SIAM J. Sci. Comput. © 2017 Society for Industrial and Applied Mathematics
Vol. 39, No. 5, pp. S379–S403

AN EFFICIENT MULTIGRID METHOD FOR GRAPH LAPLACIAN SYSTEMS II: ROBUST AGGREGATION[*]

ARTEM NAPOV[†] AND YVAN NOTAY[†]

Abstract. We consider the iterative solution of linear systems whose matrices are Laplacians of undirected graphs. Designing robust solvers for this class of problems is challenging due to the diversity of connectivity patterns encountered in practical applications. Our starting point is a recently proposed aggregation-based algebraic multigrid method that combines the recursive static elimination of the vertices of degree 1 with the *degree-aware rooted aggregation* (DRA) algorithm. The latter always produces aggregates big enough to ensure that the preconditioner cost per iteration is low. Here we further improve the robustness of the method by controlling the quality of the aggregates. More precisely, "bad" vertices are removed from the aggregates formed by the DRA algorithm until a quality test is passed. This ensures that the two-grid condition number is nicely bounded, whereas the cost per iteration is kept low by reforming too small aggregates when it happens that the mean aggregate size is not large enough. The effectiveness and the robustness of the resulting method are assessed on a large set of undirected graphs by comparing with the variant without quality control, as well as with another state-of-the art graph Laplacian solver.

Key words. multigrid, algebraic multigrid, graph Laplacian, preconditioning, aggregation, convergence analysis

AMS subject classifications. 65F08, 65F10, 65N55, 65F50, 05C50

DOI. 10.1137/16M1071420

1. Introduction.

We consider large sparse linear systems

$$A\,\mathbf{u} \;=\; \mathbf{b}, \qquad \mathbf{b} \in \mathcal{R}(A), \tag{1}$$

where A is the Laplacian matrix of an undirected graph, that is, a singular matrix with nonpositive off-diagonal entries and zero row-sum. Fast solvers for such systems are a key to the efficiency of a number of numerical methods for graph problems, including graph partitioning problems, maximum flow and minimum cut problems, etc.; see [11, 20] for further examples.

As explained with more details in [14], developing a robust and efficient solver for this class of problems is a difficult task because of the diversity of the connectivity patterns. Sometimes, but not always, simple single-level preconditioners are effective, whereas standard algebraic multigrid (AMG) methods [6, 19, 22] work well for some graphs, especially mesh graphs [4], but may fail when applied to graphs with less regular connectivity pattern.

Of course, solvers specifically tailored for graph Laplacians [5, 9, 11, 21] incur less failure, and, in particular, the lean algebraic multigrid (LAMG) method from [11] seems to be state-of-the-art with respect to robustness. However, this is at the price of some heaviness that makes LAMG sometimes significantly slower than the conjugate gradient (CG) method with mere single-level preconditioning [14].

[*]Received by the editors April 19, 2016; accepted for publication (in revised form) February 7, 2017; published electronically October 26, 2017.
http://www.siam.org/journals/sisc/39-5/M107142.html
Funding: The first author's work was suppored by FNRS through grant J.0084.16. The second author is Research Director of the Fonds de la Recherche Scientifque – FNRS.

[†]Université Libre de Bruxelles, Service de Métrologie Nucléaire (C.P. 165-84), 50 Av. F.D. Roosevelt, B-1050 Brussels, Belgium (anapov@ulb.ac.be, homepages.ulb.ac.be/∼anapov, ynotay@ulb.ac.be, homepages.ulb.ac.be/∼ynotay).

In [14], we introduce an aggregation-based AMG method, which is often faster than LAMG, while being never significantly slower than a single-level preconditioning. It exploits a hierarchy of coarse systems which are obtained by combining a recursive static elimination of the vertices of degree 1 with a novel *degree-aware rooted aggregation* (DRA) algorithm. In the context of multigrid methods, the role of an aggregation algorithm is to group unknowns into disjoint sets called aggregates, each aggregate representing an unknown in the next coarser system. As a main feature, the DRA algorithm has been designed to produce large enough aggregates independently of the connectivity pattern; here, "large enough" means that the resulting method has low memory usage (typically a fraction of the memory required by the matrix itself), fast setup stage (a fraction of a second per million of nonzero entries in the matrix), and small solution cost per iteration.

However, the relatively "crude" aggregation provided by the DRA algorithm does not guarantee robustness, and, from that viewpoint, the results in [14] do not give full satisfaction. On the one hand, the solver is fairly competitive on a large set of graphs. On the other hand, the number of iterations is occasionally above 30, showing that there is room for improvement.

In the present work, we address this lack of robustness by combining DRA with the quality control along the lines of [12, 13]. In [12], a bound is proved that relates the two-grid convergence with *aggregates quality*, a measure that is defined for each aggregate and which is local in the sense that it requires only knowing the internal connectivity of the aggregate and the global weights of its external connections. In [13], this result is exploited to design a robust method via an aggregation algorithm that forms aggregates while verifying their quality. The procedure, based on successive pairwise aggregations, is, however, not appropriate for many graphs [14]. Hence, here we keep the idea of accepting only aggregates that satisfy a given quality criterion, but we consider it within the context of the DRA algorithm.

More precisely, the quality control in the DRA algorithm is implemented through extracting high-quality subsets from aggregates produced by the DRA algorithm, which are thus only tentative. This is achieved with filtering procedures that identify and remove "bad" vertices from the tentative aggregate until the quality test is passed. Now, such a filtering should be designed with care as the repeated quality testing can easily make the setup stage prohibitively expensive, especially when (as sometimes happens) the tentative aggregates issued by the DRA algorithm have several thousands of vertices. To keep the setup cost low, we therefore develop a proper methodology, partly motivated by theoretical results.

On the other hand, the filtering has of course a negative impact on the aggregates size. Nevertheless, it is harmless in many cases, whereas, to handle the other ones, we develop a heuristic strategy that enables us to keep most of the benefit of the quality control while ensuring that the mean aggregate size is big enough to preserve the attractive features of the method without quality control.

The potentialities of the novel method are demonstrated by numerical tests performed on a large set of graphs (the same as in [14]). The results show, among other things, that the method is indeed significantly more robust than the original version from [14], whereas the unavoidable increase in the solution cost per iteration and in the memory requirements is moderate. On the other hand, the increase in the setup time may be perceptible, but it is usually compensated by a faster solve stage.

The remainder of this paper is structured as follows. First, in what remains of this section, we recall the definition of a Laplacian matrix and set some notation.

The general framework of aggregation-based multigrid preconditioners as considered in this work is described in section 2. The relation between the convergence rate and the aggregates quality, further bounds on the quality measure, and some illustrative examples are presented in section 3. These results are used in section 4 to design the quality tests and the filtering procedures proposed to enhance the DRA algorithm. Numerical results are reported in section 5, and conclusions are drawn in section 6.

Graph Laplacians. Here, by graph Laplacian matrix we mean any symmetric matrix $A = (a_{ij})$ with nonnegative off-diagonal entries and zero row-sum (i.e., $A\mathbf{1} = \mathbf{0}$). Clearly, such a matrix is a singular symmetric M-matrix, hence positive semidefinite [3].

To every $n \times n$ graph Laplacian matrix A corresponds a weighted undirected graph $G = (V, E, w)$; here $V = \{1, \dots, n\}$ is the set of vertices, $E \subset V \times V$ is the set of edges, and $w : E \mapsto \mathbb{R}^+$ is a weight function. The weighted graph $G = (V, E, w)$ is *undirected* if $(i, j) \in E$ implies $(j, i) \in E$, and $w(i, j) = w(j, i)$ for every $(i, j) \in E$. A graph $G = (V, E, w)$ corresponds to a graph Laplacian matrix $A = (a_{ij})$ if

$$a_{ij} = \begin{cases} -w(i,j) & \text{if } (i,j) \in E \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

In what follows, we assume further that the diagonal entries of a graph Laplacian $A = (a_{ij})$ satisfy $a_{ii} > 0$. This is not a restrictive assumption, since the diagonal entry is zero only if the corresponding vertex is isolated; then the entries are zero for the whole row and column, and this row and column can be safely deleted from the matrix.

Notation. For any ordered set G, $|G|$ is its size and $G(i)$ is its ith element. For any matrix B, $\mathcal{N}(B)$ is its null space, and $\mathcal{R}(B)$ is its range. For any square matrix C, $\text{tril}(C)$ and $\text{triu}(C)$ are matrices corresponding, respectively, to its lower and upper triangular parts, including the diagonal.

2. Aggregation-based algebraic multigrid.

Multigrid methods correspond to the recursive application of a *two-grid* scheme. A two-grid scheme is a combination of *smoothing iterations* and a *coarse grid correction.* Smoothing iterations are stationary iterations with a simple single-level preconditioner. The coarse grid correction is based on solving a coarse representation of the problem with a reduced number of unknowns. This coarse system is solved exactly in a two-grid scheme, whereas actual *multigrid* schemes use an approximate solution obtained by applying (few iterations of) the same multigrid method, which then requires an even smaller coarse system, and so on, until the direct solution becomes possible at a low cost.

Here we consider multigrid methods based on aggregation and the symmetrized Gauss–Seidel smoothing. The corresponding building block—aggregation-based two-grid scheme—is described by Algorithm 1 below. The smoothing iteration amounts to a single forward Gauss–Seidel sweep as presmoother (see step 1 of Algorithm 1) and a single backward Gauss–Seidel sweep as postsmoother (see step 7). Regarding the coarse grid correction (steps 3–5), it is entirely determined by a partitioning of the set of unknowns $\{1, \dots, n\}$ into $n_c < n$ disjoint subsets G_i, $i = 1, \dots, n_c$, called aggregates. Every aggregate from the initial system then corresponds to an unknown of the coarse system. In the algorithm, the residual is first restricted on this coarse variable set (step 3); next, the corresponding $n_c \times n_c$ coarse system is solved (step 4); eventually, the coarse solution is prolongated on the initial variable set (step 5).

Algorithm 1. Application of two-grid preconditioner to $\mathbf{r} \in \mathcal{R}(A):\ \mathbf{v} = \mathcal{B}_{\mathrm{TG}}\,\mathbf{r}$.

```
1. Solve $L\mathbf{v}_1 = \mathbf{r}$, where $L = \mathrm{tril}(A)$    (presmoothing)
2. $\widetilde{\mathbf{r}} = \mathbf{r} - A\,\mathbf{v}_1$    (residual update)
3. Form $\widetilde{\mathbf{r}}_c$ such that $(\widetilde{\mathbf{r}}_c)_s = \sum_{i \in G_s} (\widetilde{\mathbf{r}})_i$, $s = 1, \dots, n_c$    (restriction)
4. Solve $A_c\mathbf{v}_c = \widetilde{\mathbf{r}}_c$    (coarse grid solution)
5. Form $\mathbf{v}_2$ such that $(\mathbf{v}_2)_i = (\mathbf{v}_c)_s$ for all $i \in G_s$ , $s = 1, \dots, n_c$    (prolongation)
6. $\bar{\mathbf{r}} = \widetilde{\mathbf{r}} - A\,\mathbf{v}_2$    (residual update)
7. Solve $U\mathbf{v}_3 = \bar{\mathbf{r}}$, where $U = \mathrm{triu}(A)$    (postsmoothing)
8. $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3$    (final result of the preconditioner application)
```

The entries of the coarse level matrix A_c are obtained by summation of the entries of the original matrix, namely,

$$(A_c)_{ij} = \sum_{k \in G_i} \sum_{\ell \in G_j} a_{k\ell}. \tag{2}$$

Note that this is also the Laplacian matrix of the coarse graph associated with the partitioning into aggregates (i.e., the graph whose vertices correspond to aggregates, and for which the weight of an edge (i, j) is given by the sum of the weights of all the edges connecting the vertices from G_i to those in G_j). In particular, this means that A_c is singular. However, one may check that if $\mathbf{r} \in \mathcal{R}(A)$ (as follows for any residual associated with a compatible system (1)), then the system to solve at step 4 is compatible; moreover, which particular solution is picked up actually does not influence the convergence (even with Krylov subspace acceleration) [17].

