SIAM J. SCI. COMPUT.
Vol. 32, No. 3, pp. 1652–1671

RECURSIVELY ACCELERATED MULTILEVEL AGGREGATION FOR MARKOV CHAINS*

H. DE STERCK†, K. MILLER†, G. SANDERS‡, AND M. WINLAW†

Abstract. A recursive acceleration method is proposed for multiplicative multilevel aggregation algorithms that calculate the stationary probability vector of large, sparse, and irreducible Markov chains. Pairs of consecutive iterates at all branches and levels of a multigrid W cycle with simple, nonoverlapping aggregation are recombined to produce improved iterates at those levels. This is achieved by solving quadratic programming problems with inequality constraints: the linear combination of the two iterates is sought that has a minimal two-norm residual, under the constraint that all vector components are nonnegative. It is shown how the two-dimensional quadratic programming problems can be solved explicitly in an efficient way. The method is further enhanced by windowed top-level acceleration of the W cycles using the same constrained quadratic programming approach. Recursive acceleration is an attractive alternative to smoothing the restriction and interpolation operators, since the operator complexity is better controlled and the probabilistic interpretation of coarse-level operators is maintained on all levels. Numerical results are presented showing that the resulting recursively accelerated multilevel aggregation cycles for Markov chains, combined with top-level acceleration, converge significantly faster than W cycles and lead to close-to-linear computational complexity for challenging test problems.

Key words. multilevel method, aggregation, Markov chain, stationary probability vector, quadratic programming, algebraic multigrid

AMS subject classifications. 65C40, 60J22, 65F10, 65F15

DOI. 10.1137/090770114

1. Introduction.

In this paper, we consider multilevel iterative methods for the numerical calculation of the stationary probability vector of large, sparse, and irreducible Markov matrices [1]. Let $B \in \mathbb{R}^{n\times n}$ be a column-stochastic matrix, i.e., $0 \le b_{ij} \le 1$ for all i, j, and

$$\mathbf{1}^T B = \mathbf{1}^T, \tag{1.1}$$

with $\mathbf{1}$ the column vector of all ones. We seek a vector $\mathbf{x} \in \mathbb{R}^n$ that satisfies

$$B\,\mathbf{x} = \mathbf{x}, \qquad x_i \ge 0\ \forall i, \qquad \|\mathbf{x}\|_1 = 1. \tag{1.2}$$

If B is irreducible, then there exists a unique solution $\mathbf{x}$ to (1.2). (Matrix B is irreducible *iff* there exists a path from each node i to each node j in the directed graph of matrix B.) Moreover, this stationary probability vector $\mathbf{x}$ then satisfies the strict inequality $x_i > 0$ for all i [1]. All eigenvalues λ_i of B satisfy $|\lambda_i| \le 1$, and $\lambda_1 = 1$ is an eigenvalue of B. The power method for calculating $\mathbf{x}$ converges very slowly when the subdominant eigenvalue satisfies $|\lambda_2| \to 1$ as the problem size increases, since the error component associated with the subdominant eigenvalue is damped very slowly [2, 3]. The Markov chain is said to be *slowly mixing* in this case. Multilevel iterative methods aim to accelerate convergence for this type of problem by reducing

*Received by the editors September 3, 2009; accepted for publication (in revised form) March 25, 2010; published electronically June 9, 2010.
http://www.siam.org/journals/sisc/32-3/77011.html
†Department of Applied Mathematics, University of Waterloo, Waterloo, ON N2L 3G1, Canada (hdesterck@uwaterloo.ca, k7miller@uwaterloo.ca, mwinlaw@uwaterloo.ca).
‡Department of Applied Mathematics, University of Colorado at Boulder, Boulder, CO 80309 (sandersg@colorado.edu).

error components with different scales on progressively coarser levels. (Note that the weighted power method can sometimes be used to obtain satisfactory convergence for problems for which some eigenvalues $|\lambda| \to 1$, namely, for cases where the slowly converging components have eigenvalues with real parts that are significantly smaller than one.)

Methods based on aggregation of Markov states have proven to be a fruitful approach to accelerating convergence for slowly mixing Markov chains. In these methods, aggregates of Markov states are formed and a coarse-level transition matrix is constructed using basic probability calculus that describes the transition probabilities between the aggregated states. Iteration on the fine level is accelerated by iteration on the coarse level using the coarse-level transition matrix, followed by multiplicative correction on the fine level. The earliest work along these lines is Takahashi's two-level iterative aggregation/disaggregation method for Markov chains [4], and two-level aggregation/disaggregation has been studied extensively ever since [5, 6, 7, 8, 9, 10, 11, 3]. Convergence proofs are given for two-level aggregation/disaggregation methods in [9, 11].

Two-level iterative aggregation/disaggregation can naturally be extended to multiple levels, along the lines of multigrid methods for linear systems of equations [12]. Direct extension of two-level aggregation/disaggregation to multiple levels was first explored in [13, 14] and later in [15]. In the latter, aggregates are formed algebraically based on strength of connection in the problem matrix column-scaled by the current iterate. Coarse grids on all levels are formed adaptively, based on the current iterate, and are different in each cycle. The resulting methods are similar to the adaptive smoothed aggregation (SA) and adaptive algebraic multigrid (AMG) methods of [16] and [17]. However, numerical results in [18] show that the resulting multilevel aggregation method, while improving on two-level aggregation results, does not give satisfactory convergence for many slowly mixing Markov chains: the number of multigrid iterations required for convergence grows significantly as a function of problem size, resulting in computational complexity that is much worse than the optimal $O(n)$ complexity (linear in the number of unknowns). For these problems, so-called W cycles, in which coarser levels are visited increasingly often, do not succeed in fully restoring optimal convergence [18].

In this paper, we propose a new way to accelerate multilevel aggregation for Markov chains. We start from the multiplicative nonoverlapping aggregation method of [13, 14, 15], but we consider W cycles and use acceleration on all recursive levels by selecting the best (in a sense to be defined below) linear combination of the two iterates generated at all levels and branches of the W cycle. Our approach is inspired by the work on K-cycle (which stands for Krylov cycle) multigrid by Notay and Vassilevski [19] (see also [20]) and is also closely related to recursive Krylov acceleration methods for nonlinear multigrid proposed by Oosterlee and Washio [21]. The idea of accelerating multilevel methods by combining iterates recursively on all levels is thus not new; in fact, [21, 19] contain references to earlier work in which this was already explored. What is different in our approach is that we solve a quadratic programming problem with inequality constraints on all levels: we solve for the linear combination of the two iterates that has a minimal two-norm residual, under the constraint that all vector components are nonnegative. (We thus do not, as in [21, 19, 20], use Krylov-type linear system accelerators like conjugate gradient (CG) or generalized minimal residual (GMRES) or their so-called flexible variants.) We show how, for the case of combining two previous iterates, the constrained quadratic programming problem can

be solved explicitly in an efficient way. The method is further enhanced by windowed top-level acceleration of the W cycles using the same constrained quadratic programming approach. We present numerical results showing that the resulting recursively accelerated multilevel aggregation (RAMA) cycles for Markov chains, combined with top-level acceleration, converge significantly faster than W cycles and lead to close-to-linear computational complexity for challenging test problems. Note that we do not call our multigrid cycles K cycles, since we do not employ a Krylov method for acceleration. In a generalized sense, however, they could also be called K cycles.

