Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 816935, 15 pages
http://dx.doi.org/10.1155/2015/816935

Research Article

A Cost-Effective Smoothed Multigrid with Modified Neighborhood-Based Aggregation for Markov Chains

Zhao-Li Shen,[1,2] Ting-Zhu Huang,[1] Bruno Carpentieri,[2] and Chun Wen[1]

[1]*School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China*
[2]*Institute of Mathematics and Computational Science, University of Groningen, 9747 AG Groningen, Netherlands*

Correspondence should be addressed to Ting-Zhu Huang; tzhuang@uestc.edu.cn

Received 11 April 2015; Accepted 16 August 2015

Academic Editor: Francesco Braghin

Smoothed aggregation multigrid method is considered for computing stationary distributions of Markov chains. A judgement which determines whether to implement the whole aggregation procedure is proposed. Through this strategy, a large amount of time in the aggregation procedure is saved without affecting the convergence behavior. Besides this, we explain the shortage and irrationality of the Neighborhood-Based aggregation which is commonly used in multigrid methods. Then a modified version is presented to remedy and improve it. Numerical experiments on some typical Markov chain problems are reported to illustrate the performance of these methods.

1. Introduction

Markov chains have a large number of applications for scientific research and technological optimization in many areas, including queuing systems and statistical automata networks, web page ranking [1, 2] and gene ranking [3], risk management, and customer relationship analysis. Markov chains also have an important application for the noise and signal analysis, processing of voice, image, and screen, analog of communications network and transport phenomena, economic calculation methods, and so forth. For the needs of these practical applications, there arise different types of Markov chain models, such as the hidden Markov chain [4], multivariate Markov chain, and high-order Markov chain [5]. For Markov models, computing the stationary distribution is an important issue.

The transition matrix of a homogeneous irreducible Markov chain with n states can be denoted as B. $B \in R^{n\times n}$ is a stochastic matrix and commonly defined as a column-stochastic matrix; that is, for every $1 \le i, j \le n$,

$$\begin{aligned} 0 \le b_{i,j} \le 1, \\ e^T B = e^T, \end{aligned} \tag{1}$$

where column vector e is equal to one.

Then the stationary distribution problem is mathematically given by

$$\begin{aligned} Bx &= x, \\ \|x\|_1 &= 1, \\ x &\ge 0. \end{aligned} \tag{2}$$

In this study, we only consider the situation where B is also an irreducible matrix, which means, in its directed graph, there exists a directed path from any vertex i to any other vertex j. If B is irreducible, according to the Perron-Frobenius theorem [6], there exists a unique solution x to the linear system (2).

This linear problem (2) is often reformulated as

$$\begin{aligned} Ax &= 0, \\ \|x\|_1 &= 1, \\ x &\ge 0 \end{aligned} \tag{3}$$

with

$$A = I - B, \tag{4}$$

where I is an identity matrix. As Sterck et al. showed in their work [7], this formulation has two specific advantages. First, the solution x is not only the right eigenvector of A corresponding to eigenvalue 0, but also the right singular vector of A corresponding to singular value 0. The second advantage of formulation (3) is that the M-matrix structure of A is amenable to additive algebraic multigrid solvers designed for nonsingular linear problems [8, 9]. Hence, our interest is to solve the linear system (3), corresponding to a homogeneous irreducible Markov chain.

In many applications, the size of the Markov chain is very large, leading to the fact that direct solvers are impractical. Hence, the iterative procedures are widely used to calculate the stationary probability of Markov chains. However, traditional one-level iterative methods, for example, the Power method, (weighted) Jacobi method, and (weighted) Gauss-Seidel method, converge very slowly for calculating the solution x in the situations where the subdominant eigenvalue λ_2 of B satisfies $|\lambda_2| \sim 1$ [10, 11]. The Markov chain in this case is a slowly mixing type.

Recently, multigrid iterative methods aiming to accelerate the convergence rate for these types of Markov chains have received much attention. The multigrid methods for Markov chains have been developed and applied very successfully since the earliest work of Takahashi's two-level iterative aggregation/disaggregation method for Markov chains [12]. Direct extension of two levels to multilevels was first explored in [13]. Then, adaptive algebraic multigrid methods whose aggregates are formed algebraically based on the strength of connection in the problem matrix column-scaled by the current iterate were developed in [14, 15], and for Markov chains [16]. This method shows good potential. Following this idea, there have arisen some techniques including smoothing the interpolation and restriction operators [17], using acceleration on all recursive levels [18] or on the top level [2, 19], the on-the-fly interactive strategy [20], and so forth.

The numerical experiments in [18] show that the aggregation strategy has a certain effect on the performance of these adaptive algebraic multigrid methods. Employing reasonable and efficient aggregation strategy can improve the performance and applicability of these multigrid methods. However, in some cases, we found that the Neighborhood-Based aggregation used commonly by the above-mentioned methods generates aggregates with many states of very weak connection between each other. These unreasonable aggregates often lead to a poor convergence rate of the algebraic multigrid methods.

To remedy these problems and improve the applicability of Neighborhood-Based aggregation, this paper propose a modified version which changes some rules.

Besides the effect on the convergence rate, the computational cost of aggregation procedure accounts for a large proportion of the whole cost of these adaptive algebraic multigrid methods [20]. We study the aggregation method which is based on the connection strength and propose a judge mechanism to capture in advance the situations where the aggregates will be the same as previous. Then there is no need to implement the whole aggregation method. This strategy keeps the convergence rate the same with computing time reduced.

The remainder of this paper is organized as follows. In Section 2, the smoothed aggregation multigrid (SAM) method for the Markov chains from [17] is reviewed. In Section 3, we analyse and demonstrate the drawback of the Neighborhood-Based aggregation method and present the modified version. The judge mechanism and the cost-effective SAM are presented in Section 4. Numerical tests are presented in Section 5. Finally, conclusions are reported in Section 6.

2. Smoothed Aggregation Multigrid (SAM) Method for Markov Chains

First, we briefly review the framework of SAM method for Markov chains from [17], where the hierarchy of coarse grids is developed based on the structure of the scaled problem matrix $A\,\mathrm{diag}(x_i)$, instead of relying on the geometry of the original problem. Here, we follow the presentation of [17].

Multilevel methods aim to reduce the algebraically smooth error left by the classical iterative methods. So classical iterative methods like Power method, (weighted) Jacobi method, and (weighted) Gauss-Seidel method are often used as relaxation techniques for multilevel methods.

In this study, the weighted Jacobi iterative method is used as relaxation for (3) and it is given by

$$x \longleftarrow \left(I - \omega D^{-1} A\right) x, \tag{5}$$

where D is the diagonal matrix of A.

System (3) can be rewritten into the following form:

$$A\,\mathrm{diag}\left(x_i\right) e_i = 0, \tag{6}$$

where x_i is the approximate got by last multilevel cycle and current relaxations and e_i is the multiplicative error vector and should be all ones when convergence is achieved.

Then the aggregation matrix $Q \in R^{n\times m}$ needs to be constructed, where $q_{ij} = 1$ if the node i belongs to aggregate j and $q_{ij} = 0$ otherwise. Each column of Q represents an aggregate, and each row has only one component equal to 1 while others are 0. For example, if the nodes are ordered according to the aggregates they belong to, the matrix Q has the form like

$$Q = \begin{pmatrix} 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ 0 & 0 & 1 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{7}$$

The aggregation algorithm for constructing Q is discussed in Section 3.

When Q is constructed, the coarse-level equation of (6) is given by

$$Q^T A \operatorname{diag}(x_i) Q e_c = 0, \tag{8}$$

where e_c is the coarse-level multiplicative error corresponding to e_i.