Let us now express the two-grid preconditioner in the matrix form. We first define the $n \times n_c$ prolongation matrix

$$(P)_{ij} = \begin{cases} 1 & \text{if } i \in G_j \\ 0 & \text{otherwise} \end{cases} \qquad (1 \le i \le n,\ 1 \le j \le n_c). \tag{3}$$

The restriction at step 3 of Algorithm 1 may then be written $\widetilde{\mathbf{r}}_c = P^T\,\widetilde{\mathbf{r}}$, whereas the prolongation at step 5 corresponds to $\mathbf{v}_2 = P\,\mathbf{v}_c$. On the other hand, a particular solution to $A_c\mathbf{v}_c = \widetilde{\mathbf{r}}_c$ (step 4) may be written $\mathbf{v}_c = A_c^g\,\widetilde{\mathbf{r}}_c$, where A_c^g is a pseudo-inverse of A_c satisfying $A_c\,A_c^g A_c = A_c$ [2]. Using further $D = \mathrm{diag}(A) = L + U - A$ for $L = \mathrm{tril}(A)$ and $U = \mathrm{triu}(A)$, one may then check that the preconditioner $\mathcal{B}_{\mathrm{TG}}$ defined by Algorithm 1 satisfies

$$\mathcal{B}_{\mathrm{TG}} = \left(L\,D^{-1}U\right)^{-1} + \left(I - U^{-1}A\right)\,P\,A_c^g\,P^T\,\left(I - A\,L^{-1}\right). \tag{4}$$

The idea behind the approach is easier to see considering the iteration matrix associated with stationary iterations, namely,

$$I - \mathcal{B}_{\mathrm{TG}}A = \left(I - U^{-1}A\right)\left(I - P\,A_c^g P^T\,A\right)\left(I - L^{-1}A\right). \tag{5}$$

Thus, one such stationary iteration combines the effect of the forward and backward Gauss–Seidel iterations with the coarse grid correction represented by the term $I - P\,A_c^g P^T\,A$.

A multigrid method is obtained by a recursive application of the two-grid scheme from Algorithm 1: at step 4, instead of the exact solution, one uses the approximation obtained by performing 1 or 2 iterations with the same two-grid method, but applied this time at the coarse level. The chosen iterative scheme defines the multigrid cycle:

the V-cycle is obtained with one stationary iteration, the W-cycle with two stationary iterations, and the K-cycle [18] with two iterations accelerated by a Krylov method.

Here we consider the K-cycle which is the standard choice for aggregation-based methods [16]. In the present setting this means that the coarse grid system at step 4 is solved with two iterations of the flexible CG method [15] (more precisely FCG(1)) preconditioned with the same aggregation-based two-grid scheme. Then, to a large extent, two-grid convergence properties carry over to the multigrid scheme [18], and, hence, the control of the two-grid convergence implies in practice the control of the multigrid convergence.

The above description corresponds to standard aggregation-based multigrid methods. The solver in [14] is different in that every time the Algorithm 1 needs to be applied to a system, the unknowns corresponding to degree-one vertices are recursively eliminated from this latter, and Algorithm 1 is then applied to the resulting Schur complement system. Note that if the matrix before elimination is a graph Laplacian, then the Schur complement matrix is also a graph Laplacian, and the corresponding graphs differ only by the set of the eliminated vertices and the edges adjacent to them. The degree-one elimination is beneficial for both the convergence and the cost per iteration of the resulting preconditioner, but the improvement is perceptible only for the limited set of graphs that have a large number of degree-one vertices.

3. Convergence estimates and aggregate quality.

In this section we develop the convergence analysis of two-grid preconditioners defined in Algorithm 1. First, we show in Theorem 3.2 below that the corresponding condition number can be bounded as a function of a measure of aggregates quality. The quality measure is then further analyzed in Theorem 3.3 and the subsequent examples.

Before stating these results, we need to recall in Theorem 3.1 below the sharp two-grid convergence estimate from [8, 23] and a related bound, as extended to positive semidefinite matrices in Theorems 3.4 and 3.5 of [17]. To alleviate the presentation, we restrict the corresponding theorem below to the two-grid schemes as described in the preceding section, with one forward Gauss–Seidel sweep as presmoother and one backward Gauss–Seidel sweep as postsmoother. We also assume

$$\mathcal{N}(A) \subset \mathcal{R}(P), \tag{6}$$

since this is always satisfied with the methods considered in this paper (see [17] for a more general formulation that does not require this assumption). Indeed, if the considered graph has only one connected component, there holds $\mathcal{N}(A) = \operatorname{span}(\mathbf{1})$, which is clearly in the range of a piecewise constant prolongation as defined by (3). On the other hand, if there are multiple connected components, the condition still holds provided that no aggregate has vertices from two or more components. This latter condition is satisfied with virtually all known aggregation algorithms, which always group vertices by following the edges which connect them. This is also why we further assume below that the subgraph associated to each aggregate is connected.

THEOREM 3.1. *Let A be an $n \times n$ symmetric positive (semi)definite matrix, and let P be an $n \times n_c$ matrix of rank $n_c < n$ satisfying* (6)*. Let $\mathbf{u}_k$ be the approximate solution obtained after k steps of the CG method applied to a compatible system* (1)*, using the preconditioner* (4) *defined by Algorithm* 1*.*

Then there holds

$$\|\mathbf{u} - \mathbf{u}_k\|_A \;\leq\; 2\left(\frac{\sqrt{\kappa_{\text{eff}}} - 1}{\sqrt{\kappa_{\text{eff}}} + 1}\right)^k \|\mathbf{u} - \mathbf{u}_0\|_A,$$

where

$$\kappa_{\rm eff} \;=\; \sup_{\substack{\mathbf{v}\in\mathbb{R}^n\\ \mathbf{v}\notin\mathcal{N}(A)}} \frac{\mathbf{v}^T X(I-P(P^TX\,P)^{-1}P^TX)\,\mathbf{v}}{\mathbf{v}^T A\,\mathbf{v}} \tag{7}$$

with $X = UD^{-1}L$, $U = \mathrm{triu}(A)$, $D = \mathrm{diag}(A)$, *and* $L = \mathrm{tril}(A)$. *Moreover,*

$$\kappa_{\rm eff} \;\le\; \sup_{\substack{\mathbf{v}\in\mathbb{R}^n\\ \mathbf{v}\notin\mathcal{N}(A)}} \frac{\mathbf{v}^T \overline{X}(I-P(P^T\overline{X}\,P)^{-1}P^T\overline{X})\,\mathbf{v}}{\mathbf{v}^T A\,\mathbf{v}} \tag{8}$$

for any matrix $\overline{X}$ *such that* $\overline{X}-X$ *is nonnegative definite.*

The convergence estimate from [8, 23] is exploited in [12] to bound the condition number of aggregation-based two-grid methods applied to symmetric matrices with nonpositive offdiagonal entries and nonnegative row-sum. This bound is equal to the maximum over all aggregates of an algebraic quantity that is associated to each aggregate and that can thus be used to characterize its quality. Clearly, avoiding aggregates of bad quality allows one then to control the two-grid condition number, and a related method for PDE problems is developed in [13].

The analysis in [12], however, is restricted to positive definite matrices and to smoothers based on preconditioners that are block diagonal with respect to the partitioning in aggregates. In the following theorem, we extend the approach to semidefinite matrices and Gauss–Seidel smoothing as implemented in Algorithm 1. Regarding terminology, the quantity $\mu(G)$ defined in the theorem below is referred to as the quality measure of an aggregate G.

Theorem 3.2. *Let A be an $n\times n$ symmetric matrix with nonpositive offdiagonal entries and nonnegative row-sum. Let P be defined via* (3)*, with G_i, $i=1,\dots,n_c$, being disjoint subsets whose union is $\{1,\dots,n\}$. Let, for $i=1,\dots,n_c$, $A|_{G_i}$ be the submatrix of A corresponding to the indices in G_i.*

For $i=1,\dots,n_c$, assume that the submatrix $A|_{G_i}$ is irreducible, and let the diagonal matrices Σ_{G_i} and Γ_{G_i} be such that, for $j=1,\dots,|G_i|$,

$$(\Sigma_{G_i})_{jj} \;=\; \sum_{k\notin G_i} |a_{G_i(j)k}| \quad\text{and}\quad (\Gamma_{G_i})_{jj} \;=\; \big((U-D)D^{-1}(L-D)\mathbf{1}\big)_{G_i(j)} + 2(\Sigma_{G_i})_{jj},$$

where $U=\mathrm{triu}(A)$, $D=\mathrm{diag}(A)$, and $L=\mathrm{tril}(A)$.

Then, $\kappa_{\rm eff}$ as defined in (7) *satisfies*

$$\kappa_{\rm eff} \;\le\; \max_{1\le i\le n_c} \mu(G_i),$$

where

$$\mu(G_i) \;=\; \begin{cases} 1 & \text{if } |G_i|=1,\\ \sup_{\substack{\mathbf{v}\in\mathbb{R}^{|G_i|}\\ \mathbf{v}\notin\mathcal{N}(A_{G_i})}} \dfrac{\mathbf{v}^T X_{G_i}(I-\mathbf{1}(\mathbf{1}^TX_{G_i}\mathbf{1})^{-1}\mathbf{1}^TX_{G_i})\,\mathbf{v}}{\mathbf{v}^T A_{G_i}\,\mathbf{v}} & \text{otherwise,}\end{cases} \tag{9}$$

with $A_{G_i} = A|_{G_i} - \Sigma_{G_i}$ and $X_{G_i} = A_{G_i} + \Gamma_{G_i}$.

Proof. First, the null space of A contains at most the vectors that are constant over each connected component. Then, (6) holds for a prolongation matrix defined by (3) providing that no aggregate has vertices from two or more components, which is ensured by the assumption that all submatrices $A|_{G_i}$ are irreducible.

Then, the stated result is a straightforward corollary of (8) and Theorem 3.2 in [12] if we can show that

$$A - \text{blockdiag}(A_{G_i}) \qquad \text{and} \qquad \text{blockdiag}(X_{G_i}) - X$$

are both nonnegative definite. (The result in [12] is given for a symmetric positive definite matrix A, but its extension to a symmetric semipositive definite A is straightforward.)

Now, the definition of A_{G_i} implies that $A - \text{blockdiag}(A_{G_i})$ has zero row-sum and nonnegative offdiagonal entries (equal to a_{ij} if i and j do not belong to the same aggregate and zero otherwise); hence it is weakly diagonally dominant and therefore nonnegative definite.

