Aggregation Algorithms for K-cycle Aggregation Multigrid for Markov Chains

by

Manda Winlaw

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirements for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Hans De Sterck

Waterloo, Ontario, Canada, 2009

Abstract

Two different aggregation algorithms are proposed for a K-cycle multigrid method to efficiently solve for the stationary probability vector of an irreducible Markov chain. It is shown how the neighborhood aggregation algorithm and the double pairwise aggregation algorithm use a strength of connection measure derived from the scaled matrix problem to form aggregates. A recursively accelerated W-cycle is presented as the K-cycle multigrid method. Acceleration is performed at each level of the W-cycle. Improved iterates are produced by combining pairs of iterates that minimize the residual. A similar top-level acceleration method is also proposed to further improve convergence. Several representative test problems are used to compare the two aggregation algorithms and numerical results indicate that the neighborhood aggregation algorithm performs better and leads to nearly optimal multigrid efficiency.

Acknowledgments

I would like to thank Hans De Sterck for his excellent supervision. I would also like to thank Geoffrey Sanders for many helpful suggestions. As well, thank-you to my family for all their continued support.

1 Introduction

In this paper we are interested in using multilevel methods to calculate the stationary probability vector of large, sparse, irreducible Markov matrices in a numerically efficient way. Finding efficient solution methods is of interest across many disciplines since large, sparse, Markov chains are used in many different applications. These applications include information retrieval and web ranking, performance modeling of computer and communication systems, dependability and security analysis and analysis of biological systems. They are also prevalent in business and economics. The solution methods we examine in this paper share similarities to both algebraic multigrid (AMG) methods for sparse linear systems and to iterative aggregation/disaggregation (IAD) methods for Markov chains.

Most IAD methods are two-level methods and while multi-level methods have been previously examined, [4, 6, 8, 10], their use is not widespread. A likely factor is the far from optimal convergence properties. In contrast, multigrid methods have been developed for a large class of (nonsingular) linear systems with optimal computational complexity and since it is the multilevel component of multigrid methods that makes them so powerful and scalable there is still a lot left to explore regarding multigrid methods for Markov chains.

Another common feature of IAD methods is the prespecified choice of aggregates based on topological knowledge of the Markov chain. Instead, the aggregation procedures used in this paper are based on the strength of connection in the problem matrix, similar to AMG methods. Again, this idea is not new [4, 6, 10]; however, in a strength-based multilevel aggregation strategy, calculating the strength of connection using the original problem matrix has not proven successful for a wide class of Markov matrices. Given this difficulty we adopt the proposed method in [17], where the aggregation algorithm is based on the strength of connection in a scaled problem matrix.

Using the scaled problem matrix to calculate the strength of connection, we explore two aggregation algorithms. The first aggregation scheme which we will refer to as the "double pairwise aggregation" algorithm was developed by Notay [12] to study multigrid methods for second order discrete scalar elliptic PDEs. The procedure uses two passes of a pairwise matching algorithm applied to the scaled matrix graph. The matching algorithm favors the strongest negative couplings. The second algorithm which was first introduced in [20] will be referred to as the "neighborhood aggregation" algorithm. Aggregates are chosen so that they include so-called neighborhoods. Each neighborhood of a point is determined by the strength of connection in the scaled matrix problem. Since multigrid methods aim to accelerate convergence by reducing error components with different scales at different levels the choice of neighborhoods as aggregates is designed to create maximal aggregates where the average error of each aggregate is different but the error within each aggregate is nearly constant.

For both aggregation procedures the scalability of the multigrid method is enhanced by using a so-called K-cycle multigrid scheme, providing acceleration recursively at each

level. Acceleration is done at each level of a W-cycle by finding the linear combination of the current and past iterate that minimizes the residual. Notay [12] uses the "double pairwise aggregation" algorithm along with a K-cycle scheme where acceleration is done using a conjugate gradient method. He is able to show that for second order discrete scalar elliptic PDEs this method is significantly more robust than an unsmoothed aggregation method without recursive acceleration. Several others have also considered K-cycle multigrid methods to solve PDEs including Oosterlee and Washio [22, 14]. They propose a Krylov acceleration method in a nonlinear multigrid algorithm to efficiently solve several nonlinear PDEs.

It is interesting to note some of the recent developments in the study of efficient Markov chain solution methods. It has been shown in [15] that smoothing the interpolation and restriction operators can dramatically increase the efficiency of aggregation multigrid for Markov chains. As well, a lumped AMG method for Markov chains proposed in [16] leads to nearly optimal multigrid efficiency for a representative set of test problems for which traditional iterative methods are slow to converge. By considering our two aggregation algorithms along with the K-cycle multigrid scheme, we hope to match the performance of these two methods and we consider a number of Markov chains as examples.

The rest of the paper is organized as follows. Section 2 gives a detailed description of the problem we are trying to solve and includes several theorems important for the development of multilevel aggregation methods for Markov chains. Section 3 provides details of the standard multilevel schemes used in multigrid methods including the V-cycle, W-cycle and the K-cycle. Section 4 gives a description of the two aggregation algorithms. In Section 5 we implement our methods on several example Markov chains and discuss the results. Section 6 concludes.

2 Mathematical Formulation

A Markov transition matrix, or Markov matrix, describes the transition probabilities of a Markov chain. The (i, j)th element of a Markov matrix is the probability of moving from state i to state j. Our objective is to solve for the stationary probability vector of large, sparse, irreducible Markov matrices. Let $B \in \mathbb{R}^{n\times n}$ be a column stochastic matrix, i.e., $0 \leq b_{ij} \leq 1 \; \forall i, j$ and

$$\mathbf{1}^T B = \mathbf{1}^T \tag{2.1}$$

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

$$B\mathbf{x} = \mathbf{x}, \qquad x_i \geq 0 \; \forall i, \qquad \|\mathbf{x}\|_1 = 1. \tag{2.2}$$

If B is a Markov matrix then $\mathbf{x}$ is the stationary probability vector. From the above formulation we can see that $\mathbf{x}$ is an eigenvector of the matrix B with associated eigenvalue

1, which is an eigenvalue with maximum modulus ($|\lambda_1| = 1$). A *slowly mixing* Markov chain is one in which the modulus of the subdominant eigenvalue(s), $|\lambda_2|$, is approximately one. For these types of Markov chains, traditional, one-level iterative methods are not optimal and convergence can be significantly improved using multigrid methods. If the matrix B is irreducible then the solution to (2.2) is unique. . Matrix B is irreducible *iff* there exists a path from each vertex i to each vertex j in the directed graph of matrix B. Additionally, if B is irreducible then $\mathbf{x}$ satisfies $x_i > 0 \; \forall i$.

2.1 Theoretical Foundations

It is important for us to prove that our multigrid algorithms are well-posed and that the actual solution is a fixed point of our methods. We consider our multigrid algorithm well-posed if given a strictly positive iterate, the algorithm gives a proper definition for the next iterate. In order to ensure these properties hold we include several relevant theorems that can readably be applied to Markov matrices. The Perron-Frobenius theorem is one of the most significant. It provides important insight about the spectral radius of a nonnegative matrix and its associated eigenvector. We include the following version:

THEOREM 2.1 *(Perron-Frobenius ([1], p. 26, 27,28))*
Let $B \in \mathbb{R}^{n \times n}$, $b_{ij} \geq 0 \; \forall i, j$. *Then the following hold:*

1. $\rho(B)$ *is an eigenvalue of* B.
2. $\exists \; \boldsymbol{x} \in \mathbb{R}^n, x_i \geq 0 \; \forall i : B\boldsymbol{x} = \rho(B)\boldsymbol{x}$ *and* $\exists \; \boldsymbol{y} \in \mathbb{R}^n, y_i \geq 0 \; \forall i : \boldsymbol{y}^T B = \rho(B)\boldsymbol{y}^T$.
3. *If* B *is irreducible, then the eigenvectors* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *in (2) are unique up to scaling, and the inequalities in (2) are strict.*
4. *If* B *has a left or right eigenvector with strictly positive components, then this eigenvector has* $\rho(B)$ *as its eigenvalue.*

Singular M-matrices also play a critical role in the theory underlying multigrid methods for Markov chains. A singular M-matrix is defined as follows

DEFINITION 2.2
$A \in \mathbb{R}^{n \times n}$ *is a singular M-matrix* $\Leftrightarrow \; \exists \; B \in \mathbb{R}^{n \times n}, \; b_{ij} \geq 0 \; \forall i, j : \; A = \rho(B)I - B,$

where $\rho(B)$ is the spectral radius of B. An implication of the Perron-Frobenius theorem is that the choice of B in Definition 2.2 is not unique. According to the theorem, $\rho(B+sI) = \rho(B) + s$ for any real $s > 0$. Then $A = \rho(B)I - B = (\rho(B) + s)I - (B + sI) = \rho(B + sI)I - (B + sI)$, which means that $B + sI$ can be used instead of B in the definition. In many applications of multilevel methods, the following properties of singular M-matrices are important (see [15]):

THEOREM 2.3 *(Properties of Singular M-matrices)*

1. *Irreducible singular M-matrices have a unique solution to the problem $A\boldsymbol{x} = 0$, up to scaling. All components of $\boldsymbol{x}$ have strictly the same sign (scaling can be chosen s.t. $x_i > 0\ \forall i$). This follows directly from the Perron-Frobenius theorem.*

2. *An equivalent definition for singular M-matrices is : $A \in \mathbb{R}^{n\times n}$ is a singular M-matrix $\Leftrightarrow$ A is singular and all elements of $(A + \alpha I)^{-1}$ are nonnegative, $\forall \alpha > 0$.*

3. *Irreducible singular M-matrices have nonpositive off-diagonal elements, and strictly positive diagonal elements ($n > 1$).*

4. *If A has a strictly positive vector in its left or right nullspace and the off-diagonal elements of A are nonpositive, then A is a singular M-matrix.*

Using the above theorems and properties we can restate the problem in (2.2). Rather than solving for the stationary probability vector we can solve for a strictly positive vector of unit length which lies in the nullspace of a singular M-matrix. We seek the vector $\mathbf{x} \in \mathbb{R}^n$ such that

$$A\mathbf{x} = \mathbf{0}, \qquad x_i > 0\ \forall i, \qquad \|\mathbf{x}\|_1 = 1, \tag{2.3}$$

where $A = I - B$ and B is irreducible. According to Definition 2.2, A is a singular M-matrix and $\mathbf{1}^T A = \mathbf{0}$. Since B is irreducible, A must also be irreducible. This follows from the definition of irreducibility and because subtracting B from I cannot zero out any of the off-diagonal elements of B.

