2014

Graph Partitioning Using Matrix Values for Preconditioning Symmetric Positive Definite Systems

Eugenr Vecharynski

Yousef Saad

Masha Sosonkina
Old Dominion University

GRAPH PARTITIONING USING MATRIX VALUES FOR PRECONDITIONING SYMMETRIC POSITIVE DEFINITE SYSTEMS*

EUGENE VECHARYNSKI†, YOUSEF SAAD‡, AND MASHA SOSONKINA§

Abstract. Prior to the parallel solution of a large linear system, it is required to perform a partitioning of its equations/unknowns. Standard partitioning algorithms are designed using the considerations of the efficiency of the parallel matrix-vector multiplication, and typically disregard the information on the coefficients of the matrix. This information, however, may have a significant impact on the quality of the preconditioning procedure used within the chosen iterative scheme. In the present paper, we suggest a spectral partitioning algorithm, which takes into account the information on the matrix coefficients and constructs partitions with respect to the objective of enhancing the quality of the nonoverlapping additive Schwarz (block Jacobi) preconditioning for symmetric positive definite linear systems. For a set of test problems with large variations in magnitudes of matrix coefficients, our numerical experiments demonstrate a noticeable improvement in the convergence of the resulting solution scheme when using the new partitioning approach.

Key words. graph partitioning, iterative linear system solution, preconditioning, Cauchy–Bunyakowski–Schwarz (CBS) constant, symmetric positive definite, spectral partitioning

AMS subject classifications. 15A06, 65F08, 65F10, 65N22

DOI. 10.1137/120898760

1. Introduction. Partitioning of a linear system for its parallel solution typically aims at satisfying two standard objectives: *minimizing communication volume* and *maintaining load balance* among different processors. Both of these requirements are motivated by efficiency considerations of the parallel matrix-vector product, which lie in the heart of the iterative solution methods. Once performed the partitioning is then exploited to construct a parallel preconditioner—another crucial ingredient which contributes to the overall performance of the solver. However, the quality of the resulting preconditioner may depend significantly on the given partitioning, which generally targets the efficiency of the parallel matrix-vector multiplication, but ignores its effect on the resulting preconditioner. This preconditioner can be of poor quality, especially in the cases when the coefficient matrices have entries with large variations in magnitudes.

For the purpose of obtaining an effective preconditioner, we suggest removing the requirement on the communication volume and, instead, consider partitionings that favor the quality of the preconditioner. In particular, we focus on the *nonoverlapping*

---

†Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (eugene.vecharynski@gmail.com).

‡Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 (saad@cs.umn.edu).

§Department of Modeling, Simulation and Visualization Engineering, Old Dominion University, Norfolk, VA 23529 (msosonki@odu.edu).

additive Schwarz (AS) preconditioners for symmetric positive definite (SPD) linear systems [36, 39]. The proposed partitioning algorithm aims at optimizing the quality of the AS procedure by attempting to minimize the condition number of the preconditioned matrix, while maintaining a good load balance. The new strategy is tested on several linear systems that arise from discretizations of partial differential equations (PDEs) with *strongly varying* coefficients.

The choice of the nonoverlapping AS, which is a form of block-diagonal, or block Jacobi, preconditioning, is motivated by several factors. First, the procedure represents the simplest domain decomposition (DD) type preconditioner, which is theoretically well understood and is often of practical interest due to its high degree of parallelism. Second, block-diagonal preconditioning constitutes an important element of a number of more powerful preconditioning schemes, e.g., overlapping or multilevel Schwarz methods, substructuring type algorithms, etc. Our expectation is that the partitions that improve the quality of the nonoverlapping AS preconditioners are also capable of increasing the robustness of these preconditioning schemes.

The problem of partitioning a linear system $Ax = b$ is commonly formulated in terms of the adjacency graph $G(A) = (V, E)$ of the coefficient matrix $A = (a_{ij})$. Here, $V = \{1, 2, \ldots, n\}$ is the set of vertices (nodes) corresponding to the equations/unknowns of the system, and E is the set of edges (i, j), where $(i, j) \in E$ iff $a_{ij} \neq 0$. Throughout, we assume that A is SPD, i.e., $A = A^* \succ 0$, which, in particular, implies that the graph $G(A)$ is undirected.

The *standard* goal of graph partitioning is to partition $G(A)$ into s subgraphs $G_k = (V_k, E_k)$, where $V_k \subseteq V$ and $E_k \subseteq E$, such that

$$(1.1) \qquad \bigcup_{k=1,s} V_k = V, \qquad \bigcap_{k=1,s} V_k = \emptyset, \qquad |V_k| \approx n/s,$$

and the size of the edge cut between G_k (i.e., the size of the set of edges whose end points are in different G_k) is minimized. Equations and unknowns with numbers in V_k are then typically mapped to the same processor; s corresponds to the total number of processors. The requirement on the small edge cut aims at reducing the cost of communications related to the parallel matrix-vector multiplication. The condition $|V_k| \approx n/s$ ensures load balancing. We note that there are alternative models for graph partitioning, based, e.g., on bipartite graphs [14] or hypergraphs [2]. We do not consider these models in the present paper.

The graph partitioning problem is NP-complete. However, there exist a variety of heuristics for solving the problem; see, e.g., [8, 12, 22, 26, 29]. Efficient implementations of partitioning routines are often based on multilevel algorithms, e.g., [17, 20], and are made available in a number of graph partitioning software packages, such as Chaco [15], JOSTLE [38], MeTiS [19], SCOTCH [28], etc.

If the preconditioner quality becomes an objective of the partitioning, then along with the adjacency graph $G(A)$, it is reasonable to consider weights w_{ij} assigned to the edges $(i, j) \in E$, where w_{ij}'s are determined by the coefficients of the matrix A. The corresponding algorithm should then be able to take these weights into account and properly use them to perform graph partitioning. An example of such an algorithm has been discussed in [32].

Indeed, one may consider partitioning as part of the preprocessing phase of the preconditioner setup. Then the use of the coefficients of A at the partitioning step, e.g., through the weights w_{ij}, represents a natural option. This approach, however, faces a number of issues. For example, given a preconditioning strategy, how does

one assign the weights? What are the proper partitioning objectives? How can the partitioning be performed in practice?

In the present work, these questions are addressed for the case of SPD linear systems and nonoverlapping AS preconditioners. Our rationale is to relate partitioning to the results of the convergence theory for the preconditioned conjugate gradient (PCG) method with block-diagonal preconditioning. More specifically, we regard bipartitioning (i.e., partitioning of the graph into two parts) as an optimization problem that aims at minimizing an upper bound on the condition number of the preconditioned matrix over all possible balanced bipartitions. As a result, we derive a recursive bisection procedure that is built upon a simple weighting scheme and a *modification* of the standard partitioning objective.

A straightforward approach for dealing with the partitioning problem of this paper would be to assign edge weights as the magnitudes of the corresponding matrix entries and apply a state-of-the-art partitioning algorithm. While this heuristic indeed often improves convergence, it can be substantially outperformed by the new partitioning strategy as will be demonstrated in our numerical experiments.

The presented algorithm relies on the computation of eigenvectors corresponding to the smallest eigenvalues of a generalized eigenvalue problem, which simultaneously involves weighted and standard graph Laplacians. As such, the new strategy is a form of the *spectral recursive bisection* (RSB) based on a "nonstandard" eigenvalue problem.

Spectral graph partitioning has roots in the works of Fiedler [9, 10] and Donath and Hoffman [6, 7]. It is extensively used in many applications, including VLSI circuit design, data clustering, image segmentation; see, e.g., [4, 24, 33] and the references therein. In the scientific computing community, the spectral partitioning was popularized by Pothen, Simon, and Liou in [29] and further studied, e.g., in [16, 34, 37].

Spectral graph partitioning is typically characterized by a good quality of the resulting partitions. This can be attributed to the fact that spectral algorithms utilize global information in contrast to combinatorial algorithms that typically rely on local information. The common criticism of spectral partitioning is its large computational cost, due to the eigenvalue calculations.

Although efficient solution of a specific eigenvalue problem is not the focus of this work, we favor the use of preconditioned eigensolvers. An attractive feature of this approach is that it splits the computation into a *preconditioning phase* and *outer iterations*, which are based on the matrix-vector multiplications. Ideally, one can expect that the preconditioning scheme encapsulates multilevel strategies (e.g., graph coarsening, uncoarsening, refinement) inherent to the available combinatorial graph partitioners, combined with only a few outer iterations. In our numerical experiments, however, for demonstration purposes, we use incomplete Cholesky (IC) preconditioning.

The partitioning problem addressed in this paper is closely related to the more traditional task of *constructing block-diagonal preconditioners*. Indeed, the latter also admits graph formulations and relies on the use of the matrix coefficients; see, e.g., [11]. The main difference, however, lies in the fact that, along with improving the preconditioning quality, the presented partitioning scheme imposes the additional load balance constraint, which is not required for the conventional block-diagonal preconditioning.

This paper is organized as follows. In section 2, we briefly review several known results concerning the block-diagonal preconditioning for SPD matrices. These results motivate the new partitioning scheme, introduced in section 3. In section 4, we report

on a few numerical examples, where the presented approach is compared to a state-of-the-art partitioning algorithm, MeTiS, with different weighting schemes.

2. Block-diagonal preconditioning. Consider a block 2-by-2 matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^* & A_{22} \end{pmatrix}, \tag{2.1}$$

where the diagonal blocks A_{11} and A_{22} are square of size m and $(n-m)$, respectively; the off-diagonal block A_{12} is m-by-$(n-m)$. Let T be a block-diagonal preconditioner,

$$T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix}, \tag{2.2}$$

where $T_j = T_j^* \succ 0$, $j = 1, 2$. The dimensions of T_1 and T_2 are same as those of A_{11} and A_{22}, respectively.

Since both A and T are SPD, the convergence of an iterative method for $Ax = b$, such as PCG, is fully determined by the spectrum of the preconditioned matrix $T^{-1}A$. If no information on the exact location of eigenvalues of $T^{-1}A$ is available, then the worst-case convergence behavior of PCG is traditionally described in terms of the condition number $\kappa(T^{-1}A)$, where $\kappa(T^{-1}A) = \lambda_{\max}(T^{-1}A)/\lambda_{\min}(T^{-1}A)$ with $\lambda_{\max}(T^{-1}A)$ and $\lambda_{\min}(T^{-1}A)$ denoting the largest and the smallest eigenvalues of the preconditioned matrix, respectively. The question which arises is how we can bound $\kappa(T^{-1}A)$ for an arbitrary A and a block-diagonal T. The answer to this question is given, e.g., in [1, Chapter 9]. Below, we briefly state the main result.