On the other hand, the matrix

$$\text{blockdiag}(A|_{G_i} + \Sigma_{G_i}) - A \;=\; \text{blockdiag}(A_{G_i} + 2\,\Sigma_{G_i}) - A \tag{10}$$

is also weakly diagonally dominant. Further, since

$$X = UD^{-1}L = A + (U-D)D^{-1}(L-D),$$

the matrix

$$A + \text{diag}((U-D)D^{-1}(L-D)\mathbf{1}) - X = \text{diag}((U-D)D^{-1}(L-D)\mathbf{1}) - (U-D)D^{-1}(L-D)$$

has nonpositive offdiagonal entries and zero row-sums; i.e., it is weakly diagonally dominant as well. Summing with (10) then shows that

$$\text{blockdiag}(A_{G_i} + 2\,\Sigma_{G_i}) + \text{diag}\big((U-D)D^{-1}(L-D)\mathbf{1}\big) - X \;=\; \text{blockdiag}(X_{G_i}) - X$$

is nonnegative definite as claimed. □

We now take a closer look at the quality measure of an individual aggregate. In the particular case of a graph Laplacian, the matrix A has nonpositive offdiagonal entries and zero row-sum, which entails that the submatrices A_{G_i} as defined in Theorem 3.2 also have nonpositive offdiagonal entries and zero row-sum; that is, A_{G_i} is the Laplacian matrix of the subgraph associated with G_i. The assumption that A_{G_i} is irreducible amounts thus to the assumption that this subgraph is connected, which is naturally satisfied with known aggregation schemes. This assumption guarantees that the constant vector is the only kernel mode of A_{G_i} and hence, as shown in [12, Theorem 3.2], that μ_{G_i} is finite. This latter property is also clear from the following theorem, where we further analyze the quality indicator $\mu(G)$ in the case where A_G has zero row-sum; note that n_g in this theorem is the size of the matrix A_G and plays thus the role of the aggregate size $|G|$ in Theorem 3.2.

THEOREM 3.3. *Let $A_G = (a_{ij})$ be an irreducible symmetric $n_g \times n_g$ matrix with nonpositive offdiagonal entries and zero row-sum, and let $\Gamma_G = \text{diag}(\gamma_j)$ be a nonnegative $n_g \times n_g$ diagonal matrix. Assuming $n_g > 1$ and letting $X_G = A_G + \Gamma_G$, the quantity*

$$\mu(G) \;=\; \sup_{\substack{\mathbf{v}\in\mathbb{R}^{n_g} \\ \mathbf{v}\notin\mathcal{N}(A_G)}} \frac{\mathbf{v}^T X_G \left(I - \mathbf{1}\left(\mathbf{1}^T X_G \mathbf{1}\right)^{-1} \mathbf{1}^T X_G\right) \mathbf{v}}{\mathbf{v}^T A_G\, \mathbf{v}}$$

satisfies $\mu(G) = 1$ *if* Γ_G *is the zero matrix and, otherwise,*

$$\mu(G) \;=\; 1 + \min_{\mathbf{w}\in\mathbb{R}^{n_g}} \max_{\substack{\mathbf{v}\in\mathbb{R}^{n_g} \\ \mathbf{v}\perp\mathbf{w}}} \frac{\mathbf{v}^T\,\Gamma_G\,\mathbf{v}}{\mathbf{v}^T\,A_G\,\mathbf{v}} \;=\; 1 + \max_{\substack{\mathbf{v}\in\mathbb{R}^{n_g} \\ \mathbf{v}\perp\Gamma_G\,\mathbf{1}}} \frac{\mathbf{v}^T\,\Gamma_G\,\mathbf{v}}{\mathbf{v}^T\,A_G\,\mathbf{v}}. \tag{11}$$

Moreover,

$$1 + \max_{1\le j\le n_g} \frac{\gamma_j}{\sum_{\substack{k=1\\k\ne j}}^{n_g} |a_{jk}|}\left(1 - \frac{\gamma_j}{\sum_{k=1}^{n_g}\gamma_k}\right) \;\le\; \mu(G) \;\le\; 1 + \min_{1\le i\le n_g}\left(\max_{\substack{1\le j\le n_g\\ j\ne i}} \frac{\gamma_j}{|a_{ji}|}\right). \tag{12}$$

Proof. Because A_G is an irreducible M-matrix with zero row-sum, it is singular with a single kernel vector equal to $\mathbf{1}$. Then, we know by [12, Theorem 3.4] that $\mu(G)$ coincides with the inverse of the second smallest eigenvalue of the generalized eigenvalue problem

$$A_G\,\mathbf{z} \;=\; \lambda\,X_G\,\mathbf{z}.$$

Since the eigenvector associated with the smallest eigenvalue (i.e., the zero eigenvalue) is $\mathbf{1}$, the Courant–Fisher characterization of the second smallest eigenvalue then yields

$$\big(\mu(G)\big)^{-1} \;=\; \max_{\mathbf{w}\in\mathbb{R}^{n_g}} \min_{\substack{\mathbf{v}\in\mathbb{R}^{n_g}\\ \mathbf{v}\perp\mathbf{w}}} \frac{\mathbf{v}^T\,A_G\,\mathbf{v}}{\mathbf{v}^T\,X_G\,\mathbf{v}} \;=\; \min_{\substack{\mathbf{v}\in\mathbb{R}^{n_g}\\ \mathbf{v}\perp X_G\,\mathbf{1}}} \frac{\mathbf{v}^T\,A_G\,\mathbf{v}}{\mathbf{v}^T\,X_G\,\mathbf{v}}.$$

The equalities (11) follow using $X_G = A_G + \Gamma_G$ and $X_G\,\mathbf{1} = \Gamma_G\,\mathbf{1}$.

Now, let i be any index in $[1, n_g]$, and let $\mathbf{e}_i$ be the ith unit vector, i.e., the vector satisfying $(\mathbf{e}_i)_k = \delta_{ik}$. Considering the middle term of (11) with $\mathbf{w} = \mathbf{e}_i$, one sees that

$$\mu(G) \;\le\; 1 + \max_{\mathbf{v}\ne\mathbf{0}} \frac{\mathbf{v}^T\,\Gamma_G^{(i)}\,\mathbf{v}}{\mathbf{v}^T\,A_G^{(i)}\,\mathbf{v}}, \tag{13}$$

where $\Gamma_G^{(i)}$ and $A_G^{(i)}$ are the submatrices obtained by deleting the ith row and the ith column of Γ_G and A_G, respectively. Since the row-sum of the jth row of $A_G^{(i)}$ is now given by $|a_{ji}|$, one further has

$$\max_{\mathbf{v}\ne\mathbf{0}} \frac{\mathbf{v}^T\,\Gamma_G^{(i)}\,\mathbf{v}}{\mathbf{v}^T\,A_G^{(i)}\,\mathbf{v}} \;\le\; \max_{\mathbf{v}\ne\mathbf{0}} \frac{\mathbf{v}^T\,\Gamma_G^{(i)}\,\mathbf{v}}{\mathbf{v}^T\,\mathrm{diag}_{j\ne i}(|a_{ji}|)\,\mathbf{v}} \;=\; \max_{\substack{1\le j\le n_g\\ j\ne i}} \frac{\gamma_j}{|a_{ji}|},$$

which gives the upper bound (12).

To prove the lower bound, consider $\mathbf{v} = \mathbf{e}_j - \alpha\,\mathbf{1}$ with coefficient α such that the orthogonality condition $\mathbf{v}\perp\Gamma_G\mathbf{1}$ holds; that is, $\alpha = (\mathbf{1}^T\,\Gamma_G\,\mathbf{e}_j)/(\mathbf{1}^T\,\Gamma_G\,\mathbf{1})$. Since $A_G\mathbf{1} = \mathbf{0}$, one has, using the zero row-sum condition,

$$\mathbf{v}^T\,A_G\,\mathbf{v} \;=\; \mathbf{e}_j^T\,A_G\,\mathbf{e}_j \;=\; (A_G)_{jj} \;=\; \sum_{\substack{k=1\\k\ne j}}^{n_g} |a_{jk}|.$$

On the other hand, the orthogonality condition implies

$$\mathbf{v}^T\,\Gamma_G\,\mathbf{v} \;=\; \mathbf{e}_j^T\,\Gamma_G\,\mathbf{e}_j - \alpha\;\mathbf{1}^T\,\Gamma_G\,\mathbf{e}_j \;=\; \gamma_j - \frac{\left(\mathbf{1}^T\,\Gamma_G\,\mathbf{e}_j\right)^2}{\mathbf{1}^T\,\Gamma_G\,\mathbf{1}} \;=\; \gamma_j - \frac{\gamma_j^2}{\sum_k \gamma_k}.$$

Using these relations together with the right equality (11) yields the required result. ∎

In what remains of this section, we illustrate the above theorem with two examples. These show, among other results, that both the upper and lower bounds in (12) can be sharp. In both examples, we assume that all γ_i are equal for the sake of simplicity. More general results are easily derived with the help of (11), which implies that $\mu(G)$ for the case where $\gamma_i = \gamma$ for all i provides an upper bound for the general case setting $\gamma = \max_i \gamma_i$ and a lower bound setting $\gamma = \min_i \gamma_i$.

Example 3.1. In this example the subgraph of the aggregate is a clique and all weights are equal to a; that is, all offdiagonal entries are equal to $-a$ and

$$A_G \;=\; a\left(n_g\, I - \mathbf{1}\,\mathbf{1}^T\right).$$

With $\Gamma_G = \gamma\, I$, (11) implies $\mu(G) = 1+\gamma/(a\, n_g)$. Moreover, this value coincides with the lower bound (12), since $\sum_{k\neq j} |a_{jk}| = a(n_g-1)$, whereas $(n_g-1)^{-1}(1-n_g^{-1}) = n_g^{-1}$. Thus, in this example, the inequalities (12) become

$$\textit{lower bound} \;=\; \mu(G) \;=\; 1+\frac{\gamma}{a\, n_g} \;\leq\; 1+\frac{\gamma}{a} \;=\; \textit{upper bound.}$$

Example 3.2. In this example, we consider the aggregate with n_1+n_2+1 vertices and

$$A_G \;=\; a\begin{pmatrix} n_1+n_2 & -1 & \dots & -1 & -1 & -1 & \dots & -1 \\ -1 & n_1 & -1 & \dots & -1 & 0 & \dots & 0 \\ -1 & -1 & n_1 & & -1 & 0 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots & \vdots & \dots & \vdots \\ -1 & -1 & \dots & -1 & n_1 & 0 & \dots & 0 \\ -1 & 0 & \dots & \dots & 0 & n_2 & -1 & -1 \\ \vdots & \vdots & \dots & \dots & \vdots & -1 & \ddots & -1 \\ -1 & 0 & \dots & \dots & 0 & -1 & -1 & n_2 \end{pmatrix};$$

that is, the aggregate is the union of two cliques of size $n_1+1 \geq 2$ and $n_2+1 \geq 2$, with exactly one common vertex and no other interconnection between the cliques.

