LEAN ALGEBRAIC MULTIGRID (LAMG): FAST GRAPH LAPLACIAN LINEAR SOLVER

OREN E. LIVNE [*] AND ACHI BRANDT [†]

Dedicated to J. Brahms' Symphony No. 1 in C minor, Op. 68

Abstract. Laplacian matrices of graphs arise in large-scale computational applications such as semi-supervised machine learning; spectral clustering of images, genetic data and web pages; transportation network flows; electrical resistor circuits; and elliptic partial differential equations discretized on unstructured grids with finite elements. A Lean Algebraic Multigrid (LAMG) solver of the symmetric linear system $Ax = b$ is presented, where A is a graph Laplacian. LAMG's run time and storage are empirically demonstrated to scale linearly with the number of edges.

LAMG consists of a setup phase during which a sequence of increasingly-coarser Laplacian systems is constructed, and an iterative solve phase using multigrid cycles. General graphs pose algorithmic challenges not encountered in traditional multigrid applications. LAMG combines a lean piecewise-constant interpolation, judicious node aggregation based on a new node proximity measure (the affinity), and an energy correction of coarse-level systems. This results in fast convergence and substantial setup and memory savings. A serial LAMG implementation scaled linearly for a diverse set of 3774 real-world graphs with up to 47 million edges, with no parameter tuning. LAMG was more robust than the UMFPACK direct solver and Combinatorial Multigrid (CMG), although CMG was faster than LAMG on average. Our methodology is extensible to eigenproblems and other graph computations.

Key words. Linear-scaling numerical linear solvers, graph Laplacian, aggregation-based algebraic multigrid, piecewise-constant interpolation operator, high-performance computing.

AMS subject classifications. 65M55, 65F10, 65F50, 05C50, 68R10, 90C06, 90C35.

1. Introduction. Let $G = (\mathcal{N}, \mathcal{E}, w)$ be a connected weighted undirected graph, where $\mathcal{N}$ is a set of n nodes, $\mathcal{E}$ is a set of m edges, and $w : \mathcal{E} \to \mathbb{R}^+$ is a weight function. The Laplacian matrix $\mathbf{A}_{n \times n}$ is naturally defined by the *quadratic energy*

$$E(\mathbf{x}) := \mathbf{x}^T \mathbf{A} \mathbf{x} = \sum_{(u,v) \in \mathcal{E}} w_{uv} (x_u - x_v)^2 , \qquad \mathbf{x} \in \mathbb{R}^{\mathcal{N}} , \qquad (1.1)$$

where $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$. In matrix form,

$$\mathbf{A} = (a_{uv})_{u,v} , \quad a_{uv} := \begin{cases} \sum_{v' \in \mathcal{E}_u} w_{uv'} , & u = v , \\ -w_{uv} , & v \in \mathcal{E}_u := \{v' : (u,v') \in \mathcal{E}\} , \\ 0 , & \text{otherwise.} \end{cases} \qquad (1.2)$$

$\mathbf{A}$ is Symmetric Positive Semi-definite (SPS), and has zero row sums and $2m + n$ non-zeros. Typically, $m \ll n^2$ and $\mathbf{A}$ is sparse. Our approach also handles some SPS Laplacian matrices corresponding to *negative edge weights*, such as high-order and anisotropic grid discretizations [12, 6]; those are discussed in §5.2.

Since G is connected, $\mathbf{A}$'s null space is spanned by the vector of ones $\mathbf{1}$ (a disconnected graph can be decomposed into its components in $O(m)$ time [58, 63]). We consider the nonsingular compatible linear system [65, pp. 185–186]

$$\mathbf{A}\mathbf{x} = \mathbf{b} \qquad (1.3a)$$

$$\mathbf{1}^T \mathbf{x} = 0 , \qquad (1.3b)$$

---
[*]The University of Chicago, Department of Human Genetics, 920 E. 58th St. CLSC 431F, Chicago, IL 60637. Tel: +1-773-702-5898. Email: `livne@uchicago.edu`

[†]The Weizmann Institute of Science, Department of Mathematics and Computer Science, POB 26 Rehovot 76100, Israel. Tel. +972-8-934-3545. Email: `abrandt@math.ucla.edu`
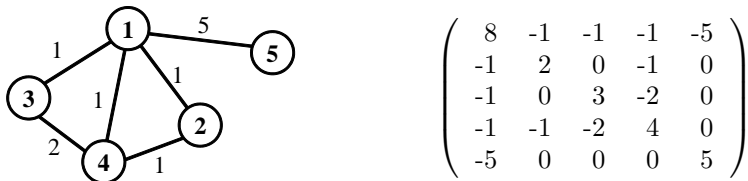
FIG. 1.1. *A 5-node graph and its corresponding Laplacian matrix.*

where $\mathbf{b} \in \mathbb{R}^{\mathcal{N}}$ is a given zero-sum vector, and $\mathbf{x} \in \mathbb{R}^{\mathcal{N}}$ is the vector of unknowns.

Our goal is to develop an iterative numerical solver of (1.3) that *requires $O(m)$ storage and $O(m \log(1/\varepsilon))$ operations to generate an ε-accurate solution, for graphs arising in real-world applications.* The hidden constants should be small (in the hundreds, not millions). The solver should require a smaller cost to re-solve the system for multiple $\mathbf{b}$'s – a useful feature for time-dependent and other applications.

Importantly, we are interested in *good empirical performance* (bounded hidden constants over a diverse set of test instances), and do not consider the problem of designing an algorithm with provably linear complexity in any graph [61, Problem 5, p. 18]. While proofs are important, they often provide unrealistic bounds or no bounds at all on the hidden constants that can be attained in practice.

1.1. Applications. The linear system (1.3) is fundamental to many applications; see Spielman's review [61, §2] for more details:
- Elliptic Partial Differential Equations (PDEs) discretized on unstructured grids by finite elements within a fluid dynamics simulation [7, 31].
- Interior-point methods for network flow linear programming [23, 32].
- Electrical flow through a resistor network G.

Additionally, $\mathbf{A}\mathbf{x} = \mathbf{b}$ is *a stepping stone toward the eigenproblem.* Our multilevel methodology can be extended to compute the smallest eigenpairs of $\mathbf{A}$ with minor adaptations (cf. §6.4). The Laplacian eigenproblem is central to graph regression and classification in machine learning [20, 67], spectral clustering of images, graph embedding [59], and dimension reduction for genetic ancestry discovery [45]. Of particular interest is the Fiedler value – the smallest non-zero eigenvalue of $\mathbf{A}$, which measures the algebraic connectivity of G [21, §1.1] and is related to minimum cuts [27]. Although we believe it is preferable to develop multiscale strategies for the original formulations of these problems, as demonstrated by the works [36, 43, 55, 60] and graph partitioning packages [37], a fast black-box eigensolver is a practical alternative.

1.2. Related Work. There are two main approaches to solving (1.3): direct, leading to an exact solution (up to round-off errors); and iterative, which typically requires a one-time setup cost, followed by a solve phase that produces successive approximations $\tilde{\mathbf{x}}$ to $\mathbf{x}$ to achieve ε-accuracy, namely,

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|_{\mathbf{A}} \le \varepsilon \|\mathbf{x}\|_{\mathbf{A}}, \qquad \|\mathbf{x}\|_{\mathbf{A}} := \sqrt{E(\mathbf{x})}. \tag{1.4}$$

1.2.1. Direct Methods. The Cholesky factorization with a clever elimination order can be applied to $\mathbf{A}$. A permutation matrix $\mathbf{P}$ is chosen and factorization $\mathbf{P}^T \mathbf{A} \mathbf{P} = \mathbf{L}\mathbf{L}^T$ constructed so that the lower triangular $\mathbf{L}$ is as sparse as possible, using Minimum or Approximate Minimum Degree Ordering [1, 64]. Except for simple graphs, direct algorithms do not scale, requiring $O(n^{1.5})$ operations for planar graphs [33, 46] and $O(n^3)$ in general. Alternatively, fast matrix inversion can be performed in $O(n^{2.376})$ or combined with Cholesky, yet yields similar complexities [61, §3.1].

1.2.2. Iterative Methods: Graph Theoretic. These are variants of the preconditioned conjugate method [35, §10.3] that achieve (1.4) in $O(\sqrt{\kappa(\mathbf{AB}^{-1})}\log(1/\varepsilon))$ iterations for a preconditioner $\mathbf{B}$; κ is the finite condition number [61, §3.3].

Spielman and Teng (S-T) [62] and subsequent works [41] have been focusing on multilevel graph-sparsifying preconditioners. The S-T setup builds increasingly smaller graphs, alternating between partial Cholesky and ultra-sparsification steps, in which the graph is partitioned into sets of high-conductance nodes without removing too many edges. The complexity is a near-linear $O(m\log^2 n\log(1/\varepsilon))$, guaranteed for any symmetric diagonally-dominant $\mathbf{A}$. Unfortunately, no implementation is available yet, nor is there a guarantee on the size of the hidden constant.

1.2.3. Iterative Methods: AMG. Algebraic Multigrid (AMG) is a class of high-performance linear solvers, originating in the early 1980s [14, §1.1], [15], [56] and under active development. During setup, AMG recursively constructs a multi-level hierarchy of increasingly coarser graphs by examining matrix entries, without relying on geometric information. The solve phase consists of multigrid cycles. AMG can be employed either as a solver or a preconditioner [65, App. A]. Open-source parallel implementations include Hypre [30] and Trilinos-PETSc [38]. In classical AMG, the coarse set is a subset of $\mathcal{N}$; alternatively aggregation AMG [65, App. A.9],[8, 4, 52] and smoothed aggregation [19] define coarse nodes as aggregates of fine nodes.

AMG mainly targets discretized PDEs, where it has been successful [31]. Advanced techniques have ventured to widen its scope by increasing the interpolation accuracy. These include Bootstrap AMG [11, §17.2], [13] adaptive smoothed aggregation [17], and interpolation energy minimization [54]. While these methods approach linear scaling for more systems, their complexity cannot be controlled in general graphs (cf. §3.1.2).

At the same time, accelerated aggregation AMG has become a hot research topic. A crude *caliber-1* (piecewise-constant) interpolation is employed between levels to reduce runtime and memory costs, at the expense of a slower cycle that is subsequently accelerated. Notay [52, 53] aggregated nodes based on matrix entries and applied multilevel CG acceleration with a large cycle index to obtain a near-optimal solver for convection-diffusion M-matrices, but the method was limited to those grid graphs. Caliber-1 interpolation was tested for a single graph by Bolten et. al within the bootstrap framework [5], but their setup cost was large and required parameter tuning.

The present work aims at generalizing AMG to graph Laplacians and addresses peculiarities not encountered in traditional AMG applications. To the best of our knowledge, the only other solver targeting general topologies is Combinatorial Multigrid (CMG) [42], a hybrid graph-theoretic-AMG preconditioner that partitions nodes into high-conductance aggregates, similarly to S-T. CMG outperformed classical AMG for a set of 3-D image segmentation applications. In our experiments over a much larger graph collection, CMG and LAMG had comparable average solve speeds, yet LAMG's performance was much more robust, with almost no outliers.