3 Aggregation Multigrid For Markov Chains

In this section, we recall the principal features of the classical AMG V-cycle, W-cycle and K-cycle. For K-cycle multigrid, we include the details of our acceleration method. We begin this section by considering several iterative methods that are often used to find the stationary probability vector of Markov chains.

3.1 Power, Jacobi, and Gauss-Seidel Methods

The Power method is a simple and commonly used iterative method for approximating the stationary probability vector of a stochastic matrix. Let $\mathbf{x}_i$ be the ith approximation, then the Power method is given by

$$\mathbf{x}_{i+1} = B\mathbf{x}_i. \tag{3.1}$$

Suppose the stochastic matrix B is both irreducible and aperiodic. Matrix B is called periodic with period $p > 1$ *iff* the lengths of all cycles in the directed graph of B are

multiples of p. A matrix B is called aperiodic if it is not periodic. If B is aperiodic and irreducible then the unique stationary probability vector $\mathbf{x}$ can be obtained from any initial vector $\mathbf{x}_0$ with nonnegative components and $\|\mathbf{x}_0\|_1 = 1$ by repeated multiplication with B:

$$\mathbf{x} = \lim_{n\to\infty} B^n \mathbf{x}_0. \tag{3.2}$$

This helps explain why the Power method is so appealing since convergence to the unique stationary probability vector is guaranteed if B is both irreducible and aperiodic.

Two other frequently used iterative methods are the Jacobi (JAC) and Gauss-Seidel (GS) methods. The Jacobi method is given by

$$\mathbf{x}_{i+1} = D^{-1}(L+U)\mathbf{x}_i, \tag{3.3}$$

and the Gauss-Seidel method by

$$\mathbf{x}_{i+1} = (L+D)^{-1}U\mathbf{x}_i. \tag{3.4}$$

where we use standard notation for the decomposition of matrix A into its lower and upper triangular parts and its diagonal part, $A = D - (L+U)$. The Jacobi and Gauss-Seidel methods may fail to converge to the unique stationary probability vector.

We can write all of the above methods in the following general form

$$\mathbf{x}_{i+1} = S\mathbf{x}_i, \tag{3.5}$$

where $S = B$ for the Power method, $S = D^{-1}(L+U)$ for the Jacobi method and $S = (L+D)^{-1}U$ for the Gauss-Seidel method. Using this general form we can construct a weighted or damped method in the following manner

$$\mathbf{x}_{i+1} = (1-w)\mathbf{x}_i + wS\mathbf{x}_i, \tag{3.6}$$

where $w \in (0,1)$. Like the Power method, the weighted Power method will converge to the unique stationary probability vector regardless of the initial condition. However, the weighted Power method does not require the stochastic matrix to be aperiodic. Both the weighted Gauss-Seidel and Jacobi methods will also always converge [17]. The periodicity of the stochastic matrix is irrelevant for convergence; however, for some initial conditions, convergence may not be to the unique stationary probability vector. Although these simple iterative methods have nice convergence properties, convergence can often be slow especially when the error is low frequency. That is when IAD methods are often employed and multilevel methods can also be considered. The simple iterative methods described above dampen oscillatory error components quickly and this explains why they are actually used in multigrid methods. In this context, they are often referred to as relaxation or smoothing

methods. In this paper we will use the weighted Jacobi method as a relaxation method and for completeness we include it here:

$$\mathbf{x}_{i+1} = (1-w)\mathbf{x}_i + w(D^{-1}(L+U)\mathbf{x}_i). \tag{3.7}$$

3.2 Two-level Aggregation

Before considering multilevel methods, we start with a simple two-level aggregation method. In developing this method we begin by examining the error that results from our approximate solution. In our two-level method, as well as in all our multigrid methods for Markov chains, the error formulation is multiplicative. This is a significant difference from standard multigrid methods for sparse linear systems where the error formulation is traditionally additive. Let $\mathbf{e}_i$ be the ith multiplicative error vector and $\mathbf{x}_i$ the ith iterate where the exact solution $\mathbf{x}$ is defined as $\mathbf{x} = \text{diag}(\mathbf{x}_i)\mathbf{e}_i$. Equation (2.3) can be rewritten in terms of the multiplicative error to get

$$A\,\text{diag}(\mathbf{x}_i)\mathbf{e}_i = 0. \tag{3.8}$$

Once convergence has been reached, $\mathbf{x}_i = \mathbf{x}$ and $\mathbf{e}_i = \mathbf{1}$. We will assume that for each iterate all the entries of $\mathbf{x}_i$ are greater than zero since our solution has this property. This property is a consequence of Theorem 2.3.1, and is also required for (3.8). Let $Q \in \mathbb{R}^{n\times m}$ be the aggregation matrix where n fine-level points are aggregated into m groups so that $q_{ij} = 1$ if fine-level node i belongs to aggregate j and $q_{ij} = 0$ otherwise. Each fine-level point belongs to one and only one aggregate (In Section 4 we will discuss how Q is actually formed). If the fine level error $\mathbf{e}_i$ is unknown we can approximate it with the coarse level error using Q, $\mathbf{e}_i \approx Q\mathbf{e}_c$. We use the subscript c to represent vectors and matrices on the coarse level. Using the fine level error approximation we can get a coarse level version of Equation (3.8) as follows

$$Q^T A\,\text{diag}(\mathbf{x}_i)Q\mathbf{e}_c = 0. \tag{3.9}$$

Let the restriction and prolongation operators, R and P, be defined as

$$R = Q^T \tag{3.10}$$

and

$$P = \text{diag}(\mathbf{x}_i)Q. \tag{3.11}$$

We can then rewrite Equation (3.9) as

$$RAP\mathbf{e}_c = 0. \tag{3.12}$$

Define the course-level operator, A_c, by

$$A_c = RAP. \tag{3.13}$$

This gives us the following coarse-level error equation

$$A_c \mathbf{e}_c = 0. \tag{3.14}$$

Using $\mathbf{1}_c^T R = \mathbf{1}^T$, we get that $\mathbf{1}_c^T A_c = 0$. $R\mathbf{x}_i$ is the restriction of current fine-level approximate $\mathbf{x}_i$ to the coarse level and $R\mathbf{x}_i = P^T\mathbf{1}$. Instead of solving the coarse-level equation, Equation (3.14), for the multiplicative error, $\mathbf{e}_c$, equivalently one can seek an improved coarse-level approximation, $\mathbf{x}_c$, of probability vector $\mathbf{x}$. This improved coarse-level approximation $\mathbf{x}_c$ is related to coarse-level error $\mathbf{e}_c$ by

$$\mathbf{x}_c = \text{diag}(R\mathbf{x}_i)\mathbf{e}_c = \text{diag}(P^T\mathbf{1})\mathbf{e}_c, \tag{3.15}$$

leading to the coarse-level probability equation

$$A_c(\text{diag}(P^T\mathbf{1}))^{-1}\mathbf{x}_c = 0. \tag{3.16}$$

Using the coarse-level probability equation, Equation (3.16), we can solve for the coarse level probability vector. We can perform the coarse-level solve approximately (e.g., by using a relaxation method, which may employ $P^T\mathbf{x}_i$ as the initial guess) or exactly. Once the coarse level probability vector has been computed we can use the prolongation matrix, P, to determine, $\mathbf{x}_i$, the improved approximate solution on the fine level. This two-level method is fully described in Algorithm 1. In the algorithm, Relax(A,$\mathbf{x}$), stands for one relaxation by a standard iterative method which in our numerical experiments will be the weighted Jacobi method. The efficiency of the algorithm is contingent on the fact that the coarse-level solve typically requires much less work than a fine-level solve. For the above two-level method, local convergence properties have been derived in [11].

Algorithm 3.1: TLA(A, ν_1, ν_2), Two-Level Aggregation Method

Choose the initial guess $\mathbf{x}$
while *The convergence criterion is NOT satisfied* **do**
  $\mathbf{x} \leftarrow$ Relax(A,$\mathbf{x}$) ν_1 times
  Build Q
  $R = Q^T$ and $P = \text{diag}(\mathbf{x})Q$
  $A_c = RAP$
  $\mathbf{x}_c \leftarrow$ Solve $A_c\text{diag}(P^T\mathbf{1})^{-1}\mathbf{x}_c = 0$, $x_{c,i} > 0\ \forall i$, $\|\mathbf{x}_c\|_1 = 1$ (Coarse-level solve)
  $\mathbf{x} \leftarrow P(\text{diag}(P^T\mathbf{1}))^{-1}\mathbf{x}_c$ (Coarse-level correction)
  $\mathbf{x} \leftarrow$ Relax(A,$\mathbf{x}$) ν_2 times
end

3.3 Multilevel Aggregation

If we recursively apply the two-level method to the coarse level probability equation, Equation (3.16), then we can obtain a multilevel aggregation method. This multilevel method will hopefully improve upon the efficiency of the two-level method. Algorithm 2 uses the simplest type of recursion which results in a so-called V-cycle. Figure 3.1 illustrates the structure of one iteration step of the V-cycle. It is obvious from this figure where this method's name is derived from.