The RAMA method proposed in this paper compares favorably with other methods that have been developed to accelerate the multilevel aggregation algorithm of [13, 14, 15], including two approaches we have described in recent work. The first of these employs smoothing of the restriction and interpolation operators using the problem matrix [18]. The second employs algebraic multigrid coarsening and interpolation [22] rather than aggregation to obtain coarse problem formulations while maintaining the adaptive multiplicative setting of the above-discussed aggregation methods for Markov chains [23]. Both of these methods lead to near-optimal computational complexity. They both can be interpreted as employing overlapping aggregates to accelerate convergence (the nonzeros in the columns of the interpolation operator overlap). While this leads in both cases to near-optimal convergence properties, these methods share two important disadvantages. First, the probabilistic interpretation of the coarse-level operators is lost, and an ad hoc lumping procedure is required to maintain the singular M-matrix nature of the coarse-level operators [18, 23], which is a somewhat unnatural fix. Second, for some problems the memory and execution time complexity per multigrid V cycle (as measured by the so-called operator complexity; see section 4) can become very large with these methods, in particular the smoothed aggregation method [18]. Our new RAMA method is attractive because it is conceptually simpler than SA or AMG for Markov chains [18, 23] and easier to implement, and it is built on a method for which two-level convergence proofs are available [9, 11]. There are no overlapping aggregates, which makes the operator complexity better controlled, and the probabilistic interpretation of coarse-level operators is maintained on all levels. As such, our RAMA method is a direct extension of the aggregation/disaggregation idea originally introduced by Takahashi [4], in that our coarse-level operators are obtained by the standard aggregation mechanism, with its probabilistic interpretation. Our recursive acceleration improves the convergence of the multilevel aggregation approach of [13, 14, 15] significantly, maintaining the probabilistic interpretation of the coarse-level operators. Finally, we also want to mention some recent different approaches for accelerating the multiplicative aggregation method for Markov chains, including the so-called square-and-stretch multigrid algorithm by Treister and Yavneh, which has shown good performance in a variety of test problems [24].

The remainder of this paper is organized as follows. In section 2, we briefly review the multilevel aggregation algorithm for Markov chains from [13, 14, 15]. The new recursively accelerated multilevel aggregation algorithm is described in section 3, and numerical results are presented in section 4. Conclusions are formulated in section 5.

2. Multilevel aggregation for Markov chains.

2.1. Multilevel aggregation algorithm. In this section, we briefly review the multilevel aggregation algorithm for Markov chains from [13, 14, 15]. We follow the presentation of [18].

Let $A = I - B$ and rewrite $B\,\mathbf{x} = \mathbf{x}$ as

$$A\,\mathbf{x} = 0. \tag{2.1}$$

We then rewrite the exact solution, $\mathbf{x}$, in terms of the current approximate, $\mathbf{x}_i$, and its multiplicative error, $\mathbf{e}_i$, as $\mathrm{diag}(\mathbf{x}_i)\,\mathbf{e}_i$, obtaining

$$A\,\mathrm{diag}(\mathbf{x}_i)\,\mathbf{e}_i = 0. \tag{2.2}$$

Note that we have to assume here that all components of the current approximate, $\mathbf{x}_i$, are nonzero (the Perron–Frobenius theory guarantees that the exact solution, $\mathbf{x}$, also has this property; see [1]). At convergence, the multiplicative error is $\mathbf{e}_i = \mathbf{1}$.

The n fine-level degrees of freedom are aggregated into m groups according to the columns of aggregation matrix $Q \in \mathbb{R}^{n\times m}$, where $q_{ij} = 1$ if fine-level node i belongs to aggregate j and $q_{ij} = 0$ otherwise. For example, if the fine-level degrees of freedom are ordered according to the aggregates they belong to, then Q has the form

$$Q = \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ \hline 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ \hline 0 & 0 & 1 & \cdots \\ 0 & 0 & 1 & \cdots \\ \hline 0 & 0 & 0 & \cdots \\ \hline \vdots & \vdots & \vdots & \ddots \end{array}\right]. \tag{2.3}$$

In our algebraic multilevel methods, the aggregates are determined at each level using strength of connection in the scaled problem matrix, $A\,\mathrm{diag}(\mathbf{x}_i)$. The details of the aggregation algorithm we use are explained below in section 2.2.

Once Q has been determined, a coarse-level version of (2.2) is constructed:

$$Q^T A\,\mathrm{diag}(\mathbf{x}_i)\,Q\,\mathbf{e}_c = 0, \tag{2.4}$$

where $\mathbf{e}_c$ represents the coarse-level approximation of unknown fine-level multiplicative error $\mathbf{e}_i$.

Define the restriction and prolongation operators, R and P, by $R = Q^T$ and $P = \mathrm{diag}(\mathbf{x}_i)\,Q$, and write $R\,A\,P\,\mathbf{e}_c = 0$ with the coarse-level operator, A_c, defined by

$$A_c = R\,A\,P. \tag{2.5}$$

Note that $P^T\,\mathbf{1} = R\,\mathbf{x}_i = Q^T\,\mathbf{x}_i$ is the restriction of current fine-level approximate $\mathbf{x}_i$ to the coarse level. The coarse-level error, $\mathbf{e}_c$, can be used to define an improved coarse-level approximation, $\mathbf{x}_c = \mathrm{diag}(Q^T\,\mathbf{x}_i)\,\mathbf{e}_c$, leading to the coarse-level probability equation

$$A_c\,(\mathrm{diag}(Q^T\,\mathbf{x}_i))^{-1}\,\mathbf{x}_c = 0. \tag{2.6}$$

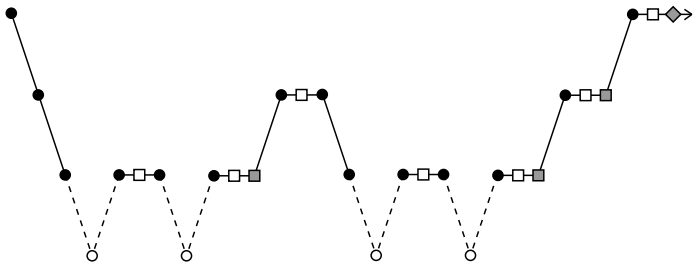

FIG. 2.1. *RAMA cycle: recursively accelerated W cycle. Fine-grid operations are represented at the top of the diagram, coarse-grid operations are on the bottom, and intermediate-grid operations are in between. The black dots represent relaxation operations on their respective grids, and the open dots represent coarse-level solves. The iterates represented by two consecutive white square boxes on each level are combined to produce a new accelerated iterate (gray square boxes). Several top-level iterates (white square box) may be recombined to produce improved iterates on the top level (gray diamond).*

We define the coarse-level stochastic matrix, B_c, as

$$B_c = Q^T B \,\mathrm{diag}(\mathbf{x}_i)\, Q\, (\mathrm{diag}(Q^T \mathbf{x}_i))^{-1}. \tag{2.7}$$

This matrix is nonnegative and satisfies $\mathbf{1}_c^T B_c = \mathbf{1}_c^T$, with the coarse-level vector of all ones denoted by $\mathbf{1}_c$. We then obtain

$$\begin{aligned} A_c\, (\mathrm{diag}(Q^T \mathbf{x}_i))^{-1} &= R\,(I - B)\, P\, (\mathrm{diag}(Q^T \mathbf{x}_i))^{-1} \\ &= Q^T \mathrm{diag}(\mathbf{x}_i)\, Q\, (\mathrm{diag}(Q^T \mathbf{x}_i))^{-1} \\ &\quad - Q^T B \,\mathrm{diag}(\mathbf{x}_i)\, Q\, (\mathrm{diag}(Q^T \mathbf{x}_i))^{-1} \\ &= I_c - B_c. \end{aligned} \tag{2.8}$$

Coarse-level equation (2.6) was first introduced in [25] and has a straightforward probabilistic interpretation (see, e.g., [14, 15]). It is well known that (2.6) can be used to accelerate simple one-level iterative methods for (2.1), like the power or weighted Jacobi relaxation methods. For example, a two-level numerical method (aggregation/disaggregation) may proceed by relaxation on (2.1) on the fine level, followed by a coarse-level solve of (2.6), a coarse-level correction according to

$$\mathbf{x}_{i+1} = P\, (\mathrm{diag}(Q^T \mathbf{x}_i))^{-1}\, \mathbf{x}_c = P\, \mathbf{e}_c, \tag{2.9}$$

and another relaxation on the fine level.

In this paper, we use the weighted Jacobi method for all relaxation operations. We split problem matrix A into its diagonal and lower and upper triangular parts as $A = D - (L + U)$, using standard notation. Weighted Jacobi relaxation with weight $w \in (0, 1)$ is given by $\mathbf{x} \leftarrow (1 - w)\, \mathbf{x} + w\, D^{-1}\, (L + U)\, \mathbf{x}$.

A multilevel method can then be obtained by recursively applying the two-level method to coarse-level equation (2.6). In this paper, we consider so-called W cycles, which are obtained by applying coarse-level correction twice (see Figure 2.1). The resulting algorithm is given by Algorithm 1.