Let $\mathbf{1}_1$ be the vector such that

$$(\mathbf{1}_1)_i \;=\; \begin{cases} 1 & \text{if } i \text{ belongs to clique 1,} \\ 0 & \text{otherwise,} \end{cases}$$

and $I_1 = \operatorname{diag}(\mathbf{1}_1)$, and define similarly $\mathbf{1}_2$ and I_2. The matrix A_G can be written

$$A_G \;=\; a\big((n_1+1)I_1 - \mathbf{1}_1\,\mathbf{1}_1^T + (n_2+1)I_2 - \mathbf{1}_2\,\mathbf{1}_2^T\big).$$

Then, consider the right equality (11) with trial vector

$$\mathbf{v} \;=\; \alpha_1\,(\mathbf{1}_1-\mathbf{e}_1) + \alpha_2\,(\mathbf{1}_2-\mathbf{e}_1)\,,$$

where $\mathbf{e}_1$ is the unit vector corresponding to the first vertex, which is the common vertex of both cliques. Choosing α_1, α_2 such that the condition $\mathbf{v}\perp\Gamma_G\,\mathbf{1}$ holds, the associate ratio $\mathbf{v}^T\,\Gamma_G\,\mathbf{v}/\mathbf{v}^T\,A_G\,\mathbf{v}$ provides a lower bound on $(\mu(G)-1)$. One finds

$$\begin{aligned}\mathbf{v}^T A_G\,\mathbf{v} &= \alpha_1^2\;(\mathbf{1}_1-\mathbf{e}_1)^T A_G\;(\mathbf{1}_1-\mathbf{e}_1) + \alpha_2^2\;(\mathbf{1}_2-\mathbf{e}_1)^T A_G\;(\mathbf{1}_2-\mathbf{e}_1) \\ &= a\,\left(\alpha_1^2\, n_1 + \alpha_2^2\, n_2\right),\end{aligned}$$

whereas, with $\Gamma_G = \gamma\, I$,

$$\mathbf{v}^T\, \Gamma_G\, \mathbf{v} \;=\; \gamma\, \left(\alpha_1^2\, n_1 + \alpha_2^2\, n_2\right).$$

Therefore, the associate lower bound on $\mu(G)$ is $1+\gamma/a$. Since it coincides with the upper bound (12), it necessarily gives the exact value. For this example, (12) becomes then

$$\begin{aligned} \textit{lower bound} &= 1 + \frac{\gamma}{a\, \min(n_1, n_2)} \left(1 - \frac{1}{n_1 + n_2 - 1}\right) \le\; \mu(G) \\ &= 1 + \frac{\gamma}{a} = \textit{upper bound}. \end{aligned}$$

If n_g is relatively large in the first example, and if n_1, n_2 are relatively large in the second example, one sees that in both cases the lower bound is significantly smaller than the upper bound. One could then be afraid that none of the bounds is really useful. However, in the next section, we show how the combined use of both bounds can help to design a procedure that builds aggregates of required quality at affordable cost.

4. DRA with quality control.

In this section, we first recall the DRA algorithm (section 4.1). Then (section 4.2), we consider how to inexpensively check whether the quality measure of a given aggregate is below a given threshold. Next (section 4.3), we explain how, starting from a tentative aggregate—as, e.g., produced by the DRA algorithm—one can progressively remove "bad" vertices until the quality test is passed. A heuristic strategy to increase mean aggregate size is further discussed (section 4.4), and the section is concluded with a global overview of the proposed approach (section 4.5).

4.1. DRA algorithm.

The DRA scheme [14], whose algorithm is recalled below, proceeds as follows. First, the vertices are ordered in decreasing order of their degree. More precisely, to avoid excessive complexity, only a partial sort is made based on the rounded value (toward smallest integer) of the logarithm of the degree in base 2. Then, at each step, a *root* vertex is picked among the unaggregated vertices, using the so defined preordering. An associated aggregate is next formed with this root, its unaggregated neighbors, and, if the so formed aggregate is too small, the unaggregated neighbors of these neighbors. If there remain unaggregated vertices, one proceeds with the next step, and so on until all the vertices are aggregated.

Algorithm 2. DRA.

```
Input:   $A = (a_{ij})$                        % graph Laplacian matrix
Output:  $n_c$                                 % number of aggregates
         $G_i$, $i = 1, \dots, n_c$            % aggregates

1.  Compute floor[$\log_2$(degree(i))] for all vertices $i$, $i = 1, \dots, n$
2.  $n_c = 0$
3.  while  there are vertices outside $\cup_{s=1}^{n_c} G_s$
4.      Select as root a vertex $r \notin \cup_{s=1}^{n_c} G_s$ with maximal value of floor[$\log_2$(degree(r))]
5.      $n_c \leftarrow n_c + 1$ and $G_{n_c} = \{r\} \cup \{j \notin \cup_{s=1}^{n_c - 1} G_s \mid a_{rj} \neq 0\}$
6.      if  $|G_{n_c}| \le 6$
          $G_{n_c} \leftarrow G_{n_c} \cup \{j \notin \cup_{s=1}^{n_c - 1} G_s \mid \exists k \in G_{n_c} : a_{kj} \neq 0\}$
        end if
7.  end while
```

The results in [14] show that in practice this algorithm always produces aggregates big enough to ensure a low computational complexity for the associated multigrid algorithm. More precisely, low computational complexity is guaranteed if the number of nonzero elements of the coarse grid matrix decreases at least by a factor 3 from one level to the next (see section 4.4 for an explanation); this is typically ensured if the average aggregate size is at least 4. The latter requirement motivates step 6 of the algorithm, which expands small root-and-neighbors aggregates.[1]

Now, this algorithm does not control the quality of the aggregates, which, as seen in [14], sometimes leads to relatively large two-grid condition numbers.

4.2. Quality tests. Computing $\mu(G)$ for a given (tentative) aggregate is too costly to be done repeatedly within the setup phase of an AMG method. Fortunately, as observed in [13], an explicit computation is not needed to check that $\mu(G)$ is below a given threshold $\overline{\kappa}$. It suffices to verify whether the matrix

$$Z_G \;=\; \overline{\kappa}\, A_G - X_G \left(I - \mathbf{1} \left(\mathbf{1}^T X_G \mathbf{1} \right)^{-1} \mathbf{1}^T X_G \right) \tag{14}$$

is nonnegative definite, which can be done performing a Cholesky factorization. More precisely, if A_G has zero row-sum (as always in this work), then Z_G has also zero row-sum (and therefore is singular). Then, the last pivot of the Cholesky factorization is always zero (in exact arithmetic), and Z_G is nonnegative definite if and only if the factorization of any principal submatrix of size $|G|-1$ yields no negative pivot.

We may further use the upper bound in Theorem 3.3 as a sufficient condition. Let r be the index of the root vertex of G and let, for all j in G,

$$\delta_j \;=\; \left((U-D)D^{-1}(L-D)\mathbf{1} \right)_j , \tag{15}$$

with, as before, $U = \operatorname{triu}(A)$, $D = \operatorname{diag}(A)$ and $L = \operatorname{tril}(A)$. From the right inequality (12) (considered with $i = r$) and the definition of $\Gamma_G = \operatorname{diag}(\gamma_j)$ in Theorem 3.2 one sees that one will have $\mu(G) < \overline{\kappa}$ if

$$\frac{2 \sum_{k \notin G} |a_{jk}| + \delta_j}{|a_{jr}|} \;\le\; \overline{\kappa} - 1 \qquad \forall\, j \in G \backslash \{r\}. \tag{16}$$

This criterion is useful in two ways. First, when it is met, one may skip the factorization of the matrix (14) and hence save some computing time. Next, and more importantly, in some graphs, there are vertices with thousands of neighbors, for which the DRA algorithm will produce huge tentative aggregates (remember that vertices of high degree have priority for being selected as root vertex). Because the cost of the factorization of the matrix (14) is $\mathcal{O}(|G|^3)$, we can in fact not afford it when $|G|$ is too large. Then the only viable way to guarantee aggregate quality is to use (16) as a necessary condition, although we know that it is only a sufficient one.

In practice, we observed that the impact of the factorization on the overall computing time was significant beyond the size of 1024. Hence, when $|G| > 1024$, a tentative aggregate will be considered passing the quality test only if (16) holds (regardless Z_G), whereas if $|G| \le 1024$, the main criterion is based on the factorization of the matrix Z_G in (14) (which is skipped only if (16) turns out to be satisfied).

[1]The expansion size threshold is set to 6, this value being observed in [14] to deliver on average the best trade-off between larger aggregates obtained with larger thresholds and the better average quality that typically results when using smaller thresholds.

Note that (16) can be passed only if all vertices are connected to the root vertex (otherwise, some of the $|a_{jr}|$ are zero). This is usually the case for large aggregates produced by the DRA algorithm, since step 6 in Algorithm 2 is skipped when the root vertex has more than five unaggregated neighbors.

4.3. Aggregate filtering. The probability that an aggregate directly produced by Algorithm 2 passes the quality test is relatively low. However, in most of the cases that we encountered, it turns out to be possible to extract a reasonably large subset of vertices that form an aggregate of acceptable quality. Here we present the two tools that we use to identify such subset of vertices in a given tentative aggregate.

The first tool, presented in section 4.3.1, detects and removes "bad" vertices, that is, vertices that fail to satisfy given criteria based on the two-sided bound (12). This tool is systematically applied to every aggregate produced by the DRA algorithm, before testing explicitly for quality as explained in the preceding subsection.

The second tool, presented in section 4.3.2, is applied when the aggregate fails the explicit quality test based on (14). It selects a subgroup of vertices in a way that allows one to properly handle the situations like in Example 3.2.

4.3.1. Bad vertices removal. Our approach is inspired by the two-sided bound (12). It is better explained by rewriting these bounds for the context of Theorem 3.2 (as already done for (16) before), with i being again set to the root index r of the tentative aggregate G. This gives

$$\text{(17)}\quad \forall j \in G\backslash\{r\} : \quad 1 + \frac{2\sum_{k\notin G}|a_{jk}| + \delta_j}{\sum_{\substack{k\in G\\ k\neq j}}|a_{jk}|}\,\xi_j \;\le\; \mu(G) \;\le\; 1 + \frac{2\sum_{k\notin G}|a_{jk}| + \delta_j}{|a_{jr}|},$$

where

$$\delta_j \;=\; \left((U-D)D^{-1}(L-D)\mathbf{1}\right)_j \qquad \text{and} \qquad \xi_j \;=\; 1 - \frac{2\sum_{\ell\notin G}|a_{j\ell}| + \delta_j}{\sum_{k\in G}\left(2\sum_{\ell\notin G}|a_{k\ell}| + \delta_k\right)}.$$

Based on this we decide that a vertex $j \neq r$ should be kept in G if

$$\text{(18)}\qquad \frac{2\sum_{k\notin G}|a_{jk}| + \delta_j}{|a_{jr}|} \;\le\; \overline{\kappa} - 1$$

or

$$\text{(19)}\qquad \frac{2\sum_{k\notin G}|a_{jk}| + \delta_j}{\sum_{\substack{k\in G\\ k\neq j}}|a_{jk}|} \;\le\; \frac{\overline{\kappa}-1}{\eta} \qquad \text{and} \qquad |G| \;\le\; 1024,$$

where $\eta > 1$ is a security factor (we typically use $\eta = 2$). In other words, only the criterion (18) applies for aggregates larger than 1024, while vertices in smaller aggregates are accepted if they satisfy either (18) or (19).