Algorithm 3.2: AM(A, $\mathbf{x}$, ν_1, ν_2), Aggregation Multigrid for Markov chains (V-cycle)

if *NOT on the coarsest level* **then**
  $\mathbf{x} \leftarrow$ Relax(A,$\mathbf{x}$) ν_1 times
  Build Q
  $R = Q^T$ and $P = \text{diag}(\mathbf{x})Q$
  $A_c = RAP$
  $\mathbf{x}_c \leftarrow$ AM($A_c\text{diag}(P^T\mathbf{1})^{-1}$, $P^T\mathbf{1}$, ν_1, ν_2) (Coarse-level solve)
  $\mathbf{x} \leftarrow P(\text{diag}(P^T\mathbf{1}))^{-1}\mathbf{x}_c$ (Coarse-level correction)
  $\mathbf{x} \leftarrow$ Relax(A,$\mathbf{x}$) ν_2 times
else
  $\mathbf{x} \leftarrow$ solve $A\mathbf{x} = 0$
end

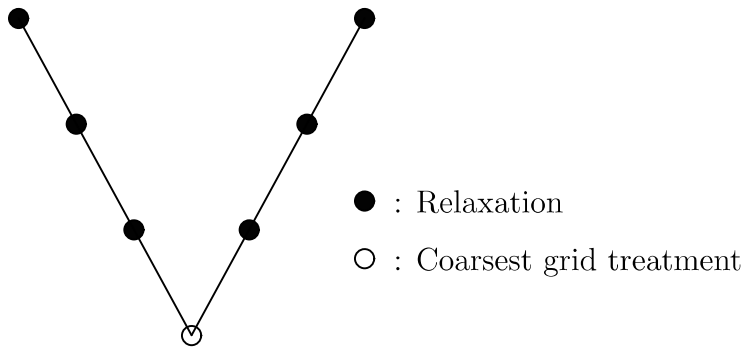


Figure 3.1: V-Cycle

Consider the following two theorems which are reproduced from [15]. The first of these theorems proves the well-posedness of the V-cycle algorithm and the second is necessary for convergence. To prove these theorems we first define the coarse-level stochastic matrix

B_c as

$$B_c = Q^T B \operatorname{diag}(\mathbf{x}_i) Q (\operatorname{diag}(Q^T \mathbf{x}_i))^{-1}. \tag{3.17}$$

The matrix satisfies $\mathbf{1}_c^T B_c = \mathbf{1}_c^T$. We then obtain

$$\begin{aligned} A_c(\operatorname{diag}(P^T\mathbf{1}))^{-1} &= R(I-B)P(\operatorname{diag}(P^T\mathbf{1}))^{-1} \\ &= Q^T \operatorname{diag}(\mathbf{x}_i) Q (\operatorname{diag}(Q^T\mathbf{x}_i))^{-1} - Q^T B \operatorname{diag}(\mathbf{x}_i) Q (\operatorname{diag}(Q^T\mathbf{x}_i))^{-1} \\ &= I_c - B_c. \end{aligned} \tag{3.18}$$

THEOREM 3.1 *(Singular M-matrix property of AM coarse-level operators).*
A_c is an irreducible singular M-matrix on all coarse levels, and thus has a unique right kernel vector $\boldsymbol{e}_c$ with strictly positive components (up to scaling) on all levels.

Proof. Equation (3.18) shows that A_c has nonpositive off-diagonal elements, and $\mathbf{1}_c^T A_c = 0$ because $\mathbf{1}_c^T R = \mathbf{1}^T$. This implies that A_c is a singular M-matrix, due to Theorem 2.3.4. Irreducibility of A_c can be proved as follows. Let fine-level node i belong to aggregate I, and fine-level node j to aggregate J. Then a link exists from J to I ($(Q^T B \operatorname{diag}(\mathbf{x}_k) Q)_{IJ} \neq 0$) if a link exists from j to i ($b_{ij} \neq 0$), by virtue of the shape of Q and the strict positivity of the components of $\mathbf{x}_k$. This implies that every aggregate J is connected to every aggregate I via a directed path, because every $i \in I$ is connected to every $j \in J$ via a directed path due to A's irreducibility. The second part of the theorem then follows directly from Theorem 2.3.1. ∎

THEOREM 3.2 *(Fixed-point property of AM V-cycle).*
Exact solution $\boldsymbol{x}$ is a fixed point of the AM V-cycle.

Proof. It is easy to see that $\mathbf{e}_c = \mathbf{1}_c$ is a solution of coarse-level equation (3.12) for $\mathbf{x}_i = \mathbf{x}$: $RAP\mathbf{e}_c = RAP\mathbf{1}_c = Q^T A \operatorname{diag}(\mathbf{x}) Q \mathbf{1}_c = Q^T A \mathbf{x} = 0$. This solution is unique (up to scaling) because A_c is an irreducible M-matrix. The coarse level correction equation then gives $\mathbf{x}_{i+1} = P\mathbf{e}_c = \operatorname{diag}(\mathbf{x}) Q \mathbf{1}_c = \operatorname{diag}(\mathbf{x})\mathbf{1} = \mathbf{x}$. ∎

We can generalize the standard V-cycle algorithm by changing the number of coarse-level solves from 1 to μ where μ can be any positive number. This gives us Algorithm 3. Of course, if $\mu = 1$ then we get the standard V-cycle and if $\mu = 2$ we get the standard W-cycle. One iteration of the standard W-cycle is depicted in Figure 3.2. W-cycles are more expensive than V-cycles; however, as long as coarsening is sufficiently fast, they can retain computational complexity that is linear in the number of unknowns. The theorems used to prove well-posedness and the fixed-point property for the V-cycle can also be used for the W-cycle since none of the equations have been modified.

Algorithm 3.3: μAM(A, $\mathbf{x}$, ν_1, ν_2, μ), μ-cycle Aggregation Multigrid for Markov chains

if *NOT on the coarsest level* **then**
  $\mathbf{x} \leftarrow$ Relax(A,$\mathbf{x}$) ν_1 times
  Build Q
  $R = Q^T$ and $P = \text{diag}(\mathbf{x})Q$
  $A_c = RAP$
  $\mathbf{x}_c \leftarrow \mu\text{AM}(A_c\text{diag}(P^T\mathbf{1})^{-1}, \mathbf{x}_c, \nu_1, \nu_2)$ μ times (Coarse-level solve)
  $\mathbf{x} \leftarrow P(\text{diag}(P^T\mathbf{1}))^{-1}\mathbf{x}_c$ (Coarse-level correction)
  $\mathbf{x} \leftarrow$ Relax(A,$\mathbf{x}$) ν_2 times
else
  $\mathbf{x} \leftarrow$ solve $A\mathbf{x} = 0$
end

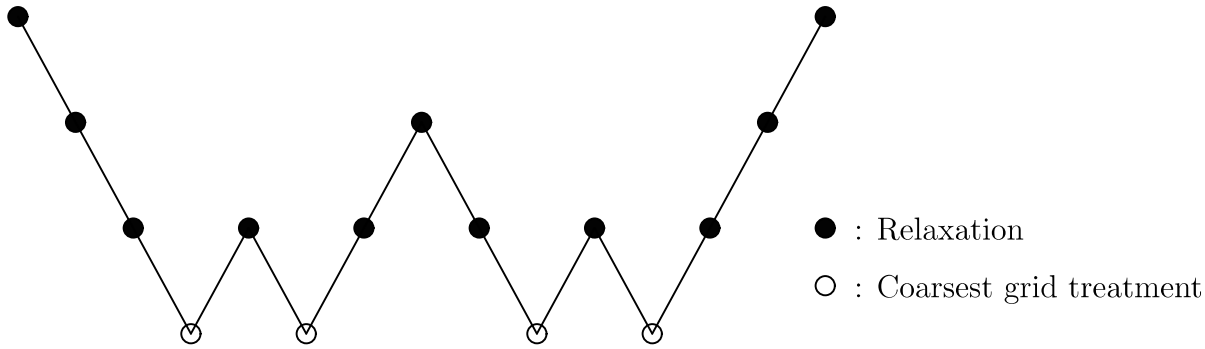


Figure 3.2: W-Cycle

3.4 K-cycle

The scalability of the general multigrid method outlined in Algorithm 3 can be enhanced or accelerated recursively on all levels by the use of K-cycle methods. The K-cycle method is so named because the acceleration of multigrid is typically done by Krylov subspace methods such as the conjugate gradient method (CG) or the generalized minimal residual method (GMRES). However, acceleration is not limited to these methods. In fact, for our numerical experiments, we use neither of these methods. Instead, we perform multigrid acceleration by a different form of iterant recombination. Typically, accelerating multigrid by a Krylov subspace method at the top level is synonymous with using multigrid as a preconditioner in connection with Krylov subspace methods. General K-cycle methods can be better understood if we consider the top-level acceleration case first. Consider Figure 3.3. In this case the standard V-cycle is being used as a preconditioner for a Krylov subspace

method. After each V-cycle, a Krylov subspace method is used in an attempt to reduce the residual (i.e. $\|A\mathbf{x}_{i,acc}\|$ should be less than $\|A\mathbf{x}_i\|$ where $\mathbf{x}_{i,acc}$ is the approximate solution computed using the Krylov subspace method). Through the use of the Krylov subspace method the number of V-cycle iterates needed to reach convergence is hopefully reduced. Although most acceleration methods only consider acceleration at the top-level there is no reason why acceleration cannot be used within each multigrid cycle. As well, there is no apparent restriction on the acceleration method. Figure 3.4 illustrates one iteration of our K-cycle algorithm (we present an accelerated W-cycle) and Algorithm 4 provides an outline of this method.

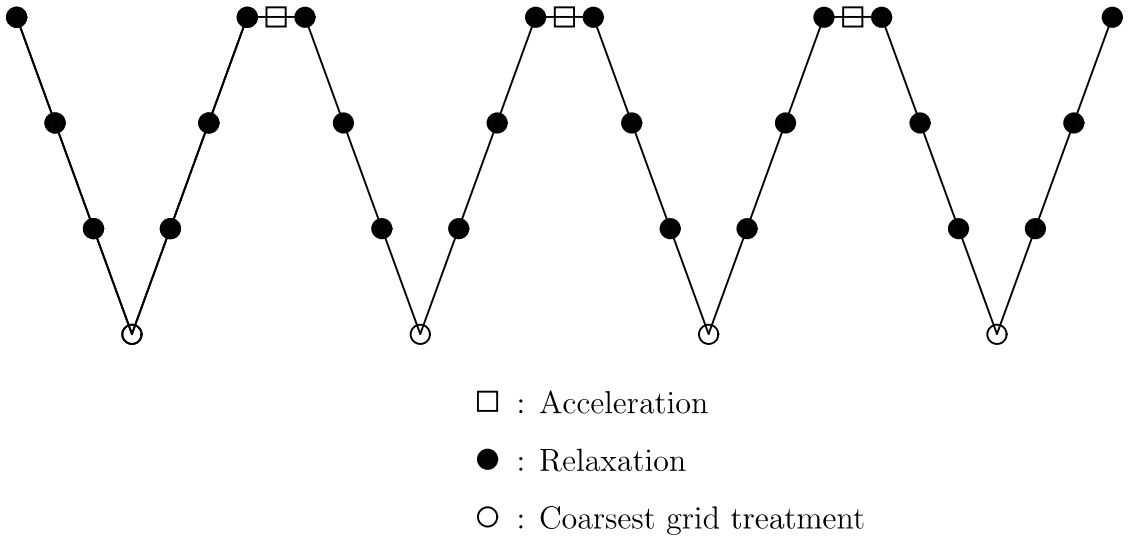
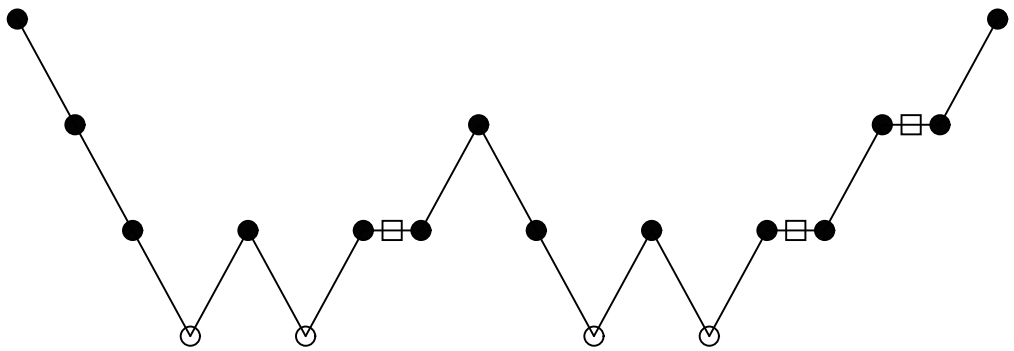


