Computing and Informatics, Vol. 30, 2011, 225–246

MULTILEVEL AGGREGATION METHODS FOR SMALL-WORLD GRAPHS WITH APPLICATION TO RANDOM-WALK RANKING

Hans de Sterck

University of Waterloo
Department of Applied Mathematics
e-mail: hdesterck@math.waterloo.ca

Van Emden Henson, Geoffrey Sanders

Lawrence Livermore National Laboratory
Center for Applied Scientific Computing
e-mail: {henson5, sanders29}@llnl.gov

Revised manuscript received 1 December 2010

Abstract. We describe multilevel aggregation in the specific context of using Markov chains to rank the nodes of graphs. More generally, aggregation is a graph coarsening technique that has a wide range of possible uses regarding information retrieval applications. Aggregation successfully generates efficient multilevel methods for solving nonsingular linear systems and various eigenproblems from discretized partial differential equations, which tend to involve mesh-like graphs. Our primary goal is to extend the applicability of aggregation to similar problems on small-world graphs, with a secondary goal of developing these methods for eventual applicability towards many other tasks such as using the information in the hierarchies for node clustering or pattern recognition. The nature of small-world graphs makes it difficult for many coarsening approaches to obtain useful hierarchies that have complexity on the order of the number of edges in the original graph while retaining the relevant properties of the original graph. Here, for a set of synthetic graphs with the small-world property, we show how multilevel hierarchies formed with non-overlapping strength-based aggregation have optimal or near optimal complexity. We also provide an example of how these hierarchies are employed to accelerate convergence of methods that calculate the stationary probability vector of large, sparse, irreducible, slowly-mixing Markov chains on such small-world graphs. The stationary probability vector of a Markov chain allows one to rank the nodes in a graph based on the likelihood that a long random walk visits each node. These ranking approaches have a wide range of applications including information retrieval and web ranking, performance modeling of computer and communication systems, analysis of social networks, dependability and security analysis, and analysis of biological systems [19].

Keywords: Markov chain, aggregation, multilevel, random graphs

1 INTRODUCTION

Information retrieval applications typically involve networks that are large, with perhaps billions of connected network members, or *nodes*, and unstructured, meaning that there is no regular connection pattern. Further, they are frequently *scale-free*, meaning the nodes cannot be arranged in a low-dimensional manifold without many edges spanning relatively long distances. This property is typified by *power-law* networks, where the fraction $g(k)$ of nodes in the network having k connections to other nodes goes for large values of k as $g(k) \sim k^{-\gamma}$ where γ is a constant whose value is typically in the range $1 < \gamma < 4$. Coarsening the underlying graphs of such networks, while retaining certain relevant properties, is useful to perform scalable calculations used to extract relevant features of the network.

For example, schemes that rapidly extract information from networks often make use of low-rank decompositions of large, sparse data structures such as matrices or tensors. These decompositions usually involve the computation of eigenvectors or singular vectors (or analagous tensor bases) and fast, scalable approximation to such vectors is important for the underlying scheme to be practical for large data sets. Multilevel hierarchies formed on coarse versions of the original graphs often allow rapid calculation of low-rank approximations.

Coarsening a scale-free graph into a hierarchy with optimal complexity that is still rich enough to approximate desired features of the original graph can be difficult, and approaches that are quite successful for simpler graphs often fail. In this work, we focus on a specific class of nonsymmetric eigenvalue problems that arises when computing the steady-state of a Markov chain, which determines popularity of each entity within a network. Many of the techniques presented here are specific to this class of problem, however, similar methods may be employed for eigenproblems relating to other calculations of interest. A few common applications include approximating the commute time between two nodes in a graph using several of the lowest eigenmodes of the graph Laplacian [15], clustering a graph using an eigenvector of the graph Laplacian corresponding to the smallest positive eigenvalue (Fiedler vector partitioning) [13, 14], and approximating the number of triangles in a graph using several of the largest eigenvalues of the adjacency matrix [33]. We further suggest that the coarsening approaches may be useful for tasks that do not necessarily involve eigenproblems, such as clustering or pattern recognition (a related approach to image segmentation is employed in [18]).

A Markov chain with n states is represented by an $n \times n$ non-negative matrix, B, that is column-stochastic, $\mathbf{1}^t B = \mathbf{1}^t$. The stationary vector that we seek, $\mathbf{x}$, satisfies the following eigenproblem with known eigenvalue:

$$B\mathbf{x} = \mathbf{x}, \qquad \|\mathbf{x}\|_1 = 1, \qquad \mathbf{x} \geq 0, \tag{1}$$

where the normalization constraint and the non-negativity constraint make $\mathbf{x}$ a *probability vector*. If every node in the underlying network is connected to every other node through a series of directed arcs, then the matrix B is called *irreducible*. We assume this property, which guarantees that there is a unique solution to (1) that is strictly positive ($\mathbf{x} > \mathbf{0}$), by the Perron-Frobenius theorem (see [2, 10] for details).

Because the sizes of the graphs of interest tend to grow very large, it is imperative that we seek solution methods that are *algorithmically scalable*. An algorithmically scalable method computes an approximate solution (to a specified error tolerance) with an amount of work proportional to the amount of information in matrix B, which for the problems we consider is proportional to the number of edges in the graph. The simplest solution method is the power method, which converges to $\mathbf{x}$ when B is *aperiodic*, meaning the lengths of all directed cycles on the graph of B have greatest common denominator equal to one. Letting $\Sigma(B)$ denote the set of eigenvalues of B, it is well-known that the rate of convergence of the power method is dependent on the*subdominant eigenvalue(s)*, λ_2, where

$$|\lambda_2| = \max |\lambda| \quad \text{ for } \lambda \in \Sigma(B) \setminus \{1\}.$$

When $|\lambda_2| \approx 1$, B is called *slowly-mixing*, and if there exist eigenvalues $\lambda \neq 1$ such that $\mathrm{Re}\,\lambda \approx 1$, then the convergence rates of the power method and of related classical iterative techniques are unacceptably close to 1 as well. For many Markov chains of interest there are nondominant eigenvalues that approach 1 as the problem size increases; for these problems the power method and its relatives are *not* algorithmically scalable. Moreover, for many of these problems, applying Krylov acceleration (such as preconditioned GMRES) to classical iterative methods does not improve the scalability. This is largely because these techniques influence the approximate solution locally, and a great many iterations are required to properly obtain the desired global solution from a poorly distributed initial guess. Multilevel iterative methods are employed to accelerate convergence for this type of problem by reducing error components at different scales on progressively coarser levels.