DEFINITION 2.1. *Let U_1 and U_2 be finite dimensional spaces, such that A in (2.1) is partitioned consistently with U_1 and U_2. The constant*

$$\gamma = \max_{w_1 \in W_1, w_2 \in W_2} \frac{|(w_1, Aw_2)|}{(w_1, Aw_1)^{1/2}(w_2, Aw_2)^{1/2}}, \tag{2.3}$$

where W_1 and W_2 are subspaces of the form

$$W_1 = \left\{ u = \begin{pmatrix} u_1 \\ \boldsymbol{0} \end{pmatrix}, u_1 \in U_1 \right\}, \quad W_2 = \left\{ u = \begin{pmatrix} \boldsymbol{0} \\ u_2 \end{pmatrix}, u_2 \in U_2 \right\}, \tag{2.4}$$

is called the Cauchy–Bunyakowski–Schwarz (CBS) constant.

In (2.3), $(u, v) = v^* u$ denotes the standard inner product. We note that γ can be interpreted as a cosine of an angle between subspaces W_1 and W_2. Thus, since, additionally, $W_1 \cap W_2 = (\boldsymbol{0}\ \boldsymbol{0})^*$, it is readily seen that $0 \le \gamma < 1$. Also we note that γ is the smallest possible constant satisfying the strengthened CBS inequality $|(w_1, Aw_2)| \le \gamma(w_1, Aw_1)^{1/2}(w_2, Aw_2)^{1/2}$, which motivates its name.

THEOREM 2.2 (see [1, Chapter 9]). *If $T_1 = A_{11}$ and $T_2 = A_{22}$ in (2.2)), and A in (2.1) is SPD, then $\kappa(T^{-1}A) \le (1+\gamma)/(1-\gamma)$.*

The bound given by Theorem 2.2 is sharp. In the subsequent sections, we use this result to derive a new partitioning algorithm.

3. Partitioning using matrix coefficients. Given decomposition $\{V_k\}_{k=1}^s$ of the set $V = \{1, 2, \ldots, n\}$ (possibly with overlapping V_k), we consider the AS preconditioning for an SPD system $Ax = b$. The preconditioning procedure is given in Algorithm 3.1. By $A(V_l, V_k)$ we denote a submatrix of A located at the intersection of rows with indices in V_l and columns with indices in V_k. Similarly, $r(V_k)$ denotes the subvector of r, containing entries from positions V_k. In this section, we focus on the case where sets (subdomains) $\{V_k\}_{k=1}^s$ are nonoverlapping, i.e., (1.1) holds. This means that Algorithm 3.1 gives a *nonoverlapping* AS preconditioner.

ALGORITHM 3.1 (AS preconditioner).
Input: A, r, $\{V_k\}_{k=1}^s$. Output: $w = T^{-1}r$.
 1. *For $k = 1, \ldots, s$, Do*
 2.   Set $A_k := A(V_k, V_k)$, $r_k := r(V_k)$, and $w_k = \boldsymbol{0} \in \mathbb{R}^n$.
 3.   Solve $A_k \delta = r_k$.
 4.   Set $w_k(V_k) := \delta$.
 5. *EndDo*
 6. $w = w_1 + \cdots + w_s$.

Let P be a permutation matrix which corresponds to the reordering of V according to the partition $\{V_k\}_{k=1}^s$, where the elements in V_1 are labeled first, in V_2 second, etc. Then the AS preconditioner T, given by Algorithm 3.1, can be written in the matrix form as $T = P^T \bar{T} P$, where $\bar{T} = \text{blockdiag}\{A_1, \ldots, A_s\}$ and $A_k = A(V_k, V_k)$. Thus, Algorithm 3.1 results in the block-diagonal, or block Jacobi, preconditioner, up to a permutation of its rows and columns.

3.1. Optimal bipartitions. Let $s = 2$, so that (1.1) corresponds to a *bipartition*

$$(3.1) \qquad V = I \cup J, \qquad I \cap J = \emptyset, \qquad |I| = |J| = n/2.$$

Here, we assume that n is even. This guarantees the existence of *fully balanced* bipartitions, such that vertex sets I and J are of the same size, $n/2$. Similarly, we assume that each connected component of $G(A)$ also has an even number of vertices. The above assumptions, however, will not be a restriction for the practical algorithm described below.

The following theorem provides a relation between a given bipartition and $\kappa(T^{-1}A)$. Theorem 3.1 is a direct consequence of Theorem 2.2 and is based on the fact that symmetric permutations preserve the spectra.

THEOREM 3.1. *Let $\{I, J\}$ in (3.1) be a bipartition of V (possibly unbalanced). Let T be the AS preconditioner for system $Ax = b$ with an SPD matrix A, given by Algorithm 3.1, with respect to the bipartition $\{I, J\}$. Then,*

$$(3.2) \qquad \kappa(T^{-1}A) \le \frac{1 + \gamma_{IJ}}{1 - \gamma_{IJ}},$$

where

$$(3.3) \qquad \gamma_{IJ} = \max_{u \in W_I, v \in W_J} \frac{|(u, Av)|}{(u, Au)^{1/2}(v, Av)^{1/2}}.$$

The spaces W_I and W_J are the subspaces of $\mathbb{R}^n$ with dimensions m and $(n - m)$, respectively, such that

$$(3.4) \qquad W_I = \{u \in \mathbb{R}^n \ : \ u(J) = \boldsymbol{0}\}, \quad W_J = \{v \in \mathbb{R}^n \ : \ v(I) = \boldsymbol{0}\}.$$

The sharp upper bound (3.2) represents a meaningful indicator of the preconditioner quality. Thus, as an optimal bipartition, we can choose $\{I, J\}$, such that $(1 + \gamma_{IJ})/(1 - \gamma_{IJ})$ or, equivalently, the CBS constant γ_{IJ}, is minimized. More precisely, we define an optimal bipartition $\{I_{opt}, J_{opt}\}$ to be such that

$$(3.5) \qquad \{I_{opt}, J_{opt}\} = \operatorname*{argmin}_{\substack{I, J \subset V = \{1, \ldots, n\}, \\ |I| = |J| = \frac{n}{2}, J = V \setminus I}} \gamma_{IJ},$$

where γ_{IJ} is defined in (3.3).

A straightforward solution of optimization problem (3.5) entails evaluating (3.3) for very large, namely $n!/[(n/2)!]^2$, possible choices of the partitions $\{I, J\}$. While this indicates that the problem is likely to be NP-hard, it is not clear whether an efficient solution to (3.5) can be found.

Therefore, our idea is to replace (3.5) by a related simpler problem, such that the minimizer of the latter *approximates* (in terms of smallness of the CBS constant) $\{I_{opt}, J_{opt}\}$ rather than determines it exactly. Below, we discuss several approaches.

3.2. The minimal averaged cut. In order to obtain a simpler optimization problem, let us replace γ_{IJ} in (3.5) by some $\tilde{\gamma}_{IJ}$, such that $\tilde{\gamma}_{IJ}$ captures information on γ_{IJ} and is easy to compute. We define $\tilde{\gamma}_{IJ}$ as follows.

Given $\{I, J\}$, instead of maximizing the ratio

$$(3.6) \qquad \frac{|(u, Av)|}{(u, Au)^{1/2}(v, Av)^{1/2}},$$

as required for computing γ_{IJ} in (3.3), we introduce a set of pairs

$$(3.7) \qquad S = \{(e_i, e_j) \; : \; i \in I, j \in J, a_{ij} \neq 0\} \subset W_I \times W_J,$$

where $e_k \in \mathbb{R}^n$ denotes the unit vector with 1 at position k, and calculate (3.6) on all $(e_i, e_j) \in S$. Note that the cardinality of S is equal to the size of the cut between I and J, further denoted by $\mathrm{cut}(I, J)$. The resulting values are averaged. This gives a constant $\tilde{\gamma}_{IJ}$, such that

$$(3.8) \qquad \tilde{\gamma}_{IJ} = \frac{w(I, J)}{\mathrm{cut}(I, J)},$$

where $w(I, J) = \sum_{i \in I, j \in J} w_{ij}$, $w_{ij} = |a_{ij}|/\sqrt{a_{ii}a_{jj}} \in (0, 1)$.

The constant $\tilde{\gamma}_{IJ}$ has a transparent meaning in terms of the adjacency graph $G(A)$. Assigning the weights w_{ij} to the edges (i, j), (3.8) can be interpreted as a ratio of the cut weight, $w(I, J)$, to the cut size; or, equivalently, as the *averaged* cut weight. At the same time, $\tilde{\gamma}_{IJ}$ is closely related to γ_{IJ}. In particular, $\tilde{\gamma}_{IJ} \leq \gamma_{IJ}$ for all $\{I, J\}$. Thus, we replace optimization problem (3.5) by

$$(3.9) \qquad \{\tilde{I}_{opt}, \tilde{J}_{opt}\} = \underset{\substack{I, J \subset V = \{1, \ldots, n\}, \\ |I| = |J| = \frac{n}{2}, J = V \setminus I}}{\mathrm{argmin}} \tilde{\gamma}_{IJ}.$$

The minimizer $\{\tilde{I}_{opt}, \tilde{J}_{opt}\}$ is expected to approximate $\{I_{opt}, J_{opt}\}$. Formally, (3.8)–(3.9) represents the problem of graph bipartitioning, where the targeted cut has the smallest, in average, weight. We call such a cut *the minimal averaged cut (Acut).*

Minimization of objective (3.8) is achieved by bipartitions that provide a balance between the two concurrent requirements on minimizing the cut weight and maximizing the number of edges included in the cut. Thus, (3.8)–(3.9) gives cuts that include a relatively large number of "light-weighted" edges. As shown in our numerical experiments, for a number of systems arising from discretizations of partial differential equations with strongly varying coefficients, this allows us to recognize the boundaries of subregions that correspond to different coefficient magnitudes. The resulting subdomains are experimentally shown to improve preconditioning quality.