Figure 3.3: Multigrid as a Preconditioner for a Krylov Subspace Method

Algorithm 3.4: KAM(A, $\mathbf{x}$, ν_1, ν_2), K-cycle Aggregation Multigrid for Markov chains

if *NOT on the coarsest level* **then**
  $\mathbf{x} \leftarrow$ Relax(A,$\mathbf{x}$) ν_1 times
  Build Q
  $R = Q^T$ and $P = \text{diag}(\mathbf{x})Q$
  $A_c = RAP$
  $\mathbf{x}_{c,0} = P^T\mathbf{1}$
  for *i=1* **to** *2* **do**
    $\mathbf{x}_{c,i} \leftarrow \text{KAM}(A_c\text{diag}(P^T\mathbf{1})^{-1}, \mathbf{x}_{c,i-1}, \nu_1, \nu_2)$ (Coarse-level solve)
    if *i equals 2* **then**
      Set $\mathbf{x}_c$ equal to the linear combination of $\mathbf{x}_{c,1}$ and $\mathbf{x}_{c,2}$ which minimizes $\|A_c\mathbf{x}_c\|_2$ s.t. $\mathbf{x}_c > 0$ and $\|\mathbf{x}_c\|_1 = 1$
    end
  end
  $\mathbf{x} \leftarrow P(\text{diag}(P^T\mathbf{1}))^{-1}\mathbf{x}_c$ (Coarse-level correction)
  $\mathbf{x} \leftarrow$ Relax(A,$\mathbf{x}$) ν_2 times
else
  $\mathbf{x} \leftarrow$ solve $A\mathbf{x} = 0$
end



□ : Acceleration

● : Relaxation

○ : Coarsest grid treatment

Figure 3.4: K-Cycle

The acceleration method we use is an iterant recombination method. Let $\mathbf{x}_{c,i}$ be the approximate solution on the coarse level after the ith coarse level solve. Since we are using the W-cycle there are two coarse level solves on any given level. Our objective is to find the linear combination of these iterates that minimizes the residual. Let $\mathbf{z} = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2$, $\hat{X} = [\mathbf{x}_1|\mathbf{x}_2]$, and $\alpha = [\alpha_1\ \alpha_2]^T$, where $\mathbf{z} = \hat{X}\alpha$. For now we will ignore the coarse level subscript. We can write the minimization problem as

$$\mathbf{z}^* = \text{argmin}_{\mathbf{z}}\|A\mathbf{z}\|_2 \tag{3.19}$$

$$\begin{aligned} \text{subject to: } & \mathbf{z} = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2, \\ & \mathbf{z} \geq 0 \text{ and} \\ & \|\mathbf{z}\|_1 = 1. \end{aligned} \tag{3.20}$$

We set the coarse-level approximation equal to the solution once it has been determined ($\mathbf{x}_c = \mathbf{z}^*$). We should also note that our multilevel method requires all iterates to have strictly positive components. This implies that the inequality in (3.19) should be strict. To account for this we replace the above inequality with the following one: $\mathbf{z} \geq \epsilon\mathbf{1}$ where $\epsilon = \delta\ \min_{i,j}(\hat{x}_{i,j})$ and $\delta = 0.1$ in our numerical experiments. Although there is a possibility that one of the components in the solution is actually less than ϵ, using the value proposed seems to work well in practice. This optimal iterant recombination formulation is also used by others in our research group for top-level acceleration of AMG methods for Markov chains.

Given the small dimensionality of our minimization problem we can develop an efficient solution algorithm. Problem (3.19) is a two dimensional problem and since there are $n > 2$ inequality constraints at most two of them will determine the feasible region. For each inequality constraint we can define the following subset,

$$\mathcal{H}_i = \{(\alpha_1, \alpha_2) : x_{i1}\alpha_1 + x_{i2}\alpha_2 \geq 0\}, \tag{3.21}$$

which lies in the (α_1, α_2) plane. Since each component of $\mathbf{x}_1$ and $\mathbf{x}_2$ is positive, $\mathcal{H}_i$ is the set of all points on the side of the line normal to $(x_{i1}, x_{i2})^T$ that contains the first quadrant and the line itself. Let

$$j = \underset{1\leq i\leq n}{\text{argmin}}\frac{x_{i2}}{x_{i1}} \text{ and } k = \underset{1\leq i\leq n}{\text{argmax}}\frac{x_{i2}}{x_{i1}}. \tag{3.22}$$

Then, the only two possibly binding constraints are

$$x_{j1}\alpha_1 + x_{j2}\alpha_2 \geq 0 \quad \text{and} \quad x_{k1}\alpha_1 + x_{k2}\alpha_2 \geq 0, \tag{3.23}$$

and

$$(\alpha_1, \alpha_2) \in \mathcal{H}_j \cap \mathcal{H}_k = \bigcap_{i=1}^{n} \mathcal{H}_i. \tag{3.24}$$

The equality constraint in (3.19) implies $\alpha_2 = 1 - \alpha_1$, and the objective function can be rewritten in terms of α_1,

$$\begin{aligned} f(\alpha_1) =& < A(\alpha_1 \mathbf{x}_1 + (1-\alpha_1)\mathbf{x}_2), A(\alpha_1 \mathbf{x}_1 + (1-\alpha_1)\mathbf{x}_2) > \\ &= \alpha_1^2 < A\mathbf{x}_1, A\mathbf{x}_1 > +2(1-\alpha_1)\alpha_1 < A\mathbf{x}_1, A\mathbf{x}_2 > +(1-\alpha_1)^2 < A\mathbf{x}_2, A\mathbf{x}_2 > . \end{aligned} \tag{3.25}$$

For any value of α_1 that minimizes (3.25), α_1 must be on the line segment given by

$$\{\alpha_1 + \alpha_2 = 1\} \cap H_i \cap H_k. \tag{3.26}$$

From the first order necessary conditions we can derive the optimal value of α_1 as

$$\alpha_1^* = \frac{< A\mathbf{x}_2, A\mathbf{x}_2 > - < A\mathbf{x}_1, A\mathbf{x}_1 >}{< A\mathbf{x}_1, A\mathbf{x}_1 > -2 < A\mathbf{x}_1, A\mathbf{x}_2 > + < A\mathbf{x}_2, A\mathbf{x}_2 >}. \tag{3.27}$$

If $(\alpha_1^*, 1-\alpha_1^*)$ satisfies the inequality constraints given in (3.23) we can use α_1^* to produce the minimal vector. However, if this is not the case we can evaluate the objective function at the values of α_1 where the constraints in (3.23) are binding. We check to ensure that these two values of α_1 satisfy both inequalities. The value of α_1 that satisfies both constraints and produces the minimal value of the objective function is chosen. We have highlighted the procedure using the inequality constraint given in problem (3.19); however, the procedure is similar if we use the inequality constraint, $\bar{X}\alpha \geq \epsilon \mathbf{1}$. As well as performing acceleration within the W-cycle, we also perform top-level acceleration in a manner similar to the method we outlined for the V-cycle. Our method of top-level acceleration is again an iterant recombination method. Rather than using two previous iterates to minimize the residual we allow the possibility of k possible iterates where k is any number greater than two. Numerical experiments have shown that $k = 3$ gives the best results and this is the number we use in our numerical experiments. It is also important to note that none of the equations in the K-cycle algorithm have been modified from the basic V-cycle. The acceleration only modifies the iteration values ($\mathbf{x}_i$). This implies that our theorems used to prove well-posedness and the fixed-point property for the V-cycle can also be applied to the K-cycle.

4 Aggregation

In the following section we describe how to build the aggregation matrix, Q, defined in Section 3.2 using two different algorithms: neighborhood aggregation and double pairwise

aggregation. We also consider small modifications of the double pairwise aggregation algorithm to improve performance on some of the test problems we consider. Neither the neighborhood aggregation algorithm nor the double pairwise algorithm requires any topological knowledge about the Markov chain. Instead both algorithms are based on the strength of connection in a scaled problem matrix ($A\,\mathrm{diag}(\mathbf{x}_i)$). A number of aggregation algorithms have been proposed which are based on the strength of connection in the original problem matrix [6, 9, 5, 10]. There are also several aggregation algorithms similar to the double pairwise aggregation algorithm that use the problem matrix to form pairs or matchings [2, 6, 7]. A detailed explanation of why we scale the columns of the original problem matrix at each recursive level with the current probability vector at that level can be found in [17]. However, there is a simple intuitive interpretation for why we would want to base our strength of connection on the scaled Markov chain. Since the steady-state probability of residing in state j is influenced by state i not just through the transition probability from state i to j but by the product of that transition probability and the probability of residing in state i it is important that we consider the product.

4.1 Neighborhood Aggregation

Algorithm 5 summarizes the neighborhood aggregation method. Before the aggregates can be built the strength matrix, S, must be calculated. Let $\bar{A} = A\,\mathrm{diag}(\mathbf{x})$ be the scaled problem matrix with matrix entries $\bar{a}_{ij}$. The strength matrix S is defined so that $s_{ij} = 1$ if $\bar{a}_{ij} < -\beta \max_{\bar{a}_{ik}<0} |\bar{a}_{ik}|$ and $s_{ij} = 0$ otherwise. If $s_{ij} = 1$ then we say that node i is strongly influenced by node j or equivalently node j strongly influences node i. Thus the ith row of S represents all of the nodes that strongly influence node i and the ith column of S represents all of the nodes that are strongly influenced by node i. For the neighborhood aggregation algorithm we want to consider the nodes that strongly influence node i and the nodes that are strongly influenced by node i. Let $\bar{S} = S + S^T$. If $\bar{s}_{ij} \neq 0$, node j belongs to the neighborhood of node i since node j either strongly influences node i or it is strongly influenced by node i. The parameter β is called the threshold parameter and it determines which nodes are considered strongly connected. In our numerical experiments $\beta = 0.25$. Once $\bar{S}$ has been calculated it can be used to form the aggregates. The first aggregate is constructed using node 1. All the nodes that belong to the neighborhood of node 1 are included as members of this aggregate. Node 1 is called a center node since it forms the center of the neighborhood aggregate. Before constructing the second aggregate all the nodes in the first aggregate are removed from the list of possible center nodes. Although we use node 1 as our first center node there is nothing special about this choice and we could use any node as the starting point. After the first aggregate has been built we move on to the next possible center node. If any of the nodes in the neighborhood of this possible center node are in the first aggregate we do not form an aggregate, we remove this node from the list of center nodes and we move on to the next possible center node.