Then, the restriction operator R and the prolongation operator P are defined by $R = Q^T$ and $P = \operatorname{diag}(x_i)Q$. Equation (8) is rewritten as

$$RAPe_c = 0, \tag{9}$$

and the coarse-level operator A_c can be defined as

$$A_c = RAP. \tag{10}$$

If e_c is known, then an improved coarse-level approximation can be obtained by

$$x_c = \operatorname{diag}\left(P^T 1\right) e_c. \tag{11}$$

With (11), (9) is rewritten as the coarse-level probability equation:

$$A_c \left(\operatorname{diag}\left(P^T 1\right)\right)^{-1} x_c = 0. \tag{12}$$

Finally, (12) is used to accelerate the classical one-level iterative methods like Power method, (weighted) Jacobi method, or (weighted) Gauss-Seidel method for (3). We take the two-level case, for example.

A two-level aggregation/disaggregation method is applied after some relaxations on (3) at the fine level. Through constructing the aggregation Q, the coarse-level equation (12) is formatted, and we solve (12) to get the coarse-level approximation x_c. Then, a coarse-level correction

$$x_{i+1} = Pe_c = P\left(\operatorname{diag}\left(P^T 1\right)\right)^{-1} x_c \tag{13}$$

is implemented followed by some relaxations again to obtain the improved approximation at fine level.

Two-level iterative aggregation/disaggregation can be naturally extended to multiple levels by recursively applying the two-level method to solve the coarse-level probability equation (12). There are many types of recursive cycles such as V cycle which solves the coarse-level probability equation (12) once at each intermediate level of a cycle and W cycle which solves it twice. Here we only describe V cycle. It is illustrated by Figure 1.

Performance of this aggregation multigrid method for Markov chains is often unsatisfactory. Inspired by smoothed aggregation (SA) multigrid methods for linear systems, smoothing the interpolation and restriction operators, P and R, is proposed to improve the convergence behavior of the aggregation multigrid method [17].

All columns of the interpolation operator, P, are smoothed by (weighted) Jacobi method with weight ω:

$$P_s = \left(I - \omega D^{-1} A\right) P. \tag{14}$$

For the restriction operator, R, all rows are smoothed as

$$R_s = R\left(I - \omega A D^{-1}\right). \tag{15}$$

The smoothed operators P_s and R_s still hold the properties:

$$\begin{aligned} 1_c^T R_s &= 1^T, \\ P_s 1_c &= x, \\ &\text{for } x_i = x. \end{aligned} \tag{16}$$

When P_s and R_s are generated, the smoothed coarse-level operator is naturally defined as

$$A_{cs} = R_s A P_s. \tag{17}$$

Then the coarse-level probability equation is given by

$$A_{cs}\left(\operatorname{diag}\left(P_s^T 1\right)\right)^{-1} x_c = 0. \tag{18}$$

Note that because the smoothed coarse-level operator A_{cs} may lose the properties of an irreducible singular M-matrix, a lumping technique is necessary to modify A_{cs} to keep these properties. This technique is implemented before the coarse-level probability equation; refer to [17] for details. This smoothed aggregation multigrid (SAM) method for Markov chain can be formulated as Algorithm 1.

It is necessary to apply a direct solver for the coarsest level and the direct solver used in Algorithm 1 is based on the following theorem.

Theorem 1 (Theorem 4.16 in [21]). *If A is an irreducible singular M-matrix, then each of its principal submatrices other than A itself is a nonsingular M-matrix.*

Let A_L denote the coarsest-level operator; then we use the direct method presented in the coarsest-level Algorithm 2 to solve the coarsest-level equation $A_L x_L = 0$.

3. Neighborhood-Based Aggregation and Our Modified Version

How to construct the aggregation matrix Q is a crucial issue in algebraic multigrid method. Here, we consider the aggregation methods which determine aggregates based on strength of connection in matrices. There exist some corresponding aggregation methods like distance-one aggregation, distance-two aggregation [16, 17], pairwise aggregation, double pairwise aggregation [22], Neighborhood-Based aggregation [23], bottom-up aggregation [24], and some other types of aggregation. Serving as the aggregation method in [7, 18, 19], Neighborhood-Based aggregation has some advantages over others. It usually generates well-balanced aggregates of approximately equal size and reduces the number of unknowns quickly in coarsening process. Coarse-level stencil sizes tend to be rather uniform and do not grow too quickly [18].

Input:
A: problem matrix,
x: iteration solution,
v_1, v_2: pre and post relaxation steps.
Output:
x: iteration solution.
(1) **if** not on the coarsest level **then**
(2) $x \leftarrow \text{Relax}(A, x)$ v_1 times.
(3) Construct Q.
(4) $R \leftarrow Q^T$ and $P \leftarrow \text{diag}(x)Q$.
(5) $R_s \leftarrow$ smooth R, $P_s \leftarrow$ smooth P.
(6) $A_c \leftarrow$ lumping $R_s A P_s$, $x_c \leftarrow P_s' 1$.
(7) $x_c = \text{SAM}(A_c \,\text{diag}\,(x_c)^{-1}, x_c, v_1, v_2)$
(8) $x \leftarrow P_s(\text{diag}(P_s^T 1))^{-1} x_c$.
(9) $x \leftarrow \text{Relax}(A, x)$ v_2 times.
(10) **else**
(11) $x \leftarrow$ the solution of $Ax = 0$, $\|x\|_1 = 1$, $x > 0$ by a direct solver.
(12) **end if**
(13) **return** x;

Algorithm 1: SAM method for Markov chain (V cycle), $x \leftarrow \text{SAM}(A, x, v_1, v_2)$.

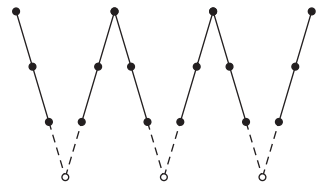


Figure 1: Multilevel V cycle: the finest-level operations are represented at the top, the coarsest-level operations are at the bottom, and intermediate-level operations are in between. The black dots represent relaxation operations on corresponding levels and the open dots represent direct solvers.

Here, the aggregation procedure is based on matrix $\overline{A} = A\,\text{diag}(x_i)$ and a node i is said to be strongly connected to node j in the graph of $\overline{A}$ if

$$\begin{aligned} -\overline{a}_{ij} &\geq \theta \max_{k \neq i} \{-\overline{a}_{ik}\} \\ \text{or} \quad -\overline{a}_{ji} &\geq \theta \max_{k \neq j} \left\{-\overline{a}_{jk}\right\}, \end{aligned} \tag{19}$$

where θ is a strength of connection parameter. The strong neighborhood N_i of node i is the set of all points which are strongly connected to i in the graph of $\overline{A}$ including node i itself. The Neighborhood-Based aggregation can be represented as Algorithm 3.

Though the Neighborhood-Based aggregation has many advantages, it is somewhat unreasonable in some detail.

In Algorithm 3, it picks $i \in \{1, 2, \ldots, n\}$ in a natural order and defines a new aggregate denoting the $(J+1)$th aggregate as the strong neighborhood of node i, if such strong neighborhood is contained in the left set: $R \setminus \bigcup_{j=1}^{J} Q_j$.

As for a strong neighborhood N_i of some node i, from (19) we can see that if the $\max_{k \neq i}\{-\overline{a}_{ik}\}$ is relatively small, then the strength of connection from other nodes in N_i to node i will be weak. Even worse, the nodes in N_i except i may be weakly connected with many others universally; that is to say, $-\overline{a}_{jk}$ is relatively small for many $j, k \in N_i$.

And if such node i is in the beginning of $\{1, 2, \ldots, n\}$, it would be aggregated earlier than most other nodes, and then it is very likely to construct an aggregate using such a neighborhood N_i described above.

This aggregate constructed through N_i is unreasonable. First, it contains many nodes which are neither strongly connected with i nor strongly connected with each other. Secondly, because it is aggregated earlier, it may separate some nodes which are truly strongly connected.