3.3. Spectral Acut computations. Optimization problem (3.9) is still hard to solve exactly since it encompasses the original graph partitioning problem which is known to be NP-complete [35]. Therefore, we address below the question of *approximating* Acut for practical applications. We propose a spectral bipartitioning technique.

Let p denote the indicator vector of size n, with the components defined as

$$(3.10) \qquad p(k) = \left\{ \begin{array}{ll} 1, & k \in I, \\ -1, & k \in J. \end{array} \right.$$

Then, for a given $\{I, J\}$,

$$4w(I, J) = \sum_{i=1}^{n} d_w(i) p(i)^2 - \sum_{i,j=1}^{n} w_{ij} p(i) p(j),$$

where $d_w(i) = \sum_{j \in N(i)} w_{ij}$ is the weighted degree of the vertex i; $N(i)$ denotes the vertices adjacent to i. Thus, $w(I, J)$ can be written as a bilinear form,

$$(3.11) \qquad 4w(I, J) = p^T L_w p, \qquad L_w = D_w - W,$$

where $D_w = \mathrm{diag}(d_w(1), \ldots, d_w(n))$ is the weighted degree matrix and $W = (w_{ij})$ denotes the weighted adjacency matrix. Similarly, setting $w_{ij} = 1$ for all edges (i, j) of $G(A)$, we get the expression for $\mathrm{cut}(I, J)$,

$$(3.12) \qquad 4\mathrm{cut}(I, J) = p^T L p, \qquad L = D - Q,$$

where D is the degree matrix and $Q = (q_{ij})$ is the adjacency matrix. The matrices L_w and L denote the weighted and unweighted *graph Laplacians*. Thus, given $\{I, J\}$, (3.11) and (3.12) allow representing (3.8) as a ratio of two bilinear forms, i.e., $\tilde{\gamma}_{IJ} = p^T L_w p / p^T L p$.

Let us assume that $G(A)$ has $q \geq 1$ connected components (V_l, E_l), where $|V_l|$ are even. We introduce vectors $z_1, \ldots, z_q$, such that

$$(3.13) \qquad z_l(k) = \left\{ \begin{array}{ll} 1, & k \in V_l, \\ 0, & k \notin V_l, \end{array} \right.$$

i.e., the entries of z_l corresponding to vertices in V_l are 1, and 0 elsewhere. Note that if $q = 1$, then we obtain a single vector of ones.

Problem (3.9) can now be written as

$$(3.14) \qquad \tilde{p}_{opt} = \underset{p}{\mathrm{argmin}} \frac{p^T L_w p}{p^T L p}, \qquad p^T z_l = 0, \qquad l = 1, \ldots, q,$$

where the minimizer $\tilde{p}_{opt}$ is searched over all feasible indicator vectors. The condition $p^T z_l = 0$ ensures that all the connected components are bipartitioned into two equal-sized sets of vertices. Hence, $|I| = |J| = n/2$, as required.

In order to approximate solution of (3.14), we relax the problem by embedding it into the real space. More specifically, we consider the minimization

$$(3.15) \qquad \min_{v \in \mathbb{R}^n} \frac{v^T L_w v}{v^T L v}, \qquad v \in \mathrm{span}\{z_1, \ldots, z_q\}^{\perp},$$

of the *generalized Rayleigh quotient* on the orthogonal complement of the subspace spanned by vectors z_l in (3.13). We expect the minimizer of (3.15) to provide an approximation to the optimal indicator vector $\tilde{p}_{opt}$ from (3.14).

Both L_w and L are symmetric positive semidefinite, with the dimension of the nullspace equal to the number of connected components of $G(A)$. In particular,

$$\text{null}(L_w) = \text{null}(L) = \text{span}\{z_1, \ldots, z_q\}.$$

Thus, $v^T L_w v / v^T L v$ in (3.15) is minimized on the orthogonal complement of the nullspace of the two matrices, where both L and L_w are SPD. This implies that the minimum in (3.15) exists. It is achieved on the eigenvector associated with smallest eigenvalue of the symmetric generalized eigenvalue problem

$$(3.16) \qquad L_w v = \lambda L v, \qquad v \in \text{span}\{z_1, \ldots, z_q\}^\perp.$$

The solution of (3.16) can be viewed as an analogue of the Fiedler vector [9, 10]. The bipartition is formed by assigning the indices of its $\lceil n/2 \rceil$ smallest components to I and the rest to J. As shown in our numerical experiments, the eigenvector of (3.16) may deliver disconnected subdomains, even though the original graph is connected. Note that the assumption on the even sizes of the vertex sets of $G(A)$ and its connected components is not restrictive any more, and is skipped for relaxed problem (3.16).

Finally, let us observe that if the weights w_{ij} are the same for all edges, then the graph Laplacians L_w and L represent multiples of each other. In this case, the solution of (3.16) is given by multiple orthogonal eigenvectors that correspond to the only nonzero eigenvalue of multiplicity $n - q$, i.e., the result of spectral Acut computations is highly uncertain. Such a situation is an indicator of the fact that the coefficient matrix A has extremely regular behavior of its entries. Therefore, if all w_{ij} are the same, or only slightly different, we suggest using *standard* partitioning criteria.

3.4. The minimal weighted cut. We now relate the problem of minimizing the CBS constant to the *standard* objective for graph partitioning. In particular, we consider replacing (3.5) by minimization of the cut weight under the load balance constraint.

Given a bipartition $\{I, J\}$ in (3.1), similarly to (3.7), we define the set

$$(3.17) \qquad \bar{S} = \{(e_i, e_j) \ : \ i \in I, j \in J\} \subset W_I \times W_J.$$

Unlike (3.7), (3.17) contains *all* pairs (e_i, e_j) with $i \in I$ and $j \in J$, including those that correspond to $a_{ij} = 0$. Instead of maximizing (3.6), as required to compute γ_{IJ}, we evaluate (3.6) on all $(n/2)^2$ pairs in (3.17) and then find the average. Thus, for a given $\{I, J\}$ we define a constant $\bar{\gamma}_{IJ} \le \gamma_{IJ}$, such that

$$(3.18) \qquad \bar{\gamma}_{IJ} = \frac{4}{n^2} w(I, J),$$

where $w(I, J)$ is the cut weight, defined as in (3.8).

Following the pattern of the previous subsections, instead of (3.5), we suggest solving the optimization problem

$$(3.19) \qquad \{\bar{I}_{opt}, \bar{J}_{opt}\} = \underset{\substack{I, J \subset V = \{1, \ldots, n\}, \\ |I| = |J| = \frac{n}{2}, J = V \setminus I}}{\operatorname{argmin}} \bar{\gamma}_{IJ},$$

where $\{\bar{I}_{opt}, \bar{J}_{opt}\}$ is expected to approximate an optimal $\{I_{opt}, J_{opt}\}$ in (3.5). It is readily seen that (3.18)–(3.19) delivers the well-known problem of graph partitioning, which aims at finding equal-sized vertex sets I and J with the minimal cut weight. The solution of (3.18)–(3.19) can be approximated by an available graph partitioning scheme which admits edge weighting. In our numerical examples, we use MeTiS.

The principal difference between (3.8)–(3.9) and (3.18)–(3.19) is that (3.18)–(3.19) delivers cuts of a *cumulatively* small weight, which may generally contain both "heavy-weighted" and "light-weighted" edges. At the same time, cuts given by (3.8)–(3.9) favor edges with small weights. For example, if A is given by a PDE with jumps in coefficients, such edges connect subdomains corresponding to different coefficient magnitudes, whereas edges with large w_{ij} are in the interior of these subdomains. Therefore, unless forced by the load balance constraint, (3.8)–(3.9) excludes the undesirable possibility of "cutting" inside jump subregions.

3.5. Acut by recursive spectral bisection. Let $\{I, J\}$ be given by Acut described in subsection 3.3. A natural way to construct further partitions (1.1) is to apply the bipartitioning process separately for subgraphs of $G(A)$ corresponding to I and J, then to the resulting partitions and so on, until all the subpartitions are sufficiently small. We summarize this in the following algorithm.

ALGORITHM 3.2 (Acut-RSB(A)).
Input: A. Output: *Partition* $\{V_i\}$.
  1. Form $G(A)$. Assign weights $w_{ij} = |a_{ij}|/\sqrt{a_{ii}a_{jj}}$.
  2. Construct $L_w = D_w - W$ and $L = D - Q$.
  3. Find connected components $\{E_l, V_l\}$. Define z_l in (3.13).
  4. Find the eigenvector associated with the smallest eigenvalue of (3.16).
  5. Define $\{I, J\}$ based on the computed eigenvector.
  6. Apply the algorithm recursively:
     If $|I| > $ maxSize*, then* call Acut-RSB*($A(I, I)$), else* return I.
     If $|J| > $ maxSize*, then* call Acut-RSB*($A(J, J)$), else* return J.

The parameter *maxSize* in Algorithm 3.2 is provided by the user, and should be chosen to ensure that $|V_k| \approx n/s$. The connected components in step 3 can be detected by standard algorithms based on the breadth-first search or the depth-first search [3]. Note that weights w_{ij} are the same on all levels of the recursion. In practice, they are assigned only once, on the top level.

Clearly, if eigenvalue problem (3.16) in step 4 is replaced by computing the Fiedler vector of either $L_w v = \lambda v$ or $Lv = \lambda v$, then Algorithm 3.2 reduces to the well-known RSB scheme; see, e.g., [30]. We suggest that before the recursive Acut-RSB calls in step 6, one checks if $A(I, I)$ and $A(J, J)$ indeed have large variations in magnitudes of their entries, e.g., by assessing the variance of the coefficients. If the entries of $A(I, I)$ or $A(J, J)$ exhibit a regular behavior, we recommend invoking one of the *standard*, unweighted, partitioning algorithms.

4. Numerical results. In this section, we apply Acut-RSB to several SPD test problems and compare the partitioning results to more traditional approaches delivered by the MeTiS algorithm with various weighting schemes. For each test problem, we use the partitions produced by all algorithms to construct the AS preconditioners, which are then supplied to the PCG iteration. We further refer to this solution scheme as PCG–AS.