1.3. Our Contribution. We present Lean Algebraic Multigrid (LAMG): a practical graph Laplacian linear solver. A MATLAB LAMG implementation scaled linearly for a set of 3774 real-world graphs with up to 47 million edges, ranging from computational fluid dynamics to web, biological and social networks. Specifically, the setup phase required on average 200 Matrix-Vector Multiplications (MVMs) and $4m$ storage bytes, and the average solve time was $\approx 27\log(1/\varepsilon)$ MVMs per right-hand side. The standard deviations were small with only three outliers. LAMG was more

robust than the UMFPACK direct solver and CMG, although CMG was faster on average (§5.1). Our methodology is extensible beyond the scope of S-T and CMG, to non-diagonally-dominant (§5.2), eigenvalue, and nonlinear problems (§6.4).

LAMG is an accelerated caliber-1 aggregation-AMG algorithm that builds upon the state-of-the-art AMG variants, yet introduces four new ideas essential to attaining optimal efficiency in general graphs:

(a) *Lean methodology* (§3.1). LAMG advocates using *minimalistic over sophisticated AMG components*. No parameter tuning should be required. In particular, we apply caliber-1 interpolation between levels, constructed using relaxed Test Vectors (TVs), but without bootstrapping them as in the papers [5, 13]. Fast asymptotic convergence is achieved by the following three ideas.

(b) *Low-degree Elimination* (§3.2). Like the S-T method [62], we eliminate low-degree nodes prior to each aggregation. However, the role of elimination here is different: *it removes the effectively-1-D part of the graph*, thereby eliminating extreme tradeoffs between complexity and accuracy in aggregating the remaining graph. Thus, unlike S-T, our elimination need not strictly reduce the number of edges (allowing us to eliminate nodes of higher degrees) nor be exact (making it also useful in the eigenproblem). The elimination and aggregation are in fact specializations of the same coarsening scheme (§6.4).

(c) *Affinity* (§3.3). Aggregation is based on a new *normalized relaxation-based node proximity* heuristic. Ron et al. [55] were the first to use TVs to measure algebraic distance between nodes, but our measure is effective for a wider variety of graph structures. The affinity admits a statistical interpretation and approximates the diffusion distance [22]. In contrast, the S-T algorithm strives to create high-conductance aggregates [62, 40].

(d) *Energy-corrected Aggregation* (§3.4). Recognizing that caliber-1 interpolation leads to an energy inflation of the coarse-level system (§3.4.3), our aggregation also *reuses test vectors to minimize coarse-to-fine energy ratios*. We offer two alternative energy corrections to accelerate the solution cycle: a flat correction to the Galerkin operator resemblant of Braess' work [8], yet resulting in a superior efficiency; and an adaptive correction to the solution via *multilevel iterate recombination* [65, §7.8.2] that is even more efficient.

Following an AMG prelude in §2, we discuss each of the main ideas in §3. They are integrated into the complete LAMG algorithm in §4 (cf. the expanded ArXiV e-print [49] for implementation details). Our development methodology emphasizes learning from examples: we studied instances for which the original design was slow to derive general aggregation rules. Testing over a large collection ensured that new rules did not spoil previous successes. The results are presented in §5. Coarsening improvements and extensions to the eigenproblem and other graph computational problems are outlined in §6.

2. Algebraic Multigrid Basics. Relaxation methods for $\mathbf{Ax} = \mathbf{b}$ such as Jacobi or Gauss-Seidel slow down asymptotically. Yet after only several sweeps, the *error* $\mathbf{e} := \mathbf{x} - \tilde{\mathbf{x}}$ in the approximation $\tilde{\mathbf{x}}$ to $\mathbf{x}$ becomes *algebraically smooth*: its normalized residuals are much smaller than its magnitude [14, §1.1] (assuming its mean has been subtracted out). In AMG, these errors are approximated by an interpolation $\mathbf{P}_{n \times n_c}$ from a coarse subspace: $\mathbf{e} \approx \mathbf{Pe}^c$, where $\mathbf{e}^c$ is a coarse vector of size n_c.

Recognizing that $\mathbf{Ax} = \mathbf{b}$ corresponds to the quadratic minimization

$$\mathbf{x} = \min_{\mathbf{y}} E_{\text{tot}}(\mathbf{y}), \quad E_{\text{tot}}(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T \mathbf{Ax} - \mathbf{x}^T \mathbf{b}, \qquad (2.1)$$

the variational correction scheme [15] seeks the optimal correction in the energy norm,

$$\mathbf{e}^c = \underset{\mathbf{y}^c}{\operatorname{argmin}} \, E_{\text{tot}} \left(\tilde{\mathbf{x}} + \mathbf{P}\mathbf{y}^c \right) . \tag{2.2}$$

The resulting two-level cycle (except for determining $\mathbf{x}$'s mean) is
   1. Perform 2-3 relaxation sweeps on $\mathbf{A}\mathbf{x} = \mathbf{b}$, resulting in $\tilde{\mathbf{x}}$.
   2. Compute an approximation $\tilde{\mathbf{e}}^c$ to the solution $\mathbf{e}^c$ of

$$\mathbf{A}^c \mathbf{e}^c = \mathbf{b}^c, \quad \mathbf{A}^c := \mathbf{P}^T \mathbf{A} \mathbf{P}, \quad \mathbf{b}^c := \mathbf{P}^T \left(\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}} \right) . \tag{2.3}$$

   3. Correct the fine-level solution:

$$\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} + \mathbf{P}\tilde{\mathbf{e}}^c . \tag{2.4}$$

Eq. (2.3), called Galerkin coarsening, is a smaller linear system. In the multilevel cycle, it is recursively solved using γ two-level cycles, where the *cycle index* γ is an input parameter. Our interpolation $\mathbf{P}$ is full-rank with unit row sums (see §3.1.4) for which it easy to verify that $\mathbf{A}^c$ is also a connected graph Laplacian. Overall, LAMG constructs a hierarchy of L increasingly coarser Laplacian systems ("levels") $\mathbf{A}^l \mathbf{x}^l = \mathbf{b}^l$, $l = 1, \dots, L$, the finest being the original system $\mathbf{A}^1 := \mathbf{A}, \mathbf{b}^1 := \mathbf{b}$. The setup phase depends on $\mathbf{A}$ only, and produces $\{(\mathbf{A}^l, \mathbf{P}^l)\}_{l=2}^L$, where $\mathbf{A}^l$ is $n_l \times n_l$ and $\mathbf{P}^l$ is the $n_{l-1} \times n_l$ interpolation matrix from level l to $l-1$. We denote by $G^l = (\mathcal{N}^l, \mathcal{E}^l)$ the graph corresponding to $\mathbf{A}^l$.

3. LAMG: Main Ideas. Our description refers to a single coarsening stage $(\mathbf{A} \to \mathbf{A}^c)$ and applies to each pair of levels.

3.1. Lean Methodology.

3.1.1. Relaxation. Our choice is Gauss-Seidel (GS) relaxation, defined by the successive updates [14, §1.1]

$$\text{For } u = 1, \dots, n, \qquad x_u \leftarrow \frac{b_u - \sum_{v \in \mathcal{E}_u} a_{uv} x_v}{a_{uu}} . \tag{3.1}$$

We picked GS because it is an effective smoother in SPS systems [14, §1] and does not require parameter tuning (such as the Jacobi relaxation damping parameter [5]).

3.1.2. Interpolation Caliber. Textbook multigrid convergence for the Poisson equation requires that the interpolation of corrections $\mathbf{P}$ be second-order [9, §3.3]. The analogous AMG theory implies a similar condition on the interpolation accuracy of low-energy errors. While a piecewise-constant $\mathbf{P}$ is acceptable in a two-level cycle, it is insufficient for V-cycles ($\gamma = 1$); W-cycles ($\gamma = 2$) are faster but costly [65, p. 471],[52].

Constructing a second-order $\mathbf{P}$ is already challenging in grid graphs [13, 19, 54]. We argue that it is infeasible in general graphs:
   (a) The graph's effective dimension d is unknown; had it been known, the required interpolation *caliber*, i.e., the number of coarse nodes used to interpolate a fine node, would grow with d and result in unbounded interpolation complexity.
   (b) Identifying a proper interpolation set (whose "convex hull" contains the fine node) is a complex and costly process [13].
   (c) The Galerkin coarse operator $\mathbf{P}^T \mathbf{A} \mathbf{P}$ fills in considerably; cf. §3.1.3.

In contrast, LAMG employs a lean caliber-1 (piecewise-constant) $\mathbf{P}$, equivalent to an *aggregation* of the nodes into coarse-level aggregates, and corrects the energy of the coarse-level Galerkin *operator* to maintain good convergence (in practice, the correction is applied to the coarse right-hand side). This could not have been achieved within the variational setting of §2, which only permits modifying $\mathbf{P}$. Here the barrier to fast convergence is the coarse-to-fine operator energy ratio. Our contribution is an algorithm that yields a small energy ratio, which translates into optimal efficiency; cf. §3.4. The compatible relaxation performance predictor [10, 47, 16], [14, §§14.2–14.3] is irrelevant for low interpolation accuracy; the energy ratio is a better predictor.

3.1.3. Managing Fill-in. Frequently, the coarse-level matrices in AMG hierarchies become increasingly dense. This is a result of a poor aggregation, a high-caliber $\mathbf{P}$, or both: many fine nodes whose neighbor sets are disjoint are aggregated, creating additional edges among coarse-level aggregates. This renders the ideally-accurate interpolation irrelevant, because the actual cycle *efficiency* (error reduction per unit work) is small albeit convergence may be rapid. While fill-in is often controllable in grid graphs because their coarsening is still local, it is detrimental in non-local graphs.

LAMG's interpolation is designed to minimize fill-in. Heuristically, the sparser $\mathbf{P}$, the sparser $\mathbf{P}^T \mathbf{A} \mathbf{P}$. We further indirectly control fill-in via our affinity criterion (§3.3), which tends to aggregate nodes that share many neighbors. The cycle work is also restrained by a fractional cycle index [14, §6.2] between 1 and 2; cf. §§4.1,4.2.

Occasionally, the interpolation caliber may be slightly increased as long as the number of coarse edges does not become too large; see §6.1.

3.1.4. Utilizing Extenuating Circumstances. Specific properties of graph Laplacians are exploited to simplify the LAMG construction.

- Since $\mathbf{A}$ has zero row sums, its null-space consists of constant vectors. A fundamental AMG assumption is that *all near*-null-space errors can be fitted by a *single* interpolation from a coarse level [14, p. 8]. Here it implies that the interpolation weights can be apriori set to 1. The unit-weight assumption is easily verified for Laplacians with bounded node degrees [65, p. 439]; in the most interesting applications of this work, however, the node degree is unbounded, and a proof of this conjecture is an open problem.
- Some graph locales are effectively one-dimensional: many nodes have degree 1–2. Such nodes can be quickly eliminated similarly to the paper [41] (§3.2).
- In other graphs, Gauss-Seidel is an efficient solver and no coarsening is required. These include complete graphs, star graphs, expander graphs [61, §1], and certain classes of random graphs. More generally, if GS converges fast for a subset of the nodes, they can all be aggregated together (§3.4.2).