Methods based on aggregation of Markov states have proven to be fruitful approaches 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 [31]. Two-level aggregation/disaggregation has been studied extensively since [19, 28, 21, 20, 23, 7, 24, 30]. Convergence proofs are given for two-level aggregation/disaggregation methods in [23, 24].

Two-level iterative aggregation/disaggregation can naturally be extended to multiple levels, along the lines of multigrid methods for linear systems of equations [6]. Direct extension of two-level aggregation/disaggregation to multiple levels was first explored in [17, 22], and later also in [10]. In the latter, aggregates are formed algebraically based on "strength of connection" in the scaled problem matrix, where each column is scaled by the value of the the current iterate at the corresponding graph vertex. Thus coarse grids on all levels are formed adaptively, based on the current iterate, and are different in each iteration. However, numerical results in [8] 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. For many types of problems, employing so-called *W-cycles* (in which coarser levels are visited increasingly often) will restore optimal convergence properties; experience shows, however, that this is not the case with existing multilevel aggregation methods on scale-free graphs. For a Markov problem posed on a *mesh-like* graphs, or a graph whose nodes can be embedded into low-dimensional manifold with very few relatively long-distance connections, smoothed aggregation (SA) may be employed to improve the approximation properties of the multilevel hierarchy and result in a scalable method [8].

Often a multilevel hierarchy may not be rich enough to provide a useful standalone method, but a simple top-level acceleration technique may be employed to greatly improve convergence. The schemes we present here use multilevel hierarchies that adapt with every cycle and standard Krylov acceleration cannot be applied to accelerate these methods because the spaces involved are not related by a fixed preconditioner. However, *flexible* acceleration is possible for methods with changing hierarchies or nonstationary preconditioners. In this paper, we do not use flexible GMRES or flexible CG, but we discuss an acceleration technique that is customized to solve problem (1), employing a constrained minimization problem as presented in [11].

This paper demonstrates the use of multilevel hierarchies within accelerated versions of the classical unsmoothed aggregation algorithm [17, 10] and the SA algorithm given in [8] in the context of Markov chains posed on scale-free networks (Figure 1). Since the *degrees* (number of edges coming from each node) of the nodes are distributed according to a power law, the number of nodes with small degree is very large while the number of nodes with large degree is very small (but nonzero) [1]. Scale-free networks frequently have the *small-world property,* where the minimal path length between any pair of nodes is small and independent of the size of the network. The power-law distribution and small-world property pose fundamental difficulties for the multilevel methods we employ, which were originally

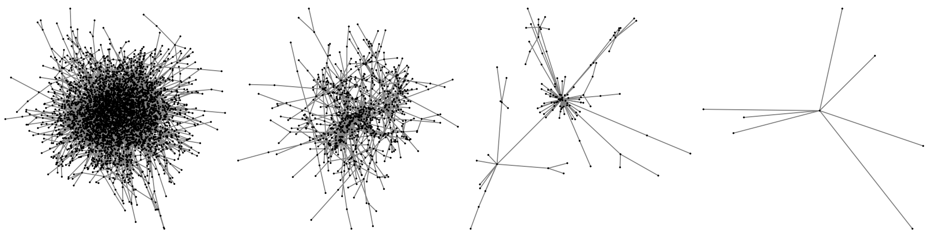


Fig. 1. A small version of the Barábasi-Albert model (described in Section 4.1), coarsened using neighborhood aggregation. To the far left is a visualization of the original graph, center left depicts the first coarsening, center right the second coarsening, and far right the third coarsening. Black dots represent nodes, and light gray lines represent bidirectional links. Visualization was performed by randomly distributing nodes in the unit circle and performing several iterations of sequentially moving the ith node's location to a weighted average of the locations of all nodes connected to the ith node and the current location of the ith node itself.

designed for graphs coming from discretized partial differential equations (which are *mesh-like* and neither scale-free nor small-world). We show, for a specific class of scale-free test problems, that a multilevel, pure aggregation approach can generate multilevel hierarchies with complexity on the order of the number of edges in the graph while essentially retaining power-law distributions on coarse levels. Applying acceleration techniques with these multilevel aggregations improves convergence to the stationary vector significantly, but the methods designed for mesh-like graphs are not automatically algorithmically scalable for scale-free graphs. Further, we replace the standard aggregation approach with a simple aggregation routine that takes advantage of tree-like structure within a graph, and scalability is achieved for a simple model problem that is highly tree-like.

The rest of this paper is organized as follows. Section 2 mostly reviews several multilevel aggregation techniques for accelerating ranking calculations, and Section 2.3 introduces a new aggregation approach that takes advantage of tree-like structure within a graph. Section 3 reviews a top-level Krylov-like acceleration that enhances the robustness of multilevel aggregation techniques. Section 4 presents numerical results and Section 5 has concluding remarks.

2 MULTILEVEL AGGREGATION FOR MARKOV CHAINS

We briefly review the multilevel aggregation algorithm for Markov chains from [17, 22, 10, 8] following the presentation in [8]. *Pure aggregation* (often called *unsmoothed aggregation*) is the process of building intergrid transfer operators from local groupings of the nodes within a graph. *Smoothed aggregation* is a commonly used technique where these intergrid transfer operators are smoothed (presented briefly in Section 2.4) to enhance approximation within coarse spaces.

2.1 Pure Aggregation Multilevel Methods

We describe the process of using aggregation to coarsen a graph. Let $A = I - B$ and rewrite $B\mathbf{x} = \mathbf{x}$ as

$$A\mathbf{x} = 0. \tag{2}$$

Rewrite the exact solution, $\mathbf{x}$, in terms of the current approximation, $\mathbf{x}_i$, and its multiplicative error, $\mathbf{e}_i$, or $\mathbf{x} = \text{diag}(\mathbf{x}_i)\mathbf{e}_i$, obtaining

$$A\text{diag}(\mathbf{x}_i)\mathbf{e}_i = 0. \tag{3}$$

We assume here that all components of $\mathbf{x}_i$ are nonzero (Perron-Frobenius theory guarantees that the exact solution, $\mathbf{x}$, also has this property, see [2]). At convergence, the multiplicative error is $\mathbf{e}_i = \mathbf{1}$, the vector of all ones.

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

$$Q = \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ \hline 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ \hline 0 & 0 & 1 & \cdots \\ 0 & 0 & 1 & \cdots \\ \hline 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{array}\right]. \tag{4}$$

Aggregates are determined using strength of connection in the scaled problem matrix, $A\text{diag}(\mathbf{x}_i)$. The details of the aggregation algorithms we use are explained below in Sections 2.2 and 2.3.