The rationale for this is as follows. If all vertices are accepted thanks to the first condition (18), the inequality (16) holds, and hence the aggregate passes the quality test. However, in view of Example 3.1, it would be too restrictive to use this sole condition, at least when $|G| \le 1024$, so that the aggregate has a chance to pass the quality test when (16) does not hold. This motivates the second criterion (19), which disregards the upper bound, but tends to ensure that the lower bound on $\mu(G)$ is significantly below the largest admissible value (we neglect the factor ξ_j, the approach being heuristic, anyway). Observe also that, apart from the term δ_j, this criterion is essentially a constraint on the ratio

$$\frac{\sum_{k\notin G} |a_{jk}|}{\sum_{\substack{k\in G\\ k\neq j}} |a_{jk}|}, \tag{20}$$

that is, on the sum of the weights of external connections divided by the sum of the weights of internal connections. Intuitively, the smaller this ratio, the larger the probability that the vertex is a "good" aggregate member.

From a practical viewpoint, it is worth noting that the removal of a vertex from G possibly implies, for the vertices j still in G , an increase of $\sum_{k\notin G} |a_{jk}|$ and decrease of $\sum_{k\in G, k\neq j} |a_{jk}|$. This negatively affects the satisfaction of the criteria (18) and (19). To take this into account, we continuously sweep through the vertices of an aggregate, assessing (18) and (19) based on the most recent tentative aggregate G, and stopping only once a complete sweep has been achieved without any removal. In practice, we observed that the number of times each vertex is visited during this procedure is most often between 2 and 4.

4.3.2. Subgroup extraction. Here we consider the treatment of tentative aggregates issued from the just exposed removal procedure but that nevertheless fail the quality test based on the factorization of (14). Such aggregates necessarily contain vertices accepted thanks to (19). Hence a further filtering is possible by increasing the security factor η to make this criterion more stringent. However, in view of situations like that of Example 3.2, this is not always sufficient.

This motivates the subgroup extraction technique presented here, which exploits the fact that the cases we are interested in are those where the Cholesky factorization of Z_G in (14) has broken down. As a by-product of the factorization, one can inexpensively retrieve a vector $\mathbf{w}$ such that

$$\mathbf{w}^T Z_G \mathbf{w} \;=\; p < 0,$$

where p is the encountered negative pivot. Because Z_G is symmetric and $Z_G \mathbf{1} = \mathbf{0}$, we have as well

$$(\mathbf{w} + \alpha \mathbf{1})^T Z_G (\mathbf{w} + \alpha \mathbf{1}) \;=\; p$$

for any α, thus also for $\mathbf{v} = \mathbf{w} + \alpha \mathbf{1}$ satisfying $\mathbf{1}^T X_G \mathbf{v} = 0$. For this specific vector we have (see (14))

$$p \;=\; \mathbf{v}^T Z_G \mathbf{v} \;=\; \overline{\kappa}\, \mathbf{v}^T A_G \mathbf{v} - \mathbf{v}^T X_G \mathbf{v} \;=\; (\overline{\kappa} - 1)\, \mathbf{v}^T A_G \mathbf{v} - \mathbf{v}^T \Gamma_G \mathbf{v},$$

where $\Gamma_G = X_G - A_G$. Hence,

$$1 + \frac{\mathbf{v}^T \Gamma_G \mathbf{v}}{\mathbf{v}^T A_G \mathbf{v}} \;=\; \overline{\kappa} - \frac{p}{\mathbf{v}^T A_G \mathbf{v}} \;>\; \overline{\kappa}.$$

Remembering that $\mathbf{1}^T X_G \mathbf{v} = 0$ and, therefore, $\mathbf{1}^T \Gamma_G \mathbf{v} = 0$ (here $\mathbf{1}^T\Gamma_G$ is always nonzero if $\mu(G) > 1$), the vector $\mathbf{v}$ may be seen as an approximation of $\arg\max_{\mathbf{v}\,\perp\,\Gamma_G \mathbf{1}} (\mathbf{v}^T \Gamma_G \mathbf{v} / \mathbf{v}^T A_G \mathbf{v})$ which provides the exact value of $\mu(G)$; see (11). On the other hand, as noted in the proof of Theorem 3.3, this latter is the eigenvector associated with the second smallest eigenvalue of the generalized eigenvalue problem

$$A_G\, \mathbf{z} \;=\; \lambda\, X_G\, \mathbf{z}. \tag{21}$$

This eigenvector may be seen as a generalized Fielder vector. Since the sign of the entries in the Fiedler vector is often used for graph partitioning, this motivates us in discriminating aggregate members based on the sign of the corresponding entry in $\mathbf{v}$. More precisely, we proceed as follows. We preselect as remaining aggregate members all vertices for which the component in $\mathbf{v}$ has the same sign as that of the root vertex. Letting G_p be the set of preselected vertices, we then effectively keep in the tentative aggregate G the vertices other than root such that

$$\frac{2\sum_{\substack{k\notin G_p \\ k\neq j}} |a_{jk}| + \delta_j}{|a_{jr}|} \leq \overline{\kappa} - 1 \qquad \text{or} \qquad \frac{2\sum_{\substack{k\notin G_p \\ k\neq j}} |a_{jk}| + \delta_j}{\sum_{\substack{k\in G_p \\ k\neq j}} |a_{jk}|} \leq \frac{\overline{\kappa}-1}{\eta}. \tag{22}$$

Note that, in this way, the vertices not initially preselected (i.e., those in $G\backslash G_p$) can finally stay in the tentative aggregate; i.e., we avoid a too crude decision based on the sole signs in $\mathbf{v}$.

Once this new tentative aggregate has been obtained, it will undergo the bad vertices removal procedure described in the preceding subsection, before a new quality test is performed. It may then reenter the procedure described here only in case of a new factorization failure, which turns out to be extremely rare.

The following example illustrates how the subgroup extraction works and can successfully face situations like that of Example 3.2.

Example 4.1. We consider a particular instance of Example 3.2, with $n_1 = 10$ and $n_2 = 12$, except that some randomly chosen edges have been removed; see Figure 1, left. To make the case more realistic, a random reordering of the vertices is applied, and the connectivity pattern of the tentative aggregate is as illustrated in Figure 1, right. All weights are equal to one, and the diagonal of Γ_G is filled with random numbers uniformly distributed in $(0, 25)$:

$$\left(2\sum_{k\notin G} |a_{jk}| + \delta_j\right)_{j=1,\dots,21} = \begin{pmatrix} 7.4 & 13.3 & 4.8 & 1.7 & 19.7 & 16.4 & 15.9 & 14.4 & 1.0 & 8.9 & 23.6 \\ 1.5 & 21.6 & 21.9 & 1.3 & 16.3 & 13.8 & 14.9 & 12.1 & 7.1 & 7.4 \end{pmatrix}.$$

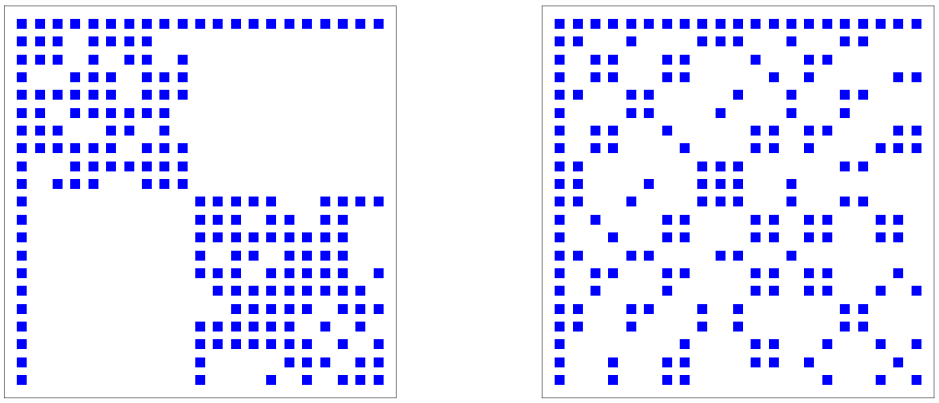

FIG. 1. *Sparsity pattern of the Laplacian matrix A_G associated to the aggregate G from Example 4.1; left and right figures represent the pattern before and after a random reordering of the vertices, respectively.*

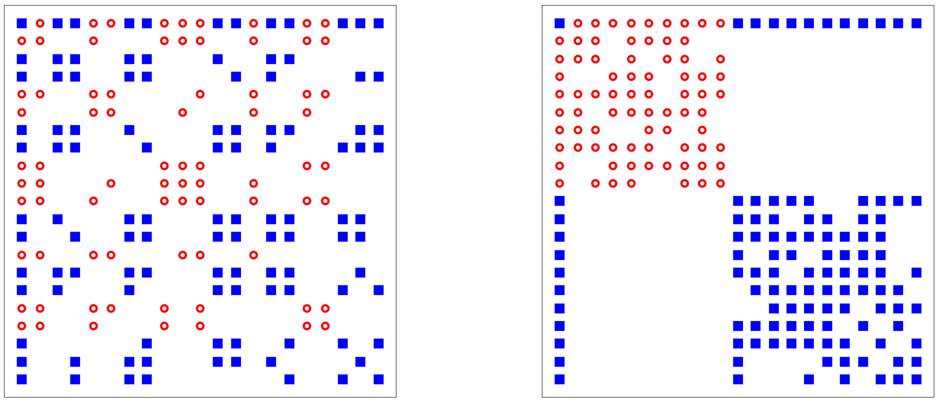

Fig. 2. *Same as Figures* 1 *(in reversed order), but with entries corresponding to* G_p *marked by red circles.*

It turns out that the largest value for the ratio in the left-hand side of (19) is 3.7, which is below $(\overline{\kappa}-1)/\eta$ for the standard values $\overline{\kappa}=10$ and $\eta=2$.

Hence, the aggregate passes the "bad vertices removal" procedure described in section 4.3.1 without any vertex removed. However, the factorization of the matrix (14) yields a negative pivot. The corresponding vector $\mathbf{v}$ is

$$\mathbf{v} = \Big(-0.24 \;\; 0.88 \;\; -1.16 \;\; -1.09 \;\; 1.09 \;\; 1.29 \;\; -1.32 \;\; -1.24 \;\; 0.60 \;\; 0.99 \;\; 1.04 \\ -1.12 \;\; -1.44 \;\; 1.42 \;\; -1.13 \;\; -1.36 \;\; 0.85 \;\; 0.01 \;\; -1.00 \;\; -1.00 \;\; -1.00 \Big).$$

One may check that it satisfies the proper orthogonality condition. On the other hand,

$$1+\frac{\mathbf{v}^T\,\Gamma_G\,\mathbf{v}}{\mathbf{v}^T\,A_G\,\mathbf{v}} \;=\; 11.2,$$

whereas a numerical computation reveals that $\mu(G)=14.4$. Moreover, $\mathbf{v}$ is relatively close to the eigenvector $\mathbf{z}_2$ associated with the second smallest eigenvalue of (21), since we obtain

$$\cos(\mathbf{v},\,\mathbf{z}_2) \;=\; 0.97.$$

Further, the preselection based on the sign of the entries in $\mathbf{v}$ yields the result illustrated in Figure 2, left. In fact, G_p is precisely the set of vertices belonging to the second group of vertices, as is better seen looking at Figure 2, right.