3.2. Low-degree Elimination. We first attempt to eliminate from $\mathcal{N}$ an independent set $\mathcal{F}$ of nodes u of degree $|\mathcal{E}_u| \leq 4$. The set is identified by initially marking all nodes as "eligible"; we then sweep through nodes, adding each eligible low-degree node to $\mathcal{F}$ and marking its neighbors as ineligible [49, Algorithm 1].

Eliminating a node connects all its neighbors; therefore, $\mathcal{F}$-nodes of degree ≤ 3 do not increase m. When $|\mathcal{E}_u| = 4$, m might be increased by at most 2. However, we assume that this is unlikely to happen for many $\mathcal{F}$-nodes and eliminate these nodes as well (in practice, we have observed that the neighbors are already connected prior to elimination). Eliminating larger degrees results in an impractical fill-in.

Let $\mathcal{C} := \mathcal{N}\backslash\mathcal{F}$. $\mathbf{A}\mathbf{x} = \mathbf{b}$ reduces to the Schur complement system

$$\mathbf{A}^c\mathbf{x}_\mathcal{C} = \mathbf{b}^c, \quad \mathbf{A}^c := \mathbf{P}^T\mathbf{A}\mathbf{P}, \quad \mathbf{b}^c := \mathbf{P}^T\mathbf{b}, \quad \mathbf{P} := \mathbf{\Pi}\left(-\mathbf{A}_{\mathcal{F}\mathcal{C}}^T\mathbf{A}_{\mathcal{F}\mathcal{F}}^{-1}, \mathbf{I}_\mathcal{C}\right)^T, \quad (3.2)$$

where $\mathbf{\Pi}$ is a permutation matrix such that $\mathbf{\Pi}^T\mathbf{x}$ lists the all $\mathcal{F}$ node values, then all $\mathcal{C}$ node values. (3.2) is a smaller Laplacian system for which we perform further elimination rounds, until $|\mathcal{F}|$ becomes small [49, Algorithm 2].

The purpose of elimination is to reduce n while incurring a small fill-in, and to remove the 1-D part of the graph, which cannot be effectively coarsened by the energy-corrected aggregation of §3.4. Note that $\mathbf{P}$'s caliber is larger than 1 here.

Viewed as a full approximation scheme (§6.4), (3.2) could be generalized to a *non-exact* elimination for approximating the lowest eigenvectors of $\mathbf{A}$, instead of forming a nonlinear Schur complement [3]. Additionally, a larger set of *loosely-coupled* nodes could also be eliminated: if $\mathbf{A}_{\mathcal{F}\mathcal{F}}$ is strongly diagonally-dominant, its inverse can be approximated by a few Jacobi relaxations. The two-level convergence rate and fill-in would need to be kept in check in that case. We plan to pursue these generalizations in a future research.

We hereafter denote the coarse system by $\mathbf{A}\mathbf{x} = \mathbf{b}$ (either (1.3a), if $q = 0$, or (3.2)), which is further coarsened by caliber-1 aggregation in §§3.3–3.4.

3.3. Affinity. The construction of an effective aggregate set hinges upon defining which nodes in $\mathcal{N}$ are "proximal", i.e., nodes whose values are strongly coupled in all smooth (i.e., low-energy) vectors [65, p. 473]. Table 3.1 lists three definitions.

Classical AMG [56]	$1 - |w_{uv}|/\max\{\max_s |w_{us}|, \max_s |w_{sv}|\}$
Algebraic Distance [55]	$\max_{k=1}^K \left| x_u^{(k)} - x_v^{(k)} \right|$
Affinity (LAMG)	$c_{uv} := 1 - |(X_u, X_v)|^2 / ((X_u, X_u)(X_v, X_v))$, $(X_u, X_v) := \sum_{k=1}^K x_u^{(k)} x_v^{(k)}$

TABLE 3.1

Comparison of node proximity measures. Nodes are defined as "close" when the measure is smaller than a threshold. $\{\mathbf{x}^{(k)}\}_{k=1}^K$ is a set of relaxed test vectors; see the text.
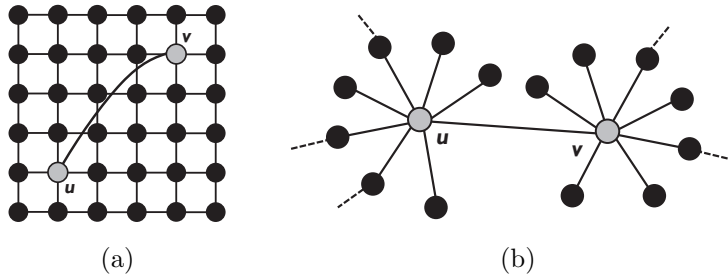

(a) (b)

FIG. 3.1. *Unweighted graph instances that present aggregation difficulties. (a) A 2-D grid with an extra link. (b) Two connected hubs. A hub is a high-degree node.*

3.3.1. Existing Proximity Measures. Classical AMG defines proximity based on edge weights (Table 3.1, top row). While this has worked well for coarsening discretized scalar elliptic PDEs, it leads to wrong aggregation decisions in non-local

graphs. In a grid graph with an extra link between distant nodes u, v (Fig. 3.1a), u and v become proximal and may be aggregated. Unless w_{uv} is outstandingly large, this is undesirable because u and v belong to unrelated milieus of the grid.

This problem is overcome by the algebraic distance measure introduced by Ron et al. [55] (Table 3.1, middle row; a related definition is used in the work [12]). Insofar as coarsening concerns the space of smooth error vectors $\mathbf{x}$, nodes u and v should be aggregated only if $\mathbf{x}_u$ and $\mathbf{x}_v$ are highly correlated for all such $\mathbf{x}$. A set of K Test Vectors (TVs) $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}$ is generated – a sample of this error space [11, §17.2], [55]. Each TV is the result of applying ν relaxation sweeps to $\mathbf{Ax} = \mathbf{0}$, starting from random$[-1, 1]$. However, Ron et al.'s definition falls prey to a graph containing two connected high-degree nodes u and v ("hubs"; Fig. 3.1b). For each k, the value $x_u^{(k)}$ is an average over a large neighborhood of random node values whose size increases with the number of sweeps, hence is small. Similarly, $x_v^{(k)}$ is small, so every such u and v turns out proximal, even though they may be distant.

3.3.2. The New Proximity Measure. LAMG's proximity measure, the *affinity* (Table 3.1, bottom), also relies on TVs, but is scale-invariant, and correctly assesses both Fig. 3.1a and b as well as many other constellations. The affinity c_{uv} between u and v is defined as the goodness of fitting the linear model $x_v \approx p\,x_u$ to TV values:

$$c_{uv} := 1 - \frac{|(X_u, X_v)|^2}{(X_u, X_u)\,(X_v, X_v)}, \quad (X, Y) := \sum_{k=1}^{K} x^{(k)} y^{(k)}, \quad X_u := \left(x_u^{(1)}, \ldots, x_u^{(K)} \right).$$
(3.3)

$c_{uu} = 0$, $0 \leq c_{uv} \leq 1$ and $c_{uv} = c_{vu}$. The affinity measures *distance*: the smaller c_{uv}, the closer u and v. In the d-D discretized Laplace operator on a grid, c_{uv} related to the *geometric distance* between the gridpoints corresponding to u and v. In general graphs, c_{uv} is an alternative definition of the algebraic distance [55], and approximates the *diffusion distance* at a short time ν [22].

3.3.3. Statistical Interpretation. x_u can be thought of as a random variable; c_{uv} is the Fraction of Variance Unexplained of linearly regressing x_u on x_v using the TV samples X_u and X_v [28]. We make several observations:

- c_{uv} is invariant to scaling X_u and X_v. This is vital in the two-hub case (Fig. 3.1b).
- Rather than subtracting the sample means $\overline{X}_u := \sum_{k=1}^{K} x_u^{(k)}$ and $\overline{X}_v$ from X_u and X_v, respectively, as in the standard statistical definition, we use their exact means over all error vectors. These are zero, since each X_u is a linear combination of some initial X_w's, each of which has a zero mean.
- A weighted inner product (X_u, X_v) could be used to account for TV variances, but since all TVs have the same level of smoothness, i.e., comparable normalized residuals [11, §17.2], [13], they were assigned equal weights.

A few vectors K and smoothing sweeps ν suffice to obtain a good enough c_{uv} estimate, which guides a coarsening of the node set by a modest factor of 2–3 (cf. §3.4.2).

3.3.4. Interpolation Accuracy. Bootstrap AMG [11, §17.2] defines a general-caliber $\mathbf{P}$ using a least-squares fit to TVs. In our case,

$$c_{uv} = \min_p \frac{\|pX_u - X_v\|^2}{\|X_v\|^2}$$
(3.4)

relates the affinity to the accuracy of the caliber-1 interpolation formula $x_v = px_u$ for TVs. As a byproduct, we obtain the interpolation coefficient $\hat{p} = (X_u, X_v)/(X_v, X_v)$.

Eq. (3.3) also works for non-zero-sum matrices, such as restricted and normalized Laplacians [61, §2], where p_{uv} is set to $\hat{p}$ (see also §§6.1,6.3). For the Laplacian, $\hat{p}$ is abandoned in favor of $p_{uv} = 1$ (cf. §3.1.4).

In the Helmholtz equation, c_{uv} is large for all u, v, indicating that all nodes are distant and that no single aggregate set can yield fast AMG convergence (indeed, multiple coarse grids are required to restore textbook multigrid efficiency [50]).

3.4. Aggregation. Aggregation levels refer to the two-level method of §2 with a caliber-1 interpolation. **P** is equivalent to partitioning $\mathcal{N}$ into n_c non-overlapping *aggregates* $\{\mathcal{T}_U\}_{U \in \mathcal{N}^c}$; $\mathcal{T}_U$ is the set of $\mathbf{e}_u$ interpolated from $\mathbf{e}_U^c$, and $\mathcal{N}^c := \{1, \dots, n_c\}$. Each aggregate consists of a *seed* node and zero or more *associate* nodes [49, Fig. 3.2].

This section explains the technical details of aggregate selection, and may be of interest to multigrid experts. Other readers may wish to skip it and assess the quality of our aggregation decisions via the numerical experiments in §5.

3.4.1. Aggregation Rules. Intuitively, nodes should be aggregated together if their values are "close"; ideal aggregates have strong internal affinities and weaker external affinities. To this end, we formulated five rules:

1. Each node can be associated with one seed.
2. A seed cannot be associated.
3. Aggregate together nodes with smaller affinities before larger affinities.
4. Favor aggregates with small energy ratios (§3.4.4).
5. A hub node should be a seed.