Once Q has been determined, a coarse-level version of Equation (3) is constructed:

$$Q^T A\text{diag}(\mathbf{x}_i)Q\mathbf{e}_c = 0, \tag{5}$$

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 $P = \text{diag}(\mathbf{x}_i)Q$, and write $RAP\mathbf{e}_c = 0$ with the coarse-level operator, A_c, defined by

$$A_c = RAP. \tag{6}$$

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 equation

$$A_c(\mathrm{diag}(Q^T\mathbf{x}_i))^{-1}\mathbf{x}_c = 0. \tag{7}$$

We define coarse-level stochastic matrix, B_c, as

$$B_c = Q^T B\mathrm{diag}(\mathbf{x}_i)Q(\mathrm{diag}(Q^T\mathbf{x}_i))^{-1}. \tag{8}$$

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 $\mathbf{1}_c$. Some algebra shows $A_c(\mathrm{diag}(Q^T\mathbf{x}_i))^{-1} = I - B_c$.

Coarse-level Equation (7) was introduced in [29] and has a straightforward probabilistic interpretation (see, e.g., [22, 10]). It is well-known that (7) can be used to accelerate simple one-level iterative methods for Equation (2), like the power method or weighted Jacobi relaxation methods. For example, a two-level numerical method (aggregation/disaggregation) may proceed by relaxation on Equation (2) on the fine level, followed by a coarse-level solve of Equation (7), a coarse-level correction according to

$$\mathbf{x}_{i+1} = P(\mathrm{diag}(Q^T\mathbf{x}_i))^{-1}\mathbf{x}_c = P\mathbf{e}_c, \tag{9}$$

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, 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

$$\mathbf{x} \leftarrow (1 - w)\mathbf{x} + wD^{-1}(L + U)\mathbf{x}.$$

A multilevel method can then be obtained by recursively applying the two-level method to coarse-level Equation (7). In multilevel cycling, a parameter μ is used to determine the type of cycle employed. From any given level, μ is the number of times the algorithm cycles to the coarsest grid and returns to the current level before moving to the next finer level. In this paper, we consider so-called V-cyles ($\mu = 1$) and W-cycles ($\mu = 2$); the latter are obtained by applying coarse-level correction twice. The resulting algorithm is printed here as Algorithm 1.

Algorithm 1: Multilevel Aggregation
$\mathbf{x} \leftarrow \mathbf{MA}(A, \mathbf{x}, \nu_1, \nu_2, \mu)$

```
if not on coarsest level then
    $\mathbf{x} \leftarrow \text{Relax}(A, \mathbf{x})$   $\nu_1$ times.;
    Build $Q$.;
    $R \leftarrow Q^T$ and $P \leftarrow \text{diag}(\mathbf{x})Q$.;
    $A_c \leftarrow RAP$.;
    /* first coarse-level solve.                                  */
    $\mathbf{x}_c \leftarrow \mathbf{MA}(A_c\text{diag}(Q^T\mathbf{x})^{-1}, Q^T\mathbf{x}, \nu_1, \nu_2)$;
    /* secondary coarse-level solves.                             */
    for $k = 2, \ldots, \mu$ do
        $\mathbf{x}_c \leftarrow \mathbf{MA}(A_c\text{diag}(Q^T\mathbf{x})^{-1}, \mathbf{x}_c, \nu_1, \nu_2)$;
    end
    /* coarse-level correction.                                   */
    $\mathbf{x} \leftarrow P(\text{diag}(Q^T\mathbf{x}))^{-1}\mathbf{x}_c$;
    $\mathbf{x} \leftarrow \text{Relax}(A, \mathbf{x})$   $\nu_2$ times.;
else
    $\mathbf{x} \leftarrow$ direct solve of $A\mathbf{x} = 0$, $\|\mathbf{x}\|_1 = 1$.
end
```

2.2 Neighborhood-Based Aggregation Routine

We determine aggregates based on strength of connection in the scaled problem matrix $\hat{A} = A\text{diag}(\mathbf{x}_i)$ [10]. In this paper, we use a symmetrized strength of connection measure and the neighborhood-based aggregation technique of [34]. Note that this type of aggregation is a more standard version, typically used for mesh-like graphs, and differs from the aggregation technique used in [10, 8].

Node i is considered to be strongly connected to node j in the graph of $\hat{A}$ if

$$-\hat{a}_{ij} \geq \theta \max_{k \neq i}\{-\hat{a}_{ik}\} \quad \text{or} \quad -\hat{a}_{ji} \geq \theta \max_{k \neq j}\{-\hat{a}_{jk}\}, \tag{10}$$

where θ is a user-selected 'threshold' parameter. The strong neighborhood of any node i, denoted $\mathcal{N}_i$, is the set of all nodes that are strongly connected to i within the graph of $\hat{A}$, including i. In the description of the algorithm, Q_J stands for the index set of the nonzero elements of the Jth column of Q, the aggregation matrix from Equation (4).

In order to address the scale-free nature of the graphs, Algorithm 2 orders the nodes by their degree. This allows the aggregation algorithm to focus on grouping the nodes of large degree first. This is slightly different from the neighborhood aggregation used in [11], where no sorting was assumed. Ordering the nodes based on their approximate popularity (nodal value of current iterate, $\mathbf{x}_i$) was explored in [10, 8] and remains another option.