We form the next aggregate when we have found a node where all of the members of its neighborhood do not belong to any previous aggregates. We continue in this manner until all possible center nodes have been considered. After this first pass, we have completely disjoint aggregates; however, some nodes have yet to be assigned since they may not have qualified as center nodes and they may not have been strongly connected to any center nodes. The second pass of the algorithm loops through these remaining nodes. For each node we calculate the number of nodes in each aggregate that are strongly connected to our test node. The node is then included in the aggregate it has the largest number of strong connections with.

The neighborhood aggregation algorithm is designed to create aggregates which are as large as possible and for which the error of each node within a particular aggregate is similar (after fine-level relaxation). For each node within a given aggregate it is important for the values of the error to be similar since this property is necessary for multilevel methods to be successful. We illustrate the results of the neighborhood aggregation algorithm for the tandem queuing network and random planar graph in Figures 4.1 and 4.2, respectively. Section 5 includes a detailed description of both problems. Both figures depict the aggregates that are formed from applying the algorithm on the finest level. For the tandem queuing network the aggregates appear regular and the number of variables is reduced from 256 on the fine level to 36 on the coarse level, a factor of approximately seven. For the random planar graph the aggregates do not appear as regular although given the unstructured nature of the problem this is not surprising. For this case, the number of variables is reduced from 221 on the fine level to 36 on the course-level, a factor of approximately six.

Algorithm 4.1: NA($\bar{A}$, β), Neighborhood Aggregation Algorithm

$U = [1, n]$ (n is the number of fine level points)
for *i=1* **to** n **do**
  $S_i = \{j \in U \backslash \{i\} \mid \bar{a}_{ij} < -\beta \max_{\bar{a}_{ik}<0} |\bar{a}_{ij}|\}$
end
for *i=1* **to** n **do**
  $\bar{S}_i = S_i \cup \{j \mid i \in S_j\}$
end
$n_c = 0$
while $U \neq \emptyset$ **do**
  Let i be the first element in U.
  if *None of the nodes in* $\bar{S}_i$ *belong to previous aggregates* **then**
    $n_c = n_c + 1$
    $Q_{n_c,i} = 1$
    $U = U \backslash \{i\}$
    forall $j \in \bar{S}_i$ **do**
      $Q_{n_c,j} = 1$
      $U = U \backslash \{j\}$
    end
  end
  else
    $R = R \cup \{i\}$
    $U = U \backslash \{i\}$
  end
end
while $R \neq \emptyset$ **do**
  Let i be the first element in R.
  $A_j = 0 \; \forall \; 1 \leq j \leq n_c$
  forall $k \in S_i$ **do**
    if *k is in aggregate j* ($Q_{kj} = 1$) **then**
      $A_j = A_j + 1$
    end
  end
  Let A_{max} be the maximum $A_j \; \forall \; 1 \leq j \leq n_c$
  $Q_{max,i} = 1$
  $R = R \backslash \{i\}$
end
return Q^T

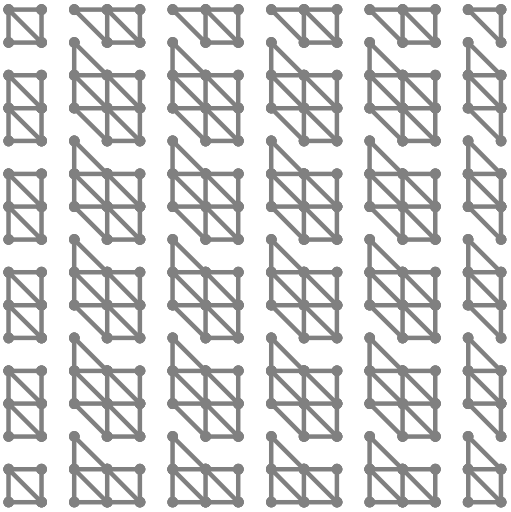


Figure 4.1: Tandem Queuing Network ($n = 15$): Aggregates formed using Neighborhood Aggregation.

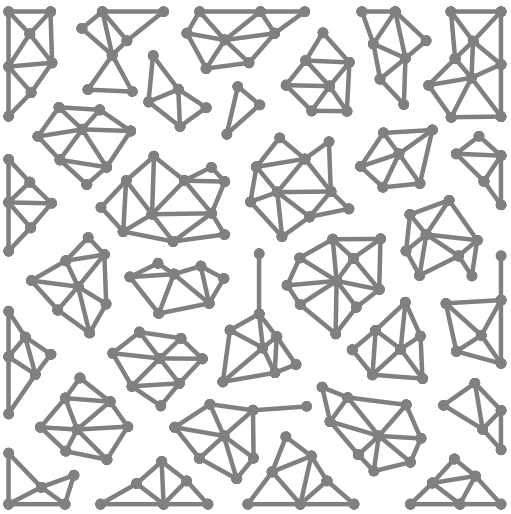


Figure 4.2: Random Planar Graph ($n = 221$): Aggregates formed using Neighborhood Aggregation.

4.2 Double Pairwise Aggregation

The double pairwise aggregation algorithm begins by forming matchings in the scaled problem matrix. Similar to the neighborhood aggregation algorithm this requires the calculation of a strength matrix to help determine which nodes are best matched. The strength matrix, S, is calculated exactly the same as it is for neighborhood aggregation. For the double pairwise algorithm we only consider the nodes that strongly influence node i (i.e., we do not use the strength matrix $\bar{S}$). Nodes are matched by comparing different neighbors of a node and by giving preference to the strongest negative coupling. The order in which the nodes are matched is determined by the variable m_i where

$$m_i = |\{j | s_{ji} = 1\}|. \tag{4.1}$$

We choose nodes with minimal m_i so nodes with the smallest number of strong connections as determined by S are considered first. This ensures that most nodes are matched. Once node i has been selected as the initial node in the aggregate, node j for which $\bar{a}_{ij}$ is minimal is considered as a possible matching. Node j is only added to the aggregate if $s_{ij} = 1$. It is possible for s_{ij} to equal zero if all the nodes that strongly influence node i have already been included in previous aggregates. If this occurs then node i is left in the aggregate alone. This pairwise aggregation scheme is given in Algorithm 6. Using just a pairwise aggregation scheme results in a relatively slow coarsening which motivates the use of a second pass. This leads to Algorithm 7, the double pairwise algorithm. In the double pairwise algorithm we simply repeat the process in the pairwise algorithm by forming pairs of aggregates from the first pass. The final aggregates are either of size one, two, three or four. In most cases using two pases of the pairwise matching algorithm results in a decrease of the number of variables by a factor of slightly less than four. We note that the auxiliary matrix A_1 used in the second pass of the algorithm is constructed using the course level probability equation, Equation (3.16). The strength of connection for the second pass of the pairwise algorithm should be determined by $A_c(\text{diag}(Q_1^T\mathbf{1}))^{-1}$ scaled by the course grid probability vector $\mathbf{x}_c$, where

$$A_c = Q_1^T A \,\text{diag}(\mathbf{x}) Q_1 = Q_1^T \bar{A} Q_1. \tag{4.2}$$

However, in this case, $\mathbf{x}_c = Q_1^T\mathbf{1}$ which implies $A_1 = Q_1^T\bar{A}Q_1$.

Algorithm 4.2: PWA($\bar{A}$, β), Pairwise Aggregation

$U = [1, n]$ (n is the number of fine level points)
for *i=1* **to** n **do**
  $S_i = \{j \in U \backslash \{i\} \mid \bar{a}_{ij} < -\beta \max_{\bar{a}_{ik}<0} |\bar{a}_{ij}|\}$
end
for *i=1* **to** n **do**
  $m_i = |\{j \mid i \in S_j\}|$
end
$n_c = 0$
while $U \neq \emptyset$ **do**
  Select $i \in U$ with minimal m_i.
  $n_c = n_c + 1$
  Select $j \in U$ such that $\bar{a}_{ij} = \min_{k \in U} \bar{a}_{ik}$.
  if $j \in S_i$ **then**
    $Q_{n_c,i} = 1$
    $Q_{n_c,j} = 1$
    $U = U \backslash \{i, j\}$
  else
    $Q_{n_c,i} = 1$
    $U = U \backslash \{i\}$
  end
  forall k *such that* $Q_{n_c,k} = 1$ **do**
    $m_l = m_l - 1$ for $l \in \bar{S}_k$
  end
end
return Q^T

Algorithm 4.3: DPWA($\bar{A}$, β), Double Pairwise Aggregation

$Q_1 = \text{PWA}(\bar{A},\beta)$
$A_1 = Q_1^T \bar{A} Q_1$
$Q_2 = \text{PWA}(A_1,\beta)$
$Q = Q_1 \cdot Q_2$
return Q

We illustrate the results of the double pairwise aggregation algorithm for the tandem queuing network and random planar graph in Figures 4.3 and 4.4, respectively. For the tandem queuing network the aggregates do not have the same regular appearance as they do in the neighborhood aggregation case. Most aggregates are of size four, while there are several size two and three aggregates. The number of variables is reduced from 256 on the fine level to 67 on the coarse level, a factor of approximately four. Like the results from the neighborhood aggregation algorithm the aggregates for the random planar graph do not appear as regular. The number of variables is reduced from 221 on the fine level to 60 on the course-level. This is a much larger number of aggregates than for neighborhood aggregation.

In our methods, an error, e, is defined to be algebraically smooth if it is slow to converge with respect to the relaxation method (i.e., WJAC). Alternatively, we call an error smooth if it has to be approximated by means of a coarser level in order to speed up convergence. The success of our multilevel methods relies on our ability to group variables into aggregates where we can take advantage of the pattern of algebraic smoothness. To do this we need to know which neighbors of each state have similar error values after relaxation. "The error at state j is influenced most strongly by the errors at states k that have large matrix elements in row j and columns k of the scaled matrix $A\,\mathrm{diag}(\mathbf{x}_i)$" [17]. It is also important for the number of aggregates to be relatively small so that the problem may be solved with relatively little expense. For our multilevel methods to be optimal any aggregation method we propose must incorporate both principles. Both neighborhood aggregation and double pairwise aggregation form aggregates based on these two principles. However, since the neighborhood aggregation algorithm forms aggregates using all of the strongly connected neighbors the number of aggregates will be smaller on average than for double pairwise and thus the computational complexity should be lower. The double pairwise aggregation algorithm forms aggregates by considering the strongest negative coupling which should give aggregates with very similar error. In the neighborhood aggregation algorithm the variance among the errors in any given aggregate might be larger which suggest convergence in the double pairwise algorithm might be better.