In our examples, we consider three types of MeTiS partitions. The first type results from applying the algorithm to the *unweighted* adjacency graph, so that the coefficient information is completely skipped. In contrast, the other two types are

obtained by running MeTiS for the weighted graphs, where the weights are based on the matrix coefficients. In particular, we use weighting schemes $y = \{y_{ij}\}$ and $t = \{t_{ij}\}$, with the edge weights $y_{ij} = \lceil \gamma |a_{ij}|/\sqrt{a_{ii}a_{jj}} \rceil$ and $t_{ij} = \lceil \delta |a_{ij}| \rceil$. Here, the parameter γ is a large integer and $\delta > 0$. We denote the corresponding partitioning methods by MeTiS(y) and MeTiS(t), respectively. Note that a closely related weighting scheme has been used in [13].

Let us remark that y_{ij} can be written as $y_{ij} = \lceil \gamma w_{ij} \rceil$, where w_{ij} are the weights in the CBS constant based partitioning formulations (3.8)–(3.9) and (3.18)–(3.19). Thus, whereas Acut-RSB targets (3.8)–(3.9), MeTiS(y) is expected to approximate partitions in (3.18)–(3.19). The constant γ and the "ceil" operation in the definition of y_{ij} are introduced to satisfy the MeTiS requirement on the integer weight values (recall that $w_{ij} \in (0,1)$).

In our numerical experiments, we have observed that MeTiS(y) gives better results than the RSB scheme based on the Fiedler vector for $L_w v = \lambda v$, which approximates solution of (3.18)–(3.19). Therefore, the latter is not reported in the examples below.

The choice of $t_{ij} = \lceil \delta |a_{ij}| \rceil$ for MeTiS(t) is motivated by the computational experience, suggesting that each edge should be weighted with (essentially) the absolute value of the corresponding matrix entry. While straightforward and well-known among practitioners, we were not able to find direct references to such a weighting scheme in the literature.

In our tests, we define t_{ij} with δ from $(0,1]$. This choice of δ "damps" the magnitudes of $|a_{ij}|$. It ensures that the weighted cut sizes computed by MeTiS(t) do not exceed the upper limit of the integer data type size, in which case the behavior of the algorithm can be unpredictable. Similar to y_{ij}, the "ceil" operation is introduced to maintain integer weights. Note that if A is diagonally scaled, y_{ij} and t_{ij} coincide.

Throughout, the reported partitioning results produced by MeTiS(y) and MeTiS(t) correspond to the best values of γ and δ observed during our experiments. In all cases, MeTiS has been invoked with the "PartGraphRecursive" option, which enables the *recursive bipartitioning* [21] and guarantees the *strict* load balance. Note that all our experiments are performed in MATLAB; the MeTiS library is accessed through the MEX interface.

Unlike MeTiS, which represents a combinatorial partitioning technique, the new Acut-RSB algorithm relies on computing eigenvectors. In all of our tests, as an eigensolver, we use locally optimal block preconditioned conjugate gradient (LOBPCG) method [23]. Our choice has been motivated mainly by the fact that, unlike the Lanczos algorithm [27], LOBPCG can solve *generalized* eigenvalue problems, such as (3.16), without requiring any factorization of the singular matrices L_w and L. Additionally, LOBPCG encapsulates preconditioning to accelerate convergence and has a relatively modest storage requirement.

The LOBPCG algorithm is a form of a (block) three-term recurrence, which locally optimizes the Rayleigh quotient; see [23] for more details. Given a suitable SPD preconditioner, the method is known to be efficient for large-scale eigenvalue computations. In our experiments, we construct preconditioners using IC factorization of matrices $L_w + \sigma I$ with a drop tolerance of 10^{-3}. To ensure that the IC procedure is correctly applied to the SPD matrices, the parameter σ is assigned a small positive value. In particular, we set $\sigma = 0.1$.

In all LOBPCG runs, we choose a random initial guess from $\mathrm{span}\{z_1, \ldots, z_q\}^{\perp}$ and at each step project the preconditioned residuals to this subspace, so that the LOBPCG iterations are kept in $\mathrm{span}\{z_1, \ldots, z_q\}^{\perp}$. Note that the amount of storage

and computations required to orthogonalize against $\text{span}\{z_1, \ldots, z_q\}^\perp$ does not depend on q, i.e., is the same as for the n-vector of all ones.

Finally, let us remark that the partitioning runtimes exhibited by MeTiS have been notably lower than those of Acut-RSB. This can be attributed to the spectral nature of the new algorithm. Additionally, the runtime difference has been exacerbated by the fact that, in contrast to the MEX-interfaced MeTiS library calls, our Acut-RSB code is purely in MATLAB.

However, as demonstrated below, despite longer runtimes, Acut-RSB often leads to a significantly smaller number of iterations of the linear solver. In this context, the partitioning cost (i.e., the preprocessing phase of the preconditioner setup) becomes less important. In fact, our goal is precisely this: to get a more reliable iterative solution procedure by paying a higher cost in the preprocessing.

Two-dimensional (2D) diffusion equation. Let us consider the diffusion equation on a unit square,

$$(4.1) \qquad -\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial u}{\partial y}\right) = f(x,y), \ (x,y) \in [0,1] \times [0,1],$$

with zero Dirichlet boundary conditions and the coefficients given by the piecewise constants

$$(4.2) \qquad a(x,y) = b(x,y) = \begin{cases} 10^5, & 0.25 < x, y < 0.75, \\ 1 & \text{otherwise}; \end{cases}$$

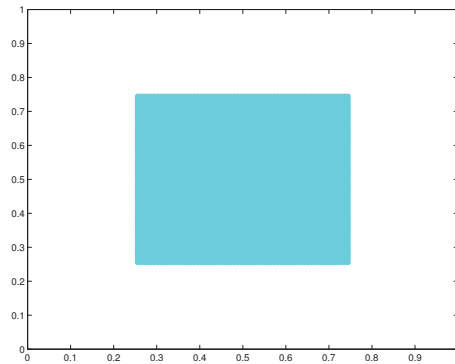which strongly vary (jump) across the two subdomains; see Figure 4.1.

FIG. 4.1. *Location of the jump in coefficients* (4.2) *for problem* (4.1). *The shaded (cyan for the color plot) subdomain corresponds to the jump region* $(0.25, 0.75) \times (0.25, 0.75)$.

In order to discretize (4.1), we introduce a 128-by-128 (interior points) uniform grid and use the standard 5-point finite difference (FD) stencil. The resulting linear system, $Ax = b$, is SPD of size $n = 16{,}384$. For testing purposes, the right-hand side b is randomly chosen. Here and below, the FD matrices have been generated using the SPARSKIT library [31].

Figure 4.2 shows the *bipartitions* produced by different methods. It is readily seen that Acut-RSB precisely detects the jump region and places most of the cut along its boundary, without "cutting" inside. In the framework of the DD type methods, the latter is consistent with a common recommendation to include the regions
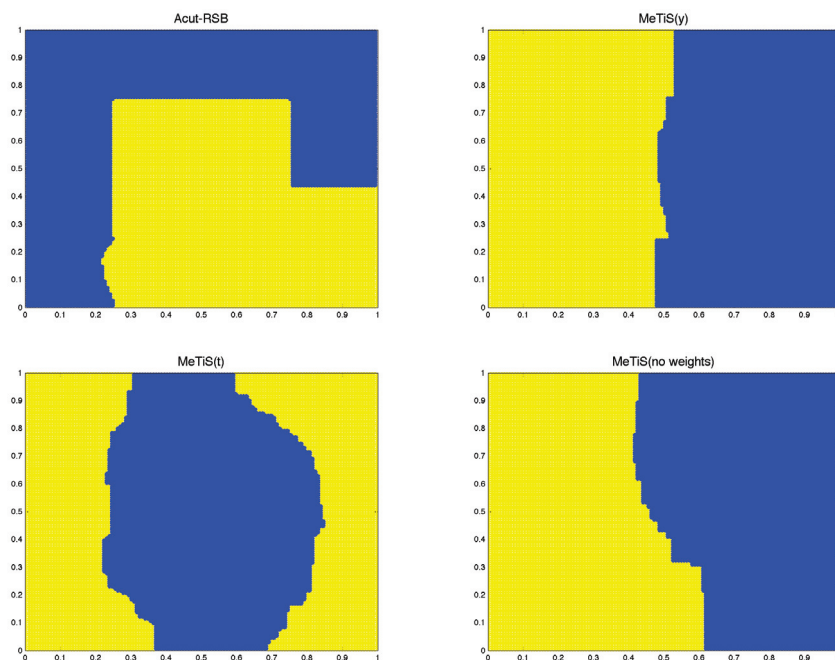
FIG. 4.2. *Bipartitions for problem* (4.1) *with coefficients in* (4.2).

corresponding to different model parameters into separate subdomains. The eigenvector, used to define the bipartition in Acut-RSB, is shown in Figure 4.8 (left). The values of γ and δ for the weights in MeTiS(y) and MeTiS(t) have been set to 10^5 and 1, respectively. The convergence tolerance for LOBPCG is 10^{-7}.

Similar to Acut-RSB, we observe that the bipartition produced by MeTiS(t) also recognizes the region of the jump and isolates it in a separate subdomain. The visual difference, however, is in that the boundaries of this subdomain are not as smooth as for the Acut and do not tightly follow the boundaries of the jump region.

In Figure 4.3, we present the results obtained after an additional step of the recursive bipartitioning (with the same γ, δ, and the LOBPCG convergence tolerance as above). In this case, the partitioning procedures deliver four subdomains. Note that, regardless of the weighting scheme, all three MeTiS runs perform long cuts within the jump region. In contrast, Acut-RSB essentially "crops" the central square, with the top right corner assigned to a different subdomain merely to ensure the strict load balance.

Since one of the two subdomains resulting from the Acut-RSB bipartitioning does not contain a jump region (see Figure 4.2, top left), we use a standard partitioning approach at the recursive step on this subdomain. In particular, we use the spectral bisection based on the Fiedler vector [29].