Algorithm 1: multilevel aggregation for Markov chains (W cycle), $\mathbf{x} \longleftarrow \mathbf{MA}(A, \mathbf{x}, \nu_1, \nu_2)$:

if *not on coarsest level* **then**
  $\mathbf{x} \leftarrow \text{Relax}(A, \mathbf{x})$ $\quad \nu_1$ times.
  Build Q.
  $R \leftarrow Q^T$ and $P \leftarrow \text{diag}(\mathbf{x})\, Q$.
  $A_c \leftarrow R\, A\, P$.
  $\mathbf{x}_c \leftarrow \text{MA}(A_c\, \text{diag}(Q^T \mathbf{x})^{-1}, Q^T \mathbf{x}, \nu_1, \nu_2)$ (first coarse-level solve).
  $\mathbf{x}_c \leftarrow \text{MA}(A_c\, \text{diag}(Q^T \mathbf{x})^{-1}, \mathbf{x}_c, \nu_1, \nu_2)$ (second coarse-level solve).
  $\mathbf{x} \leftarrow P\, (\text{diag}(Q^T \mathbf{x}))^{-1}\, \mathbf{x}_c$ (coarse-level correction).
  $\mathbf{x} \leftarrow \text{Relax}(A, \mathbf{x})$ $\quad \nu_2$ times.
else
  $\mathbf{x} \leftarrow$ direct solve of $A\, \mathbf{x} = 0$, $\|\mathbf{x}\|_1 = 1$.
end

2.2. Strength-based neighborhood aggregation. We determine aggregates based on strength of connection in the scaled problem matrix $\hat{A} = A\, \text{diag}(\mathbf{x}_i)$ [15]. In this paper, we use a symmetrized strength of connection measure and the neighborhood-based aggregation technique of [26]. Note that this aggregation is a more standard version and differs from the aggregation technique used in [15, 18]. Node i is considered to be strongly connected to node j in the graph of $\hat{A}$ if

$$-\hat{a}_{ij} \geq \theta \max_{k \neq i} \{-\hat{a}_{ik}\} \quad \text{or} \quad -\hat{a}_{ji} \geq \theta \max_{k \neq j} \{-\hat{a}_{jk}\}. \tag{2.10}$$

The strong neighborhood, $\mathcal{N}_i$, of any point i is the set of all points that are strongly connected to i within the graph of $\hat{A}$, including i. The neighborhood aggregation method is described in Algorithm 2. In the description of the algorithm, Q_J stands for the index set of the nonzero elements of the Jth column of the aggregation matrix from (2.3), Q. The problems we consider in this paper have sparse local connectivity with a small number of connections per node. For this type of problem, the neighborhood aggregation method of Algorithm 2 is attractive because it usually results in well-balanced aggregates of approximately equal size and in coarsening that reduces the number of unknowns quickly. Coarse-level stencil sizes tend to be rather uniform and do not grow too quickly.

In the numerical results to be presented below, we also briefly compare this aggregation strategy with the double pairwise aggregation that was proposed in [20] for use in a recursively accelerated multilevel aggregation method for nonsingular linear systems; see Algorithm 3. This aggregation strategy proceeds by performing pairwise aggregation (Algorithm 4) two consecutive times. Double pairwise aggregation is attractive because the maximal size of aggregates is directly controlled (the maximal size is four) and because points with few strong connections are linked to their aggregate by at least one strong connection. For unsmoothed aggregation, the approximation quality of algebraically smooth error can be significantly improved with aggregates that are smaller than neighborhoods, without increasing the operator complexity too much. While it is not our intent in this paper to perform an in-depth study of the various types of aggregation that can be considered for multilevel methods, we do find it interesting to include a brief comparison of these two aggregation methods in the numerical results section below, in part also to investigate how the performance of our recursive acceleration

Algorithm 2: neighborhood-based aggregation, $\{Q_J\}_{J=1}^m \longleftarrow$ **NeighborhoodAgg**$(A\,\mathrm{diag}(\mathbf{x}),\,\theta)$

For all points i, build strong neighborhoods $\mathcal{N}_i$ based on $A\,\mathrm{diag}(\mathbf{x})$ and θ.
Set $\mathcal{R} \leftarrow \{1,\ldots,n\}$ and $J \leftarrow 0$.
`/* 1st pass: assign entire neighborhoods to aggregates */`
for $i \in \{1,\ldots,n\}$ **do**
  if $(\mathcal{R} \cap \mathcal{N}_i) = \mathcal{N}_i$ **then**
    $J \leftarrow J + 1$.
    $Q_J \leftarrow \mathcal{N}_i$, $\hat{Q}_J \leftarrow \mathcal{N}_i$.
    $\mathcal{R} \leftarrow \mathcal{R} \setminus \mathcal{N}_i$.
  end
end
$m \leftarrow J$.
`/* 2nd pass: put remaining points in aggregates they are most connected to */`
while $\mathcal{R} \neq \emptyset$ **do**
  Pick $i \in \mathcal{R}$ and set $J \leftarrow \mathrm{argmax}_{K=1,\ldots,m}\,\mathrm{card}\,(\mathcal{N}_i \cap Q_K)$.
  Set $\hat{Q}_J \leftarrow Q_J \cup \{i\}$ and $\mathcal{R} \leftarrow \mathcal{R} \setminus \{i\}$.
end
for $J \in \{1,\ldots,m\}$ **do** $Q_J \leftarrow \hat{Q}_J$.

Algorithm 3: double pairwise aggregation, $\{Q_J\}_{J=1}^m \longleftarrow$ **DoublePairwiseAgg**$(A\,\mathrm{diag}(\mathbf{x}),\,\theta)$

$\{Q_J^{(1)}\}_{J=1}^{m_1} \leftarrow$**PairwiseAgg**$(A\,\mathrm{diag}(\mathbf{x}),\,\theta)$.
$\{Q_J^{(2)}\}_{J=1}^{m} \leftarrow$**PairwiseAgg**$(Q^{(1)T}\,A\,\mathrm{diag}(\mathbf{x})\,Q^{(1)},\,\theta)$.
$Q = Q^{(1)}\,Q^{(2)}$.

Algorithm 4: pairwise aggregation, $\{Q_J\}_{J=1}^m \longleftarrow$**PairwiseAgg**$(A\,\mathrm{diag}(\mathbf{x}),\,\theta)$

Set $\mathcal{R} \leftarrow \{1,\ldots,n\}$, $J = 0$, $\hat{A} = A\,\mathrm{diag}(\mathbf{x})$.
For all i, set $S_i \leftarrow \{j \in \mathcal{R} \,|\, \hat{a}_{ij} < -\theta \max_{i \neq k} |\hat{a}_{ik}|\}$, and $m_i \leftarrow |\{j \,|\, i \in S_j\}|$.
while $\mathcal{R} \neq \emptyset$ **do**
  Select $i \in \mathcal{R}$ with minimal m_i; $J \leftarrow J + 1$.
  `/* choose strongest remaining negative connection */`
  Select $j \in \mathcal{R}$ such that $\hat{a}_{ij} = \min_{k \in \mathcal{R}} \hat{a}_{ik}$.
  if $j \in S_i$ **then**
    $Q_J \leftarrow \{i, j\}$.
  else
    $Q_J \leftarrow \{i\}$.
  end
  $\mathcal{R} \leftarrow \mathcal{R} \setminus Q_J$.
  For all $k \in Q_J$, update: $m_l \leftarrow m_l - 1$ for $l \in S_k$.
end

method may be influenced by which aggregation method we choose from the literature.