Algorithm 2: Neighborhood-Based Aggregation
$\{Q_J\}_{J=1}^m \leftarrow$ **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 $\theta$.;
Order the nodes from highest degree to lowest degree. 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 $Q_J \leftarrow \hat{Q}_J$.
```

In Section 4, we demonstrate that it is possible for this simple aggregation technique to yield pure-aggregation multigrid hierachies with bounded complexity for a class of scale-free graphs. This grouping technique may be employed for efficient ranking calculations on mesh-like graphs [11], but the same implementation tends not give optimal ranking calculations for simple scale-free graphs. Instead, we show that merely replacing neighborhood-based aggregation with a modified aggregation routine may give better results.

2.3 Leaf-Based Aggregation Routine

Several scale-free graphs of interest are tree-like or have tree-like components. In this section we develop an aggregation strategy that is applicable to such graphs (the model we consider in Section 4 produces graphs that are highly tree-like). Define a *leaf* node to be a terminal point on the graph, or a node with only one connection. Any leaf is only connected to its *parent node*, and the leaf node's dependence is entirely captured by this connection. Therefore, it should be contained in the same aggregate as its parent. This type of grouping yields a coarse grid representing a subspace that has high approximation for all eigenvectors corresponding to eigenvalues of small magnitude, which allows the multilevel cycle to efficiently sort out these vectors on coarser grids. If node i is the leaf and node j is its parent, then consider rearranging the ith row of $(A - \lambda I)\mathbf{y} = \mathbf{0}$ for any eigenpair $(\lambda, \mathbf{y})$,

$$(a_{ii} - \lambda)y_i = -a_{ij}y_j.$$

If $\lambda < a_{ii}$ for every leaf i, then $\text{sign}(y_i) = \text{sign}(y_j)$, due to $a_{ii} > 0$ and $a_{ij} < 0$. Further, if λ is known, as it is in the ranking application, an iterative relaxation method can quickly resolve the ratio y_i/y_j.

If the underlying graph of our system is purely a tree, simply aggregating all leaves with their parents and any parents with no leaves as children by themselves yields a coarser tree. This strategy is presented in Algorithm 3. For more general graphs, this process will only coarsen the parts of the graph that have tree-like structure. For highly tree-like graphs, repeating Algorithm 3 yields a multilevel hierarchy that has excellent capability for accelerating the ranking calculation, or any calculation having to do with small eigenvectors. In Section 4, we demonstrate that this approach to aggregation gives algorithmically scalable ranking calculations for a class of highly tree-like test problems.

Algorithm 3: Leaf-Based Aggregation
$Q_J{}_{J=1}^{m} \leftarrow \mathbf{LeafAgg}(A\,\text{diag}(\mathbf{x}), \theta)$