4.3 Modified Double Pairwise Aggregation

In the double pairwise aggregation algorithm we want to form aggregates with as many points as possible to ensure sufficiently low computationally complexity while ensuring a strong connection between the matchings for convergence. However, a problem with the pairwise algorithm given in Algorithm 6 can arise because of the order in which the initial node is chosen. Suppose node i is chosen as the initial node but that all of the nodes that strongly influence node i already belong to aggregates. This implies that node i should be the only node in the aggregate. However there might exist a node j currently not in any aggregate that is strongly influenced by node i (where this is the strongest connection) but

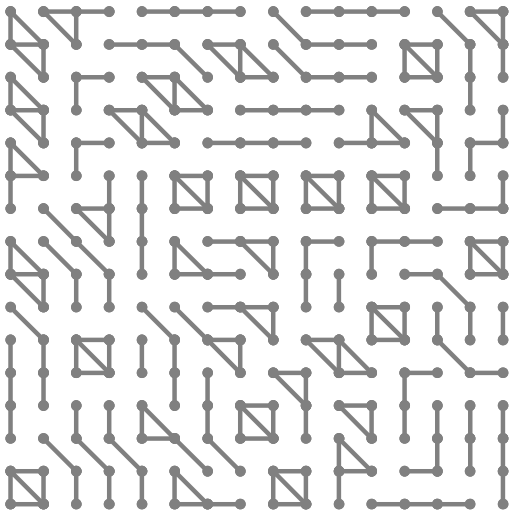


Figure 4.3: Tandem Queuing Network ($n = 15$): Aggregates formed using Double Pairwise Aggregation.

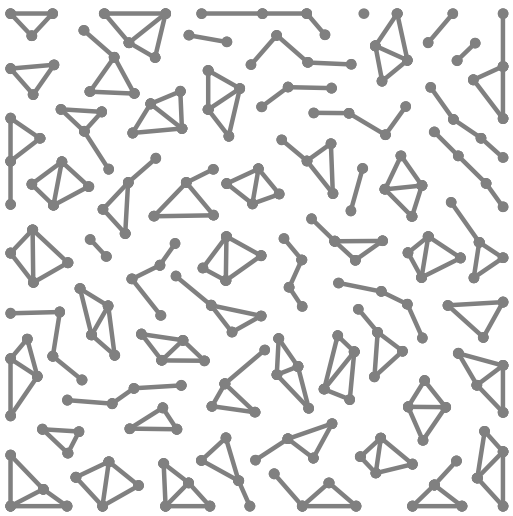


Figure 4.4: Random Planar Graph ($n = 221$): Aggregates formed using Double Pairwise Aggregation.

that does not strongly influence node i. If node j were chosen as the initial node before node i the aggregate would include both nodes. By considering nodes that both strongly influence and are strongly influenced by our initial node this may lead to more matchings. This leads us to the modified pairwise algorithm given in Algorithm 8. In this case we determine the initial node to be the node with minimal m_i where

$$m_i = |\{j|s_{ji} = 1\}| + |\{j|s_{ij} = 1\}| \tag{4.3}$$

The second node in the aggregate is determined by examining both node j where $\bar{a}_{ij}$ is minimal and node $\bar{j}$ where $\bar{a}_{\bar{j}i}$ is minimal. If the connection between node i and j is stronger and $s_{ij} = 1$ then node j is added to the aggregate and if the connection between node i and $\bar{j}$ is stronger and $s_{\bar{j}i} = 1$ then node $\bar{j}$ is added to the aggregate. If both $s_{ij} = s_{\bar{j}i} = 0$ then node i remains in the aggregate alone. This modified pairwise algorithm leads to a modified double pairwise algorithm which is identical to Algorithm 7 except we call the modified pairwise algorithm instead of the original pairwise algorithm. Our modified algorithm should have lower computational complexity than the original and should converge in fewer iterations since the connection between pairs should be the strongest possible.

Figures 4.5 and 4.6 show the results of the modified double pairwise aggregation algorithm for the tandem queuing network and random planar graph, respectively. For the tandem queuing network we can see that all of the aggregates are of size four which is an improvement over the original double pairwise algorithm for this problem. Unfortunately the aggregates still do not have the regular shape seen for the neighborhood aggregation algorithm. For the random planar graph the number of variables is reduced from 221 on the fine level to 56 on the course-level which is less than the number of aggregates for the original algoirithm.

Algorithm 4.4: MPWA($\bar{A}$, β), Modified Pairwise Aggregation

$U = [1, n]$ (n is the number of fine level points)
for *i=1* **to** n **do**
  $S_i = \{j \in U \backslash \{i\} \mid \bar{a}_{ij} < -\beta \max_{\bar{a}_{ik}<0} |\bar{a}_{ij}|\}$
end
for *i=1* **to** n **do**
  $m_i = |\{j \mid i \in S_j\}| + |\{j \mid j \in S_i\}|$
end
$n_c = 0$
while $U \neq \emptyset$ **do**
  Select $i \in U$ with minimal m_i.
  $n_c = n_c + 1$
  Select $j \in U$ such that $\bar{a}_{ij} = \min_{k \in U} \bar{a}_{ik}$.
  Select $\bar{j} \in U$ such that $\bar{a}_{\bar{j}i} = \min_{k \in U} \bar{a}_{ki}$.
  if $j \in S_i$ *and* $\bar{a}_{ij} < \bar{a}_{\bar{j}i}$ **then**
    $Q_{n_c,i} = 1$
    $Q_{n_c,j} = 1$
    $U = U \backslash \{i, j\}$
  else if $i \in S_{\bar{j}}$ **then**
    $Q_{n_c,i} = 1$
    $Q_{n_c,\bar{j}} = 1$
    $U = U \backslash \{i, \bar{j}\}$
  else
    $Q_{n_c,i} = 1$
    $U = U \backslash \{i\}$
  end
  forall k *such that* $Q_{n_c,k} = 1$ **do**
    $m_l = m_l - 1$ for $l \in S_k$
    $m_l = m_l - 1$ for $k \in S_l$
  end
end
return Q^T

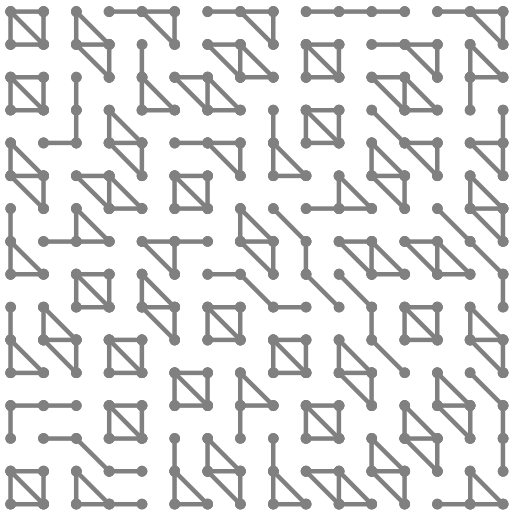


Figure 4.5: Tandem Queuing Network ($n = 15$): Aggregates formed using Modified Double Pairwise Aggregation.

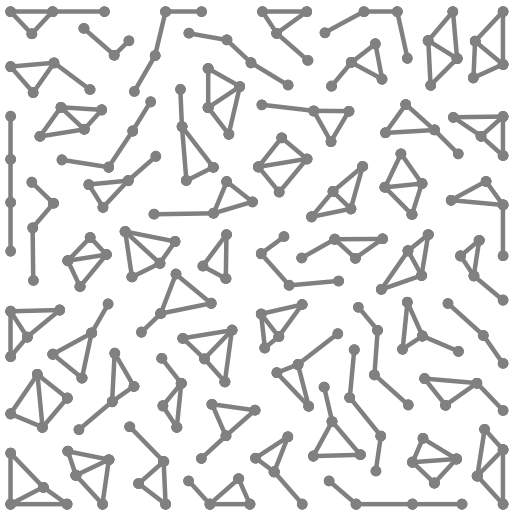


Figure 4.6: Random Planar Graph ($n = 221$): Aggregates formed using Modified Double Pairwise Aggregation.

5 Numerical Results

In this section, we compare neighborhood and double pairwise aggregation using several test problems. Using these examples, we also examine the performance of these algorithms with and without top-level acceleration in our K-cycle aggregation multigrid method. We use the shorthand notations NA for neighborhood aggregation, DPWA for double pairwise aggregation and M-DPWA for modified double pairwise aggregation. We also denote the K-cycle aggregation multigrid method with top-level acceleration by K-cycle$^+$. We consider one one-dimensional Markov chain and three two-dimensional Markov chains. The first two test problems are generated by undirected graphs with weighted edges. The transition probabilities are determined by the weights: the weight of the edge from node i to node j, divided by the sum of the weights of all outgoing edges from node i gives the transition probability from node i to node j. All of the test problems we consider have a subdominant eigenvalue approaching one as the number of states is increased. This property is important because one-level or two-level iterative methods are ineffective in this case. Across the different examples, we include many of the same variables in our discussion of the results. In the tables, 'n' is the number of degrees of freedom and 'γ' is the geometric mean of the convergence factors of the last five K-cycles. The convergence factor of a K-cycle is defined as the ratio of the one-norm of the residual, $\|A\mathbf{x}_i\|_1$, after and before the cycle. For our method to be considered scalable or optimal, γ should be bounded away from one as n increases. This also bounds the number of required iterations. The number of iterations performed is given by 'it' and 'lev' is the number of levels in the last cycle. We use 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 as a measure of operator complexity. 'C_{op}' is the operator complexity of the last cycle. Our measure of operator complexity gives a good indication of the amount of work required for one cycle. As n increases, it should be bounded by a constant not too much larger than one if our method is to be considered scalable (or optimal). To compare the overall efficiency of different methods we also provide an effective convergence factor, defined as $\gamma_{eff} = \gamma^{1\backslash C_{op}}$. The convergence factor makes it easier to compare the overall efficiency of different methods since it takes work into account. For all the numerical results presented in this paper, we start from a random, strictly positive initial guess and iterate until the residual has been reduced by a factor of 10^{-8} measured in the one-norm, or until 100 cycles have been performed, whichever comes first. We do a direct solve on the coarse level when $n < 12$. The number of pre-relaxations and post-relaxations is one ($\nu_1 = \nu_2 = 1$). For simplicity the weight in our weighted Jacobi relaxation scheme is always chosen as 0.7.