Figures 2.2 and 2.3 show results of neighborhood aggregation and double pairwise aggregation applied to a tandem queueing test problem (see section 4). For this problem, Markov states can be represented on a two-dimensional lattice, and points

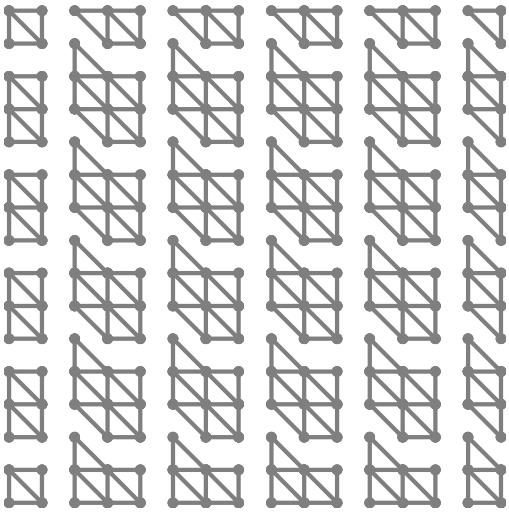

FIG. 2.2. *Tandem queuing network ($n = 15$): aggregates formed using neighborhood aggregation.*

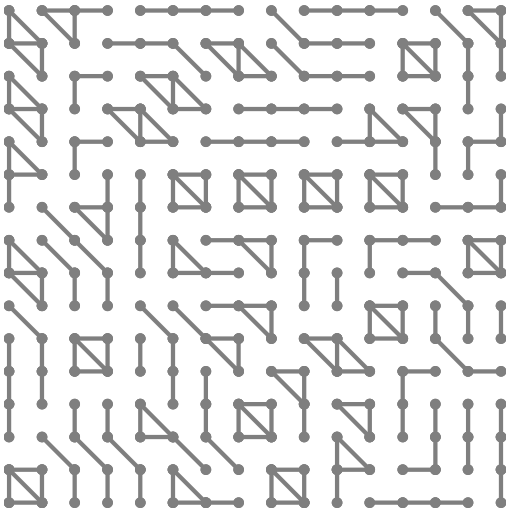

FIG. 2.3. *Tandem queuing network ($n = 15$): aggregates formed using double pairwise aggregation.*

in the interior of the domain have neighborhoods of size seven. Note that, for this problem, neighborhood aggregation generally leads to larger aggregates with a structured pattern, while double pairwise aggregation produces smaller aggregates that are more randomly structured.

3. Recursively accelerated multilevel aggregation for Markov chains. In this section, we explain how the W cycles of the Markov chain aggregation method described above can be accelerated by combining iterates recursively on all levels.

3.1. Optimal iterate recombination with inequality constraints. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the iterates obtained after the first and the second coarse-level solves in any invocation of Algorithm 1. They are represented in Figure 2.1 by any consecutive pair of small white boxes on a given level. We accelerate the cycle by considering

linear combinations $\mathbf{w}$ of $\mathbf{x}_1$ and $\mathbf{x}_2$,

$$\mathbf{w} = z_1\,\mathbf{x}_1 + z_2\,\mathbf{x}_2, \tag{3.1}$$

and choosing the linear combination such that the two-norm of the residual $A\,\mathbf{w}$ is minimized subject to constraints on the sign and size of $\mathbf{w}$. Let $\hat{X} = [\mathbf{x}_1|\mathbf{x}_2]$ and $\mathbf{z} = [z_1\,,\,z_2]^T$ such that $\mathbf{w} = \hat{X}\,\mathbf{z}$. We then seek the linear combination $\mathbf{w}^*$ of $\mathbf{x}_1$ and $\mathbf{x}_2$ that satisfies

$$\begin{aligned} &\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\ \|A\mathbf{w}\|_2 \\ \text{subject to}\quad &\mathbf{w} \geq 0 \quad \text{and} \\ &\|\mathbf{w}\|_1 = 1, \end{aligned} \tag{3.2}$$

with $\mathbf{w} = z_1\,\mathbf{x}_1 + z_2\,\mathbf{x}_2 = \hat{X}\,\mathbf{z}$. The optimal linear combination $\mathbf{w}^*$ (represented by small gray boxes in Figure 2.1) is then used for the coarse-grid correction (see Algorithm 1), so its effect propagates through the subsequent parts of the W cycle. Since $\mathbf{w} \geq 0$, we have that $\|\mathbf{w}\|_1 = \mathbf{1}^T\mathbf{w} = \mathbf{1}^T\,\hat{X}\,\mathbf{z} = \mathbf{1}^T\,\mathbf{z}$, and problem (3.2) can be rewritten in terms of $\mathbf{z}$ as

$$\begin{aligned} &\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}}\ \|(A\,\hat{X})\mathbf{z}\|_2 \\ \text{subject to}\quad &\hat{X}\,\mathbf{z} \geq 0 \quad \text{and} \\ &\mathbf{1}^T\,\mathbf{z} = 1. \end{aligned} \tag{3.3}$$

Note that this is a standard quadratic programming problem of dimension two, with n linear inequality constraints and one linear equality constraint. In the next section, it is explained how it can be solved explicitly in $O(n)$ work. Recursive acceleration of the W cycle in this way is similar in spirit and cost to the recursive acceleration of linear and nonlinear multigrid cycles using Krylov subspace methods that produced good results in [19, 20, 21]. We thus expect significantly improved convergence for our accelerated W cycles, and this will be demonstrated in the numerical convergence results of section 4. Convergence properties of recursively accelerated K cycles for symmetric positive definite linear systems are studied in [19]. Theoretical convergence study of our accelerated W cycles for Markov chains is difficult for the general nonsymmetric case that is relevant for Markov chains and is outside the scope of this paper, but we demonstrate their effectiveness by the numerical results of section 4.

One more modification needs to be made before we obtain a practically useful algorithm. Throughout the W cycle of Algorithm 1, all components of the iterates $\mathbf{x}_i$ on all levels are required to be strictly positive, due to the multiplicative nature of the algorithm (see (2.2)). However, quadratic programming problem (3.3) allows for solutions with zero components. To preclude problems with vanishing components in iterates $\mathbf{x}_i$, we modify the quadratic programming problem as follows:

$$\begin{aligned} &\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}}\ \|(A\,\hat{X})\mathbf{z}\|_2 \\ \text{subject to}\quad &\hat{X}\,\mathbf{z} \geq \delta \min_{i,j}(\hat{x}_{i,j}) \quad \text{and} \\ &\mathbf{1}^T\,\mathbf{z} = 1, \end{aligned} \tag{3.4}$$

with $\delta \in (0,1)$ a fixed parameter. We choose $\delta = 0.1$ for the numerical tests to be reported below. This means that we constrain all components of $\mathbf{w}^*$ to be at least as large as 10% of the smallest component of $\mathbf{x}_1$ and $\mathbf{x}_2$. While this ensures that all components of $\mathbf{w}^*$ are strictly positive, this can potentially slow down convergence if

components of the solution vector sought would still be more than 10 times smaller than the smallest component of current iterates $\mathbf{x}_1$ and $\mathbf{x}_2$. Our experience shows that this rarely happens, since initial relaxations and partial W cycles in most cases already bring the components of $\mathbf{x}_1$ and $\mathbf{x}_2$ close to the exact solution values in a relative sense. Note also that requiring $\hat{X}\,\mathbf{z} \geq \epsilon$ for a small ϵ of the order of machine accuracy does not lead to a practical algorithm, since some components of $\mathbf{x}$ may be much smaller than machine accuracy for many relevant Markov chain problems, and a relative bound in the constraint is thus required. The choice in (3.4) for modifying the inequality constraint worked well for all test problems considered below.

Note finally that it is important for obtaining a robust algorithm that the sign constraints on the components of $\mathbf{w}$ be enforced in the optimization problem. If $\mathbf{x}_i$ has negative or vanishing components after acceleration at any level, then these incorrect signs may propagate to finer levels via the coarse-level correction. Column-scaled operators $A\,\mathrm{diag}(\mathbf{x}_i)$ may then have entire columns that vanish or have incorrect signs, and the resulting coarse-level problems $Q^T\,A\,\mathrm{diag}(\mathbf{x}_i)\,Q\,\mathbf{e}_c = 0$ may fail to have a unique, strictly positive solution. Also, with incorrect signs in operators, relaxation may produce further negative or vanishing components. These incorrect signs lead in many cases to erratic or stalling convergence, or even divergence of the multigrid process. In summary, we have found that enforcing the sign constraints is important for the algorithm to be robust.