```
Let $\mathcal{L}$ be the set of leaves, $\{i : \text{card}(\mathcal{N}_i \setminus \{i\}) = 1\}$.;
Set $\mathcal{R} \leftarrow \{1, \dots, n\}$ and $J \leftarrow 0$.;
/* 1st pass:  group all leaves with their parents            */
for $i \in \mathcal{L}$ do
    if $i \in \mathcal{R}$ then
        Pick parent $j \in \mathcal{N}_i$.;
        $J \leftarrow J + 1$.;
        $Q_J \leftarrow (\mathcal{N}_j \cap \mathcal{R})$.;
        $\mathcal{R} \leftarrow \mathcal{R} \setminus (\mathcal{N}_j \cap \mathcal{R})$.;
    end
end
/* 2nd pass:  put remaining points in their own aggregates */
while $\mathcal{R} \neq \emptyset$ do
    $J \leftarrow J + 1$, $Q_J \leftarrow \{i\}$, $\mathcal{R} \leftarrow \mathcal{R} \setminus \{i\}$.
end
```

Remark: For a pure tree, consider performing the ranking calculation using Gauss-Seidel relaxation, with *parent-first* ordering. Then, consider a multilevel update that employs Algorithm 3. This approach is a *cyclic reduction*, and convergence is attained in one iteration. However, this superb performance will not generalize to more general graphs or to calculations involving an unknown eigenvalue. Therefore, we do not implement this in Section 4, where our intention is to merely show that a change in the aggregation mindset gives scalable results with a purely simultaneous relaxation scheme like Weighted Jacobi. It should be noted that a more sophisticated relaxation (one that updates leaf nodes last) is likely to further improve the efficiency of the calculation, with high parallelism, even for more general graphs that have some tree-like structure. More explicitly, in the first few coarsenings, leaf-based aggregation could be used to trim away the tree-like components of a graph to obtain a much coarser graph with decent approximation for a range of eigenvalues. This idea will be investigated for more complicated networks in further work.

2.4 Operator Smoothing

For multilevel aggregation methods applied to graphs that have mesh-like structure (e.g. planar graphs), the convergence of the ranking calculation is often greatly improved by applying a smoothing operator to the intergrid transfer operators, R and P. This idea is called *smoothed aggregation*, and was originally used for related algorithms applied to nonsinuglar linear systems [34] and later used for Algorithm 1 applied to steady-state Markov eigenproblems [8] on mesh-like graphs. We refer the reader to [8, 9, 32] for the additional details of this method and its relatives.

If the error propagation operator of the relaxation process is sparse, then some version of it is used for smoothing the intergrid transfer operators. For aggregation methods, the intergrid transfer operators are set to

$$R = Q^t(I - \alpha_R A D^{-1}) \tag{11}$$

and

$$P = (I - \alpha_P D^{-1} A)\mathrm{diag}(\tilde{\mathbf{x}})Q, \tag{12}$$

where (α_R, α_P) are smoothing parameters. We use $\alpha_R = \alpha_P = 0.7$ for the results in Section 4. Additionally, a process called lumping may be required to guarantee that coarse-level problem, $A_c\mathbf{x}_c = \mathbf{0}_c$, has a non-negative solution.

This unfiltered operator smoothing is generally not acceptable for scale-free graphs as smoothing dilates the aggregates, making coarse-level operators, A_c, essentially full. For a typical scale-free graph, using larger aggregates does not remedy this situation; it just decreases the ability of the coarse-grid correction to accelerate the convergence. We demonstrate the difficulties with smoothed aggregation in Section 4. A related method based on algebraic multigrid (MCAMG [9]) would have similar difficulties for scale-free graphs, if no modification to the coarsening procedure is made.

3 TOP-LEVEL ACCELERATION

Multilevel hierarchies may not be rich enough to provide useful stand-alone solvers for the stationary probability distribution. However, a simple top-level acceleration

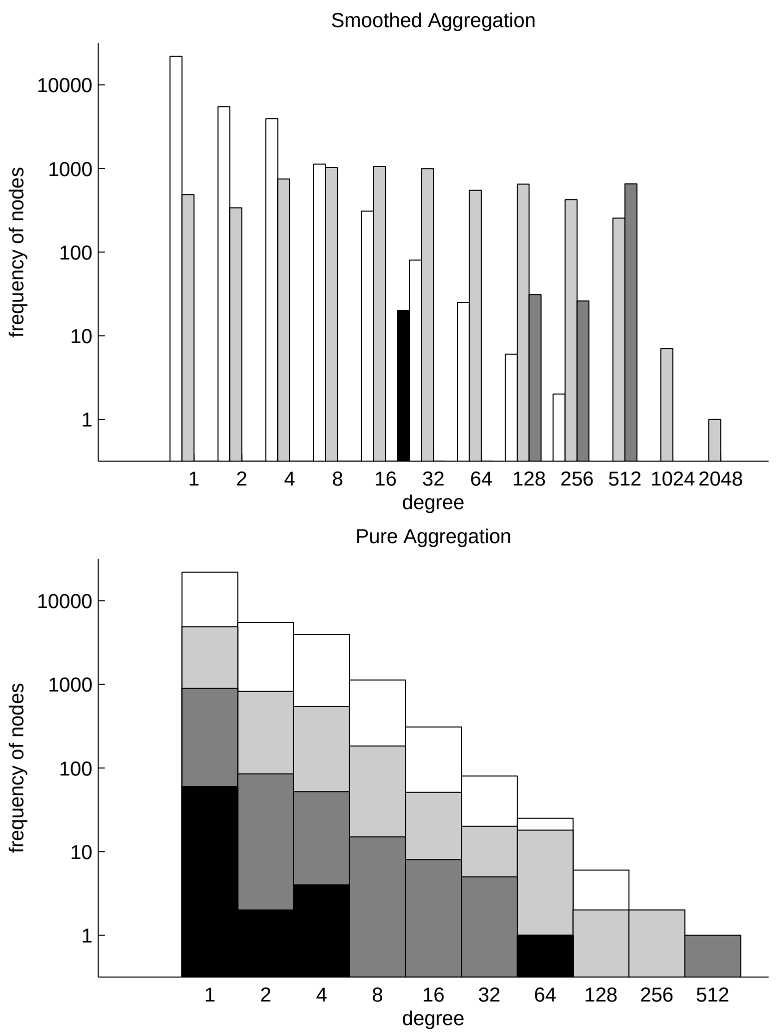


Fig. 2. Histograms of nodal degree for the first four levels of both a smoothed neighborhood aggregation (left) and a pure neighborhood aggregation (right) multigrid hierarchy applied to the $p = 1$ preferential attachment model of size $n = 32\,768$. The histogram for the original graph is shown in the white and increasingly darker shades of grey represent increasingly coarser levels. The original graph is power law in nature, but the smoothed aggregation coarse levels do not maintain the power law; after a few coarsenings the graph matrices are essentially dense. Pure aggregation does a much better job of maintaining a power law degree distribution for the initial coarsening, but the coarser levels are more hub-like.

scheme (similar to Krylov acceleration) may be used to greatly improve the convergence properties of the method. This technique enhances the robustness of this class of methods.

Assume we have some version of Algorithm 1 that produces a sequence of iterates, $\{\mathbf{x}_i\}_{i=1}^{\infty}$, designed to approximate the solution of problem (1). At the k^{th} iteration, let the last m iterates be columns of an $n \times m$ matrix,

$$X = [\mathbf{x}_k, \mathbf{x}_{k-1}, \ldots, \mathbf{x}_{k-m+2}, \mathbf{x}_{k-m+1}], \tag{13}$$

with $\mathbf{x}_k$ being the newest iterate. We call m the *window size*. All columns of X are assumed to have the following properties:

$$\mathbf{x}_i > 0 \quad \text{and} \quad \|\mathbf{x}_i\|_1 = 1, \qquad i = 1, \ldots, n. \tag{14}$$

The natural question arises: is there a linear combination of these m iterates that is optimal in some sense? If the method that produces iterates $\{\mathbf{x}_i\}_{i=1}^{\infty}$ is a stationary preconditioned residual correction, such as the weighted Jacobi iteration or a fixed and additive multigrid correction, the standard answer to this question is to use a Krylov acceleration technique. Algorithm 1, however, is a nonlinear updating scheme, where the multigrid hierarchy changes with each iteration. Therefore we take a fairly standard approach similar to that described in [35], where it is applied to nonlinear PDE problems using FAS *full approximation scheme*, a well-known multigrid technique for nonlinear problems [3, 6]. Both approaches are essentially generalized versions of Krylov acceleration that attempt to minimize the (nonlinear) residual of a linear combination of iterates, each modified for their respective problems.

We define the subset of *probability vectors* in n-dimensional space to be

$$\mathcal{P} := \{\mathbf{w} \in \mathbb{R}^n \quad \text{such that} \quad \|\mathbf{w}\|_1 = 1, \text{ and } \quad \mathbf{w} \geq 0\}.$$