5.1 Uniform chain with two weak links

The first test problem we consider is a uniform chain with two weak links. All of the weights are uniform, except two weak links with $\epsilon = 10^{-3}$ in the middle of the chain (Figure 5.1). The NA results are in Table 5.1 and the DPWA results are in Table 5.2. For both NA and DPWA, top-level acceleration improves convergence. Both with and without top-level acceleration, convergence is better for NA than for DPWA. It is interesting to note that the amount of work required for NA is higher than for DPWA. Both the computational complexity and convergence properties are contrary to what we expected. Our expectation was for the amount of work to be smaller with NA and the convergence properties to be better with DPWA. In this simple case, the neighborhoods generated by NA are relatively small which leads to more aggregates. For this problem more aggregates are generated by NA on average than by DPWA. For example, at the finest level when $n = 4374$ the number of variables is reduced from 4374 to 1458 using NA while the number of variables is reduced from 4374 to 1104 using DPWA. This helps explain why the amount of work required per cycle is higher for NA.

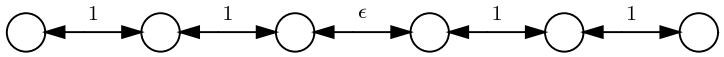


Figure 5.1: Graph for uniform chain with two weak links

	K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
54	34	1.85	3	0.66	0.80	18	1.85	3	0.44	0.64
162	34	2.22	4	0.66	0.83	18	2.22	4	0.47	0.71
486	35	2.48	5	0.68	0.85	19	2.48	5	0.47	0.74
4374	37	2.77	7	0.69	0.87	21	2.77	7	0.52	0.79
39366	38	2.90	9	0.68	0.87	20	2.90	9	0.51	0.79

Table 5.1: Uniform chain with two weak links. Neighborhood aggregation.

	K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
54	50	1.62	3	0.76	0.84	24	1.62	3	0.58	0.71
162	57	1.88	4	0.77	0.87	28	1.88	4	0.63	0.78
486	57	1.82	4	0.76	0.86	28	1.82	4	0.48	0.67
4374	56	1.96	6	0.74	0.86	29	1.96	6	0.63	0.79
39366	71	1.98	7	0.76	0.87	31	1.98	7	0.69	0.83

Table 5.2: Uniform chain with two weak links. Double pairwise aggregation.

5.2 Anisotropic 2D lattice

The first two-dimensional Markov chain we consider is an anisotropic 2D lattice. The horizontal weights are 1 while the vertical weights are $\epsilon = 10^{-6}$ (Figure 5.2). Tables 5.3 and 5.4 show the results for this problem. For NA, top-level acceleration improves convergence and we note that while the number of iterations required to reach convergence is constant both with and without top-level acceleration the computational complexity appears be growing as the state size increases. For DPWA, the computational complexity is smaller than for NA. As with the previous test problem, this can be explained by an average neighborhood size smaller than four which implies that on average the number of variables on the coarse-grid will be larger for NA than for DPWA. It is troubling to note that with DPWA when $n = 16384$ the number of iterations actually increases when we use top-level acceleration and in this case we do not get convergence. Although top-level acceleration is not guaranteed to improve convergence it is interesting that by applying this method we actually worsen convergence. Perhaps DPWA changes the aggregation too much from iteration to iteration and this causes the convergence deterioration with top-level acceleration. However, this is only one possible explanation.

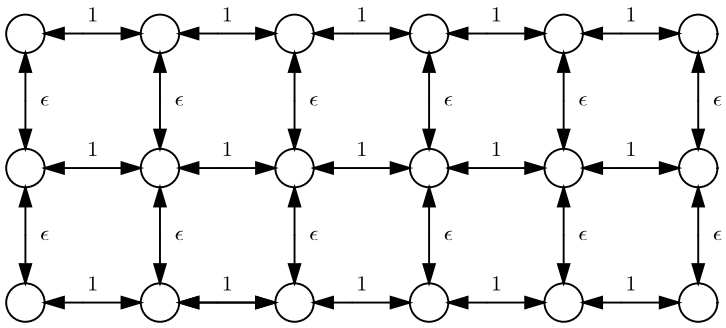


Figure 5.2: Graph for 2D lattice with anisotropic weights

	K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
64	30	1.83	3	0.59	0.75	13	1.83	3	0.31	0.52
256	33	2.53	5	0.65	0.84	17	2.53	5	0.43	0.71
1024	34	2.93	6	0.65	0.87	18	2.93	6	0.44	0.76
4096	34	2.77	7	0.66	0.86	18	2.77	7	0.46	0.76
16384	35	2.95	9	0.67	0.87	18	2.95	9	0.46	0.77
65536	35	3.18	10	0.68	0.89	19	3.18	10	0.48	0.79

Table 5.3: 2D lattice with anisotropic weights. Neighborhood aggregation.

	K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
64	>100	1.49	3	1.00	1.00	29	1.49	3	0.86	0.91
256	50	1.64	4	0.76	0.84	23	1.64	4	0.56	0.70
1024	>100	1.77	5	1.00	1.00	72	1.77	5	0.95	0.97
4096	56	1.84	6	0.77	0.87	25	1.84	6	0.43	0.63
16384	95	1.89	7	0.97	0.98	>100	1.89	7	1.00	1.00
65536	56	1.92	8	0.76	0.86	27	1.92	8	0.43	0.65

Table 5.4: 2D lattice with anisotropic weights. Double pairwise aggregation.

5.3 Tandem queuing network

Since Markov chains are important for studying queuing and manufacturing systems, the next test problem we consider is an open tandem queuing network [18]. Two finite queues with single servers are placed in tandem. 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. Figure 5.3 shows the basic structure of the tandem queuing network. Let N be the largest number of customers allowed in each queue. For our numerical tests, $N = 15$, 31, 63, 127, 255, $\mu = 10$, $\mu_1 = 11$, and $\mu_2 = 10$. The states of the system can be represented by tuples $(n_1\,, n_2)$, where n_1 is the number of customers waiting in the first queue and n_2 is the number waiting in the second queue. The total number of states is given by $(N + 1)^2$. A 2D regular lattice can be used to represent the states and the transition rates are as indicated on Figure 5.4. The transition probabilities are determined the same way as in all previous problems using the transition rates as weights.

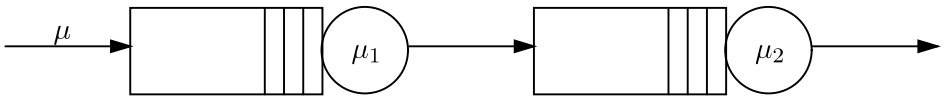


Figure 5.3: Tandem queuing network

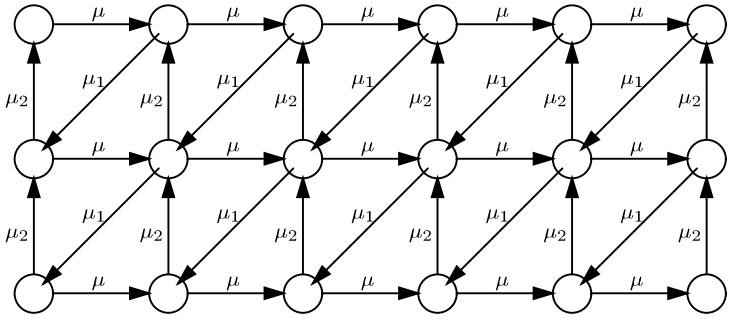


Figure 5.4: Graph for tandem queuing network

The numerical results for the tandem queuing network with NA are in Table 5.5. In these tables, we include W-cycles as well as K-cycles with and without top-level acceleration. From the results it is clear that W-cycles do not scale optimally and that K-cycles greatly improve convergence. This is also true when DPWA is used, which can be seen from Table 5.6. If we compare the two aggregation algorithms for K-cycles without top-level acceleration, we note that NA operator complexity is less than DPWA operator complexity but as n increases, C_{op}, appears to be bounded in both cases. The convergence factors are very similar for the two aggregation schemes and the number of iterations required to reach convergence is comparable except when $n = 31$ where DPWA performs noticeably better and when $n = 255$ where NA performs noticeably better. For NA, performing top-level acceleration significantly improves convergence. However, with DPWA performing top-level acceleration actually increases the number of iterations required to reach convergence for large problems. As mentioned previously, the reason why convergence deteriorates using top-level acceleration might be because DPWA changes the aggregation too much from iteration to iteration. To test this hypothesis, we modify the K-cycle multigrid method. The aggregation at the finest level is changed only every five iterations of the K-cycle, as apposed to every iteration of the K-cycle. For coarser levels, the aggregation is determined as usual. The results are in Table 5.7 and it is obvious that in this case top-level acceleration improves convergence which confirms our hypothesis.

We propose one addition method to try to improve our DPWA results for this problem. Suppose that the problem matrix is used to determine the strength of connection for this

problem (instead of the scaled problem matrix). Then, from Figure 5.4, we can see that the strongest connection in the graph should be between diagonal elements since the transition probability will be the largest between these nodes. However, including these diagonal nodes in a particular aggregate is strongly dependent on the order on which the initial nodes are chosen. This suggests that we might be able to improve the properties of DPWA if we consider M-DPWA. The results from M-DPWA are in Table 5.8. If we compare M-DPWA and DPWA without top-level acceleration we note that the computational complexity is lower for M-DPWA in all but one case and the number of iterations required to reach convergence is lower in all cases. For problems with a relatively small number of states, M-PDWA outperforms NA with and without top-level acceleration. Without top-level acceleration, M-DPWA is competitive with NA for problems with relatively large state size in terms of convergence; however, C_{op} is larger which is not unexpected given that NA produces fewer aggregates on average than M-DPWA. With top-level acceleration, M-DPWA is outperformed by NA for large problems.

	W-cycle					K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
15	67	1.47	3	0.84	0.88	42	1.47	3	0.72	0.80	27	1.47	3	0.56	0.68
31	>100	1.47	4	0.91	0.94	48	1.47	4	0.74	0.81	31	1.47	4	0.60	0.71
63	>100	1.50	4	0.91	0.94	50	1.50	4	0.73	0.81	33	1.50	4	0.64	0.74
127	>100	1.51	5	0.95	0.96	57	1.51	5	0.76	0.83	32	1.51	5	0.63	0.73
255	>100	1.50	6	0.95	0.97	63	1.50	6	0.77	0.84	37	1.50	6	0.56	0.68

Table 5.5: Tandem queuing network. Neighborhood aggregation.