Rules 1 and 2 prevent an associate from being transitively with a distant seed. Otherwise, long chains might be aggregated together, creating aggregates with weak internal connections and very large energy ratios. Rule 3 favors strongly-connected aggregates. Rule 4 has a dual purpose: (a) Ultimately, the energy ratio determines the AMG asymptotic convergence factor; hence, this rule ensures good convergence. (b) Affinities are based on local information (relaxed TVs); their quantitative value becomes fuzzier as nodes grow apart [55, §5]. Since small energy ratios usually dictate small aggregates, affinities are thus used only for *local* aggregation decisions. Rule 5 avoids costly aggregation decisions due to traversing the large neighbor sets of a hubs, which would increase the computational work (cf. §3.4.4). On the other hand, aggressively coarsening a large clique into a single node is desirable, as all its nodes are strongly correlated. Thus hubs are defined as *locally-high degree* nodes.[1]

A typical coarsening ratio in our algorithm ranges between .3–.5.

3.4.2. Aggregation Algorithm. The algorithm requires the cycle index γ as an input. Each node is marked as a seed, associate or undecided. First, hubs are identified and marked as seeds. Second, edges with very small $|w_{uv}|$ are discarded during aggregation. Since relaxation converges fast at the nodes that become disconnected, they need not be coarsened at all; however, to keep the coarse-level matrix a Laplacian, we aggregate all of them into a single (dummy) aggregate. All other nodes are marked as undecided.

Aggregation is performed in r stages: aggregate sets $S_1, \dots, S_r$ are generated such that each S_i-aggregate is contained in some S_{i+1}-aggregate. The set whose coarsening ratio $\alpha := |S_i|/n$ is closest to $\alpha_{\max} := .7/\gamma$ is selected as the final set, so that the total cycle work would be bounded by $\approx 1 + \gamma\alpha_{\max} + (\gamma\alpha_{\max})^2 + \cdots \approx \frac{10}{3}$ finest-level units,

---

[1] In our code, a hub is a node whose degree is significantly larger than a weighted mean of its neighbors' degrees [34]: $|\mathcal{E}_u| \geq 8 \sum_{v \in \mathcal{E}_u} |w_{uv}||\mathcal{E}_v| / \sum_{v \in \mathcal{E}_u} |w_{uv}|$.

had the same fill-in occurred at all levels (this is a practical guideline for selecting a good coarse set; there exist many sets that yield similar cycle complexities). In our code, at most two stages are performed per coarsening level in the cycle.

In each stage, we scan undecided nodes u and decide whether to aggregate each one with a neighbor s that is either an existing seed, or an undecided node that thereby becomes a new seed. s is the non-associate neighbor of u with the smallest affinity c_{us}. At the end of the last stage, still-undecided nodes are converted to seeds. The complete algorithm is described in the ArXiV e-print [49, §3.4.3].

3.4.3. Energy Inflation. The Galerkin coarse-level correction $\mathbf{e}^c$ (Eq. (2.2)) is the best approximation to a smooth error $\mathbf{e}$ in the energy norm. Braess [8] noted that this does not guarantee a good approximation in the l_2 norm. For example, if $\mathbf{e}$ is a piecewise linear function in a path graph (1-D grid with $w \equiv 1$) coarsened by aggregates of size two, $\mathbf{Pe}^c$ is constant on each aggregate and matches $\mathbf{e}$'s slope across aggregates, resulting in about half the fine-level magnitude. See Fig. 3.2a.
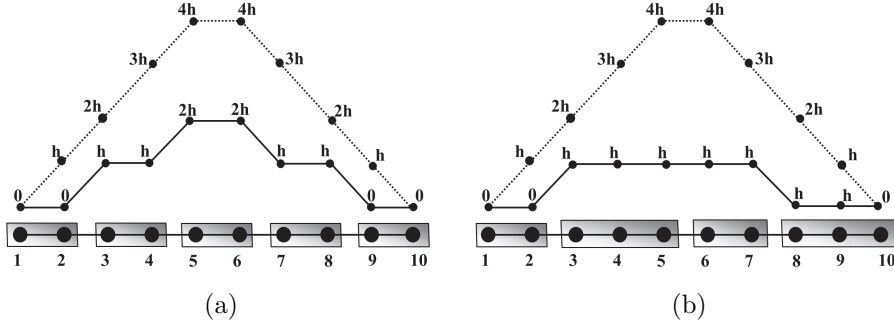

FIG. 3.2. *The Galerkin correction $P\mathbf{e}^c$ to a piecewise linear error $\mathbf{e}$ in a path graph for (a) uniform 1:2 aggregation and (b) variable aggregate size. The meshsize $h > 0$ is arbitrary.*

An equivalent and more useful observation is that the energy of $\mathbf{PTe}$ is twice larger than $\mathbf{e}$'s, where $\mathbf{Te}$ is some coarse representation of $\mathbf{e}$, say,

$$(\mathbf{Te})_U := \frac{1}{|\mathcal{T}_U|} \sum_{u \in \mathcal{T}_U} e_u\,, \qquad U \in \mathcal{N}^c\,. \tag{3.5}$$

$\mathbf{T}$ is called the *aggregate type* operator [16, §2]. (2.3) can be rewritten as

$$\min_{\mathbf{y}^c} \left\{ \frac{1}{2}(\mathbf{y}^c)^T \mathbf{A}^c \mathbf{y}^c - \mathbf{y}^c \mathbf{P}^T \mathbf{r} \right\}\,, \qquad \mathbf{r} := \mathbf{Ae}\,. \tag{3.6}$$

For an ideal interpolation $\mathbf{P}$ that satisfies $\mathbf{PTe} = \mathbf{e}$, (3.6) is minimized by $\mathbf{y}^c = \mathbf{Te}$. A caliber-1 interpolation $\mathbf{P}$ still satisfies $\mathbf{PTe} \approx \mathbf{e}$, but the first term in (3.6) is multiplied by the *energy inflation factor*

$$q(\mathbf{e}) := \frac{E^c(\mathbf{Te})}{E(\mathbf{e})} = \frac{E(\mathbf{PTe})}{E(\mathbf{e})}\,, \qquad E^c(\mathbf{e}^c) := \frac{1}{2}(\mathbf{e}^c)^T \mathbf{A}^c \mathbf{e}^c\,. \tag{3.7}$$

Now (3.6) is minimized by $\mathbf{y}^c \approx q^{-1}\mathbf{Te}$. As $\mathbf{e}$ is not significantly changed by relaxation, its two-level Asymptotic Convergence Factor (ACF) will be $\rho \approx 1 - 1/q$. In Fig. 3.2a, $q \approx 2$ and $\rho \approx .5$.

Several inflation remedies can be pursued:

(A) Increase the **P**'s caliber and accuracy. This leads to fill-in troubles (§3.1.2).
(B) Accept an inferior two-level ACF of $1 - 1/q$ and increase the cycle index γ to maintain it in a multilevel cycle. Unfortunately, not only does this increase complexity, the examples in §3.4.4 demonstrate that q can be arbitrarily large. This ACF cannot be improved by additional smoothing steps either, because it is governed by smooth mode convergence.
(C) Correct the coarse level operator $\mathbf{A}^c$ to match the fine level operator's energy during the setup phase. This option is considered in §3.4.4.
(D) Modify the coarse level correction $\mathbf{P}\mathbf{e}^c$ to match the fine level error $\mathbf{e}$ during the solve phase. This option is pursued in §3.4.5.

3.4.4. Flat Energy Correction. In this scheme, (2.3) is modified to

$$\mathbf{P}^T \mathbf{A} \mathbf{P} \, \mathbf{e}^c = \mu \mathbf{P}^T \left(\mathbf{b} - \mathbf{A} \tilde{\mathbf{x}} \right) . \tag{3.8}$$

The key question is how to choose μ. Motivated by Fig. 3.2a and its two-dimensional analogue, Braess used $\mu = 1.8$, but his V-cycle convergence for 2-D grid graphs was mesh-independent only if a fixed number of levels were used per cycle, and if AMG was used as a preconditioner. In fact, no *predetermined global* factor exists that fits all error *corrections* in scenarios such as Fig. 3.2b, because the coarse-level solution depends on all local inflation ratios, which vary among graph nodes.

On the other hand, a *local energy* correction factor μ does exist. Indeed, the fine-level and coarse-level quadratic energies are separable to nodal energies:

$$E(\mathbf{x}) = \sum_{u \in \mathcal{N}} E_u(\mathbf{x}) , \qquad E_u(\mathbf{x}) := -\frac{1}{2} \sum_{v:v \neq u} a_{uv} \left(x_u - x_v \right)^2 , \tag{3.9a}$$

$$E^c(\mathbf{x}^c) = \sum_{U \in \mathcal{N}^c} E_U^c(\mathbf{x}^c) , \quad E_U^c(\mathbf{x}^c) := -\frac{1}{2} \sum_{V:V \neq U} a_{UV}^c \left(x_U^c - x_V^c \right)^2 . \tag{3.9b}$$

Here $E_u(\mathbf{x})$ is the nodal energy at node u, and $E_U^c(\mathbf{x}^c)$ is the nodal energy at aggregate U. The *local inflation factor* at aggregate U is defined by

$$q_U(\mathbf{x}) := \frac{E_U^c \left(\mathbf{T}\mathbf{x} \right)}{\sum_{u \in U} E_u \left(\mathbf{x} \right)} . \tag{3.10}$$

In principle, a *local μ_U* can be designed using our TVs to at least partially offset q_U; unfortunately, new difficulties arise (cf. §6.2). Thus we chose to still scale the right-hand side by a global μ, but *modify the aggregation so that $q_U(\mathbf{x}) \lessgtr Q$ for all smooth vectors $\mathbf{x}$ and all $U \in \mathcal{N}^c$*, where $Q > 1$ is a parameter. Under this condition a global factor is effective, whose optimal value minimizes the overall convergence factor:

$$\mu_{\mathrm{opt}} = \operatorname*{argmin}_{\mu > 0} \max_{1 \leq q \leq Q} \left| 1 - \frac{\mu}{q} \right| = \operatorname*{argmin}_{\mu > 0} \max \left\{ |1 - \mu| , \left| 1 - \frac{\mu}{Q} \right| \right\} = \frac{2Q}{Q + 1} . \tag{3.11}$$

In LAMG, we shoot for $Q = 2$; hence $\mu_{\mathrm{opt}} = \frac{4}{3}$, and the expected ACF of smooth errors is $Q/(Q + 1) = \frac{1}{3}$.