It is easily seen that the functional $\mathcal{F}(\mathbf{w}) = \langle A\mathbf{w}, A\mathbf{w}\rangle$ is uniquely minimal in $\mathcal{P}$ at the solution to (1). Our goal is to minimize this functional within a subset, $\mathcal{V}$, of the range of X, with additional constraints $\|\mathbf{w}\|_1 = 1$ and $\mathbf{w} \geq 0$, ensuring that $\mathbf{w}$ is a probability vector. Formally, this is

$$\text{minimize } \mathcal{F}(\mathbf{w}) \text{ within } \mathcal{V} := \mathcal{P} \cap \mathcal{R}(X). \tag{15}$$

We label the requirements imposed on set $\mathcal{V}$ in the following way:

(C1)	(Normalization Constraint)	$\|\mathbf{w}\|_1 = 1$
(C2)	(Nonnegativity Constraints)	$\mathbf{w} \geq 0$
(C3)	(Subspace Constraint)	$\mathbf{w} \in \mathcal{R}(X)$

Note that **(C1)** is a single *equality constraint* while **(C2)** is a set of *inequality constraints*. Also, **(C3)** is technically a set of equality constraints which determine a linear subspace of $\mathbb{R}^n$. However, because $m << n$ and $\dim(\mathcal{R}(X)^{\perp}) \approx n$, it is more convenient (and equivalent) to use the fact that there exists a vector $\mathbf{z}$ such that $\mathbf{w} = X\mathbf{z}$ for any $\mathbf{w}$ satisfying **(C3)**. This approach is preferred versus explicitly addressing the constraint equations, which are less accessible and inefficient to deal with.

If **(C2)** holds, then the absolute values in $\|\mathbf{w}\|_1$ are unnecessary. Thus, **(C1)** is a linear constraint, $\sum_i w_i = 1$. Furthermore, because $\mathbf{w} \in \mathcal{R}(X)$, there exists a vector $\mathbf{z}$ such that $\mathbf{w} = X\mathbf{z}$. This implies that

$$\|\mathbf{w}\|_1 = \sum_{i}^{n} w_i = \sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij} z_j = \sum_{j=1}^{m} z_j \sum_{i=1}^{n} X_{ij} = \sum_{j=1}^{m} z_j,$$

due to each column in X being a probability vector. Therefore, the constrained subset is equivalently written as

$$\mathcal{R}(X) \cap \mathcal{P} = \left\{ \mathbf{w} = X\mathbf{z} : \sum_{i=1}^{m} z_i = 1, X\mathbf{z} \geq \mathbf{0} \right\}. \tag{16}$$

This is a *convex subset* of $\mathbb{R}^m$ defined by a single equality constraint and a large number, n, of inequality constraints. Formally, we rewrite the minimization problem as

$$\begin{array}{rrcll} \text{minimize:} & \multicolumn{3}{c}{\mathbf{z}^t(X^tA^tAX)\mathbf{z},} & \\ \text{subject to:} & \mathbf{1}^t\mathbf{z} & = & 1, & \text{and} \\ & X\mathbf{z} & \geq & \mathbf{0}. & \end{array} \tag{17}$$

A solution to (17) is given by any vector

$$\mathbf{x}^* = X\mathbf{z} = z_1\mathbf{x}_k + z_2\mathbf{x}_{k-1} + \ldots + z_m\mathbf{x}_{k-m+1}, \tag{18}$$

where coefficients z_i are selected to minimize $\langle AX\mathbf{z}, AX\mathbf{z}\rangle$ with the equality constraint satisfied, $\sum_{j=1}^{m} z_j = 1$, and the full set of inequality constraints satisfied, $\sum_{j=1}^{m} x_{ij} z_j \geq 0$, for any $1 \leq i \leq n$.

It is shown in [12] that for the $m = 2$ case, we are guaranteed that only two constraints are necessary, and the other $n - 2$ constraints may be ignored when solving (17). For slightly larger sets of the iterates, say $m = 3$ or $m = 4$, we assume that the constrained minimization is performed in $\mathcal{O}(n)$ operations, near the cost of a cycle of Algorithm 1 itself, which is consistent with what we have observed in the experiments. The implementation for the experiments in this paper employs the active set method from the *quadprog* function in Matlab® [16].

4 NUMERICAL RESULTS

In this section, we assess the use of aggregation-based multilevel hierarchies for some scale-free graphs. First we consider C_{op}, the *operator complexity* of the algorithm. C_{op} is defined as the sum of the number of nonzero elements in all problem matrices, A_c, on all levels, divided by the number of nonzero elements in the fine-level matrix, A. Note that all operators on each level are counted in C_{op} (i.e., for W-cycles, we count two operators on level two, four on level three, eight on level four, etc.). Since most of the work in our algorithms consist of relaxations, whose work is proportional to the number of nonzeros in the operator matrix, C_{op} 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. In the tables 'n' is the number of nodes, 'levs' is the number of levels used in the multilevel hierachy, 'iteration counts' is the number of iterations until the convergence tolerance is reached, and 'C_{op}' is the operator complexity of the last cycle.

We then show numerical performance tests for calculating the stationary probability distribution of a Markov chain with a few versions of Algorithm 1: both standalone and accelerated cycles involving neighborhood-based aggregation (smoothed and unsmoothed) and unsmoothed leaf-based aggregation. For all the numerical results presented, we start from a random, strictly positive, initial guess and iterate until the 1-norm residual, $\|A\mathbf{x}_i\|_1$, has been reduced by a factor of 10^{-8}. We do a direct solve on the coarsest level. All multilevel cycles used are $(1,1)$ cycles ($\nu_1 = \nu_2 = 1$ in Algorithm 1), and we use strength threshhold parameter $\theta = 0.25$ for all test problems on all levels, as in [8, 9]. For simplicity, the weight in our weighted Jacobi relaxation scheme is always chosen as 0.7. Accelerated methods use window size $m = 3$.

	Smoothed Aggregation		Pure Aggregation	
n	$p = 1$	$p = 2$	$p = 1$	$p = 2$
1 024	3.23	3.89	1.23	1.34
2 048	3.37	5.17	1.22	1.36
4 096	4.34	7.03	1.23	1.38
8 192	5.63	9.14	1.24	1.41
16 384	7.48	11.42	1.24	1.42
32 768	9.54	14.62	1.24	1.44

Table 1. Operator complexities for multilevel hierarchies created to coarsen graphs generated using the preferential attachment model adding either one edge per node ($p = 1$) or two edges per node ($p = 2$). For the $p = 2$ cases, each hierarchy has only 3 or 4 levels. The hierarchies for the $p = 1$ case are used in the other tables below to solve for steady state probability distributions.

4.1 Example: Barabási-Albert Model

We present a test problem associated with a synthetic, small-world graph generated using a common random graph model, a version of the *preferential attachment* model

			Iteration Counts	
n	C_{op}	levs	SAM	SAM+
1 024	3.26	4	166	28
2 048	3.53	4	224	31
4 096	4.82	4	303	38
8 192	5.63	4	430	63
16 384	7.48	4	670	79
32 768	9.54	5	862	83

Table 2. Calculating stationary probability distributions on preferential attachment graphs using stand-alone and accelerated smoothed neighborhood aggregation versions of Algorithm 1 with $\mu = 1$ (V-cycles)

			Iteration Counts	
n	C_{op}	levs	V	V+
1 024	1.23	4	355	59