It follows that the test (22) yields the same discrimination as the test based on the signs in $\mathbf{v}$, hence G_p is also the new tentative aggregate we restart the bad vertices removal procedure with. Moreover, all vertices satisfy (19) (the values of the ratio (20) are unchanged from the previous step), whereas the factorization test is here successful. Hence, G_p is the finally accepted aggregate. A further numerical computation reveals that $\mu(G_p)=4.2$, which is indeed below $\overline{\kappa}=10$ and about 3.5 times smaller than the initial value $\mu(G)=14.4$.

4.4. Enhancing complexity. As stated in the introduction (see also section 4.1), one of the attractive features of the DRA algorithm lies in its ability to produce "large enough" aggregates independently of the connectivity pattern of the graph. Clearly, the quality control may have a negative impact on aggregate size, which needs to be investigated before going further.

To develop the discussion, it is useful to recall how aggregate size is related to the complexity of the multigrid algorithm. Because we use the K-cycle, the work per iteration and the memory requirements are controlled, respectively, through the weighted and operator complexities [13, 14], defined by

$$\mathcal{C}_W \;=\; 1+\sum_{\ell=2}^{L} 2^{\ell-1}\,\frac{nnz(A_\ell)}{nnz(A)} \quad \text{and} \quad \mathcal{C}_A \;=\; 1+\sum_{\ell=2}^{L} \frac{nnz(A_\ell)}{nnz(A)}, \tag{23}$$

where $A_1 = A$ and where $A_2, \ldots, A_L$ are the successive coarse grid matrices, L being the number of levels. More precisely, $\mathcal{C}_W$ indicates the cost of the preconditioner application relative to the cost of smoothing iterations at fine grid level, whereas $\mathcal{C}_A$ provides the memory needed to store the preconditioner relative to that needed for the system matrix A. Note that in our case the cost of smoothing iterations at the fine grid level, including the residual updates at steps 2 and 6 of Algorithm 1, is equal to[2] the cost of 2.5 matrix-vector products by the matrix A. Both parameters are kept under control by ensuring that

$$nnz(A_{\ell-1})/nnz(A_\ell) \;\geq\; \tau, \quad \ell = 2, \ldots, L,$$

with τ well beyond 2; for instance, if $\tau = 3$, then $\mathcal{C}_W \leq 3$ and $\mathcal{C}_A \leq 3/2$.

Clearly, larger aggregates imply smaller coarse grid matrices with, on average, fewer nonzero entries, i.e., larger *coarsening ratios* $nnz(A_{\ell-1})/nnz(A_\ell)$. When we state that the DRA algorithm produces "large enough" aggregates, we mean that it consistently succeeds in achieving low weighted and operator complexities, the coarsening ratios being never significantly below 3 [14]. Regarding the quality control, we therefore need to check whether it has a significant impact on these ratios.

In this regard, one may consider the first two groups of columns in Table 1 (the third group is addressed later on), where relevant quantities are shown for a sample of six test problems that are representative of the different situations met in practice. With no surprise, when using the DRA algorithm alone, the coarsening ratios are always above 3 but the condition number is sometimes large, whereas, adding the quality control, some coarsening ratios become small, while, in agreement with the theory, the condition number is always below the prescribed limit $\overline{\kappa} = 10$.

Going into detail, the first two problems are representative of situations where the quality control has little impact: the condition number is already nice without it, and, although the coarsening ratio is significantly reduced, it is still far above the limit that guarantees low complexities. The third and fourth problems show cases where the condition number is significantly improved thanks to the quality control, while the impact on the coarsening speed is again marginal. The difficult cases are represented by the fifth and sixth problems. In the fifth one, one would better skip the quality control, since it has only a marginal effect on the already nice condition number, but it has a dramatic impact on the coarsening ratio. The situation is more mixed in the sixth case, where the quality control brings the needed improvement of the condition number, whereas the coarsening ratio, although small, is not that small.

[2]The value 2.5 takes into account the solution steps 1 and 7 of Algorithm 1 as well as the residual evaluation at steps 2 and 6, the former evaluation being performed via $\widetilde{\mathbf{r}} = (D-U)\,\mathbf{v}_1$ with $U = \mathrm{triu}(A)$ and $D = \mathrm{diag}(A)$.

TABLE 1

Condition number and coarsening ratios for a two-grid method when using the DRA algorithm alone (DRA), the DRA algorithm with quality control (DRA-QC), and the DRA algorithm with both quality control and complexity enhancement (DRA-QC-CE). The dagger symbol † *indicates that the Laplacian matrix of the graph comes from the University of Florida Sparse Matrix Collection* [7]*; a double dagger* ‡ *indicates matrices from the LAMG test set suite* [10]*; the subscript* $_{\mathrm{lcc}}$ *means that the experiment was performed with the largest connected component of the matrix; and* n_c *represents the number of rows and columns of* A_2*, that is, the number of aggregates.*

	DRA			DRA-QC			DRA-QC-CE		
Graph	κ_{eff}	$\frac{n}{n_c}$	$\frac{nnz(A)}{nnz(A_2)}$	κ_{eff}	$\frac{n}{n_c}$	$\frac{nnz(A)}{nnz(A_2)}$	κ_{eff}	$\frac{n}{n_c}$	$\frac{nnz(A)}{nnz(A_2)}$
`bcsstk29`‡	2.8	19.8	121.6	2.3	5.7	14.1	(2.3)	(5.7)	(14.1)
`nopoly`†	6.7	7.1	8.0	6.0	4.8	5.0	(6.0)	(4.8)	(5.0)
`Oregon-2`$^{\dagger}_{\mathrm{lcc}}$	4.7	3.6	6.1	2.1	10.0	5.8	(2.1)	(10.0)	(5.8)
`web-NotreDame`$^{\ddagger}_{\mathrm{lcc}}$	58.2	13.4	12.3	7.2	13.1	8.5	(7.2)	(13.1)	(8.5)
`t60k`† ‡	4.0	4.1	3.0	3.5	2.2	1.7	3.8	4.9	2.9
`web-BerkStan`$^{\ddagger}_{\mathrm{lcc}}$	85.5	15.8	52.3	8.8	3.2	2.9	18.2	13.7	27.8

To improve the situation when the coarsening ratio becomes small while keeping (hopefully) most of the benefit of the quality control, we then propose the following strategy. If, after the DRA algorithm with control has completed, it turns out that the ratio n/n_c between the number of variables and the number of aggregates is below 4,[3] then the vertices of aggregates of size 3 or less are marked as unaggregated, and the DRA aggregation procedure is applied to them without any quality control—the aggregates of size 4 or more obtained during the first pass being kept unchanged. Note that the decision is based on the ratio n/n_c because the coarsening ratio (based on the number of nonzero entries) is known only when the coarse grid matrix is formed, which is usually done only after the aggregation has been finalized. On the other hand, satisfactory complexity values are typically obtained if the ratio n/n_c is above 4 (see Table 1 for an illustration); this in turn explains the threshold value.

This strategy, referred to as DRA with quality control and complexity enhancement (DRA-QC-CE), is illustrated with the last group of columns in Table 1. In the first four problems, the ratio n/n_c is above the target, hence no extra step is performed, and the results are the same as for DRA with quality control. The extra step is performed in the remaining two cases, and one may observe that the coarsening ratios become indeed acceptable. More precisely, in the fifth problem, one actually recovers almost the situation one had without quality control, and which is optimal in this case. In the last problem, one gets in some sense the best of two worlds: the coarsening ratio is pretty large as it was without quality control, but the condition number, although above the limit $\overline{\kappa} = 10$, remains far below the one obtained without quality control.

4.5. Summary. We summarize the resulting aggregation scheme with Algorithm 3 below. The aggregates are initially formed as in Algorithm 2 above (steps 4–6): first a root vertex is chosen among those with the highest value of floor[$\log_2$(degree)]; then the neighboring vertices are added and, if the aggregate size is at most 6, neighbors of neighbors are appended.

However, unlike in Algorithm 2, the resulting aggregates are only tentative. They further undergo the bad vertices removal described in section 4.3.1 (step 10). Next, the aggregates are tested for quality, first via the criterion (16) (step 11), and, if

[3]This ratio is also the mean aggregate size.

needed, via the factorization of the matrix Z_G in (14) (step 12)—as written above, this is in fact never done when the size of the tentative aggregate is above 1024, since then the removal procedure only keeps vertices satisfying (18). If the factorization of Z_G yields a negative pivot, the subgraph extraction tool described in section 4.3.2 is further applied (steps 13–16). The resulting new tentative aggregate then reenters the procedure at step 9; i.e., bad vertices removal is again applied (with a slightly larger value of η) before a new explicit quality test based on (14). And so on, until the tentative aggregate G satisfies either the upper bound (16) (step 11) or the explicit quality criterion based on (14) (step 12); in either case it satisfies $\mu(G) < \overline{\kappa}$ when it is accepted (step 19).

Eventually, after all vertices have been assigned to an aggregate with verified quality, we check if the number of coarse variables (i.e., the number of aggregates) is at least four times smaller than the number of vertices (step 21). If not, the complexity enhancement procedure described in section 4.4 is applied (steps 22–24); that is, small aggregates are deleted, and the DRA algorithm without control is executed to group the so released vertices into additional aggregates.

Algorithm 3. DRA-QC-CE.

```
Input:   $A = (a_{ij})$                      % a graph Laplacian matrix
         $\overline{\kappa}$                              % quality threshold (here set to 10)
Output:  $n_c$                              % number of aggregates
         $G_i$, $i = 1, \dots, n_c$                % aggregates

1.   Compute floor[$\log_2$(degree(i))] for all vertices $i$, $i = 1, \dots, n$
2.   $n_c = 0$
3.   while  there are vertices outside $\cup_{s=1}^{n_c} G_s$
       % compute next tentative aggregates using DRA (steps 4-6)
4.     Select as root a vertex $r \notin \cup_{s=1}^{n_c} G_s$ with maximal value of floor[$\log_2$(degree(r))]
5.     $G = \{r\} \cup \{j \notin \cup_{s=1}^{n_c} G_s \mid a_{rj} \neq 0\}$
6.     if  $|G| \leq 6$
         $G \leftarrow G \cup \{j \notin \cup_{s=1}^{n_c} G_s \mid \exists k \in G : a_{kj} \neq 0\}$
       end if
       % extract a good-quality aggregate (steps 7-18)
7.     $\eta \leftarrow 2$
8.     accept $\leftarrow$ false
9.     while not accept
         cleaned $\leftarrow$ false
         % filter bad vertices (step 10)
10.      while not cleaned
           cleaned $\leftarrow$ true
           for $j \in G$, $j \neq r$
             if criteria (18) and (19) not satisfied for $j$
               $G \leftarrow G \backslash \{j\}$
               cleaned $\leftarrow$ false
             end if
           end for
         end while
         % test quality (steps 11, 12)
```

```
11.         if criterion (16) is satisfied
              accept ← true
            else
12.           if Cholesky factorization of $Z_G$ defined by (14) fails
                % subgraph extraction (steps 13-16)
13.             $\eta \leftarrow \eta + 0.5$
14.             $G_p \leftarrow \{ j \in G \,|\, \mathbf{v}(j)\,\mathbf{v}(r) \geq 0 \}$ with $\mathbf{v}$ as defined in section 4.3.2
15.             if $\mathbf{v}(r) = 0$: $G_p \leftarrow \{ j \in G \,|\, \mathbf{v}(j) \geq 0 \}$ end if
16.             $G \leftarrow \{ j \in G \,| j = r$ or criterion (22) satisfied with respect to $G_p \}$
17.           else
                accept ← true
              end if
            end if
18.     end while
        % form aggregate (step 19)
19.     $n_c \leftarrow n_c + 1$ and $G_{n_c} \leftarrow G$
20. end while
      % if needed, reform small aggregates (steps 21-25)
21. if $n_c > n/4$
22.     delete all $G_j$ such that $|G_j| \leq 3$;
23.     reset $n_c$ equal to the number of nondeleted aggregates;
24.     add aggregates obtained by running Algorithm 2 from step 3 with
        $G_j$, $j = 1, \ldots, n_c$ being the nondeleted aggregates
25. end if
```

5. Numerical experiments. We begin by describing in section 5.1 the extensive test set used for the experiments, providing details of the experimental setting, and commenting on the representation of the reported results. We then assess the new method in section 5.2, comparing with the former variant without quality control. Eventually, we provide in section 5.3 a comparison with the LAMG solver from [11].