Here, we take a toy example to illustrate.

Example 2. Let $\overline{A}$ be an 8×8 matrix:

$$\begin{pmatrix} 0.07 & -0.01 & -0.01 & -0.01 & -0.01 & -0.01 & -0.01 & -0.01 \\ -0.01 & 0.94 & -0.3 & -0.3 & -0.3 & -0.01 & -0.01 & -0.01 \\ -0.01 & -0.3 & 0.94 & -0.3 & -0.3 & -0.01 & -0.01 & -0.01 \\ -0.01 & -0.3 & -0.3 & 0.94 & -0.3 & -0.01 & -0.01 & -0.01 \\ -0.01 & -0.3 & -0.3 & -0.3 & 0.94 & -0.01 & -0.01 & -0.01 \\ -0.01 & -0.01 & -0.01 & -0.01 & -0.01 & 0.95 & -0.45 & -0.45 \\ -0.01 & -0.01 & -0.01 & -0.01 & -0.01 & -0.45 & 0.95 & -0.45 \\ -0.01 & -0.01 & -0.01 & -0.01 & -0.01 & -0.45 & -0.45 & 0.95 \end{pmatrix}. \tag{20}$$

Input:
A: problem matrix,
Output:
x: the solution with $\|x\|_1 = 1$.
(1) Compute $N_L := \text{size}(A_L, 1)$;
(2) Determine $A_{Lp} := A_L(1 : N_L - 1, 1 : N_L - 1)$;
(3) Determine $b_{Lp} := -A_L(1 : N_L - 1, N_L)$;
(4) Compute $x_{Lp}(1 : N_L - 1) := A_{Lp} \setminus b_{Lp}$; let $x_{Lp}(N_L) = 1$;
(5) Set the coarsest-level solution $x = x_{Lp}/\|x_{Lp}\|_1$.
(6) **return** x;

Algorithm 2: Coarsest-level algorithm.

Input:
$\overline{A}$: scaled problem matrix,
θ: tolerance.
Output:
Q: aggregation matrix.
1st process: **generate the initial aggregates**
(1) Set $R \leftarrow \{1, 2, \ldots, n\}$ and $J \leftarrow 0$.
(2) **for** $i \in \{1, 2, \ldots, n\}$ **do**
(3) construct strong neighborhoods N_i based on (19).
(4) **if** $N_i \cap R = N_i$ **then**
(5) $J \leftarrow J + 1$.
(6) $Q_J \leftarrow N_i$
(7) $R \leftarrow R \setminus N_i$.
(8) **end if**
(9) **end for**
2nd process: **connect the remaining nodes to their most connected aggregates**
(10) **while** $R \neq \emptyset$ **do**
(11) pick $i \in R$ and set $j = \text{argmax}_{k=1,\ldots,J}\, card(N_i \cap Q_k)$.
(12) $Q_j \leftarrow Q_j \cup \{i\}$ and $R \leftarrow R \setminus \{i\}$.
(13) **end while**
3rd process: **construct the aggregation matrix** $Q \in R^{n \times J}$
(14) **for** $i \in \{1, 2, \ldots, n\}$ **do**
(15) **if** $i \in Q_j$, $j = 1, 2, \ldots, J$ **then**
(16) $Q_{ij} = 1$.
(17) **else**
(18) $Q_{ij} = 0$.
(19) **end if**
(20) **end for**
(21) **return** Q;

Algorithm 3: Neighborhood-Based aggregation, $\{Q_j\}_{j=1}^{J} \leftarrow$ Neighborhood-Based$(\overline{A}, \theta)$.

We can write this matrix as its block form:

$$\begin{pmatrix} \overline{a}_{11} & O & \\ & T_{2-5} & O \\ O & & \\ & O & T_{6-8} \end{pmatrix}, \tag{21}$$

where T_{2-5} denotes the square block containing elements in the columns and rows from 2nd to 5th of $\overline{A}$, T_{6-8} is defined correspondingly, and O means the elements of it are relatively very small.

This block form (21) can directly drive the reasonable aggregates for matrix $\overline{A}$. With the restriction that each aggregate must have at least two nodes, the reasonable aggregates should be constructed as follows: 2nd to 5th nodes form an aggregate; the 6th to 8th nodes form another aggregate; the first node is added to either of these two aggregates and where it is added makes little difference.

However, let us check which aggregates of this matrix will be constructed by Neighborhood-Based algorithm. We define the strong connection matrix S as follows:

For $i \neq j$,

$$s_{ij} = \begin{cases} 1, & \text{if } -\overline{a}_{ij} \geq \theta \max_{k \neq i} \{-\overline{a}_{ik}\} \text{ or } -\overline{a}_{ji} \geq \theta \max_{k \neq j} \{-\overline{a}_{jk}\}, \\ 0, & \text{otherwise.} \end{cases} \quad (22)$$

For $i = j$,

$$s_{ii} = 1. \quad (23)$$

So, $S_{ij} = 1$ means that node j is strongly connected with node i and is included in N_i. S is a symmetric matrix and the ith row or ith column represents the strong neighborhood N_i. In this example, S is

$$S = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}. \quad (24)$$

In Neighborhood-Based Algorithm 3, nodes are picked in a natural order, so the 1st node is first picked and its strong neighborhood N_1 is used as 1st aggregate. It is seen from S in (24) that all the other nodes are strongly connected to node 1. Adding 1st node itself, the strong neighborhood N_1 contains all the nodes of $\overline{A}$. The 1st aggregate is constructed containing all the nodes and the algorithm ends. It is too aggressive and unreasonable.

We consider that picking nodes in a natural order to construct aggregates is unreasonable. Here we consider picking nodes in another order.

We define a measurement size_nei(i) of node i as

$$\text{size\_nei}\,(i) := \text{length}\,(N_i), \quad (25)$$

which means the number of nodes in the strong neighborhood N_i of i.

We can suppose that the more the nodes the strong neighborhood N_i contains, the more likely the probabilities to be distributed evenly and be relatively smaller. So, we preferentially pick the nodes with smaller size_nei value to construct aggregates. We apply this strategy by reordering R by size_nei value to make the reordered set increasing in size_nei. This strategy can avoid generating the unreasonable aggregate in the 1st process: generate the initial aggregates, as the example shows.

However the Neighborhood-Based aggregation also has some defects in the 2nd process that may lead to bad aggregates.

As seen from the 2nd process of Algorithm 3, it picks the remaining nodes of 1st process and adds them to the "most connected" aggregates. We conclude that the reason one node i is left is because its strong neighborhood N_i has some common nodes with at least one of the existing aggregates and the "most connected" aggregate is defined as the aggregate which has the most common nodes with N_i.

However, when the algorithm considers which aggregate i should be added to, it ignores N_i itself. Consider a situation that N_i has a small number of common nodes with the existing aggregates compared to size of N_i. If we still add i to the "most connected" aggregate as Algorithm 3 does, it may cause some bad aggregates containing many nodes which should be separated because of the weak connection between each other.

We use Example 2 to illustrate again.

We change the order of picking nodes to avoid the problem caused by the 1st process discussed above. First the 6th node should be picked to construct aggregate in 1st process of Algorithm 3; because N_6 contains the 1st node and nodes from 6th to 8th, the 1st aggregate is constructed as $\{1, 6, 7, 8\}$. Then we pick nodes from 2nd to 5th. It can be seen from S in (24) that the strong neighborhoods of them are the same and contain the nodes from 1st to 5th. These strong neighborhoods have a common node with the 1st aggregate: 1st node, so the nodes from 2nd to 5th are all remaining nodes and should be added to the "most connected" aggregate. According to 2nd process of Algorithm 3, each node should be added to the aggregate which has most common nodes with its strong neighborhood. Thus, the result of the algorithm is also an aggregate with all the nodes, still too aggressive and unreasonable.