The worst energy ratio $Q := \max_{\mathbf{x}} q_U(\mathbf{x})$ varies considerably with aggregate size, shape and alignment. Fig. 3.3 depicts four constellations that may arise in an unweighted 2-D grid graph. (While these examples do not represent every scenario that can occur in graphs, they provide necessary conditions under which the algorithm must work.) Limiting the aggregate size to 2, for instance, would not prevent case

(d), whose d-dimensional analogue yields an unbounded $Q = d + 1$. Fortuitously, we already possess the tool to signal and avoid bad aggregates: test vectors. The algorithm of §3.4.2 is modified so that u is only aggregated with a seed s *if the local energy ratios of all test vectors are sufficiently small.*
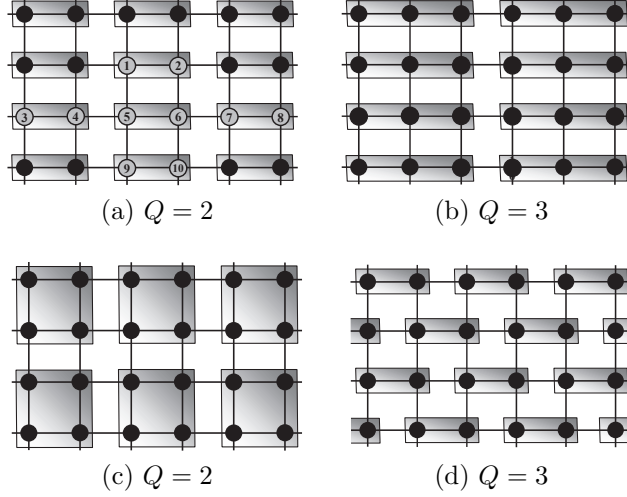

FIG. 3.3. *Coarsening patterns. (a) 1:2 semi-coarsening. The energy ratio of aggregate $\{5,6\}$ depends on $\{x_u\}_{u=1}^{10}$. (b) 1:3 semi-coarsening. (c) 1:2 full coarsening. (d) Staggered semi-coarsening.*

Specifically, we compare the nodal energy E_u before and after aggregation for each TV. Note that the nodal energy (3.9a) is a quadratic in x_u and $\{x_v\}_{v \in \mathcal{E}_u}$. Define

$$E_u(\mathbf{x}; y) := \frac{1}{2} a_{uu} y^2 - B_u(\mathbf{x}) y + C_u(\mathbf{x}), B_u(\mathbf{x}) := \sum_{v \in \mathcal{E}_u} w_{uv} x_v, C_u(\mathbf{x}) := \frac{1}{2} \sum_{v \in \mathcal{E}_u} w_{uv} x_v^2,$$

so $E_u(\mathbf{x}) = E_u(\mathbf{x}; x_u)$. The energy inflation that would occur upon aggregating u with a seed s is estimated by

$$q_{us} := \max_{1 \leq k \leq K} \frac{\min_{y} E_u\left(\mathbf{x}^{(k)}; y\right)}{E_u\left(\mathbf{x}^{(k)}; x_s^{(k)}\right)} . \tag{3.12}$$

The numerator is the local energy after a temporary relaxation step is performed at u (since the coarse-level correction is executed on a relaxed iterate during the cycle, this is the energy it aims to approximate; the papers [5, 18] use a similar idea). The denominator is the energy obtained when $x_u^{(k)}$ is set to $x_s^{(k)}$, simulating the caliber-1 aggregation; more accurate coarse-level energy estimates could be used, but we have not pursued them in the lean spirit of LAMG. We aggregate u with the seed s whose c_{us} is minimal of all seeds t with $q_{ut} \leq 2.5$; if none exist, u is not aggregated at all. (Ratios slightly greater than the target $Q = 2$ are accepted because TVs also contain high-energy modes, for which strict ratios are neither attainable nor necessary.) The complexity of the aggregation decision is $O(K|\mathcal{E}_u|)$ [49, §3.5.4].

Low-degree elimination (§3.2) is advantageous because (a) it largely prevents worst case 1-D scenarios such as Fig. 3.2b, where it is impossible to obtain low energy ratios without excessively increasing the coarsening ratio; (b) it increases the number of neighbors of u and the chance of locating a seed s with small energy inflation.

3.4.5. Iterate Recombination. Instead of fixing μ by (3.8), an effectively-*adaptive energy correction* is obtained by modifying the correction to smooth errors during the solution cycle. Let l be any level such that $l+1$ is an AGGREGATION level. When the cycle switches from level $l-1$ to level l, ϑ sub-cycles are applied to $\mathbf{A}^l \mathbf{x}^l = \mathbf{b}^l$, where ϑ is 1 or 2. We save the iterates $\mathbf{x}_i^l$ obtained after the pre-relaxation of sub-cycle i, and, before switching back to level $l-1$, replace the final iterate $\mathbf{x}^l$ by

$$\mathbf{y}^l = \mathbf{x}^l + \boldsymbol{\alpha}_1 \left(\mathbf{x}_1^l - \mathbf{x}^l \right) + \cdots + \boldsymbol{\alpha}_\vartheta \left(\mathbf{x}_\vartheta^l - \mathbf{x}^l \right) , \qquad (3.13)$$

where $\{\alpha_i\}_{i=1}^\vartheta$ are chosen so that $\|\mathbf{b}^l - \mathbf{A}^l \mathbf{y}^l\|_2$ is minimized (this is an $n_l \times \vartheta$ least-squares problem solved in $O(n_l)$ time). This *iterate recombination* [65, §7.8.2] diminishes smooth errors $\mathbf{x}_i^l - \mathbf{x}^l$ that were not eliminated by $(l+1)^{\text{th}}$-level corrections. Since the initial *residuals* obtained after interpolation from level $l+1$ are not smooth, the residual minimization is only effective after $\mathbf{x}_i^l - \mathbf{x}^l$ is smoothed. To maximize iterate smoothness, we perform more post- than pre-smoothing relaxations. The optimal splitting turned out to be a (1,2)-cycle; cf. §4.

This acceleration is superior to CG because it is performed at all levels. Iterate recombination at coarse levels has been long recognized as an effective tool in the multigrid literature [65, Remark 7.8.5]. In LAMG, recombination occurs more frequently at coarser levels because $\gamma > 1$. Notay's K-cycle [52, 53] employs a similar multilevel CG acceleration, however with a much larger cycle index (up to $\gamma = 4$), which increases the solver's complexity.

The aggregation is still modified here as in §3.4.4, to ensure small energy ratios and maximum reduction in the residual norm after recombination.

4. The LAMG Algorithm.

4.1. Setup Phase. The setup flow is depicted in Fig. 4.1. Its sole input is the cycle index $\gamma \geq 1$ to be employed at most levels of subsequent solution cycles. In our program, $\gamma = 1.5$; this choice is discussed in §4.2. The original problem ($l = 1$) is repeatedly coarsened by either elimination or caliber-1 aggregation until the number of nodes drops below 150, or until relaxation converges rapidly. We employ $K = 4$ TVs at the finest level, and increase K up to 10 at coarser levels. Each TV is smoothed by $\nu = 3$ relaxation sweeps.
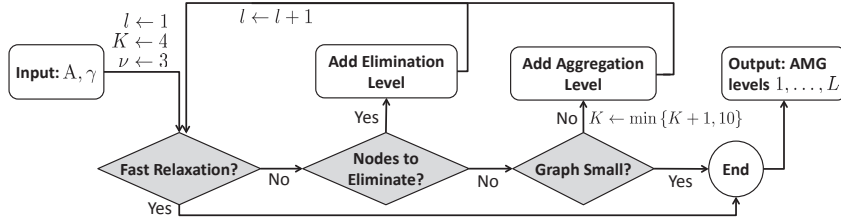

FIG. 4.1. *LAMG setup phase flowchart.*

4.2. Solve Phase. The solve phase consists of multigrid cycles [14, §1.4]. Each $l < L$ is assigned a cycle index γ^l and pre- and post-relaxation sweep numbers ν_1^l, ν_2^l. If level $l+1$ is the result of elimination, $\gamma^l = 1$ and $\nu_1^l = \nu_2^l = 0$; otherwise,

$$\gamma^l := \begin{cases} \gamma, & |\mathcal{E}^l| > .1|\mathcal{E}|, \\ \min\{2, .7\,|\mathcal{E}^{l+1}|/|\mathcal{E}^l|\}, & \text{otherwise}, \end{cases} \qquad \nu_1^l = 1, \quad \nu_2^l = 2. \qquad (4.1)$$

At fine levels, $\gamma = 1.5$ is employed. This value is theoretically marginal for attaining a bounded multilevel ACF if the smoothest error two-level ACF is $\approx \frac{1}{3}$ [14, §6.2], as implied by (3.11) for $Q = 2$. Notwithstanding, worst-case energy ratios occur infrequently, and in practice a smaller ACF is obtained. This issue is further diminished by the adaptive energy correction. At coarse levels, γ^l is increased to maximize error reduction while incurring a bounded work increase.

Three relaxation sweeps per level provide adequate smoothing, especially in light of the coarse-level correction's crudeness. The coarsest problem is solved by relaxation (if it is fast) or a direct solver on an augmented system [49, §3.6.3]. Finally, (1.3b) is enforced by subtracting the mean of $\mathbf{x}$ from $\mathbf{x}$ at the end of the cycle.
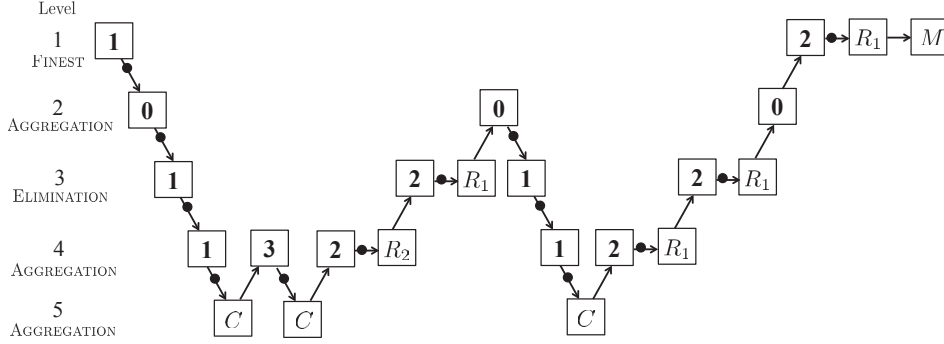

FIG. 4.2. *A four-level cycle. A boxed number denotes a number of relaxations. C: the coarsest-level solver. M: subtracting the iterate mean. Down-arrows: right-hand side coarsening (3.2) or (3.8). Up-arrows: coarse-level corrections [49, Eq. (3.4a)] or (2.4). R_ϑ: a $(\vartheta + 1)$-iterate recombination (3.13). Iterates are saved at the black dots before coarsening.*

All cycle parameters are fixed: no fine tuning or parameter optimization is required for specific graphs. The total cycle work is equivalent to about 10 relaxations.

5. Numerical Results. We provide supporting evidence for LAMG's practical efficiency for a wide range of graphs.

5.1. Smorgasbord. An object-oriented MATLAB 7.13 (R2011b) serial LAMG implementation was developed and is freely available online [48]. The time-intensive functions were implemented in C and ported with the mex compiler [24]. It was tested on a diverse set of 3774 real-world graphs with up to 47 million edges, collected from The University of Florida Sparse Matrix collection (UF) [25], C. Walshaw's graph partitioning archive [66], I. Safro's MLogA results archive at Argonne National Laboratory [57], and the FTP site of the DIMACS Implementation Challenges [26].