5.1. General setting. The experiments have been performed on all the graph Laplacian matrices of undirected graphs that come from either the University of Florida Sparse Matrix Collection [7] or the LAMG test set suite [10]. In both cases, only the graphs with more than 10^4 vertices and with limited variation of edge weights have been considered. This gave a total of 142 graphs. For graphs with several connected components, the tests were run on the largest component.

The elapsed time is reported only for the 113 graphs that have more than $2 \cdot 10^5$ nonzero entries in their Laplacian matrix. (The elapsed times for the remaining 29 graphs are too small and hence subject to large relative variations from run to run.) Time experiments were performed by running a Fortran 90 implementation of the method on a single core of a computing node with two Intel Xeon L5420 processors at 2.50 GHz and 16 GB RAM memory. When considering the absolute value of the reported times, it is good to remember that this machine dates back to 2009.

In all cases, we used the flexible conjugate gradient method (FCG(1)) with the zero vector as initial approximation, the stopping criterion being a 10^{-6} reduction in the residual norm. Right-hand sides were generated randomly, the compatibility condition being enforced by an explicit projection onto $\mathcal{R}(A) = \mathbf{1}^{\perp}$.

Numerical results are reported here using bar diagrams, see Figure 3 for an example. Each abscissa segment corresponds to an individual graph from the test set, and each bar based on this segment is a reported value. All bar diagrams simultaneously report two parameters, which then correspond to superposed bars of a different color. More precisely, the semitransparent green (light gray) bar is always printed over the opaque blue (dark gray) bar; since the former bar is semitransparent, the latter bar is clearly visible even when it is completely covered by the other bar; the color of the overlapping area is dark green (gray). Hence, if the semitransparent green bars are higher than the blue ones, the resulting diagram looks like the one in Figure 3, bottom. However, if the blue bars are higher, as is the case for most of the bars in Figure 3, top and center, the blue bar will remain partly uncovered.

The problems in the diagram are gathered into three groups, depending on the ratio p between the maximal and the average number of nonzeros per row: group g_1 corresponds to $p < 1.2$ and mainly contains the regular mesh graphs; group g_2 corresponds to $1.2 < p < 12$ and includes some unstructured finite element meshes, road network graphs, as well as some low-degree referencing graphs; and group g_3 corresponds to $p > 12$ and contains social networks, e-mail, citation, and web referencing graphs—this group contains in principle all scale-free type of graphs [1] from our test set. We refer to [14] (where the same test suite is considered) for more details on the rationale of this classification. Within each group the problems are ordered by increasing value of their average number of nonzeros per row (or nnz/n).

5.2. Assessment of the new method. Let us first summarize how the ingredients of the new method are combined. It first recursively eliminates all the vertices of degree 1 in the system matrix, yielding a reduced Schur complement system (this applies only to graphs having such vertices, and the importance of this preprocessing step is discussed in [14]). This system is solved with FCG(1), using the preconditioner defined by Algorithm 1, in which step 4 is modified when $n_{\ell+1} > n_1^{1/3}$, the vertices of degree 1 being first eliminated, and the reduced Schur complement system being approximately solved with 2 FCG(1) using the same two-grid preconditioner at this coarse level. The aggregates G_i, $i = 1, \dots, n_c$, used at each level are obtained with the DRA-QC-EC algorithm (Algorithm 3) applied to the Schur complement matrix resulting from the elimination of degree 1 vertices. The only difference with the method in [14] lies in this use of the DRA-QC-EC algorithm instead of the original DRA algorithm (Algorithm 2).

The difference between both methods is illustrated in Figure 3, where we display the weighted and operator complexities as defined by (23), as well as the number of iterations needed to solve the test problems. The results are along the lines expected from the discussion in the preceding section. On the one hand, thanks, in some cases, to the complexity enhancement, the impact of the quality control on the complexities is fairly moderate, and, in particular, the weighted complexity remains below 3 as desired. On the other hand, the quality control clearly brings the robustness it has been designed for, the number of iterations being stabilized in the interval $15-30$ independently of the problem.

The timing results are displayed in Figure 4 in seconds per million nonzeros. The new aggregation algorithm is significantly more costly during the setup phase, but the setup time remains fairly low in absolute value, exceeding rarely and never dramatically half a second per million nonzeros. Regarding the solve times, they are very close to each other in roughly two-thirds of the problems, whereas the new method brings a significant improvement in the remaining one-third. Hence it is always a good

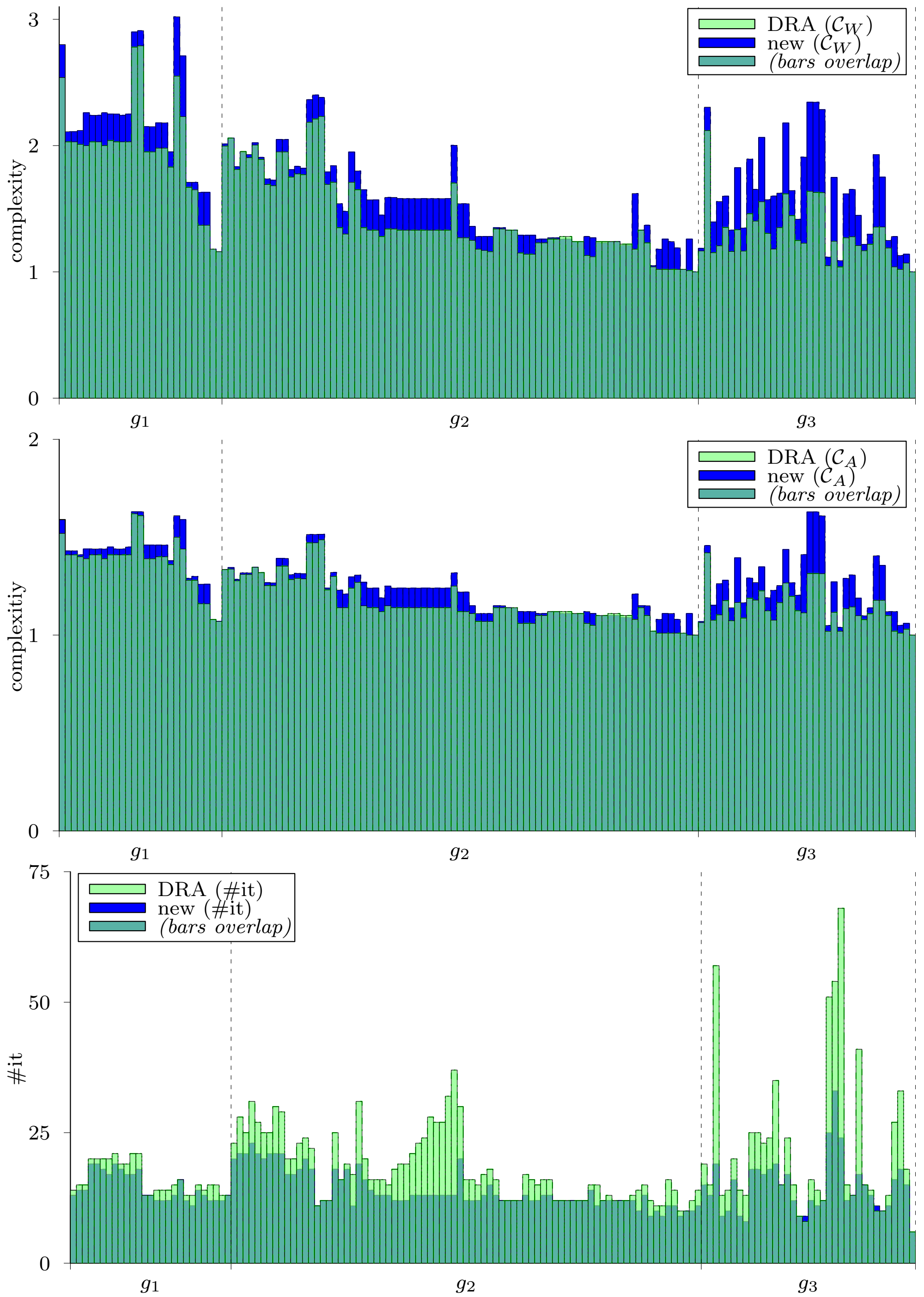

FIG. 3. *Weighted (top) and operator (center) complexities, as well as iteration counts (bottom) for the solver* [14] *based on DRA and on the new DRA-QC-EC aggregation scheme. The test problems are gathered into three groups–g_1, g_2, and g_3–as described at the end of section* 5.1*; consecutive groups are separated with a dashed line.*