Throughout all of our experiments, we set the number of targeted subdomains to be small, i.e., at most 16 as in the three-dimensional (3D) elasticity example considered below. Due to simplicity of the geometries of the tested problems, convergence of the Acut based solver deteriorates if the number of subdomains is increased. In this case, Acut-RSB begins to partition *inside* the regions with similar coefficient magnitudes,
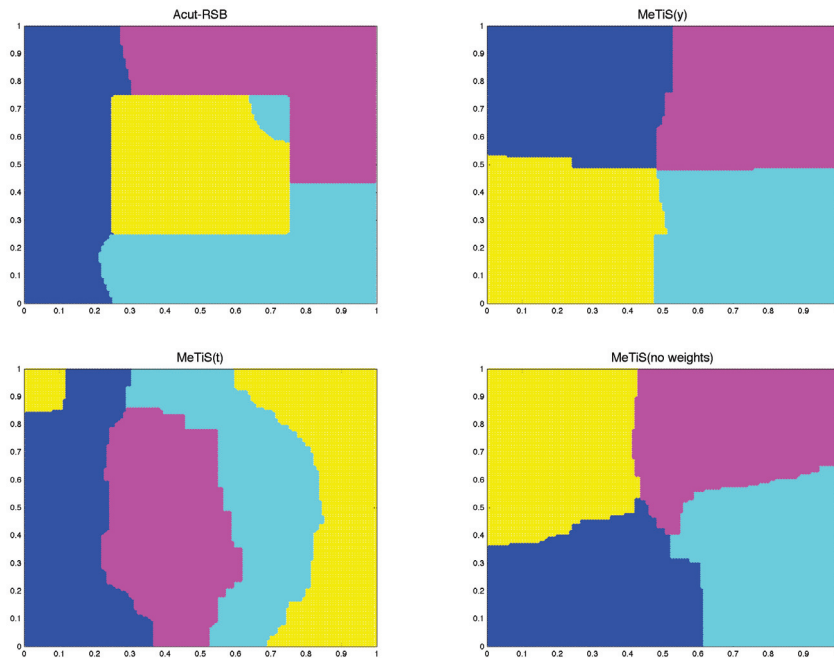
Fig. 4.3. *Partitioning into four subdomains for problem* (4.1) *with coefficients in* (4.2).

i.e., it is forced by the load balance constraint to discard "heavy" edges. At some point, this hinders the convergence. Generally, we expect that the number of subdomains, and hence of processors, will depend on the problem geometry and coefficients.

In Figure 4.4, we show the effects of partitioning on the convergence of a preconditioned iterative scheme. In particular, we plot the convergence curves of PCG–AS with the preconditioners defined on different partitions. In both cases, with two and four subdomains, we observe that PCG–AS based on the Acut-RSB partitioning in Algorithm 3.2 performs a significantly smaller number of iterations.
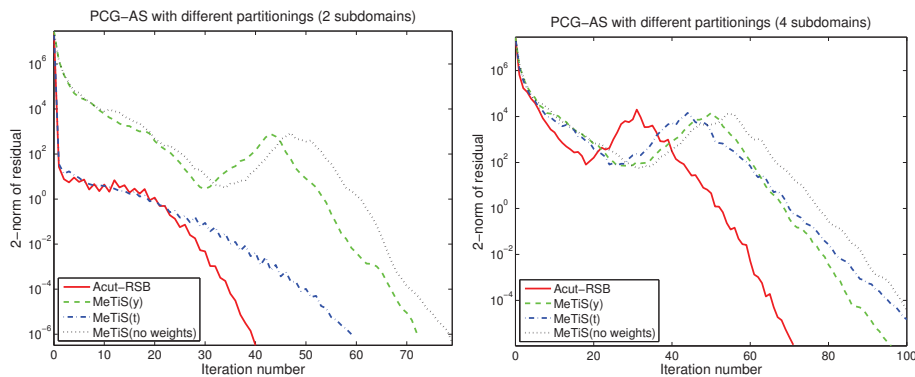

Fig. 4.4. *Convergence of PCG–AS with different partitions for problem* (4.1) *with coefficients in* (4.2). *The linear system of size* $n = 16{,}384$ *is partitioned into two (left) and four (right) subdomains.*

TABLE 4.1

Relative cut sizes and amounts of the coefficient information discarded to construct preconditioners for problem (4.1) with coefficients in (4.2).

Partitioning	Two subdomains		Four subdomains	
	relcut	relcoef	relcut	relcoef
Acut-RSB	0.76	3×10^{-5}	1.25	0.42
MeTiS(y)	0.44	0.85	0.86	1.62
MeTiS(t)	1.06	4×10^{-5}	1.67	0.88
MeTiS(no w.)	0.48	1.07	1.01	2.13

In Table 4.1, we show the relative cut sizes and amounts of the coefficient information discarded from the matrix after partitioning to construct the preconditioners. In particular, we report the quantities

$$\text{relcut} = \frac{\text{cut size}}{\text{nnz}} \times 100\%, \qquad \text{relcoef} = \frac{\sum_{(i,j)\in\text{cut}} |a_{ij}|}{\sum_{i,j} |a_{ij}|} \times 100\%,$$

where "cut size" is the number of edges in the given cut and "nnz" denotes the number of nonzeros in A, i.e., the total number of edges in the adjacency graph; "cut" is the edge cut of interest.

We see that the cut sizes corresponding to Acut-RSB are relatively large in terms of the edge count, compared to the unweighted MeTiS and MeTiS(y). At the same time, the amount of the coefficient information discarded for preconditioning is significantly smaller. The similar properties are exhibited by MeTiS(t). Note that, in the case of bipartitioning, "relcoef" for Acut-RSB is only slightly smaller than that for MeTiS(t), whereas the quality of the associated preconditioner is much higher; see Figure 4.4 (left).

To the best of our knowledge, there are no available convergence bounds that are based on "relcoef." However, from the practical point of view, it is reasonable to expect that the smaller "relcoef," the higher the quality of the corresponding preconditioner. In particular, if a standard graph partitioner is at hand, then common approaches for assigning the edge weights, such as in MeTiS(t), are motivated exactly by this heuristic. While often achieving the smallest values of "relcoef" indeed leads to better results, we demonstrate that the dependence does not hold in general.

More precisely, we keep track of "relcoef" to show that the preconditioning quality is affected not only by the sum of (absolute values of) the discarded coefficients, but also by the *actual combination of edges* in the cut. While the traditional MeTiS based approaches can succeed in the former, Acut-RSB is capable of choosing more suitable edge combinations for the resulting cuts. This point is further elaborated on in the following examples.

2D diffusion equation: The "checkerboard" jump location. Let us now consider a discretization of (4.1) with zero Dirichlet boundary conditions, where the jumps in coefficients are located in the "checkerboard" fashion,

$$(4.3) \qquad a(x,y) = b(x,y) = \begin{cases} 10^5 & \text{"black"}, \\ 1 & \text{"white"}, \end{cases}$$

as shown in Figure 4.5. As in the above example, we use a 5-point FD stencil on a 128-by-128 uniform grid, which leads to an SPD linear system of size $n = 16{,}384$.

The results of bipartitioning produced by different methods are presented in Figure 4.6. Similar to the previous example, Acut-RSB perfectly detects the jump regions
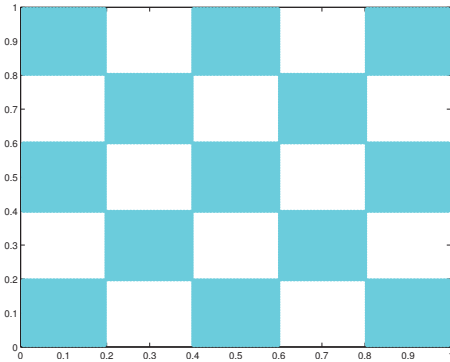
FIG. 4.5. *The "checkerboard" location of the jump in coefficients* (4.3) *for problem* (4.1)*. The shaded (cyan for the color plot) subdomains correspond to the "black" jump regions.*

and avoids "cutting" inside. Note that the resulting two subdomains are disconnected. The eigenvector of (3.16), used to define the Acut-RSB bipartition, is shown in Figure 4.8 (right). The parameters γ and δ are the same as in the previous example, i.e., 10^5 and 1, respectively. The LOBPCG convergence tolerance is set to 10^{-7}.
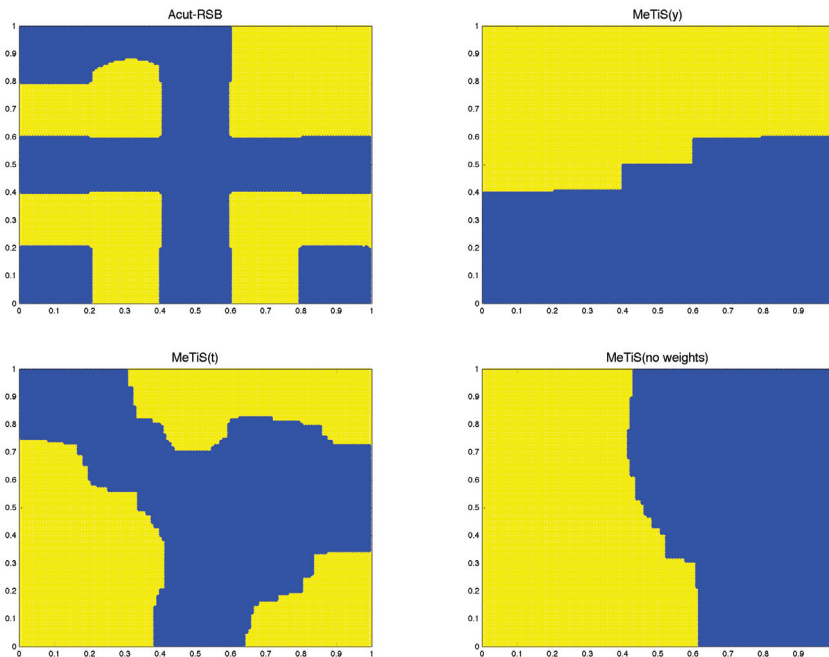

FIG. 4.6. *Bipartitions for problem* (4.1) *with coefficients in* (4.3)*.*

Figure 4.6 demonstrates that, unlike the unweighted MeTiS, the runs of MeTiS(y) and MeTiS(t) deliver partitions that adapt to the geometry of the jumps, i.e., attempt to follow the boundaries of the "black" subregions. The advantage, with respect to convergence of PCG–AS, of using matrix coefficients at the partitioning stage is further confirmed in Figure 4.7.
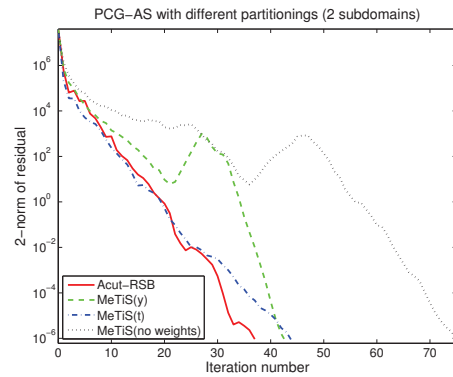
FIG. 4.7. *Convergence of PCG–AS with different bipartitions for problem* (4.1) *with coefficients in* (4.3). *The linear system is of size* $n = 16{,}384$.