3.2. Efficient explicit solution of the two-dimensional quadratic programming problem. For simplicity, we first consider optimization problem (3.3), which has inequality constraint $\hat{X}\,\mathbf{z} \geq 0$. Since problem (3.3) is a two-dimensional optimization problem, the set of n inequality constraints $\hat{X}\,\mathbf{z} \geq 0$ has at most two relevant constraints (see Figure 3.1). Each inequality constraint defines a subset in the (z_1, z_2) plane:

$$\mathcal{H}_i = \{(z_1, z_2) \,:\, x_{i1}\,z_1 + x_{i2}\,z_2 \geq 0\}\,. \tag{3.5}$$

For any i, we have $x_{i1}, x_{i2} > 0$, so $\mathcal{H}_i$ is the set of all points on the side of the line normal to $(x_{i1}, x_{i2})^T$ that contains the first quadrant, including the line itself. The n constraints imply that $(z_1, z_2) \in \bigcap_{i=1}^{n} \mathcal{H}_i$. To form this set, however, only two constraints are required:

$$x_{j1}\,z_1 + x_{j2}\,z_2 \geq 0 \quad \text{and} \quad x_{k1}\,z_1 + x_{k2}\,z_2 \geq 0, \tag{3.6}$$

where

$$j = \operatorname*{argmin}_{1\leq i\leq n} \frac{x_{i2}}{x_{i1}} \quad \text{and} \quad k = \operatorname*{argmax}_{1\leq i\leq n} \frac{x_{i2}}{x_{i1}}. \tag{3.7}$$

Then,

$$\mathcal{H}_j \cap \mathcal{H}_k = \bigcap_{i=1}^{n} \mathcal{H}_i. \tag{3.8}$$

Coefficients $\mathbf{z}$ are selected by minimizing a scalar quadratic function over a (possibly infinite) line segment. Using the equality constraint in (3.3), we have $z_2 = 1 - z_1$ and we obtain an objective function $f(z_1) = \|(A\,\hat{X})\;[z_1\,,\,1-z_1]^T\,\|_2$ that is quadratic in z_1:

$$\begin{aligned} f(z_1) &= \langle A\,(z_1\mathbf{x}_1 + (1-z_1)\,\mathbf{x}_2)\,, A\,(z_1\mathbf{x}_1 + (1-z_1)\,\mathbf{x}_2)\rangle \\ &= z_1^2\,\langle A\mathbf{x}_1, A\mathbf{x}_1\rangle + 2\,(1-z_1)\,z_1\,\langle A\mathbf{x}_1, A\mathbf{x}_2\rangle + (1-z_1)^2\,\langle A\mathbf{x}_2, A\mathbf{x}_2\rangle\,. \end{aligned}$$

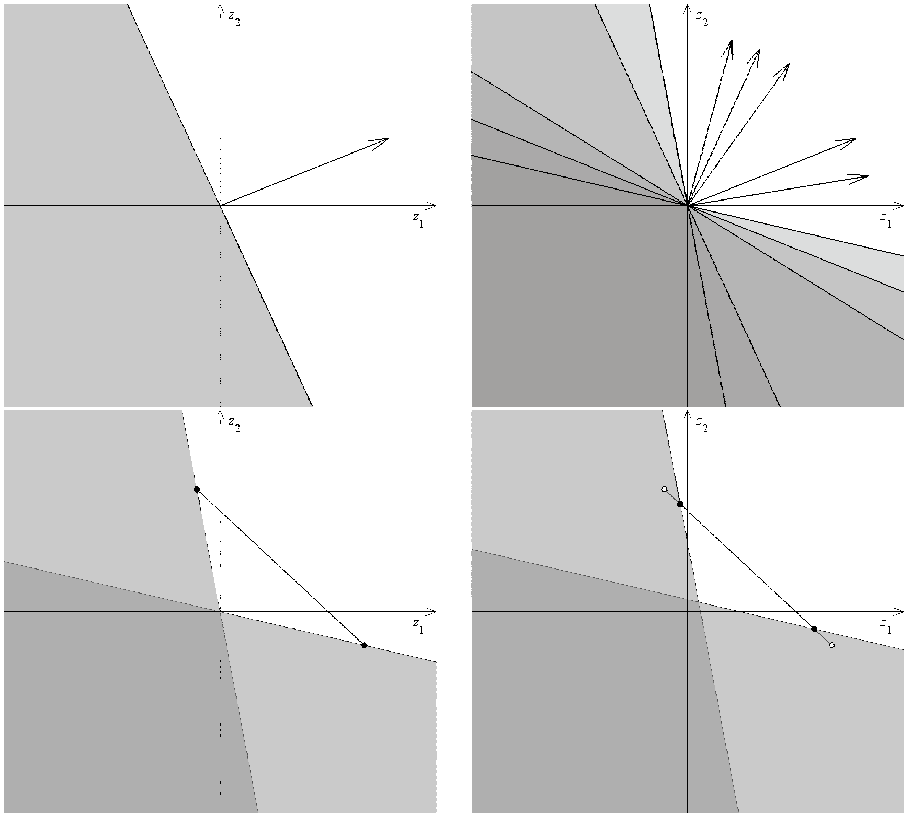

FIG. 3.1. *Optimal iterate recombination with inequality constraints. The top-left panel shows how a single inequality constraint from* (3.5) *limits* (z_1, z_2). *The shaded region is infeasible. The top-right panel shows a combination of several constraints. The bottom-left panel shows that the infeasible region of the two most extreme constraints is equivalent to that of all constraints and shows where the equality constraint line intersects the boundaries of the feasible region. The bottom-right panel shows the two relevant constraints for the modified inequality constraint* $\hat{X}\,\mathbf{z} \geq \delta\, \min_{i,j}(\hat{x}_{i,j})$, *and the intersection points with the equality constraint are indicated by black dots.*

Function $f(z_1)$ is easily minimized over the line segment specified by

$$\{z_1 + z_2 = 1\} \cap \mathcal{H}_j \cap \mathcal{H}_k. \tag{3.9}$$

This segment normally has two finite boundary points (the intersection points of $\{z_1+z_2 = 1\}$ with $\mathcal{H}_j$ and $\mathcal{H}_k$), but it can also be semi-infinite if one of the intersection points does not lie in the feasible region, or it can be infinite if both intersection points lie at infinity. The minimum of $f(z_1)$ is attained at

$$z_1^* = \frac{\langle A\mathbf{x}_2, A\mathbf{x}_2\rangle - \langle A\mathbf{x}_1, A\mathbf{x}_2\rangle}{\langle A\mathbf{x}_1, A\mathbf{x}_1\rangle - 2\,\langle A\mathbf{x}_1, A\mathbf{x}_2\rangle + \langle A\mathbf{x}_2, A\mathbf{x}_2\rangle}, \tag{3.10}$$

and we can check to see if $(z_1^*, 1 - z_1^*)$ lies on segment (3.9) by verifying that

$$x_{j1}\, z_1^* + x_{j2}\,(1 - z_1^*) \geq 0 \quad \text{and} \quad x_{k1}\, z_1^* + x_{k2}\,(1 - z_1^*) \geq 0. \tag{3.11}$$

If so, z_1^* is used to produce the minimal vector. Otherwise, $f(z_1)$ is evaluated at the points where $\{z_1 + z_2 = 1\}$ intersects the boundaries of $\mathcal{H}_j$ and $\mathcal{H}_k$:

$$z_1 = \frac{x_{j2}}{x_{j2} - x_{j1}} \quad \text{and} \quad z_1 = \frac{x_{k2}}{x_{k2} - x_{k1}}. \tag{3.12}$$

(If $x_{j2} = x_{j1}$ or $x_{k2} = x_{k1}$, the corresponding intersection point lies at infinity.) The intersection point that gives the minimal value should also be checked with (3.6), since one of the intersection points may lie on the wrong side of one of the inequality constraint boundaries (leading to a semi-infinite segment (3.9)). If both constraints are satisfied in the intersection point with the minimal $f(z_1)$ value, then this point is selected; otherwise the other point is selected. The procedure and expressions are similar for the modified inequality constraint $\hat{X}\,\mathbf{z} \geq \delta \min_{i,j}(\hat{x}_{i,j})$ of optimization problem (3.4).

The cost of obtaining the accelerated iterate $\mathbf{w}^*$ given $\mathbf{x}_1$ and $\mathbf{x}_2$ is modest: all steps can be performed in $O(n)$ work (with n the problem size at the current level). The extra work is dominated by two residual calculations, $A\mathbf{x}_1$ and $A\mathbf{x}_2$, and three inner products, $\langle A\mathbf{x}_1, A\mathbf{x}_1\rangle$, $\langle A\mathbf{x}_1, A\mathbf{x}_2\rangle$, and $\langle A\mathbf{x}_2, A\mathbf{x}_2\rangle$. In fact, in a careful implementation the calculation of $A\mathbf{x}_1$ does not incur extra work, since it is also calculated in the relaxation at the beginning of the second coarse-level solve call, and it can be saved. So in practice the extra work does not require much more than one residual calculation and a few inner products at each level in each branch of the W cycle. Note that optimized iterates in RAMA cycles are computed only starting from the first coarse level, so there is no extra work on the finest level (for RAMA cycles without top-level acceleration).