Graphs originated from a plethora of applications: airplane and car finite-element meshes; RF electrical circuits; combinatorial optimization; model reduction benchmarks; social networks; and web and biological networks. If the graph was directed, it was converted to undirected by summing the weights of both directions between each two nodes. Then, if it contained a large negative edge weight with $w_{uv} < -10^{-5} \sum_{v' \in \mathcal{E}_u} |w_{uv'}|$, all weights were made positive by taking their absolute values. Finally, the Laplacian matrix $\mathbf{A}$ was formed and used.

Runs were performed on Beagle, a 150 teraflops, 18,000-core Cray XE6 supercomputer at The University of Chicago (we only took advantage of parallelism by dividing the collection into equal parts, each of which ran a single AMD node with 2.2 GHz

CPU and 32GB RAM). For each graph, a zero-sum random $\mathbf{b}$ was generated. LAMG setup, followed by a linear solve that started from a random guess and proceeded until the residual l_2-norm was reduced by 10^{10}. Six performance measures were computed:

- *Setup time per edge t_{setup}*.
- *Solve time per edge per significant figure t_{solve}*. If the residual norm after i iterations was r_i and p iterations were executed, $t_{\text{solve}} := t/(m \log_{10}(r_0/r_p))$, where t was the solve time.
- *Total time per edge $t_{\text{total}} = t_{\text{setup}} + 10 t_{\text{solve}}$* to solve $\mathbf{Ax} = \mathbf{b}$ to 10 significant figures for a single $\mathbf{b}$.
- Storage per edge.
- *Asymptotic convergence factor*, estimated by $(r_p/r_0)^{1/p}$.
- *Percentage spent on setup $t_{\text{setup}}/t_{\text{total}}$*.

LAMG scaled linearly with graph size: both t_{setup} and t_{solve} were approximately constant (Fig. 5.1a; Table 5.1). Times were measured in terms of the most basic sparse matrix operation: a matrix-vector multiplication (MVM), because even MVM time scaled slightly superlinearly for $m \geq 5 \times 10^6$ due to loss of memory locality in the MATLAB compressed-column format [2]. In wall clock time, the total time per edge was 5.6×10^{-6} on average, i.e., LAMG performed a linear solve to 10 significant figures at $178{,}000$ edges per second. The LAMG hierarchy required the equivalent of storing $\approx 4m$ edges in memory (Fig. 5.1b). Adaptive energy correction provided a 20% speed up over a flat $\mu = \frac{4}{3}$ and was thus employed in all reported experiments. The ACF was better than the expected .33 for flat correction (§3.4.4).

We compared LAMG with MATLAB's direct solver (the '\' operator). Since the direct solver ran out of memory for many graphs with over 10^5 edges, we did not include it in the plots. LAMG aims at robustness for a wide variety of graphs, and should be compared against solvers that do not often break down or require tuning, even if they are faster for a subset of the graphs (in analogy, many graphs could be solved much faster with a tailored geometric multigrid or classical AMG algorithm).

An advantage of iterative solvers over direct is their tunable solution accuracy ε. Since $\mathbf{A}$'s entries often incur measurement or modeling errors in applications, it does not make sense to solve $\mathbf{Ax} = \mathbf{b}$ to more than 2–3 significant figures. Furthermore, a *single* multigrid cycle is typically sufficient to solve a nonlinear problems as well as time-dependent problems to the level of discretization errors [14, Chaps. 7,15].

We also compared LAMG against a MATLAB implementation of CMG, a hybrid graph-theoretic-AMG solver [42]. Since CMG doesn't run yet on the Cray architecture, experiments were performed on a smaller 64-bit Dell Inspiron 580 (3.2 GHz CPU; 8GB RAM) for 2668 graphs with up to 10^7 edges. Both algorithms successfully solved all graphs. Solve times were similar, while LAMG's setup time was thrice larger than CMG's. On the other hand, LAMG was much more robust than CMG: it had only 3 outliers whose solve time was large, as opposed to 26 CMG outliers whose relative magnitude was much larger (4 in setup and 22 in solve; see Fig. 5.1c-d and Table 5.2).

5.1.1. LAMG's Outliers. The three solve-time outliers (Table 5.2) were characterized by a large portion of small edge weights, which were carried over to all coarse matrices and increased coarsening ratios. Since LAMG's work was controlled by decreasing γ via (4.1), a slower cycle resulted. We plan to improve those cases in the future by appropriately ignoring weak edges at each level; cf. §6.1.
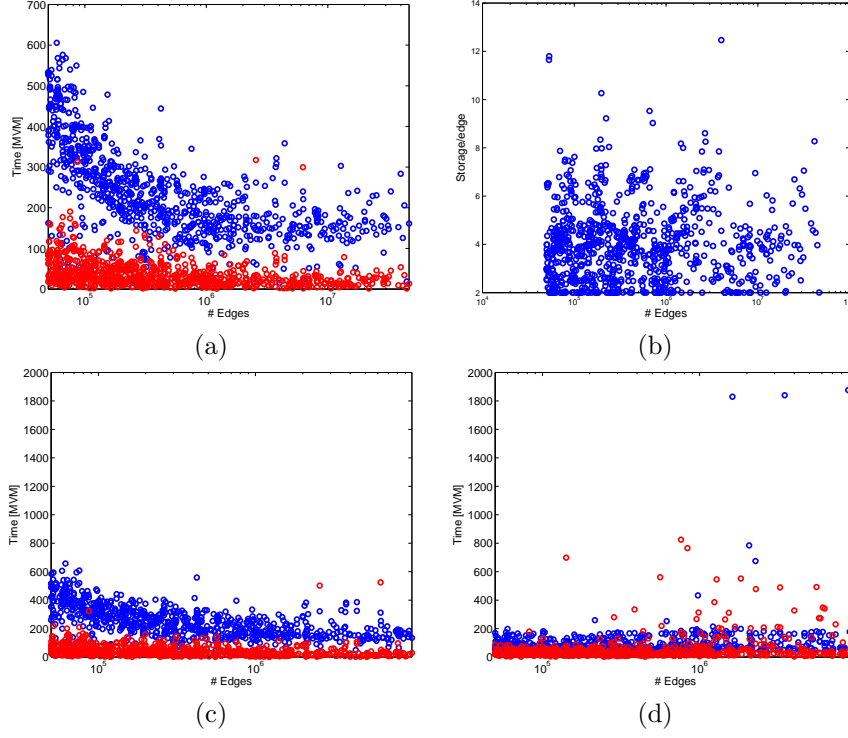
---

[2]T. Davis, private communication.

Fig. 5.1. *(a) LAMG setup time (blue) and solve time (red) per edge on Beagle (up to 4.7×10^7 edges). (b) LAMG storage per edge on Beagle. (c) LAMG setup and solve time per edge on a Dell Inspiron (up to 10^7 edges). (d) CMG setup and solve time per edge on a Dell Inspiron.*

Measure	LAMG (Beagle)		LAMG (Dell)		CMG (Dell)	
	Median	Mean $\pm$ Std.	Median	Mean $\pm$ Std.	Median	Mean $\pm$ Std.
t_{total}	482.9	585.4 ± 406	558.1	680.5 ± 497	342.2	582.6 ± 1088
t_{setup}	199.6	222.5 ± 108	234.9	248.4 ± 113	66.3	89.3 ± 111
t_{solve}	27.3	36.3 ± 33	31.6	43.2 ± 42.3	25.5	47.9 ± 106
ACF	.107	$.128 \pm .12$	.112	$.132 \pm .11$	.500	$.495 \pm .21$
%Setup	43.8%	$43.7\% \pm 13\%$	42.4%	$43.4\% \pm 13\%$	21.5%	$22.9\% \pm 12\%$

TABLE 5.1

Left column: median and mean LAMG performance on the Beagle Cray for 892 graphs with $50,000 \leq m \leq 4.7 \times 10^7$. Middle and right: LAMG vs. CMG performance on a Dell Inspiron for 794 graphs with $50,000 \leq m \leq 10^7$. Times are measured in matrix-vector multiplications.

Name	n	m	LAMG (Dell)			CMG (Dell)	
			ACF	t_{setup}	t_{solve}	t_{setup}	t_{solve}
Ill-conditioned Stokes	20896	87010	**.66**	420	**323**	66	25
Large basis	440020	2560040	**.88**	322	**502**	70	107
RF circuit simulation	4690002	6251251	**.72**	312	**525**	65	304
Law citation network	925340	6675561	.24	169	15	152	**2037**
Berkeley-Stanford web	512501	3480880	.17	168	18	**1585**	126
Molecule pseudopotential	268096	8833823	.13	167	21	**1879**	41

TABLE 5.2

Top section: the three LAMG outliers. Bottom section: three of CMG's 26 outliers. Times are measured in matrix-vector multiplications.
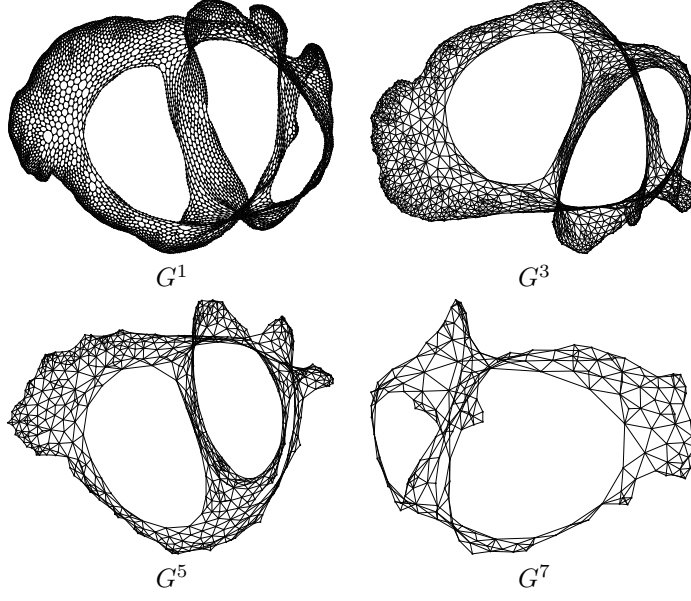
FIG. 5.2. *The four finest aggregation levels for the UF 2-D airfoil finite-element planar graph* `AG-Monien/airfoil1-dual`*. Graphs were drawn using GraphViz with the SFDP algorithm [29].*
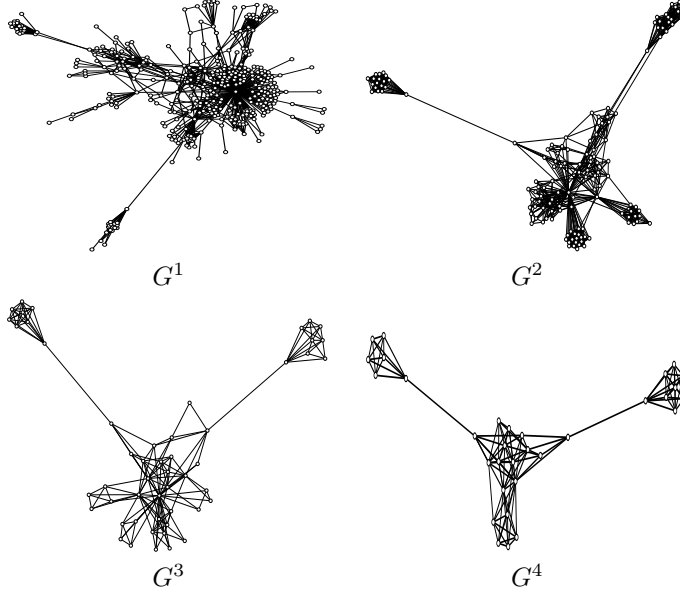

FIG. 5.3. *The four finest levels for the UF Harvard 500 non-planar web graph [51].*

5.2. Grids with Negative Weights. Unlike CMG, LAMG is not restricted to diagonally-dominant systems, and can also be applied to some graphs with negative edge weights w_{uv}, as long as the Laplacian matrix is (or is very close to being) positive semi-definite. To demonstrate this capability, we tested LAMG on the following SPS 2-D grid Laplacians, whose stencils are depicted in Fig. 5.4:

(a) The standard 5-point finite-difference discretization of $U_{xx} + U_{yy}$ on the unit square with Neumann boundary conditions.

