochbruck M. On the use of two QMR algorithms for solving singular systems and applications in Markov chain modeling. Numer Linear Algebra Appl 1994;1(4):403–20.
- [7] De Sterck H, Manteuffel TA, McCormick SF, Miller K, Pearson J, Ruge J, et al. Smoothed aggregation multigrid for Markov chains. SIAM J Sci Comput 2010;32(1):40–61.
- [8] Romanovsky VI. Discrete Markov chains. Wolters-Noordhoff Publishing; 1970.
- [9] De Sterck H, Miller K, Manteuffel T, Sanders G. Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains. Adv Comput Math 2011;35(2–4):375–403.
- [10] Philippe B, Saad Y, Stewart WJ. Numerical methods in Markov chain modeling. Oper Res 1992;40(6):1156–79.
- [11] Takahashi Y. A lumping method for numerical calculations of stationary distributions of Markov chains. Research Report B-18. Department of Information Sciences, Tokyo Institute of Technology; 1975.
- [12] Horton G, Leutenegger ST. A multi-level solution algorithm for steady-state Markov chains. In: Proceedings of the ACM SIGMETRICS conference on measurement and modeling of computer systems (Nashville, TN); 1994. p. 191–200.
- [13] Leutenegger ST, Horton G. On the utility of the multi-level algorithm for the solution of nearly completely decomposable Markov chains. In: Computations with Markov chains. US, Springer; 1995. p. 425–43.
- [14] De Sterck H, Manteuffel TA, McCormick SF, Nguyen Q, Ruge J. Multilevel adaptive aggregation for Markov chains, with application to web ranking. SIAM J Sci Comput 2008;30(5):2235–62.
- [15] De Sterck H, Miller K, Sanders G, Winlaw M. Recursively accelerated multilevel aggregation for Markov chains. SIAM J Sci Comput 2010;32(3):1652–71.
- [16] Wen C, Huang TZ, Wu DA, Li L. The finest level acceleration of multilevel aggregation for Markov chains. Int J Numer Anal Model Ser B 2011;2(1):27–41.

- [17] Shen ZL, Huang TZ, Carpentieri B, Wen C. A cost-effective smoothed multigrid with modified neighborhood-based aggregation for Markov chains. *Math Prob Eng* 2015;2015.
- [18] Treister E, Yavneh I. On-the-fly adaptive smoothed aggregation multigrid for Markov chains. *SIAM J Sci Comput* 2011;33(5):2927–49.
- [19] Brezina M, Falgout R, MacLachlan S, Manteuffel T, McCormick S, Ruge J. Adaptive algebraic multigrid. *SIAM J Sci Comput* 2006;27(4):1261–86.
- [20] Cai XC, Saad Y. Overlapping domain decomposition algorithms for general sparse matrices. *Numer Linear Algebra Appl* 1996;3(3):221–37.
- [21] MathWorks T., Inc. *MATLAB reference guide*. 1992. Natick, USA.
- [22] Kamvar S, Haveliwala T, Manning C, Golub G. Exploiting the block structure of the web for computing pagerank. Technical Report. Stanford University; 2003.
- [23] Berman A, Plemmons RJ. Nonnegative matrices. *The mathematical sciences, classics in applied mathematics*, 9; 1979.
- [24] Bru R, Pedroche F, Szyld DB. Additive schwarz iterations for Markov chains. *SIAM J Matrix Anal Appl* 2005;27(2):445–58.
- [25] Benzi M. Preconditioning techniques for large linear systems: a survey. *J Comput Phys* 2002;182(2):418–77.
- [26] Diaz J. *Mathematics for large scale computing*. CRC Press; 1989.
- [27] Benzi M, Kuhlmann V. Restricted additive Schwarz methods for markov chains. *Numer Linear Algebra Appl* 2011;18(6):1011–29.
- [28] Davis TA, Hu Y. The university of florida sparse matrix collection. *ACM Trans Math Softw (TOMS)* 2011;38(1):1.
- [29] Gleich DF, Gray AP, Greif C, Lau T. An inner-outer iteration for computing pagerank. *SIAM J Sci Comput* 2010;32(1):349–71.
- [30] Le L, Hossain E. Tandem queue models with applications to qos routing in multihop wireless networks. *IEEE Trans Mob Comput* 2008;7(8):1025–40.
- [31] Gleich DF. Pagerank beyond the web. *SIAM Rev* 2015;57(3):321–63.
- [32] Langville AN, Meyer CD. *Google's pagerank and beyond: the science of search engine rankings*. Princeton, USA: Princeton Univ. Press; 2006.
- [33] Egidi N, Maponi P. A Sherman-Morrison approach to the solution of linear systems. *J Comput Appl Math* 2006;189(1):703–18.
- [34] Wen C, Huang TZ, Shen ZL. A note on the two-step matrix splitting iteration for computing pagerank. *J Comput Appl Math* 2017;315(1):87–97.