3.3. Top-level acceleration. Recombination of iterates can also be done at the top level to further speed up convergence. We now briefly discuss top-level acceleration of our recursively accelerated W cycle. A more detailed study of various types of top-level acceleration methods applied to several variants of multiplicative multilevel algorithms for Markov chains is presented in [28].

Recombination of top-level iterates is used often, for example in accelerating multigrid for nonsingular linear systems using CG or GMRES processes on the top level. (One can equivalently say that, in this case, multigrid is used as a preconditioner for CG or GMRES.) In the case of CG or GMRES acceleration of stationary multigrid cycles, excellent convergence properties are often obtained because the approximation spaces are nested. For nonstationary cycles (our RAMA cycles are nonstationary), this nesting cannot be taken advantage of, but generalized acceleration methods like flexible CG and GMRES can still provide significant speedup in this case. An approach like this was, for example, used in top-level acceleration of nonlinear multigrid [27] and in top-level acceleration of K cycles [19, 20].

Similarly, we employ top-level acceleration in this paper: a top-level iterate resulting from a RAMA cycle (top-level white box in Figure 2.1) can be combined with top-level iterates obtained by previous RAMA cycles using our constrained minimization approach (3.3) to produce an improved approximation at the top level (top-level gray diamond in Figure 2.1). On the top level, we can combine more than two iterates: we use a windowed approach in which we take an optimal linear combination of k previous iterates (except in the initial phase of the process, where we use fewer than k previous iterates until k RAMA cycles have been executed). The optimization problem for the case of window size k is still given by (3.3), with $\mathbf{z}$ now a k-dimensional vector and the columns of $\hat{X}$ now containing the iterates resulting from the k previous

RAMA cycles. Note that the columns of $\hat{X}$ do not contain the optimally top-level accelerated linear combinations of RAMA cycle results from previous steps but rather the results of the previous RAMA cycles themselves. This gives columns of $\hat{X}$ that are more linearly independent, and it saves work since most of the products $\langle A\mathbf{x}_i, A\mathbf{x}_j\rangle$ that were used to calculate the current optimal linear combination can be reused in the next optimization step.

Numerical experiments have shown that top-level window size $k = 3$ is a good choice for the test problems discussed in the next section, so our numerical results all employ $k = 3$. Note that larger values of k typically do not further improve convergence. This is most likely due to the lack of nested spaces for our nonstationary algorithm. The n inequality constraints of problem (3.3) can be reduced only as described above for the $k = 2$ case (for window size $k \geq 3$, there is the possibility of all n inequality constraints being relevant), so for $k \neq 2$ we use a general solver for the quadratic programming problem. For our numerical results, we use the active set method that is provided by the quadprog function of MATLAB [29]. While there is no known theoretical complexity bound proven for this algorithm, we have performed tests indicating that the execution time complexity is $O(n)$. As above, to preclude problems with zero components in the iterates, we replace the inequality constraint in problem (3.3) by $\hat{X}\,\mathbf{z} \geq \delta \min(\hat{X})$, with $\delta = 0.1$. This also takes care of small negative components that may occur in numerical solutions of problem (3.3) due to rounding.

Finally, note that, instead of RAMA cycles based on W cycles, recursive acceleration of so-called super-W cycles can also be considered. For example, three consecutive coarse-level solves can be performed in Algorithm 1, and the resulting iterates in each branch and on each level can be accelerated using a general solver for the quadratic programming problem ($k > 2$). However, as in [19, 20], we have found in numerical tests that recursively accelerated W cycles ($k = 2$) are more efficient than recursively accelerated super-W cycles with three or more coarse-level solves in each branch and on each level.

4. Numerical results. In this section, we present numerical performance tests for recursively accelerated W cycles (RAMA cycles), compared with unaccelerated W cycles. We also include results with additional top-level acceleration (window size 3) for both W cycles and RAMA cycles. (We call the cycles with top-level acceleration W+ and RAMA+ cycles.)

For all the numerical results presented in this paper, we start from a random, strictly positive initial guess and iterate until the one-norm residual, $\|A\,\mathbf{x}_i\|_1$, has been reduced by a factor of 10^{-8}. We do a direct solve on the coarse level when $n < 12$ [23]. All multilevel cycles used are (1,1) cycles ($\nu_1 = \nu_2 = 1$ in Algorithm 1), and we use strength parameter $\theta = 0.25$ for all test problems on all levels, as in [18, 23]. For simplicity, the weight in our weighted Jacobi relaxation scheme is always chosen as 0.7. This value works well for all tests considered, but, if desired, convergence can be further improved by choosing problem-dependent optimal weights.

In the tables with numerical results, "n" is the number of unknowns, "it" is the number of iterations until the convergence tolerance is reached, and "C_{op}" is the operator complexity of the last cycle. C_{op} is defined as the sum of the number of nonzero elements in all operators A on all levels divided by the number of nonzero elements in the fine-level operator. Note that all operators on each level are counted in C_{op} (i.e., two operators on level two, four on level three, eight on level four, etc.). Since most of the work in our algorithms consists of relaxations, this number gives a

good indication of the amount of work required for a cycle: the work per cycle scales linearly as a function of problem size if C_{op} is bounded by a constant. More generally, a multigrid method attains optimal scalability (linear in the number of unknowns) when the number of iterations is bounded as a function of problem size, and the work per iteration scales linearly as a function of problem size.

We present five test problems. The test problems are described extensively in [18, 23] and are introduced here only briefly. All test problems are slowly mixing: $|\lambda_2| \to 1$ as n increases. The first and fourth test problems are random walks on (undirected) graphs, and they lead to problem matrices that are similar to symmetric matrices and thus have real eigenvalue spectra; their solution can be determined by an inexpensive, local calculation. The other problems are more challenging: they are nonsymmetric and have complex eigenvalue spectra [23].

Table 4.1 shows results for a random walk on a two-dimensional square lattice [18, 23]. The eigenvalue spectrum for this problem is real [23]. It can be observed that W cycles require a large number of iterations. Recursive acceleration reduces the number of iterations significantly. Top-level acceleration with window size 3 improves both the W cycle and RAMA cycle results significantly. Operator complexity is nicely bounded. The number of iterations is relatively constant, but it increases for the largest problem size. Scalability is thus not fully obtained for this problem, but recursive and top-level acceleration improve convergence significantly.

Table 4.2 shows results for a queueing problem (two queues in tandem) from [18, 3], which can be represented on a two-dimensional lattice. This problem has a complex spectrum [23]. As in the previous test problem, a large number of W cycles is required and recursive acceleration improves convergence. Top-level acceleration again improves both the W cycle and RAMA cycle results.

Table 4.3 shows results for a Petri net problem from [13] (the original problem is described in [30]). This problem has a complex spectrum [23]. W cycles are scalable for this problem, but they require more iterations than RAMA cycles. Since the RAMA results without top-level acceleration are already scalable, top-level recombination does not provide significant further acceleration.

TABLE 4.1
Random walk on a two-dimensional lattice (neighborhood aggregation).

	W cycles		RAMA cycles		W+ cycles		RAMA+ cycles	
n	it	C_{op}	it	C_{op}	it	C_{op}	it	C_{op}
64	39	1.47	35	1.47	18	1.47	18	1.47
256	62	1.51	40	1.51	26	1.51	20	1.51
1024	106	1.57	41	1.57	36	1.57	22	1.57
4096	104	1.60	41	1.61	36	1.60	21	1.61
16384	166	1.60	42	1.60	47	1.60	21	1.60
65536	187	1.60	68	1.60	50	1.60	28	1.61

TABLE 4.2
Tandem queueing problem (neighborhood aggregation).