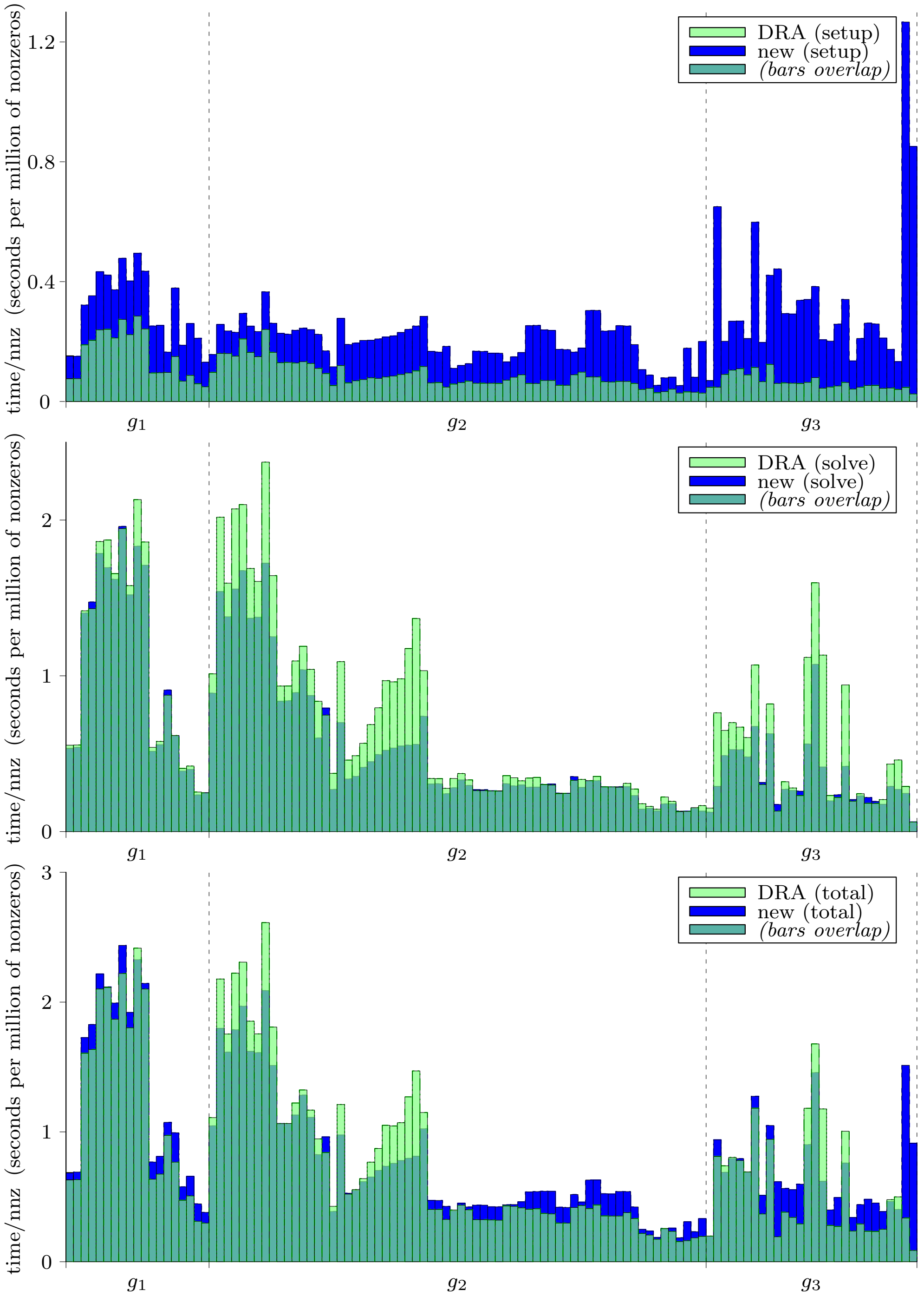

Fig. 4. *Setup (top), solution (center), and total time (bottom) for the solvers based on DRA and on the new DRA-QC-EC aggregation scheme; only graph Laplacians with more than* $2 \cdot 10^5$ *nonzero entries are considered.*

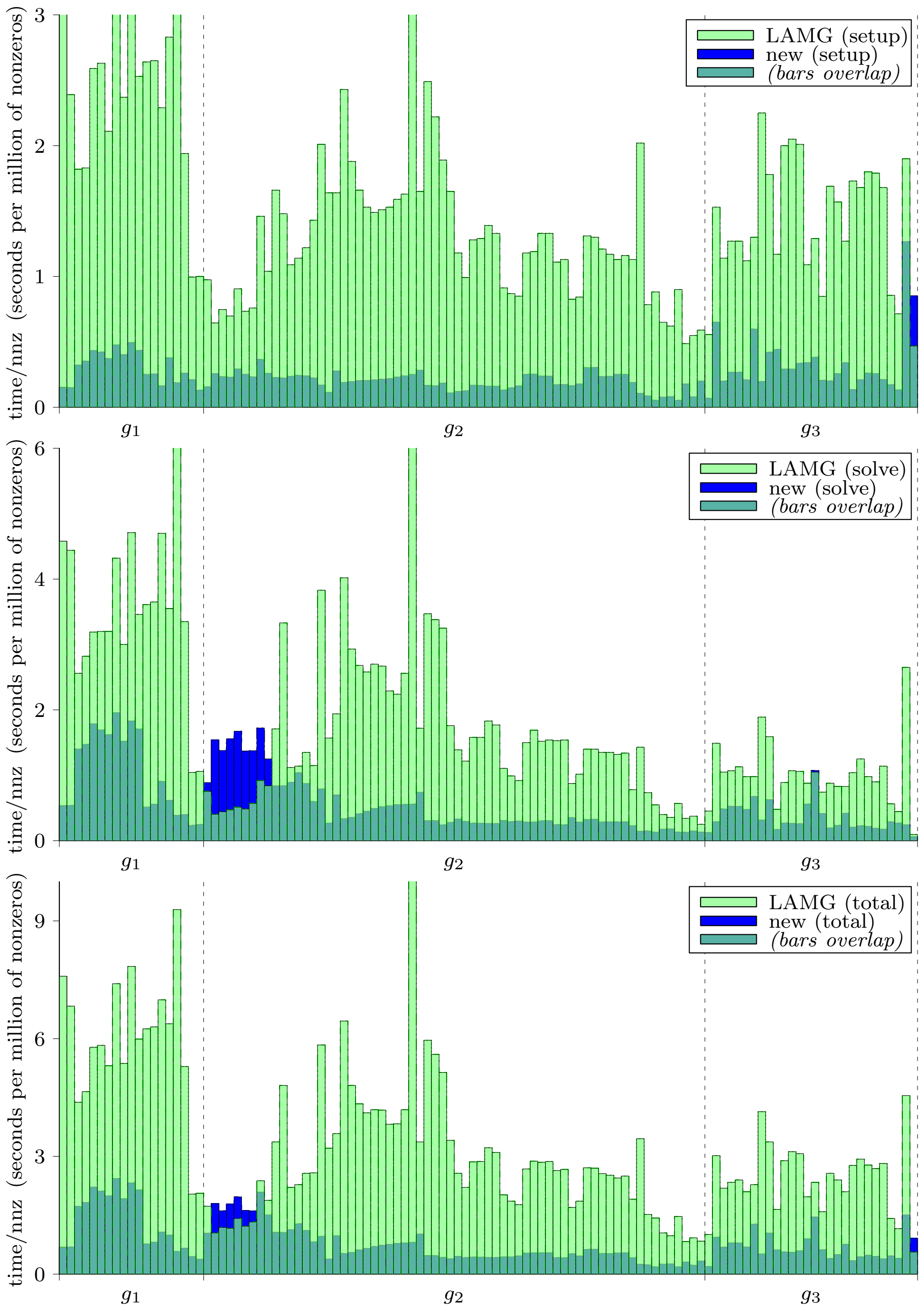

Fig. 5. *Setup (top), solution (center), and total time (bottom) for the solvers based on the new DRA-QC-EC aggregation scheme and for the LAMG code; only graph Laplacians with more than* $2 \cdot 10^5$ *nonzero entries are considered.*

idea to use the new DRA-QC-CE algorithm when emphasis is on the solve time, for instance, because several systems have to be solved with the same matrix. For one-shot applications, the results are more mixed, as illustrated by the reported total times.

5.3. Comparison with LAMG. We now report on the comparison between the solver based on the new aggregation scheme and the LAMG solver [10] implementing the method by Livne and Brandt [11]. The main LAMG driver is a MATLAB function, but computationally intensive routines are written in C. The code was run on a single computing node, using the standard MATLAB environment. Note that [11], regarding numerical experiments, mainly reports timing results, leading us to think that the LAMG code has been optimized enough with respect to speed and hence that a comparison with our code based on time measurements is reasonably fair although programming languages are different.

Timing results are displayed in Figure 5. Up to a few exceptions, both the setup and solution times are lower for the new solver and, as a result, the total times are also typically two to three times lower. In fact, LAMG is better mainly for graphs with few nonzero entries per row. Most of these graphs correspond to road networks with the number of nonzero entries per row below 4, meaning that the average degree of the vertices is below 3. In these cases, LAMG benefits from an extended preprocessing step that eliminates vertices with degree up to 4, whereas our method only eliminates vertices of degree 1. Such an extended elimination is thus worth considering if the focus is on graphs with very small average degree.

6. Conclusions.

We considered the solution of graph Laplacian systems with aggregation-based multigrid. We incorporated quality control in the DRA algorithm proposed in [14] in order to improve its robustness. Moreover, we successfully faced the two inherent obstacles. First, we avoided an excessive increase of the setup time thanks to a clever filtering procedure that extracts an aggregate of verified quality from a given tentative aggregate without using any time-consuming step. Second, we avoided a dramatic impact on the complexity by running, in case of need, a second pass where vertices in too small aggregates (of size 1, 2, or 3) are regrouped into bigger ones. This results in a new aggregation scheme, referred to as degree-aware rooted aggregation with quality control and complexity enhancement.

The numerical results demonstrate the robustness and effectiveness of the solver based on this new aggregation scheme: the number of iterations needed for a six orders of magnitude reduction in the residual norm remains below 33 for all the 142 graphs in the considered test set. The new solver also compares favorably to the LAMG solver [10], being significantly faster in most cases, whereas, in the few cases where LAMG is better, it brings only a relatively marginal improvement.

REFERENCES

[1] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.

[2] A. Ben Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*, Wiley, New York, 1974.

[3] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.

[4] M. Bolten, S. Friedhoff, A. Frommer, M. Heming, and K. Kahl, *Algebraic multigrid methods for Laplacians of graphs*, Linear Algegra Appl., 434 (2011), pp. 2225–2243.

[5] E. G. Boman and B. Hendrickson, *Support theory for preconditioning*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 694–717.

[6] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Application, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1984, pp. 257–284.

[7] T. A. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), pp. 87–94, https://www.cise.ufl.edu/research/sparse/matrices/.

[8] R. D. Falgout and P. S. Vassilevski, *On generalizing the algebraic multigrid framework*, SIAM J. Numer. Anal., 42 (2005), pp. 1669–1693.

[9] I. Koutis, G. L. Miller, and D. Tolliver, *Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing*, Comput. Vis. Image Understanding, 115 (2011), pp. 1638–1646.

[10] O. E. Livne, *Lean Algebraic Multigrid (LAMG) MATLAB software*, Release 2.1.1., https://lamg.googlecode.com (2012).

[11] O. E. Livne and A. Brandt, *Lean Algebraic Multigrid (LAMG): Fast graph Laplacian linear solver*, SIAM J. Sci. Comput., 34 (2012), pp. B449–B522.

[12] A. Napov and Y. Notay, *Algebraic analysis of aggregation-based multigrid*, Numer. Linear Algebra Appl., 18 (2011), pp. 539–564.

[13] A. Napov and Y. Notay, *An algebraic multigrid method with guaranteed convergence rate*, SIAM J. Sci. Comput., 43 (2012), pp. A1079–A1109.

[14] A. Napov and Y. Notay, *An efficient multigrid method for graph Laplacian systems*, Electron. Trans. Numer. Anal., 45 (2016), pp. 201–218.

[15] Y. Notay, *Flexible conjugate gradients*, SIAM J. Sci. Comput., 22 (2000), pp. 1444–1460.

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

[17] Y. Notay, *Algebraic two-level convergence theory for singular systems*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1419–1439.

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

[19] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers in Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.

[20] D. A. Spielman, *Algorithms, graph theory, and linear equations in Laplacian matrices*, in Proceedings of the International Congress of Mathematicians, Vol. 4, World Scientific, Singapore, 2010, pp. 2698–2722.

[21] P. M. Vaidya, *Solving Linear Equations with Symmetric Diagonally Dominant Matrices by Constructing Good Preconditioners*, manuscript, 1990.

[22] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, Computing, 56 (1996), pp. 179–196.

[23] P. S. Vassilevski, *Multilevel Block Factorization Preconditioners*, Springer, New York, 2008.