Here we propose an improved scheme to the 2nd process in Algorithm 3. Let RN_i denote the remaining strong neighborhood of the remaining note i as

$$RN_i = R \cap N_i, \quad (26)$$

which means removing the common nodes with the existing aggregates from N_i. And we consider RN_i as a candidate aggregate. Then we find the aggregates with most common nodes with N_i as previous. If the aggregate founded is in the existing aggregates, we add the node i to the corresponding aggregate. Otherwise, we use the remaining strong neighborhood RN_i which added the node i as a new aggregate, the $(J + 1)$th aggregate (J is the number of existing aggregates).

The whole algorithm is represented in Algorithm 4.

Note that the main extra work of our modified Neighborhood-Based Algorithm 4 compared to the old version consists of reordering the set R which can be solved in $O(\log_2 n)$ work in the first process, and the loss of the parallelization of the second process.

4. Cost-Effective SAM for Markov Chains

The smoothed aggregation multigrid for Markov chains described in Algorithm 1 is one type of adaptive algebraic multigrid algorithm. It constructs aggregation matrix Q based on the strength of connection in the scaled problem matrix $\overline{A} = A\,\text{diag}(x_i)$, without any need for advance

Input:
$\overline{A}$: scaled problem matrix,
θ: tolerance.
Output:
Q: aggregation matrix.
1st process: **generate the initial aggregates**
(1) Set $R \leftarrow \{1, 2, \ldots, n\}$ and $J \leftarrow 0$.
(2) **for** $i \in \{1, 2, \ldots, n\}$ **do**
(3) compute size_nei(i) based on (25).
(4) **end for**
(5) Reorder R by increasing size_nei values:
(6) $R \leftarrow$ new reordered R.
(7) **for** $i \in R$ **do**
(8) construct strong neighborhoods N_i based on (19)
(9) **if** $N_i \cap R = N_i$ **then**
(10) $J \leftarrow J + 1$.
(11) $Q_J \leftarrow N_i$.
(12) $R \leftarrow R \setminus N_i$.
(13) **end if**
(14) **end for**
2nd process: **connect the remaining nodes to their most connected aggregates**
(15) **while** $R \neq \emptyset$ **do**
(16) Pick $i \in R$ and set RN_i according to (26)
(17) $Q_{J+1} \leftarrow RN_i \setminus \{i\}$
(18) Set $j = \text{argmax}_{k=1,\ldots,J+1}\, card(N_i \cap Q_k)$.
(19) **if** $j == J + 1$ **then**
(20) $J = J + 1$.
(21) $Q_j \leftarrow Q_j \cup \{i\}$ and $R \leftarrow R \setminus RN_i$.
(22) **else**
(23) $Q_j \leftarrow Q_j \cup \{i\}$ and $R \leftarrow R \setminus \{i\}$.
(24) $Q_{J+1} = \{\}$
(25) **end if**
(26) **end while**
3rd process: construct the aggregation matrix $Q \in R^{n \times J}$
(27) **for** $i \in \{1, 2, \ldots, n\}$ **do**
(28) **if** $i \in Q_j$, $j = 1, 2, \ldots, J$ **then**
(29) $Q_{ij} = 1$.
(30) **else**
(31) $Q_{ij} = 0$.
(32) **end if**
(33) **end for**
(34) **return** Q;

Algorithm 4: Modified Neighborhood-Based aggregation, $Q \leftarrow$ MN-BA($\overline{A}, \theta$).

knowledge of the topology of the Markov chain. The aggregation process is fully adaptive, with aggregates constructed adaptively in each iteration and at all recursive levels. This type of multigrid method is powerful and scalable for many Markov chain problems compared to the classical AMG which use the operators without any update. But they are relatively more expensive because they construct the whole multigrid operators in every cycle and unfortunately the computational cost of constructing operators is accounted for a large proportion of the whole cost in a cycle. In [24], it is noted that, empirically, more than 50% of the computation time of each cycle is spent on coarse matrix construction. And through our observation, the most costly procedure is the aggregation procedure which constructs the matrix Q.

In the numerical experiments, there are some cases where the aggregation matrix Q at a certain level of current cycle is the same as the corresponding one in the previous cycle, so implementing the aggregation method to update Q is totally a waste. The Neighborhood-Based aggregation and our modified version depend on the strong neighborhood N_i of each node i and the ordering of picking nodes. And because the strong neighborhood N_i just depends on the strong connection matrix S in (22) and so does the reordering in our modified version, it can conclude that, with the same aggregation matrix S, the results after Neighborhood-Based aggregation or our modified version would be the same.

Actually, in Neighborhood-Based aggregation, it is necessary to construct strong neighborhood N_i of each node

Input:
A: problem matrix,
x: iteration solution,
ν_1, ν_2: pre and post relaxation steps.
Output:
x: iteration solution.
(1) **if** not on the coarsest level **then**
(2) $x \leftarrow \text{Relax}(A, x)$ ν_1 times.
(3) Construct S according to (22).
(4) **if** in the first cycle **then**
(5) $S_{\text{pre}} \leftarrow S$.
(6) Construct Q based on S, and $Q_{\text{pre}} \leftarrow Q$.
(7) **else if** $S == S_{\text{pre}}$ **then**
(8) $Q \leftarrow Q_{\text{pre}}$.
(9) **else**
(10) $S_{\text{pre}} \leftarrow S$.
(11) Construct Q based on S, and $Q_{\text{pre}} \leftarrow Q$.
(12) **end if**
(13) $R \leftarrow Q^T$ and $P \leftarrow \text{diag}(x)Q$.
(14) $R_s \leftarrow$ smooth R, $P_s \leftarrow$ smooth P.
(15) $A_c \leftarrow$ lumping $R_s A P_s$, $x_c \leftarrow P_s' 1$.
(16) $x_c = \text{CESAM}(A_c\, \text{diag}(x_c)^{-1}, x_c, \nu_1, \nu_2)$
(17) $x \leftarrow P_s(\text{diag}(P_s^T 1))^{-1} x_c$
(18) $x \leftarrow \text{Relax}(A, x)$ ν_2 times.
(19) **else**
(20) $x \leftarrow$ the solution of $Ax = 0$, $\|x\|_1 = 1$, $x > 0$ by a direct solver.
(21) **end if**
(22) **return** x;

Algorithm 5: Cost-effective SAM for Markov chains (CESAM), $x \leftarrow \text{CESAM}(A, x, \nu_1, \nu_2)$.

i and that is equal to constructing the strong connection matrix S because $N_i = S(i, :)$ according to the definition of N_i and S. So there is no extra work to construct S and there is only a need to add a judgment step. If the current S is the same as the corresponding one in previous cycle, there is no need to implement the remaining part of Neighborhood-Based aggregation and the corresponding aggregation matrix Q can still be used, leading to much time being saved in this procedure. Meanwhile, the storage will be increased; for that we also need to store the strong connection matrix S and the aggregation matrix Q at each level of previous cycle. The aggregation matrix Q is very sparse with only n nonzeros; n is the size of current level. The strong connection matrix S is symmetric, only the upper (lower) triangular part and the diagonal elements should be stored. The whole algorithm is presented in Algorithm 5.

Note that the aggregation methods based on the connection strength including Neighborhood-Based aggregation, our modified Neighborhood-Based aggregation, and distance-one (distance-two) aggregation are all suitable for Algorithm 5. And this cost-effective SAM (Algorithm 5) is mathematically equal to SAM (Algorithm 1).

5. Numerical Experiments

In this section, we report numerical results obtained using a Matlab R2014a implementation on 64-bit Windows-7 with a core i5-3470 processor and 8 GB RAM memory.