	W-cycle					K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
15	51	2.18	4	0.78	0.89	40	2.18	4	0.75	0.88	26	2.14	4	0.54	0.75
31	57	2.36	5	0.81	0.91	39	2.39	5	0.74	0.88	35	2.35	5	0.54	0.77
63	70	2.45	6	0.84	0.93	48	2.43	6	0.73	0.88	64	2.45	6	0.72	0.88
127	82	2.54	7	0.85	0.94	58	2.51	7	0.72	0.88	76	2.51	7	0.73	0.88
255	97	2.56	8	0.87	0.95	73	2.56	8	0.78	0.91	99	2.55	8	0.79	0.91

Table 5.6: Tandem queuing network. Double pairwise aggregation.

	K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
15	37	2.18	4	0.71	0.86	23	2.19	4	0.46	0.70
31	40	2.38	5	0.70	0.86	25	2.39	5	0.53	0.77
63	52	2.45	6	0.77	0.90	29	2.47	6	0.54	0.78
127	62	2.52	7	0.75	0.89	34	2.50	7	0.54	0.78
255	75	2.51	8	0.77	0.90	46	2.57	8	0.64	0.84

Table 5.7: Tandem queuing network. Double pairwise aggregation with fixed fine-level aggregation.

	K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
15	28	2.14	4	0.57	0.77	21	2.06	4	0.46	0.69
31	30	2.30	5	0.58	0.79	25	2.30	5	0.42	0.69
63	35	2.42	6	0.61	0.81	45	2.41	6	0.48	0.73
127	50	2.47	7	0.63	0.83	82	2.49	7	0.75	0.89
255	69	2.56	8	0.70	0.87	>100	2.56	8	0.88	0.95

Table 5.8: Tandem queuing network. Modified double pairwise aggregation.

5.4 Random Planar graph

As our final test problem, we consider an unstructured planar (undirected) graph and calculate the stationary probability distribution of a random walk on the graph. The graph is generated by choosing n random points in the unit square and triangulating them using Delaunay triangulation. The random walk on the graph is modeled by a Markov chain with the transition probability form node i to node j given by the reciprocal of the number of edges incident on node i (equal weights).

Table 5.9 shows the results using NA. As in the tandem queuing network, we see significant improvement once we move from W-cycles to K-cycles. Table 5.10 shows the results using DPWA. With DPWA, as apposed to NA, W-cycles are scalable and although K-cycles without top-level acceleration require fewer iterations to reach convergence, when $n = 16384$ W-cycles actually perform better than K-cycles with top-level acceleration. The DPWA results indicate that DPWA for K-cycles without top-level acceleration is scalable; however, it is obvious from the NA results that NA for K-cycles with top-level acceleration is optimal.

For DPWA, convergence is worse with top-level acceleration than without so we again modify the K-cycle multigrid algorithm so the aggregation on the finest level is only up-

dated every five iterations of the K-cycle. The results are in Table 5.11 and it is clear from the results that K-cycle multigrid with top-level acceleration now improves convergence and in fact the number of iterations required to reach convergence is lower than for NA. As with the tandem queuing network, we also include the results using M-DPWA. Given the random structure of the graph there is no reason to assume that M-DPWA will perform better than DPWA and from Table 5.12 we see that the results are slightly worse in most cases than those given for DPWA.

	W-cycle					K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
1024	90	1.24	4	0.88	0.90	56	1.24	4	0.80	0.84	26	1.24	4	0.61	0.67
2048	>100	1.23	4	0.91	0.93	62	1.23	4	0.81	0.85	27	1.23	4	0.62	0.68
4096	>100	1.24	4	0.92	0.93	66	1.24	4	0.84	0.87	28	1.24	4	0.61	0.67
8192	>100	1.24	4	0.93	0.94	73	1.24	4	0.87	0.90	28	1.24	4	0.48	0.55
16384	>100	1.25	5	0.94	0.95	79	1.25	5	0.88	0.90	31	1.25	5	0.67	0.72

Table 5.9: Random planar graph. Neighborhood aggregation.

	W-cycle					K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
1024	37	1.87	5	0.72	0.84	30	1.88	5	0.67	0.81	24	1.88	5	0.59	0.75
2048	41	1.89	5	0.77	0.87	31	1.90	5	0.67	0.81	27	1.90	5	0.70	0.83
4096	45	1.95	6	0.80	0.89	35	1.95	6	0.73	0.85	35	1.95	6	0.77	0.88
8192	46	1.96	6	0.80	0.89	36	1.96	6	0.73	0.85	41	1.95	6	0.84	0.92
16384	49	1.98	7	0.81	0.90	39	1.98	7	0.75	0.87	49	1.98	7	0.85	0.92

Table 5.10: Random planar graph. Double pairwise aggregation.

	K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
1024	34	1.88	5	0.68	0.81	19	1.87	5	0.44	0.65
2048	35	1.89	5	0.69	0.82	18	1.90	5	0.45	0.66
4096	39	1.94	6	0.73	0.85	19	1.95	6	0.49	0.69
8192	40	1.96	6	0.75	0.86	20	1.96	6	0.48	0.69
16384	42	1.98	7	0.78	0.88	21	1.98	7	0.57	0.75

Table 5.11: Random planar graph. Double pairwise aggregation with fixed fine-level aggregation.

	K-cycle					K-cycle$^+$				
n	it	C_{op}	lev	γ	γ_{eff}	it	C_{op}	lev	γ	γ_{eff}
1024	31	1.88	5	0.66	0.80	26	1.88	5	0.69	0.82
2048	33	1.89	5	0.70	0.83	26	1.89	5	0.73	0.85
4096	35	1.94	6	0.70	0.83	41	1.94	6	0.80	0.89
8192	36	1.95	6	0.76	0.87	36	1.96	6	0.82	0.90
16384	40	1.98	7	0.79	0.89	49	1.98	7	0.87	0.93

Table 5.12: Random planar graph. Modified double pairwise aggregation.

6 Conclusion

We have presented two different aggregation algorithms for a K-cycle multigrid method for Markov chains. Both the neighborhood aggregation algorithm and double pairwise aggregation algorithm use the information in the scaled matrix problem and determine a strength of connection measure to form aggregates. While the neighborhood aggregation algorithm forms aggregates using strongly connected neighborhoods, double pairwise aggregation uses two passes of a pairwise matching algorithm where the strongest connection is favored in forming pairs. The K-cycle scheme we propose is a recursively accelerated W-cycle where acceleration is performed by finding the optimal linear combination of two iterates. Given the small dimensionality of our minimization problem we are able to develop an efficient solution method. We also implement a top-level acceleration algorithm where the proposed acceleration finds the optimal linear combination of three K-cycle iterates.

Using several test problems we are able to show how our two aggregation algorithms compare. Our results indicate that the neighborhood aggregation algorithm performs better than the double pairwise aggregation algorithm. For some problems the performance of double pairwise aggregation can be improved when we consider a modified version. In

the modified version, nodes are considered strongly connected if they strongly influence our initial node or if they are strongly influenced by our initial node. In the neighborhood aggregation algorithm strength of connection is determined exactly the same way as it is for our modified double pairwise aggregation algorithm. For neighborhood aggregation, top-level acceleration is able to improve convergence; however for double pairwise aggregation top-level acceleration convergence actually becomes worse for some problems. It is not entirely clear why this occurs although if we fix the top-level aggregates for a number of iterations we are able to improve convergence using top-level acceleration. Future work could include a further exploration of this phenomenon as well as finding ways to fix it. It is also not entirely clear why neighborhood aggregation performs better the double pairwise aggregation and a further examination of this could lead to better aggregation algorithms.

References

[1] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, SIAM, Philadelphia, 1987. 3

[2] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing, 55 (1995), pp. 379–393. 15

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

[4] G. Horton and S. T. Leutenegger, *A Multi-Level Solution Algorithm for Steady-State Markov Chains*, ACM SIGMETRICS, New York, 1994. 1

[5] C. Isensee, *Aggregationsstrategien für ein Multilevel-Verfahren zur Lösung von Markov-Ketten*, PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2003. 15

[6] C. Isensee and G. Horton, *A Multi-Level Method for the Steady Sate Solution of Markov Chains*, Simulation and Visualisierung, SCS European Publishing House, Bonn, 2004. 1, 15

[7] H. Kim, J. Xu, and L. Zikatanov, *A multigrid method based on graph matching for convectiondiffusion equations*, Numer. Lin. Alg. Appl., 10 (2003), pp. 181–195. 15

[8] U. R. Krieger, *Numerical solution of large finite markov chains by algebraic multigrid techniques*, in Numerical Solution of Markov Chains, W. Stewart, ed., Kluwer, Norwell, MA, 1995, pp. 425–443. 1

[9] U. Labsik, *Algorithmische Erweiterung des Multi-Level-Verfahrens zum Lösen von Markov-Ketten*, PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1997. 15

[10] 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, Norwell, MA, 1995, pp. 425–443. 1, 15

[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. 7

[12] Y. Notay, *An aggregation-based algebraic multigrid method*, Tech. Report GANMN 0802, Université Libre de Bruxelles, Brussels, Belgium, 2008. http://homepages.ulb.ac.be/~ynotay. 1, 2

[13] Y. Notay and P. S. Vassilevski, *Recursive krylov-based multigrid cycles*, Numer. Linear Algebra Appl., 0 (2006), pp. 1–20.

[14] C. W. Oosterlee and T. Washio, *Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows*, SIAM J. Sci. Comp., 21 (2000), pp. 1670–1690. 2

[15] H. D. 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. Comp., accepted, (2009). 2, 3, 8

[16] H. D. Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Ruge, and G. Sanders, *Algebraic multigrid for markov chains*, SIAM J. Sci. Comp., submitted, (2009). 2

[17] 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 J. Sci. Comp., 30 (2008), pp. 2235–2262. 1, 5, 15, 21

[18] W. J. Stewart, *An Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, 1994. 29

[19] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.

[20] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid on unstructured meshes*, Tech. Report UCD/CCM Report No. 34, University of Colorado at Denver, Denver, Colorado, 1994. 1

[21] E. Virnik, *An algebraic multigrid preconditioner for a class of singular m-matrices*, SIAM J. Sci. Comp., 29 (2007), pp. 1982–1991.

[22] T. Washio and C. W. Oosterlee, *Krylov subspace acceleration for nonlinear multigrid schemes*, Electronic Transactions on Numerical Analysis, 6 (1997), pp. 271–290. 2