2 048	1.23	4	366	58
4 096	1.22	4	696	69
8 192	1.24	5	745	82
16 384	1.24	5	> 999	127
32 768	1.23	5	> 999	142

Table 3. Calculating stationary probability distributions on preferential attachment graphs using stand-alone and accelerated pure neighborhood aggregation versions of Algorithm 1 with $\mu = 1$ (V-cycles)

proposed in [1]. Random graphs are generated by starting with a small graph with 5 nodes and successively adding new nodes with a fixed number of edges, p. These edges are randomly attached to an old node with probability that is proportional to the old node's degree. An example is shown in the leftmost panel of Figure 1.

Table 1 shows that pure neighborhood-based aggregation methods coarsen the graphs into hierarchies appearing to have bounded C_{op}, independent of problem

			Iteration Counts	
n	C_{op}	levs	W	W+
1 024	1.52	4	233	37
2 048	1.50	4	319	47
4 096	1.53	4	326	51
8 192	1.55	5	492	63
16 384	1.55	5	> 999	96
32 768	1.54	5	> 999	109

Table 4. Calculating stationary probability distributions on preferential attachment graphs using stand-alone and accelerated pure neighborhood aggregation versions of Algorithm 1 with $\mu = 2$ (W-cycles)

			Iteration Counts	
n	C_{op}	levs	V	V+
1 024	1.55	4	12	8
2 048	1.57	5	12	8
4 096	1.59	6	12	8
8 192	1.59	6	12	8
16 384	1.61	8	12	8
32 768	1.61	9	12	8

Table 5. Calculating stationary probability distributions on preferential attachment graphs using stand-alone and accelerated pure leaf-based aggregation versions of Algorithm 1 with $\mu = 1$ (V-cycles)

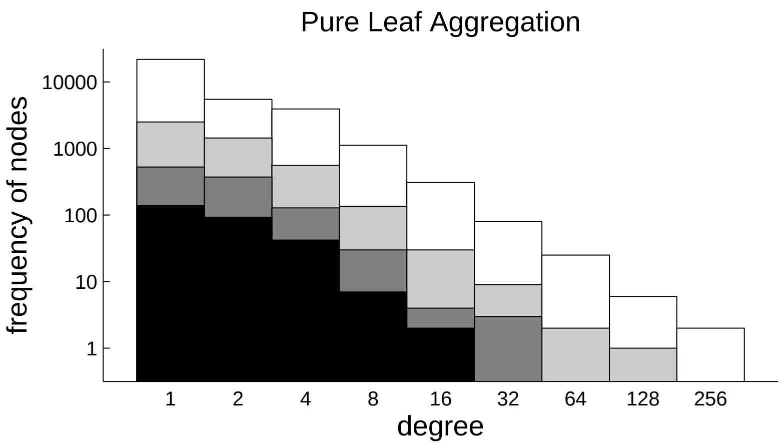


Fig. 3. Histograms of nodal degree for levels 1, 3, 5, and 7 of a pure, leaf-based aggregation multigrid hierarchy for the same graph as in Figure 2. The histogram for the original graph is shown in the white and increasingly darker shades of grey represent increasingly coarser levels. The leaf-based aggregation routine maintains the power law degree distribution on all levels.

size, whereas the smoothed neighborhood-based aggregation methods demonstrate complexities that grow dramatically. Thus the complexity of multilevel hierarchies is acceptable for pure aggregation for the graphs generated by adding either one ($p = 1$) or two ($p = 2$) edges per new node (see Table 1). Additionally, the power law degree distributions are better maintained on coarser levels for pure aggregation (see Figure 2). If $p = 1$, the stochastic matrices associated with the graphs tend to be slowly mixing: there are subdominant eigenvalues with $\mathrm{Re}\lambda \approx 1$ for the matrices we generated. If similar graphs are generated with more than one edge per new node ($p > 1$), our observations suggest that the associated stochastic matrices have high probability to *not* be slowly mixing ($|\lambda_2| < 0.9$ for all of a small number of randomly sampled graphs of various size), and multilevel methods are not required for calculating the stationary vector. Therefore, we only apply the multilevel eigensolver techniques to the graphs generated by adding one edge per node.

We form stochastic matrices associated with random walks on these random graphs. Note that the steady-state solution for these test problems can be easily calculated using the relative size of the degree of each node, because they are unweighted random walks on undirected graphs. Still, this scale-free example is a good test problem for our methods, which can also handle directed graphs without such simple solutions.

The results of applying the smoothed neighborhood aggregation version of Algorithm 1 to this example problem are reported in Table 2. The iteration counts for the unaccelerated method (labeled SAM) increase rapidly with the problem size. The iteration counts for the accelerated method (labeled SAM+) are significantly reduced, but are not bounded independent of the problem size. Additionally, the operator complexity is growing rapidly for large problems.

Better results are obtained by applying the accelerator to the pure neighborhood aggregation version of Algorithm 1. Table 3 contains results for V-cycles and Table 4 contains results for W-cycles. The iteration counts for the unaccelerated method (labeled V and W) increase rapidly with the problem size. The iteration counts for the accelerated method (labeled V+ and W+) are again significantly reduced, but are still not bounded independent of the problem size. The operator complexities, however, do not increase for larger problem sizes as noted earlier. W-cycles have slightly better performance than V-cycles.

The leaf-based aggregation provides scalable ranking calculations, as displayed in Table 5. Independent of the problem size, the multilevel hierarchies have bounded operator complexity and the iteration counts are bounded as well. Also, for a specific problem, Figure 3 shows that the power-law structure of the original graph is maintained on coarse levels of the multilevel hierarchy.

5 CONCLUSIONS AND FUTURE WORK

We have shown examples of small-world graphs where pure neighborhood aggregation achieved multilevel hierarchies of optimal complexity and retained power-law distribution on coarse-grids, whereas smoothed neighborhood aggregation did neither. Additionally, these multilevel hierarchies were used to calculate stationary probability vectors for Markov chains, where a Krylov-like acceleration technique was developed and employed to significantly reduce iteration counts.

Using an aggregation routine that takes advantage of tree-like structure within graphs allows pure aggregation solvers that are scalable for a class of test problems. Next, we intend to generalize this algorithm to be robust and scalable for a wider class of complicated network graphs. Additionally, we are developing related algorithms that compute multiple eigenvectors that are to be used for several types of network calculations relevant to information retrieval.

REFERENCES

[1] Barabási, A. L.—Albert, R.: Emergence of Scaling in Random Networks. Science 286, 509, 1999.

[2] Berman, A.—Plemmons, R. J.: Nonnegative Matrices in the Mathematical Sciences. SIAM, Philadelphia 1987.

[3] Brandt, A.: Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics. Gesselschaft für Mathematik und Datenverarbeitung, St. Augustin, GMD-Studien Nr. 85, 1984.

[4] Brezina, M.—Falgout, R.—MacLachlan, S.—Manteuffel, T.—McCormick, S.—Ruge, J.: Adaptive Smoothed Aggregation (aSA) Multigrid. SIAM Review Vol. 47, pp. 317–346, 2005.

[5] Brezina, M.—Falgout, R.—MacLachlan, S.—Manteuffel, T.—McCormick, S.—Ruge, J.: Adaptive Algebraic Multigrid. SIAM J. Sci. Comp., Vol. 27, pp. 1261–1286, 2006.