One purpose of our numerical tests is to examine whether our modified Neighborhood-Based aggregation can generate aggregates in a more reasonable way and bring accuracy improvement compared to the original version and to see if it is applicable for some case where the original Neighborhood-Based aggregation performs badly. Another purpose is to test the efficiency of our cost-effective SAM (referred as CESAM) for Markov chains. We also compared CESAM method with two standard methods, Power method and Gmres(50) method, to demonstrate the efficiency of it. Four Markov chain problems studied in [7] including 2 structured problems and 2 random walk on unstructured planar graph, two Markov chain examples from queueing models, and a typical Markov chain constructed by ourselves like Example 2 in Section 3 have been employed in our experiments. In our experiments, we use V cycle only. For convenience, we let CESAM_O and SAM_O denote the applications of CESAM and SAM with original Neighborhood-Based aggregation, respectively, and CESAM_M and SAM_M denote the applications of CESAM and SAM with our modified Neighborhood-Based aggregation, respectively.

Without loss of generality, special sets of the parameters are employed and they are taken from [7]. As mentioned early, the weighted Jacobi method is used as the pre- and postsmoothing in SAM (Algorithm 1) and CESAM (Algorithm 5). Let $\nu_1 = \nu_2 = 1$ and set the relaxation parameter $\omega = 0.7$ in the experiments. Note that, even though the values of parameters work well for all tests

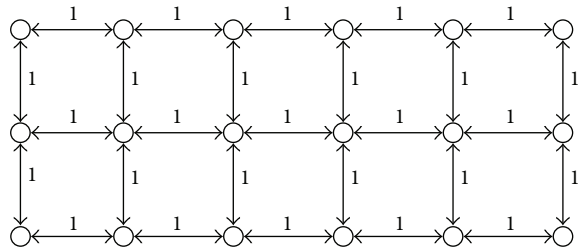


Figure 2: Graph for a 2D lattice with uniform weights.

Table 1: Numerical results for 2D lattice with uniform weights.

n	SAM_O			SAM_M			CESAM_O		CESAM_M	
	C_{op}	it	CPU_t	C_{op}	it	CPU_t	it	CPU_t	it	CPU_t
4096	1.34	32	1.532	1.39	31	1.643	32	0.554	31	0.642
16384	1.33	36	6.338	1.38	31	6.121	36	1.751	31	2.094
65536	1.34	37	26.477	1.38	31	24.882	37	8.992	31	8.837

Table 2: Numerical results for 2D lattice with uniform weights.

n	CESAM_M		Power		Gmres(50)	
	it	CPU_t	it	CPU_t	it	CPU_t
4096	31	0.642	$>10^6$	>254.7	734	2.264
16384	31	2.094	$>10^6$	>33.74	2087	17.44
65536	31	8.837	$>10^6$	>254.7	>5000	>108.6

considered here, they are likely to be problem-dependent. The coarsest-level solution is implemented by the coarsest-level Algorithm 2 when the size is below 50; the strength of connection parameter is chosen as $\theta = 0.25$. The initial guess is generated by random sampling with a uniform $(0, 1)$ distribution and normalized to one in the one norm for the first four Markov chains and is all equal to $1/n$ (n is the dimension of the Markov chain) in the last Markov chain. Iterations are terminated when $\text{res} = \|Ax_k\|_1/\|Ax_0\|_1 \leq 10^{-12}$ with x_k the current approximate solution and x_0 the initial guess solution. The max number of iterations is set as 200 for SAM and CESAM methods, 10^6 for Power method, and 5000 for Gmres(50) method.

Numerical results are reported in the tables, where "n" is the problem size, "it" denotes the iteration counts, and "CPU_t" denotes the total computing time in seconds. "C_{op}" is the operator complexity of the last cycle, which is defined as the sum of the number of nonzero entries in all operator A_l on all levels divided by the number of nonzero entries in the fine-level operator A_1; that is, $C_{\text{op}} = \sum_{l=1}^{L} nnz(A_l)/nnz(A_1)$ with $nnz(A_l)$ being the number of nonzero entries in A_l.

5.1. Structured Problems

5.1.1. Uniform 2D Lattice. The first test problem is a Markov chain on 2D lattice with uniform weights. Matrix A is essentially a scaled graph-Laplacian on a 2D uniform quadrilateral lattice with 5-point stencil (see Figure 2). It is a typically structured problem. Tables 1 and 2 give the numerical results.

Table 1 shows that, for this simple and highly structured problem, there is little difference in iterations between the Neighborhood-Based aggregation and our modified version. The operator complexity "C_{op}" of SAM_M is a little larger than that of SAM_O for the fact that our modified Neighborhood-Based aggregation is less aggressive than the original version. This larger operator complexity and the addition work of our modified Neighborhood-Based aggregation lead to a larger computing time per cycle. For our CESAM, it has the same "C_{op}" (which is not shown in the table) and iterations with SAM but converges in a much smaller time. The CESAM can save the calls of much work in aggregation procedure and this is problem-dependent. For example, for the size of 4096, the CESAM_O needs 35 times of implementing the remaining work of aggregation procedure while SAM_O needs 96 times. In addition, these savings most happen in finer level, resulting in a decrease of 63.8% proportion in computing time by CESAM_O when compared to SAM_O. It is similar for other sizes and for CESAM_M compared to SAM_M. Seen from Table 2, when compared to Power and Gmres(50) method, the CESAM_M shows better performance in both iterations and computing time.

5.1.2. Tandem Queuing Network. The next test problem is an open tandem queuing network from [10]. Tandem queuing network exists in some practical applications, for example, in wireless networks [25] and blood screening procedures [26]. Following the settings and description in [17], we consider the situation that there are two finite queues with single servers, customers arrive according to a Poisson distribution with rate μ, and the service time distribution at the two single-server stations is Poisson with rates μ_1 and μ_2. In our numerical simulation, we set the number of customers in the queues as $N = 63, 127, 255$. We set $\mu = 10$, $\mu_1 = 11$, and $\mu_2 = 10$. The states of this system can be represented by tuples (n_1, n_2), with n_1 denoting the number of customers waiting in the first queue and n_2 in the second queue. Then the total number of states is given by $(N + 1)^2$. It is also a structured problem and illustrated in Figure 3. The numerical results for this problem are presented in Tables 3 and 4.

Seen from Table 3, like the previous example, "C_{op}" of SAM with our modified Neighborhood-Based aggregation is still larger than that of SAM with Neighborhood-Based aggregation. This is mainly caused by the difference in the 2nd process of these two aggregation methods because these two test problems are very regular which means the strong

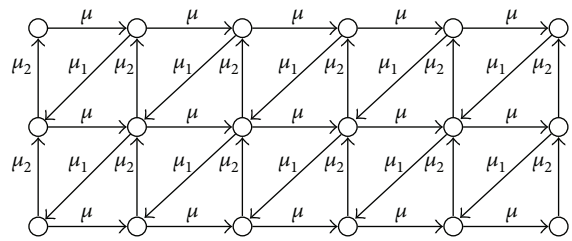


Figure 3: Graph for tandem queuing network.