TABLE 4.2
Relative cut sizes and amounts of the coefficient information discarded to construct preconditioners for bipartitioning problem (4.1) *with coefficients in* (4.3).

Partitioning	relcut	relcoef
Acut-RSB	1.78	0.06
MeTiS(y)	0.47	0.17
MeTiS(t)	1.23	0.02
MeTiS(no w.)	0.48	0.47

Figure 4.7 shows that, in terms of iteration count, PCG–AS with Acut-RSB slightly outperforms the analogues based on MeTiS(y) and MeTiS(t). Interestingly, however, the coefficient sum of (absolute values of) the matrix coefficients discarded to construct the preconditioner, reported in Table 4.2, is *not* the smallest for Acut-RSB. More precisely, the "relcoef" value corresponding to Acut-RSB is three times larger than that of MeTiS(t). This observation clearly allows one to conclude that the preconditioning quality is affected not only by the magnitudes of the discarded matrix entries, but also by the specific combination of edges selected to the cut that defines the underlying partitions.
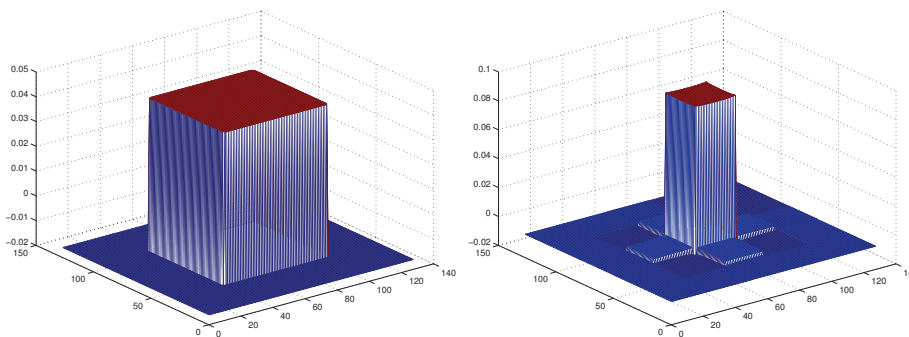

FIG. 4.8. *"Mesh" plot of the eigenvector* v *of* (3.16) *used for bipartitioning problem* (4.1) *with coefficients in* (4.2) *(left) and* (4.3) *(right). Both eigenvectors capture the discontinuities in coefficients of the corresponding problems.*

3D linear elasticity. Our next experiment is based on the example constructed by Mandel, Sousedík, and Šístek in [25] to test the performance of the adaptive balancing domain decomposition by constraints (BDDC) [5] in three dimensions. In this example, the authors consider the 3D linear elasticity problem (see, e.g., [18]) in a cube with material parameters $E = 10^6$ Pa and $\nu = 0.45$, penetrated by four bars with parameters $E = 2.1 \times 10^{11}$ Pa and $\nu = 0.3$; see Figure 4.9. Zero Dirichlet boundary conditions are assumed.
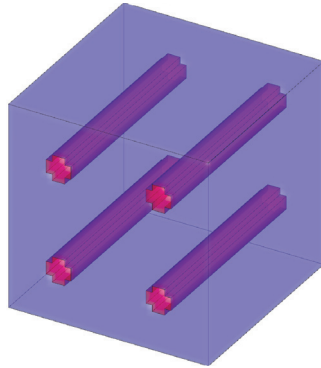

FIG. 4.9. *3D linear elasticity: cube with jumps in coefficients. Example by Mandel, Sousedík, and Šístek* [25].

The problem has been discretized using bilinear finite elements (FE), resulting in 107,811 degrees of freedom. In our tests, we apply different schemes to partition the problem into 8 and 16 subdomains, and assess the quality of the obtained preconditioners according to the number of PCG–AS iterations. The parameters γ and δ for MeTiS(y) and MeTiS(t) have been set to 10^4 and 10^{-4}, respectively. The LOBPCG convergence tolerance is 10^{-4}. In the case of 16 subdomains, on the bottom level of the Acut-RSB recursion (after producing 8 subdomains), we decrease the tolerance to 10^{-7}. The right-hand side b is chosen as a random unit vector.

Figure 4.10 shows that Acut-RSB leads to the best convergence of PCG–AS with both 8 and 16 subdomains. Note that, according to the results reported in Table 4.3,
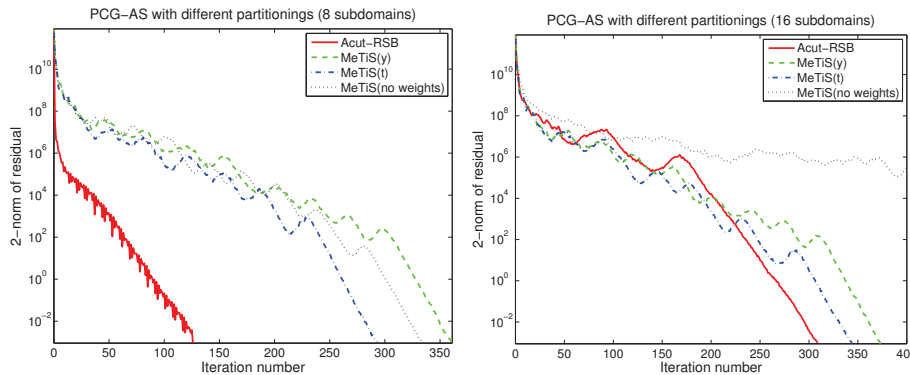

FIG. 4.10. *Convergence of PCG–AS with different partitions (8 (left) and 16 (right) subdomains) for the 3D linear elasticity in a cube with large jumps in coefficients; see Figure* 4.9. *The size of the linear system arising from the FE discretization is $n = 107,811$.*

TABLE 4.3
*Relative cut sizes and amounts of the coefficient information discarded to construct precondi-
tioners for 3D linear elasticity problem in a cube with jumps in coefficients; see Figure 4.9.*

Partitioning	8 subdomains		16 subdomains	
	relcut	relcoef	relcut	relcoef
Acut-RSB	19.98	0.01	25.39	1.55
MeTiS(y)	8.03	2.51	11.14	2.51
MeTiS(t)	7.30	1.50	11.99	1.51
MeTiS(no w.)	6.77	2.24	10.61	10.28

the cut size produced by Acut-RSB is about 2–2.5 times larger than that delivered by
the MeTiS runs, implying extra communication overhead if implemented in parallel.
The gain from this "loss of parallelism," however, is the (approximately) 10% to 45%
decrease in the iteration count compared to the closest competitor, MeTiS(t), and
more than (approximately) 60% decrease compared to the unweighted MeTiS. Once
again, from Table 4.3, we note that for 16 subdomains "relcoef" for Acut-RSB is
larger than that for MeTiS(t), although the convergence of the former is better.

2D diffusion equation: Unstructured grid. In this concluding example, we
consider an FE discretization of diffusion equation (4.1) on an unstructured grid.
We assume Dirichlet boundary conditions and $f(x, y) = 1$. The problem domain
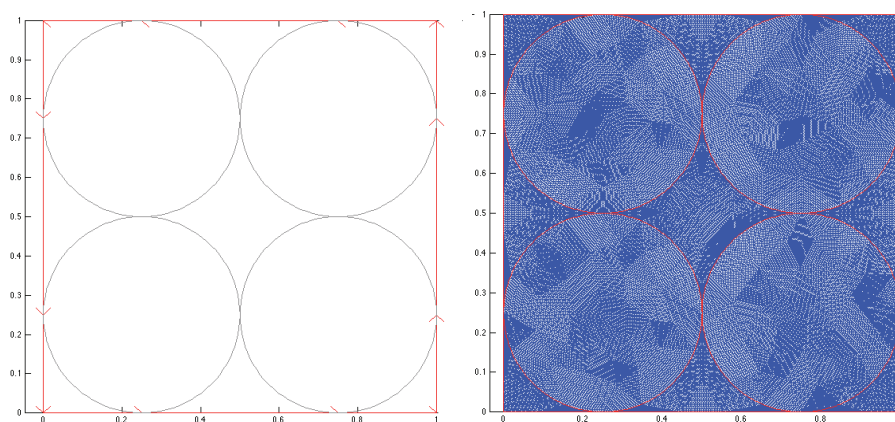represents a unit square with four inscribed circles of the same radius; see Figure 4.11
(left).

FIG. 4.11. *Problem domain (left) and the corresponding FE mesh (right) for* (4.1).

In order to discretize the equation on an unstructured grid, we use MATLAB
PDE TOOLBOX. The toolbox allows one to define the problem geometry as well as the
PDE coefficients and to introduce an initial triangulation, which is further modified
by a few refinement steps. The latter results in an unstructured FE mesh with 47,713
degrees of freedom, shown in Figure 4.11 (right).

The goal of the current example is twofold. On the one hand, we show that Acut-
RSB can be successfully applied to problems on unstructured grids. On the other
hand, the example reveals potential difficulties with the new partitioning strategy,
which should be addressed in future research.