	W cycles		RAMA cycles		W+ cycles		RAMA+ cycles	
n	it	C_{op}	it	C_{op}	it	C_{op}	it	C_{op}
256	67	1.47	42	1.47	33	1.47	27	1.47
1024	121	1.47	48	1.47	53	1.47	31	1.47
4096	142	1.50	50	1.50	50	1.50	33	1.50
16384	212	1.51	57	1.51	63	1.51	32	1.51
65536	229	1.50	63	1.50	78	1.50	37	1.50

TABLE 4.3
Petri net problem (neighborhood aggregation).

	W cycles		RAMA cycles		W+ cycles		RAMA+ cycles	
n	it	C_{op}	it	C_{op}	it	C_{op}	it	C_{op}
819	61	1.79	38	1.79	26	1.79	24	1.79
5525	62	1.86	36	1.86	28	1.87	26	1.87
17575	62	1.91	38	1.91	31	1.94	36	1.92
23821	62	1.92	38	1.93	29	1.94	31	1.92

TABLE 4.4
Random walk on planar random graph (neighborhood aggregation).

	W cycles		RAMA cycles		W+ cycles		RAMA+ cycles	
n	it	C_{op}	it	C_{op}	it	C_{op}	it	C_{op}
1024	90	1.24	56	1.24	33	1.24	26	1.24
2048	108	1.23	62	1.23	37	1.23	27	1.23
4096	107	1.24	66	1.24	38	1.24	28	1.24
8192	127	1.24	73	1.24	41	1.24	28	1.24
16384	150	1.25	79	1.25	45	1.25	31	1.25
32768	145	1.24	81	1.24	44	1.24	28	1.24

Table 4.4 shows results for a random walk on a planar unstructured graph that is generated by Delaunay triangulation of uniformly distributed random points in the unit square [18]. The eigenvalue spectrum for this problem is real [23]. Again, W cycles require many iterations and recursive acceleration improves convergence. Top-level acceleration further improves both the W cycle and RAMA cycle results, and near-linear convergence is obtained.

Here it is also interesting to report on the relative cost in terms of runtime of the W, RAMA, W+, and RAMA+ cycles. For the problem of Table 4.4 with problem size $n = 32768$, we find in our MATLAB-based implementation that the runtime for a RAMA cycle is only about 0.5% longer than the runtime for a W cycle. The additional runtime is small because there is no extra work on the finest level and because we implement an efficient exact solution of the quadratic programming problem on all coarse levels. Top-level acceleration with window size 3, which occurs on the finest level and for which there is no efficient exact solution procedure, adds approximately 5% of runtime to each cycle. Overall, we find that, for this problem, the total runtime until convergence for the RAMA cycles is about 56% of the total W cycle runtime, while W+ reduces the runtime to 32% of the total W cycle runtime, and RAMA+ reduces it to 20%.

Finally, Table 4.5 shows results for a random walk on a planar unstructured graph that is generated as in the previous example [18], but now we delete directed edges from some of the triangles in a way that maintains the irreducibility of the graph. This leads to a more challenging, nonsymmetric unstructured problem with a complex eigenvalue spectrum. Specifically, a single directed edge is removed from each triangle in a subset of triangles that is selected from the Delaunay triangulation such that no two triangles in the set share an edge. This is done by randomly selecting a triangle in the triangulation, marking it as "deletable," and marking all of its three neighbors as "undeletable." This process is repeated until all triangles are marked. One edge of each "deletable" triangle is then made unidirectional by randomly deleting one of the six directed edges that connect the three nodes of the triangle. This process ensures that the Markov chain remains irreducible. A random walk is performed on

TABLE 4.5
Random walk on planar random graph with some edges deleted (neighborhood aggregation).

	W cycles		RAMA cycles		W+ cycles		RAMA+ cycles	
n	it	C_{op}	it	C_{op}	it	C_{op}	it	C_{op}
1024	113	1.32	61	1.32	38	1.32	28	1.32
2048	152	1.33	70	1.33	35	1.33	27	1.33
4096	180	1.35	75	1.35	52	1.35	31	1.35
8192	201	1.36	78	1.36	39	1.36	26	1.36
16384	214	1.36	67	1.36	43	1.36	27	1.36
32768	301	1.37	87	1.37	47	1.37	28	1.37

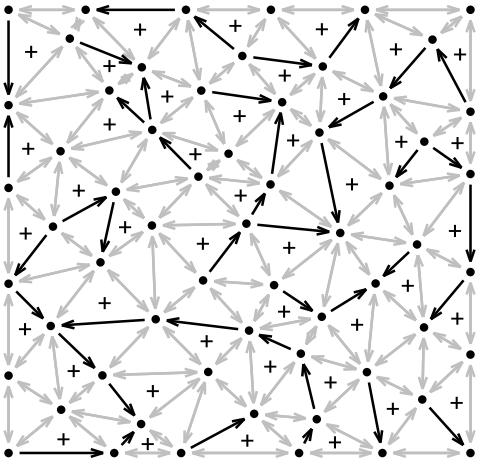

FIG. 4.1. *Planar random graph with some edges deleted. Black dots represent nodes, light gray segments represent bidirectional edges, and black segments represent directed edges. Triangles that were marked as "deletable" are indicated by "+" signs.*

the resulting graph: the transition probability from node i to node j is given by the reciprocal of the number of outward links from node i. See Figure 4.1 for an example graph for this problem. (In order to obtain a clearer picture, the graph shown here has a more regular distribution of points than the actual points used to build the Markov chains. For example, we have artificially located a certain number of points on the edges of the unit square, and there are no very small triangles that would be hard to discern.) As in Table 4.4, Table 4.5 shows that W cycles require many iterations and recursive acceleration improves convergence significantly for this test problem. Top-level acceleration further improves both the W cycle and RAMA cycle results, and near-linear convergence is also obtained for this unstructured problem with nonsymmetric sparsity structure.

4.1. Comparison with double pairwise aggregation. We now briefly report on numerical results obtained with the double pairwise aggregation (DPA) algorithm explained in section 2.2 and compare with the neighborhood aggregation (NA) results presented above. Tables 4.6 and 4.7 show DPA results for the tandem queueing problem and the random walk on a planar random graph problem. (For brevity, we

TABLE 4.6

Tandem queueing problem (DPA). In the rightmost columns, top-level aggregates are frozen for five RAMA cycles at a time.

	W cycles		RAMA cycles		RAMA+ cycles		RAMA+ cycles (frozen)	
n	it	C_{op}	it	C_{op}	it	C_{op}	it	C_{op}
256	51	2.18	40	2.18	26	2.14	23	2.19
1024	57	2.36	39	2.39	35	2.35	25	2.39
4096	70	2.45	48	2.43	64	2.45	29	2.47
16384	82	2.54	58	2.51	76	2.51	34	2.50
65536	97	2.56	73	2.56	99	2.55	46	2.57

TABLE 4.7

Random walk on planar random graph (DPA). In the rightmost columns, top-level aggregates are frozen for five RAMA cycles at a time.

	W cycles		RAMA cycles		RAMA+ cycles		RAMA+ cycles (frozen)	
n	it	C_{op}	it	C_{op}	it	C_{op}	it	C_{op}
1024	37	1.87	30	1.88	24	1.88	19	1.87
2048	41	1.89	31	1.90	27	1.90	18	1.90
4096	45	1.95	35	1.95	35	1.95	19	1.95
8192	46	1.96	36	1.96	41	1.95	20	1.96
16384	49	1.98	39	1.98	49	1.98	21	1.98

discuss only two of the test problems here; the other problems discussed above behave similarly.)

One can first observe that W cycles appear to work somewhat better for DPA than for NA: the number of iterations is lower, and the product of the number of iterations and the operator complexity (which gives a measure of the total amount of work) is also smaller. Not unexpectedly, the operator complexity of DPA is somewhat larger (its aggregates are smaller, so more levels are needed, and the shapes of the aggregates result in stencil growth on coarser levels). A partial explanation for the smaller number of iterations required may be that DPA has a more careful approach in trying to group states with the strongest available connections. For RAMA cycles, NA gives somewhat better results than DPA for the tandem queueing problem, but for the random walk problem DPA performs a bit better.