Table 3: Numerical results for tandem queuing network.

n	SAM_O			SAM_M			CESAM_O		CESAM_M	
	C_{op}	it	CPU_t	C_{op}	it	CPU_t	it	CPU_t	it	CPU_t
4096	1.24	34	1.459	1.27	34	1.576	34	0.450	34	0.442
16384	1.24	33	5.007	1.27	32	5.494	33	1.188	32	1.226
65536	1.24	43	26.51	1.25	33	22.69	43	6.529	33	5.190

Table 4: Numerical results for tandem queuing network.

n	CESAM_M		Power		Gmres(50)	
	it	CPU_t	it	CPU_t	it	CPU_t
4096	34	0.442	$>10^6$	>9.469	1652	4.631
16384	32	1.226	$>10^6$	>28.44	>5000	>40.53
65536	33	5.190	$>10^6$	>193.0	>5000	>86.82

neighborhood of each node is almost with the same size. For this problem, the number of iterations is the same for these methods when the size is 16384 or 65536. For the same reason explained above, our SAM_M costs more time per cycle than SAM_O and this leads to a larger computing time of sizes 4096 and 16384 where the iterations are almost the same. For the size of 65536, the number of iterations is reduced by a larger proportion and this offsets the larger computing time per cycle, resulting in a smaller computing time. Similar to the previous example, CESAM converges in a much smaller time than SAM, with 69.2%–76.3% proportion of time saved by CESAM_O compared to SAM_O and 72.0%–77.7% proportion of time saved by CESAM_M compared to SAM_M. Note that because the CESAM reduces the calls of much work in aggregation procedure, the higher cost of modified Neighborhood-Based aggregation has less impact. If the saving in calls is almost the same, the CESAM_M with less iterations will be more efficient than CESAM_O; this can be verified by "CPU_t" of these two methods. The CESAM_M method still shows better performance than that of Power method and Gmres(50) method, which can be seen in Table 4.

5.1.3. M/H2/1 Queues. The next test problem is M/H2/1 queues from [10]. M/H2/1 queues are used in mobile communications [27]. M/H2/1 queue can be illustrated in Figure 4. Arrivals are generated according to a Poisson distribution at rate λ, while the service is represented by a two-phase hyperexponential distribution. With probability α a customer entering service receives service at rate μ_1, while with probability $1 - \alpha$ this customer receives service at rate μ_2. The numerical results for this problem are presented in Tables 5 and 6.

Table 5: Numerical results for M/H2/1 queues.

n	SAM_O			SAM_M			CESAM_O		CESAM_M	
	C_{op}	it	CPU_t	C_{op}	it	CPU_t	it	CPU_t	it	CPU_t
4001	1.43	32	1.706	1.25	28	1.507	32	0.438	28	0.374
20481	1.44	33	8.042	1.25	27	6.691	33	1.825	27	1.191

Table 6: Numerical results for M/H2/1 queues.

n	CESAM_M		Power		Gmres(50)	
	it	CPU_t	it	CPU_t	it	CPU_t
4001	28	0.374	$>10^6$	>9.906	>5000	>16.73
20481	27	1.191	$>10^6$	>39.36	>5000	>52.60

Table 5 shows that the iterations needed for SAM_M are less than those needed for SAM_O. This means the modified Neighbour-Based aggregation can generate more suitable aggregates than the original version. Less iterations also result in a smaller computing time of SAM_M and CESAM_M compared to SAM_O and CESAM_O, respectively. For the efficiency of the cost-effective strategy, we can see that CESAM converges in a much smaller time than SAM, with 74.3%–75.2% proportion of time saved by CESAM_O compared to SAM_O and 75.2%–82.2% proportion of time saved by CESAM_M compared to SAM_M. Finally, as Table 6 shows, the CESAM_M method still obtains much better performance for this problem than that of Power and Gmres(50) methods.

5.1.4. H2/E3/1 Queues. The final structured test problem is H2/E3/1 queues from [10]. H2/E3/1 queues are used in network engineering including computer communications networks [28]. The arrival process of the H2/E3/1 queue is a two-phase hyperexponential distribution while the service process is an Erlang-3 distribution. This queueing system can be shown graphically in Figure 5. For more details, see [10]. The numerical results for this problem are presented in Tables 7 and 8.

As seen from Table 7, the iterations needed for SAM_M are still less than those needed for SAM_O, which results in a smaller computing time of SAM_M and CESAM_M compared to SAM_O and CESAM_O, respectively. For the

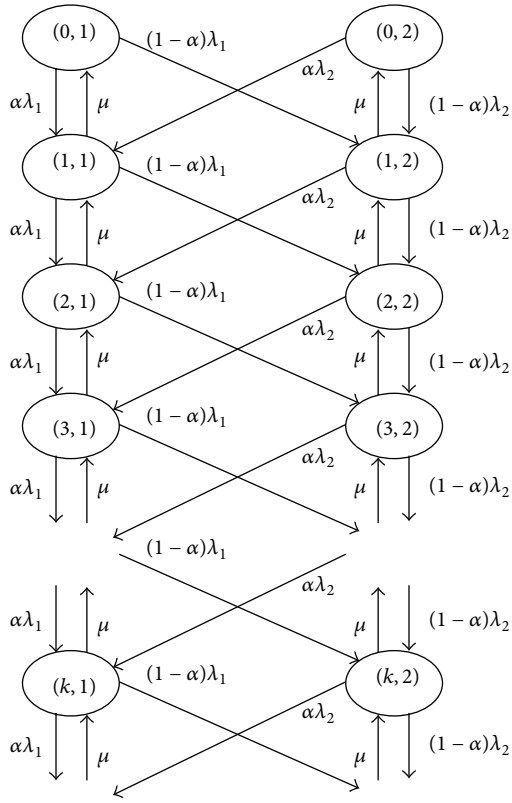


Figure 4: Graph for M/H2/1 queues.

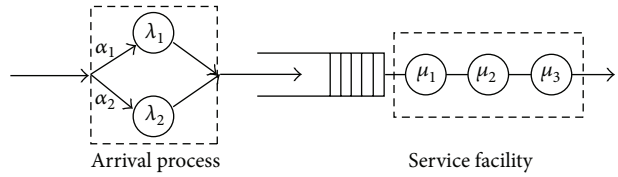


Figure 5: Graph for H2/E3/1 queues.

efficiency of the cost-effective strategy, we can see that CESAM converges in a much smaller time than SAM, with 67.5%–72.0% proportion of time saved by CESAM_O compared to SAM_O and 60.3%–60.4% proportion of time saved by CESAM_M compared to SAM_M. Finally, as Table 8 shows, the CESAM_M method still obtains much better performance for this problem than that of Power and Gmres(50) methods. For these four unstructured test problems, the SAM_M shows higher numerical scalability than SAM_O since the iterations needed almost keep the same when the sizes of the problems increase.

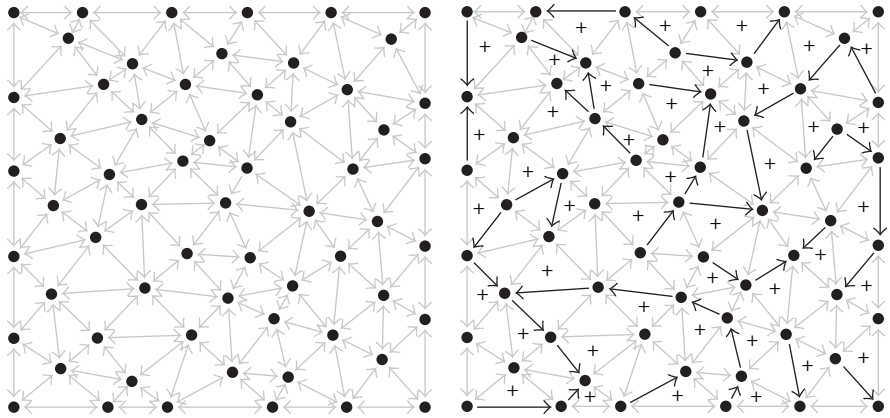
5.2. Unstructured Problems

5.2.1. Random Planar Graph (Undirected). Random walks on graphs play important roles in several fields, including information retrieval and statistical physics. For example, many ranking problems, for example, PageRank [2] and GeneRank [3] corresponding to directed and undirected graphs, respectively, can be modeled by random walks on graphs which represent relations between the items. First, we consider an unstructured planar (undirected) graph and calculate the stationary probability distribution of the random walk on the graph. We randomly distribute n nodes in $(0, 1)^2$. Then a planar graph connecting these nodes is constructed by using Delaunay triangulation and each node that shares an edge within the triangulation is connected by bidirectional links. The probability of transition from node i to node j is given by the reciprocal of the number of outward links from node i. See Figure 6(a) for a small version of this example.