Let us first consider the following definition of the coefficients:

$$(4.4) \qquad a(x,y) = b(x,y) = \begin{cases} 10^7 & \text{"outside circles and on circles' boundaries",} \\ 1 & \text{"inside circles";} \end{cases}$$

i.e., $a(x,y)$ and $b(x,y)$ are the piecewise constants taking a large value outside of the four circles and a small value inside. We partition the problem into two and four subdomains using different partitioning schemes and observe the convergence of the corresponding PCG–AS runs. The parameters γ and δ for MeTiS(y) and MeTiS(t) are set to 10^4 and 10^{-2}, respectively. The LOBPCG convergence tolerance is 10^{-7} for the case of two, and 10^{-8} for the case of four, subdomains. The partitioning results are given in Figures 4.12 and 4.13.
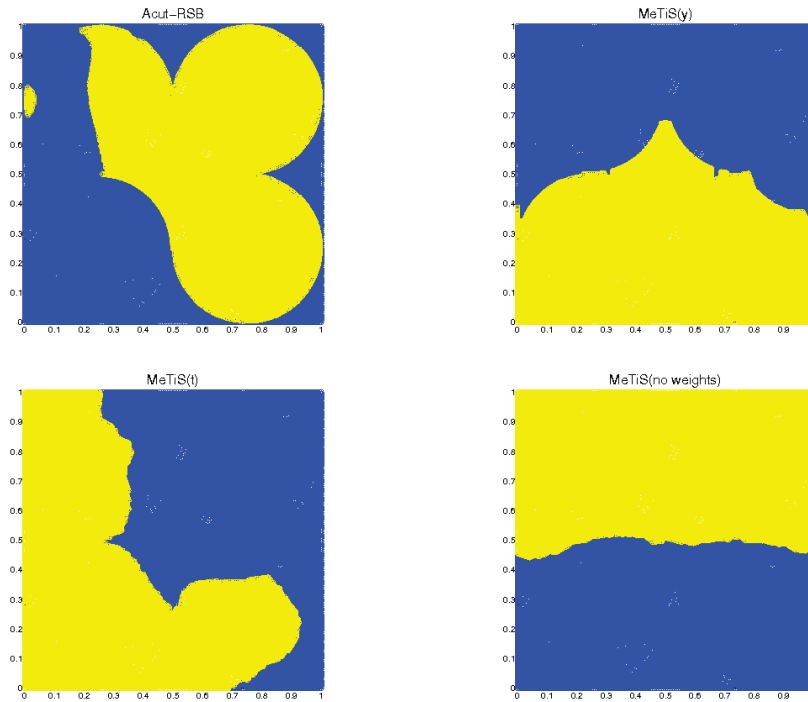

FIG. 4.12. *Bipartitions of unstructured mesh (Figure* 4.11*) for problem* (4.1) *with coefficients in* (4.4).

Figure 4.14 shows that the use of Acut-RSB reduces the iteration count by approximately 15% compared to the weighted MeTiS runs for the case of two subdomains, and gives a result comparable to MeTiS(t) for four subdomains. Note that in the latter case the comparable convergence results are produced even though the "relcoef" for MeTiS(t) is six times larger than that of Acut-RSB, as can be seen from Table 4.4.

Now, let us "invert" the definition of the coefficients, so that

$$(4.5) \qquad a(x,y) = b(x,y) = \begin{cases} 10^7 & \text{"inside circles and on circles' boundaries",} \\ 1 & \text{"outside circles";} \end{cases}$$

i.e., $a(x,y)$ and $b(x,y)$ are the piecewise constants taking a small value outside of the four circles and a large value inside. The problem geometry and the mesh are the
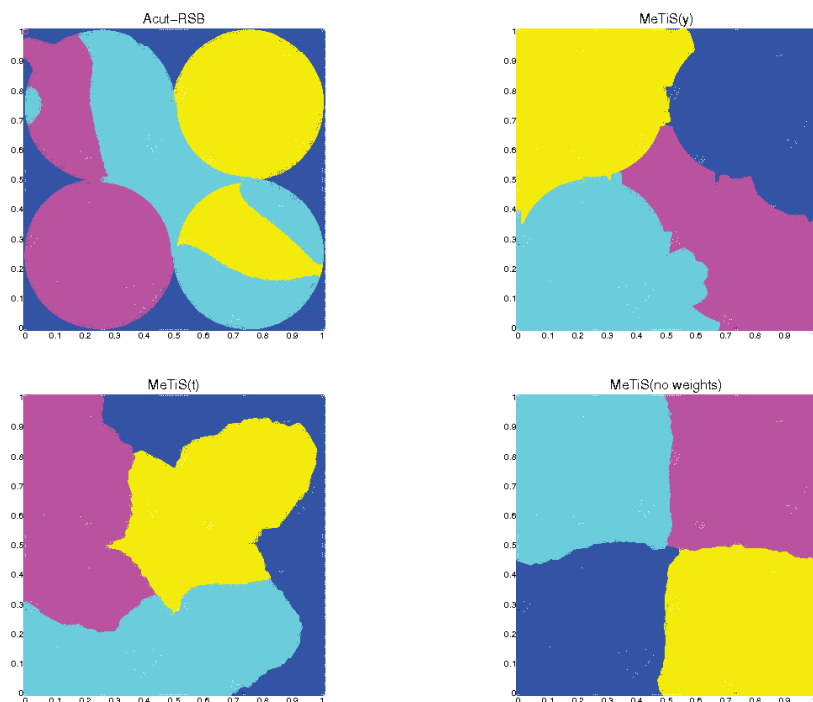
FIG. 4.13. *Partitioning of unstructured mesh (Figure* 4.11*) into four subdomains for problem* (4.1) *with coefficients in* (4.4)*.*
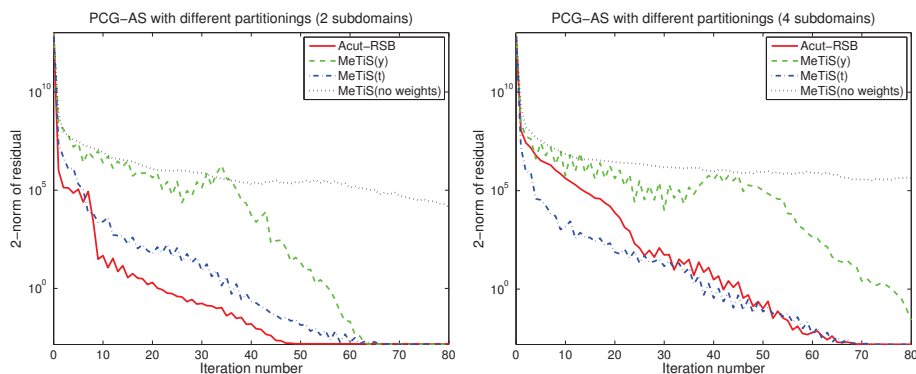

FIG. 4.14. *Convergence of PCG–AS with different partitions for problem* (4.1) *with coefficients in* (4.4)*. The domain and the corresponding FE mesh are given in Figure* 4.11*. The linear system of size* $n = 47{,}713$ *is partitioned into two (left) and four (right) subdomains.*

same as in Figure 4.11. The parameters γ, δ, and LOBPCG convergence tolerance remain unchanged. As above, the mesh is partitioned into two and four subdomains. The results of bipartitioning are shown in Figure 4.15.

Figure 4.16 demonstrates that for the above choice of problem coefficients Acut-RSB delivers partitions that fail to produce a satisfactory preconditioning quality. The convergence of the corresponding PCG–AS with Acut-RSB runs is noticeably inferior to that of PCG–AS with MeTiS partitions.

TABLE 4.4

Relative cut sizes and amounts of the coefficient information discarded to construct preconditioners for problem (4.1) *with coefficients in* (4.4); *see Figure* 4.11.

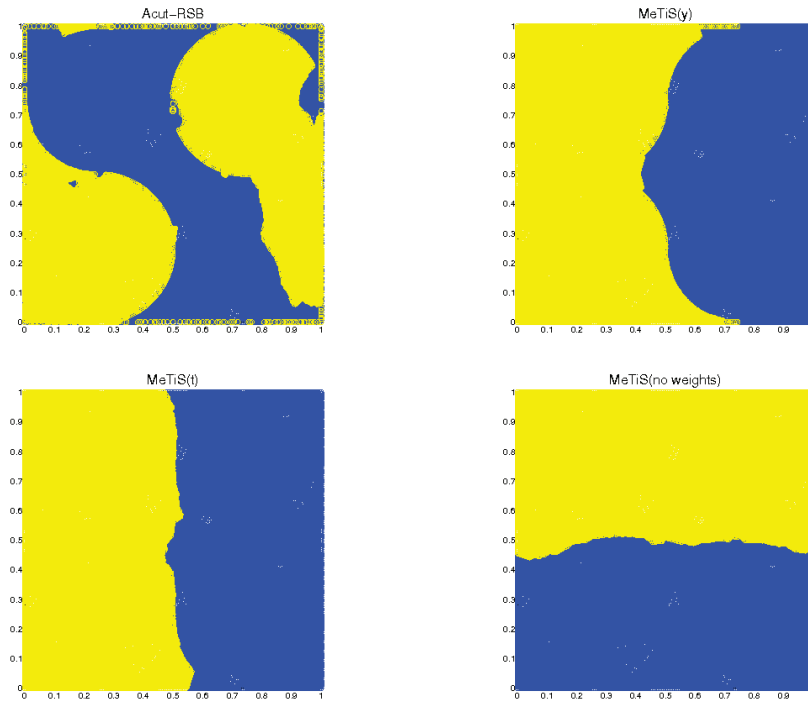| Partitioning	Two subdomains		Four subdomains	
	relcut	relcoef	relcut	relcoef
Acut-RSB	1.03	3×10^{-4}	1.96	0.06
MeTiS(y)	0.46	0.10	0.85	0.16
MeTiS(t)	0.53	5×10^{-4}	1.04	0.01
MeTiS(no w.)	0.28	0.47	0.56	1.24

FIG. 4.15. *Bipartitions of unstructured mesh (Figure* 4.11) *for problem* (4.1) *with coefficients in* (4.5).