(b) The 13-point 4$^{\text{th}}$-order finite-difference stencil of $U_{xx} + U_{yy}$.

(c) The discretized anisotropic-rotated Laplace operator

$$\left(\cos^2 \alpha + \varepsilon \sin^2 \alpha\right) U_{xx} + (1 - \varepsilon) \sin(2\alpha) U_{xy} + \left(\varepsilon \cos^2 \alpha + \sin^2 \alpha\right) U_{yy}, \quad (5.1)$$

with $\alpha = -\pi/4$, $\varepsilon = 10^{-2}$, standard 5-point stencil of U_{xx}, U_{yy}, and an alignment-agnostic cross-term

$$U_{xy} \approx \frac{1}{4h^2} \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix},$$

where h is the grid meshsize. Neumann boundary conditions were used.

(d) The same as (c), but aligning U_{xy} with the northeast and southwest neighbors:

$$U_{xy} \approx \frac{1}{2h^2} \begin{bmatrix} 0 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 0 \end{bmatrix}.$$

$$\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}$$
(a)

$$\begin{bmatrix} & & -1 & & \\ & & -16 & & \\ 1 & -16 & 60 & -16 & 1 \\ & & -16 & & \\ & & 1 & & \end{bmatrix}$$
(b)

$$\begin{bmatrix} -.12375 & -.25250 & .12375 \\ -.25250 & 1.5050 & -.25250 \\ .12375 & -.25250 & -.12375 \end{bmatrix}$$
(c)

$$\begin{bmatrix} & -0.5000 & .24750 \\ -.5000 & 1.5050 & -0.5000 \\ .24750 & -0.5000 & \end{bmatrix}$$
(d)

FIG. 5.4. *Stencils of grid Laplacians with negative weights. Entries are normalized to a meshsize-independent sum and rounded to five significant figures.*

Problems (c) and (d) are bad discretizations that do not align with the characteristic direction of (5.1), and are considered hard for AMG [12]. Performance figures for the Dell Inspiron are given in Table 5.3.

Problem	m	L	ACF	%Setup	t_{total}
(a) 5-point	2095104	19	0.216	29%	902
(b) 13-point 4$^{\text{th}}$ order	4188160	20	0.262	22%	1355
(c) Anis. rot. agnostic	4188162	19	**0.816**	5%	**5453**
(d) Anis. rot. misaligned	3141633	20	**0.870**	4%	**8136**

TABLE 5.3
LAMG performance for grid graphs on a 1024×1024 grid with $n = 1048576$ nodes.

LAMG exhibited mesh-independent convergence and run time in all cases and scaled linearly with grid size, albeit its convergence was much slower for cases (c) and (d), whose negative edge weights are more significant. Compared with the Bootstrap AMG method [12], which focused on accurately finding the characteristic directions without sparing setup costs and only presented two-level experiments, LAMG is a full multi-level method with a far shorter setup time, although its ACF could also be significantly reduced using bootstrap tools. These results are certainly preliminary.

5.3. Lean Geometric Multigrid. Higher performance for the Poisson equation discretized on a uniform grid can be obtained by a standard 1:2 coarsening in every dimension at all levels and employing Gauss-Seidel relaxation in red-black ordering [14, §3.6]. LAMG reduces to *Lean Geometric Multigrid*: standard multigrid cycle with index $\gamma = 1.5$, first-order transfers and energy-corrected coarsening. Since the energy ratio is 2 for all error modes, a flat correction $\mu = 2$ is employed in (3.8).

For the 2-D *periodic* Poisson problem, this cycle turns out to be a record-breaking Poisson solver in terms of asymptotic efficiency: it achieves a convergence factor of .5 per unit work, versus .67 for the classical multigrid V(1,1) cycle with linear interpolation and second-order full weighting [49, §4.1]. For other boundary conditions, finding the right μ is not as easy; while supplementary local relaxations near boundaries theoretically ensure attaining the two-level rates [14, §5], it would be more beneficial to study the performance of adaptive energy correction in geometric LAMG.

6. Future Research. Enhancements and adaptations of the LAMG approach to related computational problems are outlined below.

6.1. Coarsening Improvements. The LAMG algorithm of §4 is by no means final and may be improved in various ways.
- The average setup time could be reduced by employing classical AMG with no test vectors, and switching to the LAMG strategy only when the former fails.
- In graphs with many weak edges (such as the outliers in Table. 5.2), efficiency may be increased by temporarily ignoring them in the Galerkin operator computation, yet keeping track of their total contribution to each aggregate's stencil. If a level is reached at which this total is no longer small compared with the aggregate's other edge weights, it is reactivated.
- Currently, a node u can only be aggregated with a direct neighbor s. In some problems, u's second-degree neighbors should also be searched to ensure a good aggregation. For instance, in the anisotropic-rotated problem Fig. 5.4d, u should be aggregated along the characteristic direction, i.e., with its southeast or northwest neighbor, neither of which is contained in $\mathcal{E}_u$.
- If no small energy ratio can be found, or if subsequent cycle convergence is slow, isolated bottleneck nodes can be de-aggregated. Alternatively, one can up the interpolation caliber at these troublesome nodes, provided that this does not substantially increase the total coarse edges.
- Adaptive local relaxation sweeps may improve efficiency in various problems such as PDEs with structural singularities [2].

Additionally, user-defined parameters could be supplied to treat special graph families more efficiently. For instance, if node coordinates are available, they can be used to generate smoother initial TVs than the default random initial guess. Optimizing the coarsening is most advantageous when $\mathbf{Ax} = \mathbf{b}$ is solved for multiple $\mathbf{b}$'s, since a larger setup cost is tolerable. Such is the case in time-dependent problems [31].

6.2. Local Energy Correction. Instead of a flat $\mu = \frac{4}{3}$ factor in (3.8), one can apply different μ's to different aggregates. We experimented with different energy correction schemes, some based on fitting the coarse nodal energies of TVs to their fine counterparts (cf. (3.9b)). While this can dramatically curtail energy inflation, care must be taken to avert over-fitting that ultimately results in the coarse-level correction operator's instability.

Analogously, one can define a local adaptive μ in the iterate recombination (§3.4.5)

at each level l, provided that it is properly smoothed [49, §5.3]. It is unclear whether the extra work would be justified in either case. It may turn out useful for problems re-solved for many $\mathbf{b}$ vectors, or when a larger setup overhead is tolerable.

6.3. Other Linear Systems. The LAMG caliber-1 aggregation can be applied to non-zero row sum matrices, except that the interpolation weights are no longer 1. The affinity definition (3.3) remains intact, while the corresponding $\mathbf{P}$ entry is set to $p_{uv} := (X_u, X_v)/(X_v, X_v)$ (see (3.4)). Normally, relaxed TVs yield an accurate enough p_{uv}; in problems with almost-zero modes, e.g., the QCD gauge Laplacian, TVs may need to be improved by a bootstrap cycle [13].

Further research should be conducted for negative-weight graphs such as the high-order finite element and anisotropic grid graphs of §5.2. The reported convergence factors can be improved by producing bootstrapped TVs via applying multilevel cycles to $\mathbf{Ax} = \mathbf{0}$. The cycle is far more powerful than plain relaxation in damping smooth characteristic components, which should lead to more meaningful algebraic distances and to correct anisotropic coarsening in a second setup round (much larger spacing in the characteristic direction and no coarsening in the cross-characteristic direction). The bootstrap procedure should be useful in many other graphs.

6.4. LAMG Eigensolver. The LAMG hierarchy can be combined with the Full Approximation Scheme (FAS) [14, Chap. 8] to find the K lowest eigenpairs of $\mathbf{A}$, similarly to the work [44]. We perform the variable substitution $\mathbf{x}^c = \mathbf{e}^c + \mathbf{R}\tilde{\mathbf{x}}$, transforming the coarse equation (2.3) into

$$\mathbf{A}^c \mathbf{x}^c = \mathbf{P}^T \mathbf{b} + \boldsymbol{\tau}\tilde{\mathbf{x}}, \qquad \boldsymbol{\tau} := \mathbf{A}^c \mathbf{R} - \mathbf{P}^T \mathbf{A}, \tag{6.1}$$

followed by the fine-level correction $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} + \mathbf{P}(\tilde{\mathbf{x}}^c - \mathbf{R}\tilde{\mathbf{x}})$. The elimination and aggregation are both special cases of (6.1), with $\mathbf{R}\tilde{\mathbf{x}} := \tilde{\mathbf{x}}_\mathcal{C}$ and $\mathbf{R} = \mathbf{0}$, respectively.

The analogue of (2.3) for coarsening $(\mathbf{A} - \lambda_k \mathbf{I})\mathbf{x}_k = \mathbf{0}$ is

$$(\mathbf{A}^c - \lambda_k \mathbf{B}^c)\mathbf{x}_k^c = \boldsymbol{\tau}_k \tilde{\mathbf{x}}, \qquad \boldsymbol{\tau}_k = (\mathbf{A}^c - \lambda_k \mathbf{B}^c)\mathbf{R} - \mathbf{P}^T(\mathbf{A} - \lambda_k \mathbf{B}^c). \tag{6.2}$$

Thus a separate affine term appears in the coarse equation of each approximate eigenvector $\mathbf{x}_k^c$, $k = 1, \ldots, K$. In particular, the elimination of §3.2 becomes approximate, yet (6.2) remains linear in λ_k, as opposed to the exact non-linear Schur complement formed by the AMLS method [3]. Gauss-Seidel may be replaced by Kaczmarz relaxation at very coarse levels to prevent the divergence of smooth error modes [44].

Alternatively, one can incorporate the LAMG linear solver into a Rayleigh quotient iteration [35, §8.2],[39]. However, FAS is attractive because it also applies to general nonlinear problems [14, §8], e.g., quadratic and linear programming.