[6] Briggs, W. L.—Henson, V. E.—McCormick, S. F.: A Multigrid Tutorial. SIAM, Philadelphia 2000.

[7] Dayar, T.—Stewart, W. J.: Comparison of Partitioning Techniques for Two-Level Iterative Solvers on Large, Sparse Markov Chains. SIAM J. Sci. Comp., Vol. 21, pp. 1691, 2000.

[8] de Sterck, H.—Manteuffel, T. A.—McCormick, S. F.—Miller, K.—Pearson, J.—Ruge, J.—Sanders, G.: Smoothed Aggregation Multigrid for Markov Chains. SIAM J. Sci. Comp. Vol. 32, pp. 40–61, 2010.

[9] de Sterck, H.—Manteuffel, T. A.—McCormick, S. F.—Miller, K.—Pearson, J.—Ruge, J.—Sanders, G.: Algebraic Multigrid for Markov Chains. SIAM J. Sci. Comp., Vol. 32, pp. 544–562, 2010.

[10] de Sterck, H.—Manteuffel, T. A.—McCormick, S. F.—Nguyen, Q.—Ruge, J.: Multilevel Adaptive Aggregation for Markov Chains, with Application to Web Ranking. SIAM J. Sci. Comp., Vol. 30, pp. 2235–2262, 2008.

[11] de Sterck, H.—Manteuffel, T. A.—Miller, K.—Sanders, G.: Top-Level Acceleration of Adaptive Algebraic Multilevel Methods for Steady-State Solution to Markov Chains. Submitted to Advances in Computational Mathematics, 2009.

[12] de Sterck, H.—Miller, K.—Sanders, G.—Winlaw, M.: Recursively Accelerated Multilevel Aggregation for Markov Chains. Submitted to SIAM J. Sci. Comput., September 2009.

[13] Fiedler, M.: Algebraic Connectivity of Graphs. Czechoslovak Mathematical Journal, Vol. 23, pp. 298–305, 1973.

[14] Fiedler, M.: A Property of Eigenvectors of Non-Negative Symmetric Matrices and Its Application to Graph Theory. Czechoslovak Mathematical Journal, Vol. 25, pp. 619–632, 1975.

[15] Fouss, F.—Pirotte, A.—Renders, J. M.—Saerens, M.: Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation. IEEE Transactions on Knowledge and Data Engineering, Vol. 19, 2007, No. 3.

[16] Gill, P. E.—Murray, W.—Wright, M. H.: Practical Optimization. Academic Press, London, UK 1981.

[17] Horton, G.—Leutenegger, S. T.: A Multi-Level Solution Algorithm for Steady-State Markov Chains. ACM SIGMETRICS, pp. 191–200, 1994.

[18] Inglis, T.—De Sterck, H.—Sanders, G.—Djambazian, H.—Sladek, R.—Sundararajan, S.—Hudson, T. J.: Multilevel Space-Time Aggregation for Bright Field Cell Microscopy Segmentation and Tracking. International Journal of Biomedical Imaging, accepted, 2010.

[19] Koury, J. R.—McAllister, D. F.—Stewart, W. J.: Iterative Methods for Computing Stationary Distributions of Nearly Completely Decomposable Markov Chains. SIAM Journal of Algebraic and Discrete Methods, Vol. 5, pp. 164–186, 1984.

[20] Krieger, U. R.: On a Two-Level Multigrid Solution Method for Finite Markov Chains. Linear Algebra and its Applications 223/224, pp. 415–438, 1995.

[21] Krieger, U. R.—Müller-Clostermann, B. M.—Sczittnick, M.: Modeling and Analysis of Communication Systems Based on Computational Methods For Markov Chains. IEEE Journal on Selected Areas in Communication, No. 8-9, pp. 1630–1648, 1990.

[22] Leutenegger, S. T.—Horton, G.: On the Utility of the Multi-Level Algorithm for the Solution of Nearly Completely Decomposable Markov Chains. In: W. Stewart, Ed.: Numerical solution of Markov chains, Kluwer Publishers, pp. 425–443, 1995.

[23] Marek, I.—Mayer, P.: Convergence Analysis of an Iterative Aggregation/Disaggregation Method for Computing Stationary Probability Vectors of Stochastic Matrices. Numerical Linear Algebra with Applications, Vol. 5, pp. 253–274, 1998.

[24] Marek, I.—Mayer, P.: Convergence Theory of Some Classes of Iterative Aggregation/Disaggregation Methods for Computing Stationary Probability Vectors of Stochastic Matrices. Linear Algebra and its Applications, Vol. 363, pp. 177–200, 2003.

[25] Molloy, K. M.: Performance Analysis Using Stochastic Petri Nets. IEEE Transactions on Computers, Vol. C-31, pp. 913–917, 1982.

[26] Philippe, B.—Saad, Y.—Stewart, W. J.: Numerical Methods in Markov Chain Modeling. Operations Research, Vol. 40, pp. 1156–1179, 1992.

[27] Ruge, J.—Stueben, K.: Algebraic Multigrid. In: S. F. McCormick (Ed.): Multigrid Methods, Frontiers in Applied Mathematics, Vol. 3, SIAM, Philadelphia, PA, pp. 73–130, 1987.

[28] Schweitzer, P. J.—Kindle, K. W.: An Iterative Aggregation-Disaggregation Algorithm for Solving Linear Equations. Appl. Math. Comp., Vol. 18, pp. 313–354, 1986.

[29] Simon, H. A.—Ando, A.: Aggregation of Variables in Dynamic Systems.

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

[31] Takahashi, Y.: A Lumping Method for Numerical Calculations of Stationary Distributions of Markov Chains. Research Report B-18, Department of Information Sciences, Tokyo Institute of Technology, 1975.

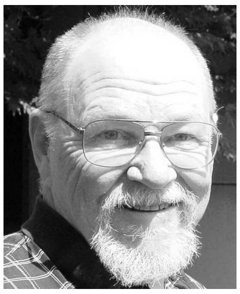
[32] Treister, E.—Yavneh, I.: Square & Stretch Multigrid for Stochastic Matrix Eigenproblems. Preprint, 2009.

[33] Tsourakakis, C. E.: Fast Counting of Triangles in Large Real Networks, without Counting: Algorithms and Laws. In: IEEE International Conference on Data Mining (ICDM 2008).

[34] Vaněk, P.—Mandel, J.—Brezina, M.: Algebraic Multigrid on Unstructured Meshes. Technical Report UCD-CCM-034, University of Colorado at Denver, Mathematics Department, 1994.

[35] Washio, T.—Oosterlee, C. W.: Krylov Subspace Acceleration for Nonlinear Multigrid Schemes. Electronic Transactions on Numerical Analysis, Vol. 6, pp. 271–290, 1997.

Hans de Sterck is Associate Professor in the Department of Applied Mathematics of the University of Waterloo. He obtained his Ph. D. at the Catholic University Leuven, followed by postdocs at the von Karman Institute and the University of Colorado at Boulder. His area of research is scientific computing methods and applications. He is an associate editor of the SIAM Journal on Scientific Computing.



Van Emden Henson is a computational mathematician with the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory, a position he has held since 1997. After earning his Ph. D. in Applied Mathematics in 1990 from the University of Colorado at Denver, he served on the mathematics faculty of the United States Naval Postgraduate School in Monterey, California until joining the staff at LLNL. His research areas have emphasized algebraic multigrid, high-performance computing, and Fourier analysis; his current research is in large-scale linear algebra for problems in data mining, cybersecurity, and intelligence.

Geoffrey Sanders is currently working as a Post Doctoral Researcher at the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. He received his Ph. D. in applied mathematics from the University of Colorado, Boulder in 2008 and continued there with a research and teaching position until 2010. His research has been in scientific computing with a focus on extending multilevel linear algebra techniques to a broader scope of applications.