Table 7: Numerical results for H2/E3/1 queues.

n	SAM_O			SAM_M			CESAM_O		CESAM_M	
	C_{op}	it	CPU_t	C_{op}	it	CPU_t	it	CPU_t	it	CPU_t
6146	1.95	29	2.382	1.80	24	1.999	29	0.775	24	0.794
24578	1.95	39	12.10	1.81	23	7.368	39	3.386	23	2.915
120002	2.00	61	101.01	1.83	52	93.07	61	30.16	52	21.01

Table 9 shows that, for this unstructured problem, much iterations are saved using our modified Neighborhood-Based



(a) (b)

Figure 6: Graphs for small versions of *random planar graph* (a) and *random planar graph*, *nonsymmetric* (b). Black dots represent nodes, and light gray arrows represent bidirectional links. For (b), black arrows represent unidirectional links and triangles with "+" inside have a single link that was made unidirectional. For easier visualization, the graphs shown here have a more regular distribution of points than the actual points used to build the Markov chains.

Table 8: Numerical results for H2/E3/1 queues.

n	CESAM_M		Power		Gmres(50)	
	it	CPU_t	it	CPU_t	it	CPU_t
6146	24	0.794	$>10^6$	>13.55	>5000	>18.77
24578	23	2.915	$>10^6$	>46.98	>5000	>64.08
120002	52	21.01	$>10^6$	>469.2	>5000	>201.3

Table 9: Numerical results for random planar graph (undirected).

n	SAM_O			SAM_M			CESAM_O		CESAM_M	
	C_{op}	it	CPU_t	C_{op}	it	CPU_t	it	CPU_t	it	CPU_t
4096	1.25	59	2.506	1.66	30	1.923	59	1.034	30	0.748
16384	1.25	81	13.08	1.72	31	8.464	81	5.190	31	3.030
65536	1.25	92	64.68	1.74	31	45.05	92	27.94	31	16.31

aggregation and this saving is more remarkable than that in previous examples of structured problems. Also note that the difference between the operator complexity "C_{op}" of SAM_O and SAM_M is larger than that in previous examples. We think the reason is that, for unstructured problems, modified Neighborhood-Based aggregation can generate more reasonable aggregates with less strong connections that are separated and less weak connections that are gathered, which leads to a much faster convergence rate and usually much more aggregates with smaller size. For this problem, 23.3%–35.3% proportion of computing time is reduced by our modified Neighborhood-Based aggregation compared to Neighborhood-Based aggregation within SAM. Similar to previous examples, there is a considerable saving in computing time by our CESAM, and this saving is more remarkable in the case of using modified Neighborhood-Based aggregation. For the size of 65536, the computing time of SAM_M is 69.7% as large as that of SAM_O, but the computing time of CESAM_M is 58.4% as large as that of CESAM_O. As shown in Table 10, when compared with Power and Gmres(50) methods, the CESAM_M still gets the best performance.

Table 10: Numerical results for random planar graph (undirected).

n	CESAM_M		Power		Gmres(50)	
	it	CPU_t	it	CPU_t	it	CPU_t
4096	30	0.748	14434	2.586	430	1.309
16384	31	3.030	50070	36.59	995	8.634
65536	31	16.31	$>10^6$	>545.8	2831	47.70

5.2.2. Random Planar Graph (Directed). Then, we consider directed unstructured problems. The unstructured planar graphs from the previous example are used to form a similar problem with nonsymmetric sparsity structure. Based on the graphs described in the previous example, we select a subset of triangles from the triangulation such that no two triangles in the set are neighbored. This is done by randomly selecting a triangle, marking it with "+" and marking all of its three neighbors with "−". Then repeat this process for the next unmarked triangle until all triangles are marked. Then we randomly delete one of the six directed arcs that connect the three nodes in each "+" triangle to destroy the symmetry. This process ensures that the resulting Markov chain is still irreducible. The probability of transition from node i to node j is given by the reciprocal of the number of outward links from node i. See Figure 6(b) for a small version of this example with the "+" triangles marked.

Seen from Table 11, for this nonsymmetric unstructured case, the behavior of these methods is very similar to that in previous example. Also much iterations are saved using our modified Neighborhood-Based aggregation. And the difference between the operator complexity "C_{op}" of SAM_O and SAM_M is still large. For this problem,

Table 11: Numerical results for random planar graph (directed).

n	SAM_O			SAM_M			CESAM_O		CESAM_M	
	C_{op}	it	CPU_t	C_{op}	it	CPU_t	it	CPU_t	it	CPU_t
1024	1.20	69	0.947	1.61	27	0.528	69	0.461	27	0.246
4096	1.24	69	2.917	1.66	29	1.872	69	1.184	29	0.742
16384	1.24	72	11.50	1.70	31	8.232	72	4.549	31	2.883

Table 12: Numerical results for random planar graph (directed).

n	CESAM_M		Power		Gmres(50)	
	it	CPU_t	it	CPU_t	it	CPU_t
1024	27	0.246	4375	0.242	173	0.139
4096	29	0.742	17030	3.077	377	1.286
16384	31	2.883	63051	45.93	1146	10.72

28.4%–44.2% proportion of computing time is reduced by our modified Neighborhood-Based aggregation compared to Neighborhood-Based aggregation within SAM. When CESAM is used, computing time is reduced by 51.3%–65.0% and this proportion increases with the problem size. As shown in Table 12, when the problem size is as small as 1024, the CESAM_M gets the worst performance compared with Power and Gmres(50) methods. When the size comes to 4096 and 16384, the CESAM_M outperforms the Power and Gmres(50) methods and this advantage increases with the problem size since CESAM_M is much more scalable.

5.3. Extreme Example. The last test problem is a Markov chain matrix constructed by ourselves. It is similar to Example 2 in Section 3 but we expand the size to 1024. It is a typical NCD (nearly completely decomposable) Markov chain which consists of groups of nodes that are strongly connected with each other and very weakly connected to the nodes in other groups. Here we suppose that there exists a node which is very weakly connected to all the other nodes, and the transition probabilities between it and other nodes are distributed evenly. For convenience, we constructed this matrix as

$$\begin{pmatrix} 0.9 & 1e-4 & 1e-4 \\ \frac{0.1}{1023} & A & O_1 \\ \frac{0.1}{1023} & O_2 & B \end{pmatrix}, \tag{27}$$

where

$$A = \begin{pmatrix} 3.9e-3 & \cdots & \left(\frac{9.9}{599}\right)e-1 \\ \vdots & \ddots & \vdots \\ \left(\frac{9.9}{599}\right)e-1 & \cdots & 3.9e-3 \end{pmatrix}, \tag{28}$$

Table 13: Numerical results of this extreme example.

n	CESAM_O				CESAM_M			
	C_{op}	Coarsest-size	it	CPU_t	C_{op}	Coarsest-size	it	CPU_t
1024	1.00	1	>200	>50.93	1.00	2	6	1.66

with size of 600×600,

$$B = \begin{pmatrix} 1.9e-3 & \cdots & \left(\frac{9.9}{422}\right)e-1 \\ \vdots & \ddots & \vdots \\ \left(\frac{9.9}{422}\right)e-1 & \cdots & 1.9e-3 \end{pmatrix}, \tag{29}$$

with size of 423×423,

$$O_1 = \begin{pmatrix} \left(\frac{4}{300}\right)e-3 & \cdots & \left(\frac{4}{300}\right)e-3 \\ \vdots & \ddots & \vdots \\ \left(\frac{4}{300}\right)e-3 & \cdots & \left(\frac{4}{300}\right)e-3 \end{pmatrix}, \tag{30}$$

with size of 600×423,

$$O_2 = \begin{pmatrix} \left(\frac{6}{423}\right)e-3 & \cdots & \left(\frac{6}{423}\right)e-3 \\ \vdots & \ddots & \vdots \\ \left(\frac{6}{423}\right)e-3 & \cdots & \left(\frac{6}{423}\right)e-3 \end{pmatrix}, \tag{31}$$

with size of 423×600.