7. Conclusion. Laplacian matrices underlie a plethora of graph computational applications ranging from genetic data clustering to social networks to fluid dynamics. To the best of our knowledge, the presented algorithm, Lean Algebraic Multigrid (LAMG), is the first graph Laplacian linear solver whose empirical performance approaches linear scaling for a wide variety of real-world graphs. Combinatorial Multigrid was also quite successful, performing faster on average, yet with many more outliers. The LAMG approach can also be generalized to non-diagonally-dominant, eigenvalue and nonlinear problems.

REFERENCES

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Mat. Anal. Appl., 17 (1996), pp. 886–905.

[2] D. Bai and A. Brandt, *Local mesh refinement multilevel techniques*, SIAM J. Sci. Stat. Comp., 8 (1987), pp. 109–134.

[3] Constantine Bekas and Yousef Saad, *Computation of smallest eigenvalues using spectral schur complements*, SIAM J. Sci. Comput., 27 (2005), pp. 458–481.

[4] M. Blatt, *A parallel algebraic multigrid method for elliptic problems with highly discontinuous coefficients*, PhD thesis, Universität Heidelberg, 2010.

[5] M Bolten, A Brandt, J Brannick, A Frommer, K Kahl, and I Livshits, *A bootstrap algebraic multilevel method for Markov chains*, (2010), p. 16.

[6] E. G. Boman, D. Chen, B. Hendrickson, and S. Toledo, *Maximum-weight-basis preconditioners*, Applications, 29 (2002), pp. 695–721.

[7] E. G. Boman, B. Hendrickson, and S. Vavasis, *Solving elliptic finite element systems in near-linear time with support preconditioners*, SIAM J. Num. Anal., 46 (2008), pp. 3264–3284.

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

[9] A. Brandt, *Rigorous quantitative analysis of multigrid, I. Constant coefficients two-level cycle with l_2-norm*, Siam J. Num Anal., 31 (1994).

[10] ———, *General highly accurate algebraic coarsening*, J. Electron. Trans. Num. Anal., 10 (2000), pp. 1–20. Multilevel methods (Copper Mountain, CO, 1999).

[11] ———, *Multiscale scientific computation: Review 2001*, in Multiscale and Multiresolution Methods, T. Barth, T. Chan, and R. Haimes, eds., Springer-Verlag, 2002, pp. 3–96.

[12] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *An algebraic distances measure of AMG strength of connection*. ArXiV e-prints, http://arxiv.org/abs/1106.5990v1, 2011.

[13] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *Bootstrap amg*, SIAM J. Sci. Comp., 33 (2011), pp. 612–632.

[14] A. Brandt and O. E. Livne, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, Classics in Applied Mathematics, SIAM, revised ed., 2011.

[15] A. Brandt, S. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations*, tech. report, Colorado State University, Fort Collins, Colorado, 1983.

[16] J. J. Brannick and R. D. Falgout, *Compatible relaxation and coarsening in algebraic multigrid*, SIAM J. Sci. Comp., 32 (2010), pp. 1393–1416.

[17] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (ASA)*, SIAM J. Sci. Comp., 25 (2004), p. 2004.

[18] M. Brezina, C. Ketelsen, T. Manteuffel, S. McCormick, M. Park, and J. Ruge, *Relaxation-corrected bootstrap algebraic multigrid (rBAMG)*, Numer. Lin. Alg. Appl., 19 (2012), pp. 178–193.

[19] M. Brezina, P. Vanek, and P. S. Vassilevski, *An improved convergence analysis of smoothed aggregation algebraic multigrid*, Num. Lin. Alg. Appl., (2011).

[20] H. Chang and D. Yeung, *Graph Laplacian kernels for object classification from a single example*, in In CVPR (2, 2006, pp. 2011–2016.

[21] F. R. K. Chung, *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*, American Mathematical Society, Feb. 1997.

[22] R.R. Coifman and S. Lafon, *Diffusion maps*, Applied Comput. Harmon. Anal., 21 (2006), pp. 5–30.

[23] S. I. Daitch and D. A. Spielman, *Faster approximate lossy generalized flow via interior point algorithms*, CoRR, abs/0803.0988 (2008).

[24] T.A. Davis, *MATLAB Primer*, Taylor and Francis, 2010.

[25] T. A. Davis, *University of Florida sparse matrix collection*, NA Digest, 92 (1994).

[26] DIMACS Center at Rutgers Univesity, *Dimacs implementation challenges*. Available online at http://dimacs.rutgers.edu/Challenges/, 2011.

[27] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, *A min-max cut algorithm for graph partitioning and data clustering*, in Proceedings of ICDM 2001, 2001, pp. 107–114.

[28] N.R. Draper and H. Smith, *Applied Regression Analysis*, Wiley-Interscience, 1998.

[29] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull, *Graphviz - open source graph drawing tools*, Graph Drawing, (2001), pp. 483–484.

[30] R. Falgout, A. Cleary, J. Jones, E. Chow, V. Henson, C. Baldwin, P. Brown, P. Vassilevski, and U. Meier Yang, *Hypre reference manual 2.7.0b*. https://computation.

llnl.gov/casc/hypre/download/hypre-2.7.0b_ref_manual.pdf, 2011.

[31] P. Fischer, J. Lottes, D. Pointer, and A. Siegel, *Petascale algorithms for reactor hydro-dynamics*, J. Physics: Conference Series, 125 (2008), p. 012076.

[32] A. Frangioni and C. Gentile, *Prim-based support-graph preconditioners for min-cost flow problems*, Comput. Optim. Appl., 36 (2007), pp. 271–287.

[33] Alan George, *Nested dissection of a regular finite element mesh*, SIAM Journal on Numerical Analysis, 10 (1973), pp. 345–363.

[34] D. Gleich and C. Seshadhri, *Neighborhoods are good communities*, CoRR, abs/1112.0031 (2011).

[35] G.H. Golub and C.F.V. Loan, *Matrix Computations*, Johns Hopkins studies in the mathematical sciences, Johns Hopkins University Press, third ed., 1996.

[36] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt, *Shape representation and classification using the poisson equation*, in In In Proc. of CVPR04, 2004, pp. 61–67.

[37] B. Hendrickson and R. Leland, *The Chaco user's guide: Version 2.0*, Tech. Report SAND94–2692, Sandia Mational Laboratory, 1994.

[38] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An overview of the Trilinos project*, ACM Trans. Math. Softw., 31 (2005), pp. 397–423.

[39] Andrew V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput, 23 (2001), pp. 517–541.

[40] I. Koutis and G. L. Miller, *Graph partitioning into isolated, high conductance clusters: Theory, computation and applications to preconditioning*, 2008, pp. 137–145.

[41] I. Koutis, G. L. Miller, and R. Peng, *Approaching optimality for solving SDD linear systems*, in Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, FOCS '10, Washington, DC, USA, 2010, IEEE Computer Society, pp. 235–244.

[42] I. Koutis, G. L. Miller, and D. Tolliver, *Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing*, in Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I, ISVC '09, Berlin, Heidelberg, 2009, Springer-Verlag, pp. 1067–1078.

[43] D. Kushnir, M. Galun, and A. Brandt, *Fast multiscale clustering and manifold identification*, Pattern Recognition, 39 (2006), pp. 1876–1891.

[44] ———, *Efficient multilevel eigensolvers with applications to data analysis tasks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32 (2010), pp. 1377–1391.

[45] A. B. Lee, D. Luca, L. Klei, B. Devlin, and K. Roeder, *Discovering genetic ancestry using spectral graph theory*, Genet. Epidem., 34 (2010), pp. 51–59.

[46] R. J. Lipton, D. J. Rose, and R. E. Tarjan, *Generalized nested dissection*, SIAM Journal on Numerical Analysis, 16 (1979), pp. pp. 346–358.

[47] O. E. Livne, *Coarsening by compatible relaxation*, Num. Lin. Alg. Appl., 11 (2004), pp. 205–227.

[48] ———, *Lean Algebraic Multigrid (LAMG) Matlab software*, 2012. Release 2.1.1. Freely available at http://lamg.googlecode.com.

[49] O. E. Livne and A. Brandt, *Lean algebraic multigrid (LAMG): Fast graph Laplacian linear solver*. ArXiV e-prints, http://arxiv.org/abs/1108.0123, 2011.

[50] I. Livshits and A. Brandt, *Accuracy properties of the wave-ray multigrid algorithm for Helmholtz equations*, SIAM J. on Sci. Comp., 28 (2006), pp. 1228–1251.

[51] C. B. Moler, *Numerical Computing with Matlab*, SIAM, 2004.

[52] Y. Notay, *An aggregation-based algebraic multigrid method*, Elec. Trans. Num. Anal., 37 (2010), pp. 123–146.

[53] ———, *Aggregation-based algebraic multigrid for convection-diffusion equations*, Tech. Report GANMN 11-01, Université Libre de Bruxelles, Brussels, Belgium, 2011.

[54] L. N. Olson, J. B. Schroder, and R. S. Tuminaro, *A general interpolation strategy for algebraic multigrid using energy minimization*, SIAM J. Sci. Comp., 33 (2011), pp. 966–991.

[55] D. Ron, I. Safro, and A. Brandt, *Relaxation-based coarsening and multiscale graph organization*, Multiscale Model. Sim., 9 (2011), pp. 407–423.

[56] J. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, Frontiers in Applied Mathematics, S. F. McCormick, ed., SIAM, 1987, pp. 73–130.

[57] I. Safro, *Minimum logarithmic arrangement (MinLogA) results archive*. Available online at http://www.mcs.anl.gov/~safro/mloga.html, 2011.

[58] R. Sedgewick, *Algorithms in C++, Part 5: Graph Algorithms*, Addison-Wesley, 2002.

[59] A. Sharma, R. P. Horaud, D. Knossow, and E. von Lavante, *Mesh segmentation using Laplacian eigenvectors and Gaussian mixtures*, in Proceedings of AAAI Fall Symposium on Manifold Learning and its Applications, Fall Symposium Series Technical Reports, Arlington, VA, November 2009, AAAI Press.

[60] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt, *Hierarchy and adaptivity in segmenting visual scenes*, Nature, 442 (2006), pp. 810–813.

[61] D. A. Spielman, *Algorithms, graph theory, and linear equations in Laplacian matrices*, in Proceedings of the International Congress of Mathematicians 2010 (ICM 2010), World Scientific, 2010, pp. 2698–2722.

[62] D. A. Spielman and S. Teng, *Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems*, CoRR, abs/cs/0607105 (2006).

[63] R. E. Tarjan, *Depth first search and linear graph algorithms*, SIAM J. Computing, 1 (1972), pp. 146–160.

[64] W. F. Tinney and J. W. Walker, *Direct solutions of sparse network equations by optimally ordered triangular factorization*, Proc. IEEE, 55 (1967), pp. 1801–1809.

[65] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2000.

[66] C. Walshaw, *Multilevel refinement for combinatorial optimisation problems*, Annals Oper. Res., 131 (2004), pp. 325–372.

[67] X. Zhu, Z. Ghahramani, and J. D. Lafferty, *Semi-supervised learning using Gaussian fields and harmonic functions.*, in ICML'03, 2003, pp. 912–919.