A surprising observation is that, for both the tandem queueing and random walk on a planar random graph problems, our top-level acceleration approach of RAMA cycles with DPA does not reduce the number of iterations, but rather it leads to significantly worse convergence. (We never observed such an increase for NA.) We were puzzled by this at first, until we found the apparent cause of this undesirable effect. We found that the DPA algorithm results in top-level aggregates that change significantly from one RAMA cycle to the next, even after many cycles when convergence is nearly attained. This means that the range of the fine-level coarse-grid correction operator is very different in each cycle, implying that the low-frequency error components that are most efficiently removed by coarse-grid correction change from iteration to iteration, which apparently has a detrimental effect on the convergence rate of the iterative process. The reason why top-level aggregates change in DPA is that seed nodes are grouped with the most strongly connected available neighbor, and the sizes of the connections in the scaled problem matrix, $\hat{A} = A\,\mathrm{diag}(\mathbf{x}_i)$, can change slightly from iteration to iteration. When some connections in $A\,\mathrm{diag}(\mathbf{x})$ (with $\mathbf{x}$ the exact solution) are equal or very close, the sizes of the corresponding entries in $A\,\mathrm{diag}(\mathbf{x}_i)$ may change slightly from iteration to iteration, and the relative magnitude is likely to change from

step to step, resulting in changes in the aggregation. This problem does not occur for NA for the range of problems tested (top-level aggregates remain fixed after a few cycles). Once this problem was realized, it was not difficult to come up with a simple solution that involves freezing the aggregates for a few cycles at a time. In the rightmost columns of Tables 4.6 and 4.7, we show how freezing the top-level aggregates for five RAMA cycles at a time results in a significant reduction in iterations of these "frozen" RAMA+ cycles compared to unaccelerated RAMA cycles. For the resulting "frozen" RAMA+ cycles, DPA performs somewhat worse than NA RAMA+ cycles for the tandem queueing problem and performs similarly to NA RAMA+ cycles for the random walk problem. (Note that our multilevel aggregation cycles with NA can also be made more efficient by freezing the aggregation hierarchies after a few cycles, but this has not been done for the NA results presented in this paper.)

Overall, we can say that both the DPA and NA methods for aggregation perform well. Not surprisingly, which aggregation method is better appears to be problem dependent. RAMA significantly speeds up W cycles for both NA and DPA, and top-level acceleration leads to further improvements for both aggregation methods, provided that slightly modified RAMA+ cycles are used for DPA that prevent top-level aggregates from changing too much.

5. Discussion and conclusion. We have shown how recursively accelerated W cycles of multiplicative multilevel algorithms for Markov chains that use simple nonoverlapping aggregation [13, 14, 15], combined with top-level acceleration, converge significantly faster than W cycles and lead to close-to-linear computational complexity for challenging test problems. In our approach, pairs of consecutive iterates at all branches and levels of the multigrid W cycle are recombined to produce an improved iterate by solving a quadratic programming problem with inequality constraints. We have shown how the two-dimensional quadratic programming problem can be solved explicitly in an efficient way. The additional top-level acceleration of the W cycles uses the same constrained quadratic programming approach. Our approach is inspired by the work on K-cycle multigrid by Notay and Vassilevski [19] and Notay [20], and is also closely related to recursive Krylov acceleration methods for nonlinear multigrid proposed by Oosterlee and Washio [21].

Our RAMA method forms an alternative to other recently proposed ways of overcoming the slow convergence of simple nonoverlapping multilevel aggregation methods for Markov chains [18, 23, 24]. As in [19, 20] for linear systems arising from elliptic PDE discretization, the recursively accelerated method does for many problems not lead to smaller iteration counts than competing approaches (and thus does not necessarily perform better for simple problems), but its value lies in its conceptual simplicity, probabilistic interpretation, and operator complexity, which may be lower for difficult problems. Both our SA method [18] and our AMG method for Markov chains [23] can be interpreted as employing overlapping aggregates to accelerate convergence (the nonzeros in the columns of the interpolation operator overlap). While this leads in both cases to near-optimal convergence properties, these methods have several important disadvantages. First, the probabilistic interpretation of the coarse-level operators is lost, which necessitates an ad hoc lumping procedure to maintain the singular M-matrix nature of the coarse-level operators. Second, for some problems the memory and execution time complexity per multigrid V cycle (as measured by the operator complexity) can be large, in particular for the smoothed aggregation method [18]. Our new RAMA method is attractive because it is conceptually simpler than SA or AMG for Markov chains [18, 23] and easier to implement. There are no over-

lapping aggregates, which makes the operator complexity better controlled, and the probabilistic interpretation (2.8) of coarse-level operators is automatically maintained on all levels.

REFERENCES

[1] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, SIAM, Philadelphia, 1994.

[2] B. Philippe, Y. Saad, and W. J. Stewart, *Numerical methods in Markov chain modeling*, Oper. Res., 40 (1992), pp. 1156–1179.

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

[4] 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.

[5] J. R. Koury, D. F. McAllister, and W. J. Stewart, *Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains*, SIAM J. Algebraic Discrete Methods, 5 (1984), pp. 164–186.

[6] P. J. Schweitzer and K. W. Kindle, *An iterative aggregation-disaggregation algorithm for solving linear equations*, Appl. Math. Comput., 18 (1986), pp. 313–353.

[7] U. R. Krieger, B. Miller-Clostermann, and M. Sczittnick, *Modeling and analysis of communication systems based on computational methods For Markov chains*, IEEE J. Sel. Areas Commun., 8 (1990), pp. 1630–1648.

[8] U. R. Krieger, *On a two-level multigrid solution method for finite Markov chains*, Linear Algebra Appl., 223/224 (1995), pp. 415–438.

[9] I. Marek and P. Mayer, *Convergence analysis of an iterative aggregation/disaggregation method for computing stationary probability vectors of stochastic matrices*, Numer. Linear Algebra Appl., 5 (1998), pp. 253–274.

[10] T. Dayar and W. J. Stewart, *Comparison of partitioning techniques for two-level iterative solvers on large, sparse Markov chains*, SIAM J. Sci. Comput., 21 (2000), pp. 1691–1705.

[11] I. Marek and P. Mayer, *Convergence theory of some classes of iterative aggregation/disaggregation methods for computing stationary probability vectors of stochastic matrices*, Linear Algebra Appl., 363 (2003), pp. 177–200.

[12] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[13] G. Horton and S. T. Leutenegger, *A multi-level solution algorithm for steady-state Markov chains*, in Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, 1994, pp. 191–200.

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

[15] H. De Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, and J. Ruge, *Multilevel adaptive aggregation for Markov chains, with application to web ranking*, SIAM J. Sci. Comput., 30 (2008), pp. 2235–2262.

[16] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA) multigrid*, SIAM Rev., 47 (2005), pp. 317–346.

[17] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.

[18] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Pearson, J. Ruge, and G. Sanders, *Smoothed aggregation multigrid for Markov chains*, SIAM J. Sci. Comput., 32 (2010), pp. 40–61.

[19] Y. Notay and P. S. Vassilevski, *Recursive Krylov-based multigrid cycles*, Numer. Linear Algebra Appl., 15 (2008), pp. 473–487.

[20] Y. Notay, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal., to appear.

[21] C. W. Oosterlee and T. Washio, *Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows*, SIAM J. Sci. Comput., 21 (2000), pp. 1670–1690.

[22] J. Ruge and K. Stueben, *Algebraic multigrid*, in Multigrid Methods, Frontiers Appl. Math. 3, S. F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.

[23] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Ruge, and G. Sanders, *Algebraic multigrid for Markov chains*, SIAM J. Sci. Comput., 32 (2010), pp. 544–562.
[24] E. Treister and I. Yavneh, *Square and stretch multigrid for stochastic matrix eigenproblems*, Numer. Linear Algebra Appl., 17 (2010), pp. 229–251.
[25] H. A. Simon and A. Ando, *Aggregation of variables in dynamic systems*, Econometrica, 29 (1961), pp. 111–138.
[26] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic Multigrid on Unstructured Meshes*, Technical report UCD-CCM-034, Mathematics Department, University of Colorado at Denver, Denver, CO, 1994.
[27] T. Washio and C. W. Oosterlee, *Krylov subspace acceleration for nonlinear multigrid schemes*, Electron. Trans. Numer. Anal., 6 (1997), pp. 271–290.
[28] H. De Sterck, T. A. Manteuffel, K. Miller, and G. Sanders, *Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains*, Adv. Comput. Math., to appear.
[29] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.
[30] M. K. Molloy, *Performance analysis using stochastic Petri nets*, IEEE Trans. Comput., C-31 (1982), pp. 913–917.