Although this situation seems extreme, it represents the similar cases which will possibly happen in the matrices of Markov chains locally. So we hope to evaluate the rationality and the universality of the original Neighborhood-Based aggregation and our modified version through this simple test matrix. Result of numerical experiments is presented in Table 5. Since we focus on the convergence rate affected by original Neighborhood-Based aggregation and modified Neighborhood-Based aggregation and there is no difference between CESAM and SAM, only the CESAM with these two aggregation methods are implemented for this example.

From Table 13, we can observe that the computing time and the iteration counts are reduced tremendously when our modified Neighborhood-Based aggregation is used. The performance of the original Neighborhood-Based aggregation over this problem is unacceptable which means it is not applicable for this problem, but our modified Neighborhood-Based aggregation can figure it out efficiently. The "coarsest-size" in Table 5 shows that the original Neighborhood-Based aggregation generates 1 aggregate containing all the nodes for this test matrix and our modified Neighborhood-Based aggregation generates 2 aggregates. Seen from (27), this test matrix should be divided into 2 aggregates as our modified Neighborhood-Based aggregation did. Because many nodes with weak connections are gathered together

by Neighborhood-Based aggregation, the restriction and prolongation operators are very rough, leading to a poor performance as shown in Table 5. In application examples, especially in unstructured problems, the situations like this may exist locally and our modified Neighborhood-Based aggregation will be more suitable.

6. Conclusions

In this paper, we have studied the Neighborhood-Based aggregation and explained the shortage and the irrationality of it. Then a modified Neighborhood-Based aggregation is presented. We also propose a CESAM method which reduces the calls of much work in aggregation procedure with the same convergence rate of SAM. The numerical results confirm that the modified Neighborhood-Based aggregation method is much more efficient than Neighborhood-Based aggregation for unstructured problems and with stronger applicability, and a significant saving in computing time is achieved by CESAM method. The CESAM_M method also shows good potential when compared with other methods for Markov chain problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is supported by NSFC (61370147, 61170309) and the Fundamental Research Funds for the Central Universities (ZYGX2013Z005).

References

[1] B.-Y. Pu, T.-Z. Huang, and C. Wen, "A preconditioned and extrapolation-accelerated GMRES method for PageRank," *Applied Mathematics Letters*, vol. 37, pp. 95–100, 2014.

[2] B.-Y. Pu, T.-Z. Huang, C. Wen, and Y.-Q. Lin, "The extrapolation-accelerated multilevel aggregation method in PageRank Computation," *Mathematical Problems in Engineering*, vol. 2013, Article ID 525313, 8 pages, 2013.

[3] M. Benzi and V. Kuhlemann, "Chebyshev acceleration of the GeneRank algorithm," *Electronic Transactions on Numerical Analysis*, vol. 40, pp. 311–320, 2013.

[4] L. R. Rabiner, "Tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[5] A. E. Raftery, "A model for high-order Markov chains," *Journal of the Royal Statistical Society—Series B: Methodological*, vol. 47, no. 3, pp. 528–539, 1985.

[6] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, NY, USA, 1985.

[7] H. D. Sterck, K. Miller, T. Manteuffel, and G. Sanders, "Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains," *Advances in Computational Mathematics*, vol. 35, no. 2, pp. 375–403, 2011.

[8] A. Brandt, S. McCormick, and J. Ruge, "Algebraic multigrid (AMG) for sparse matrix equations," in *Sparsity and Its Applications*, D. J. Evans, Ed., 1984.

[9] A. Brandt, "Algebraic multigrid theory: the symmetric case," *Applied Mathematics and Computation*, vol. 19, no. 1–4, pp. 23–56, 1986.

[10] W. J. Stewart, *An Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, USA, 1994.

[11] B. Philippe, Y. Saad, and W. J. Stewart, "Numerical methods in Markov chain modeling," *Operations Research*, vol. 40, no. 6, pp. 1156–1179, 1992.

[12] Y. Takahashi, "A lumping method for numerical calculations of stationary distributions of Markov chains," Research Report B-18, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1975.

[13] S. T. Leutenegger and G. Horton, "On the utility of the multilevel algorithm for the solution of nearly completely decomposable Markov chains," in *Numerical Solution of Markov Chains*, W. Stewart, Ed., pp. 425–443, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[14] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, "Adaptive smoothed aggregation (αSA) multigrid," *SIAM Review*, vol. 47, no. 2, pp. 317–346, 2005.

[15] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. McCormick, and J. Ruge, "Adaptive algebraic multigrid," *SIAM Journal on Scientific Computing*, vol. 27, no. 4, pp. 1261–1286, 2006.

[16] H. D. Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, and J. Ruge, "Multilevel adaptive aggregation for Markov chains, with application to web ranking," *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2235–2262, 2007.

[17] H. de Sterck, T. A. Manteuffel, S. F. McCormick et al., "Smoothed aggregation multigrid for markov chains," *SIAM Journal on Scientific Computing*, vol. 32, no. 1, pp. 40–61, 2010.

[18] H. de Sterck, K. Miller, G. Sanders, and M. Winlaw, "Recursively accelerated multilevel aggregation for markov chains," *SIAM Journal on Scientific Computing*, vol. 32, no. 3, pp. 1652–1671, 2010.

[19] C. Wen, T.-Z. Huang, D.-A. Wu, and L. Li, "The finest level acceleration of multilevel aggregation for Markov chains," *International Journal of Numerical Analysis and Modeling. Series B*, vol. 2, no. 1, pp. 27–41, 2011.

[20] E. Treister and I. Yavneh, "On-the-fly adaptive smoothed aggregation multigrid for Markov chains," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2927–2949, 2011.

[21] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, SIAM, Philadelphia, Pa, USA, 1987.

[22] Y. Notay, "An aggregation-based algebraic multigrid method," *Electronic Transactions on Numerical Analysis*, vol. 37, pp. 123–146, 2010.

[23] P. Vaněk, J. Mandel, and M. Brezina, "Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems," *Computing*, vol. 56, no. 3, pp. 179–196, 1996.

[24] E. Treister and I. Yavneh, "Square and stretch multigrid for stochastic matrix eigenproblems," *Numerical Linear Algebra with Applications*, vol. 17, no. 2-3, pp. 229–251, 2010.

[25] L. Le and E. Hossain, "Tandem queue models with applications to QoS routing in multihop wireless networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 8, pp. 1025–1040, 2008.

[26] S. K. Bar-Lev, H. Blanc, O. Boxma, G. Janssen, and D. Perry, "Tandem queues with impatient customers for blood screening procedures," *Methodology and Computing in Applied Probability*, vol. 15, no. 2, pp. 423–451, 2013.

[27] F. Barcelb, J. Paradells, and M. Aguilar, "Mean waiting time in the $M/H_2/s$ queue: application to mobile communications systems," in *Proceedings of the 15th IMACS World Congress (IMACS '97)*, pp. 24–29, Berlin, Germany, August 1997.

[28] N. Kryvinska and D. V. Thanh, "Enabling voice services in hybrid/cross network framework," in *Proceedings of the London Communications Symposium (LCS '06)*, pp. 57–60, London, UK, September 2006.