We explain this poor behavior by the presence of a large number of edges that connect vertices on the circles' boundaries and those outside the circles; see finer mesh regions along the exterior of the circles' boundaries in Figure 4.11 (right). According to (4.5), such vertices belong to subdomains with different coefficient magnitudes. Therefore, by definition of weights w_{ij}, the corresponding edges are the ones targeted by Acut-RSB; see Figure 4.15 (top left). As a result, the cuts produced by Acut-RSB turn out to be significantly larger (about 5–6 times) than those made by MeTiS; see "relcut" in Table 4.5. Even though the entries $|a_{ij}|$ corresponding to the edges in Acut are relatively small, they accumulate into an excessively large amount of information discarded from A (see "relcoef" in Table 4.5), thereby leading to the inferior convergence of PCG–AS. In other words, in this example, minimization of (3.8) is mainly contributed by the increase of the number of cut edges rather than the decrease of their weight, which appears to hinder the convergence.
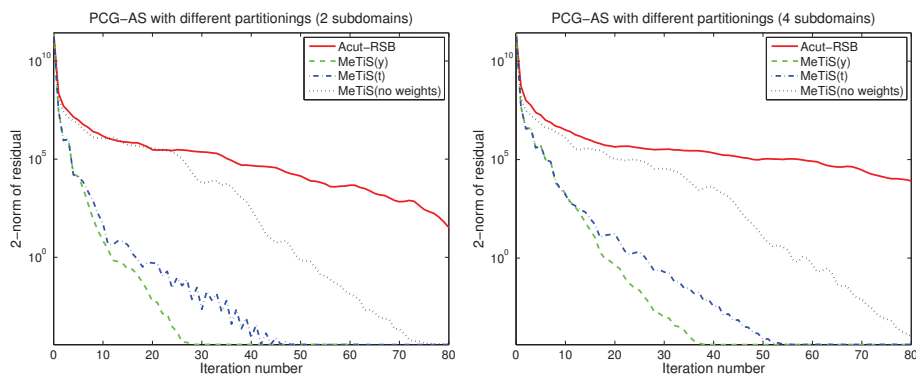
FIG. 4.16. *Convergence of PCG–AS with different partitions for problem* (4.1) *with coefficients in* (4.5). *The domain and the corresponding FE mesh are given in Figure* 4.11. *The linear system of size* $n = 47{,}713$ *is partitioned into two (left) and four (right) subdomains.*

TABLE 4.5
Relative cut sizes and amounts of the coefficient information discarded to construct preconditioners for problem (4.1) *with coefficients in* (4.5); *see Figure* 4.11.

Partitioning	Two subdomains		Four subdomains	
	relcut	relcoef	relcut	relcoef
Acut-RSB	1.80	0.44	3.67	1.51
MeTiS(y)	0.38	0.01	0.76	0.01
MeTiS(t)	0.27	0.01	0.54	0.01
MeTiS(no w.)	0.28	0.10	0.56	0.11

In contrast, the number of edges that connect vertices on the circles' boundaries and those inside the circles is relatively small; see coarser mesh regions along the interior of the circles' boundaries in Figure 4.11 (right). If problem coefficients are defined by (4.4), then these are the edges targeted by Acut-RSB; see Figure 4.12 (top left). However, their number is smaller than that of the edges discarded by the (unsuccessful) run of Acut-RSB with coefficients (4.5). Therefore, as reported in Table 4.4, the values of "relcut" and "relcoef" corresponding to Acut-RSB are not large, i.e., minimization of (3.8) is given by a suitable balance between the cut weight and the number of cut edges. As a result, a better convergence behavior is observed in Figure 4.14.

Finally, let us remark that, unlike in all the previous examples, the best convergence in Figure 4.16 is given by PCG–AS with MeTiS(y). In particular, this shows that it is not clear how to *optimally* define MeTiS weights if the preconditioning quality becomes an objective. We also note that the values of "relcoef" in Table 4.5 for MeTiS(y) and MeTiS(t) are essentially the same (up to the sixth decimal digit), while the convergence of the former is noticeably superior.

5. Conclusion. This paper introduces a new approach for partitioning SPD linear systems. The suggested technique is based on approximating the so-called Acut of the matrix adjacency graph. The information about matrix coefficients is utilized through the graph's edge weights.

The resulting partitioning procedure represents a form of RSB, where each step of the recursion requires solving a generalized eigenvalue problem that simultaneously involves weighted and standard graph Laplacians. It is shown that, for a number of

test problems, the new partitioning significantly improves the quality of the associated nonoverlapping AS preconditioners, compared to MeTiS with several different weighting schemes. In the context of parallel solution, the observed increase in the robustness of the iterative method occurs at a price of extra communication overhead.

The new partitioning strategy is shown to be effective for test linear systems with large variations in matrix coefficients. We have observed that the quality of the result strongly depends on the difference in the magnitude of coefficients, as well as (in terms of PDEs) on the problem geometry and mesh structure. Future research should address the development of practical recommendations on when the proposed partitioning method is preferable to the existing techniques. Since at the current exploratory stage the test problems have been relatively small and simple, it is of interest to benchmark the partitioning quality and performance of the new approach at a larger scale and for more complex geometries.

While in this work we have disregarded the requirement on minimizing communication volume, future research should also address the trade-off between preconditioning quality and parallel efficiency. In particular, we hope that this will lead to the containment of the large cut sizes produced by Acut.

This paper considers only the case of nonoverlapping AS (block Jacobi) preconditioners. However, it is of interest to apply the same partitioning approach for other preconditioning strategies. For example, our experiments (not reported here) suggest that results for the *overlapping* AS are similar to the above reported *nonoverlapping* AS. The overlaps have been introduced by growing a few layers of nodes for each nonoverlapping subdomain.

The new partitioning procedure is heavily rooted in the SPD properties of the coefficient matrix. Therefore, it is not clear if successful results can be obtained for symmetric indefinite or only structurally symmetric matrices.

Finally, we hope that applications of the Acut based partitioning can be found in other areas of science and engineering, such as data clustering and network analysis and logistics.

REFERENCES

[1] O. AXELSSON, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[2] U. V. ÇATALYÜREK AND C. AYKANAT, *Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication*, IEEE Trans. Parallel Distrib. Syst., 10 (1999), pp. 673–693.

[3] T. H. CORMEN, C. E. LEISERSON, R. RIVEST, AND C. STEIN, *Introduction to Algorithms*, 2nd ed., MIT Press and McGraw-Hill, Cambridge, MA, and Boston, MA, 2001.

[4] 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 Proc. 2001 IEEE International Conference on Data Mining (ICDM), IEEE Computer Society, Washington, D.C., 2001, pp. 107–114.

[5] C. R. DOHRMANN, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258.

[6] W. DONATH AND A. HOFFMAN, *Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices*, IBM Technical Disclosure Bulletin, 15 (1972), pp. 938–944.

[7] W. DONATH AND A. HOFFMAN, *Lower bounds for the partitioning of graphs*, IBM J. Res. Develop., 17 (1973), pp. 420–425.

[8] C. M. FIDUCCIA AND R. M. MATTHEYSES, *A linear time heuristic for improving network partitions*, in Proceedings of the 19th IEEE Design Automation Conference, 1982, pp. 175–181.

[9] M. FIEDLER, *Algebraic connectivity of graphs*, Czechoslovak Math. J., 23 (1973), pp. 298–305.

[10] M. FIEDLER, *A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory*, Czechoslovak Math. J., 25 (1975), pp. 619–633.

[11] D. FRITZSCHE, A. FROMMER, AND D. B. SZYLD, *Extensions of certain graph-based algorithms for preconditioning*, SIAM J. Sci. Comput., 29 (2007), pp. 2144–2161.

[12] T. GOEHRING AND Y. SAAD, *Heuristic Algorithms for Automatic Graph Partitioning*, Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1994.

[13] P. HAVE, R. MASSON, F. NATAF, M. SZYDLARSKI, AND T. ZHAO, *Algebraic Domain Decomposition Methods for Highly Heterogeneous Problems*, Technical report, INRIA, hal-00611997, Ver. 1, Le Chesnay, France, 2011.

[14] B. HENDRICKSON AND T. G. KOLDA, *Graph partitioning models for parallel computing*, Parallel Comput., 26 (2000), pp. 1519–1534.

[15] B. HENDRICKSON AND R. LELAND, *The Chaco User's Guide: Version* 2.0, 1994.

[16] B. HENDRICKSON AND R. LELAND, *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16 (1995), pp. 452–469.

[17] B. HENDRICKSON AND R. LELAND, *A multilevel algorithm for partitioning graphs*, in Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (CDROM), 1995.

[18] T. J. R. HUGHES, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover, New York, 2000.

[19] G. KARYPIS AND V. KUMAR, *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version* 4.0, University of Minnesota, Minneapolis, MN, 2009.

[20] G. KARYPIS AND V. KUMAR, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[21] G. KARYPIS AND V. KUMAR, *Multilevel k-way partitioning scheme for irregular graphs*, J. Parallel Distrib. Comput., 48 (1998), pp. 96–129.

[22] B. W. KERNIGHAN AND S. LIN, *An efficient heuristic procedure for partitioning graphs*, Bell System Tech. J., 49 (1970), pp. 291–307.

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

[24] U. LUXBURG, *A tutorial on spectral clustering*, Stat. Comput., 17 (2007), pp. 395–416.

[25] J. MANDEL, B. SOUSEDÍK, AND J. ŠÍSTEK, *Adaptive BDDC in three dimensions*, Math. Comput. Simulation, 82 (2012), pp. 1812–1831.

[26] G. L. MILLER, S. H. TENG, W. THURSTON, AND S. A. VAVASIS, *Automatic mesh partitioning*, in Graph Theory and Sparse Matrix Computations: Graph Theory Issues and Algorithms, A. George, J. Gilbert, and J. Liu, eds., IMA Vol. Math. Appl. 56, Springer, New York, 1993, pp. 57–84.

[27] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.

[28] F. PELLEGRINI, *SCOTCH* 5.1 *User's guide*, Technical report, LaBRI, Talence, France, 2008.

[29] A. POTHEN, H. D. SIMON, AND K.-P. LIOU, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.

[30] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003.

[31] Y. SAAD, *SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations*, http://www-users.cs.umn.edu/~saad/software/SPARSKIT/index.html, 1994.

[32] Y. SAAD AND M. SOSONKINA, *Non-standard parallel solution strategies for distributed sparse linear systems*, in Parallel Computation: Proceedings of ACPC'99, A. U. P. Zinterhof and M. Vajtersic, eds., Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1999.

[33] J. SHI AND J. MALIK, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 22 (2000), pp. 888–905.

[34] H. SIMON, *Partitioning of unstructured problems for parallel processing*, Comput. Syst. Eng., 2 (1991), pp. 135–148.

[35] H. SIMON AND S. H. TENG, *How Good is Recursive Bisection?*, SIAM J. Sci. Comput., 18 (1997), pp. 1436–1445.

[36] B. SMITH, P. E. BJØRSTAD AND W. GROPP, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[37] D. A. SPIELMAN AND S.-H. TENG, *Spectral partitioning works: Planar graphs and finite element meshes*, in 37th Annual Symposium on Foundations of Computer Science (Burlington, VT,

1996), IEEE Comput. Soc. Press, Los Alamitos, CA, 1996, pp. 96–105.

[38] C. WALSHAW AND M. CROSS, *JOSTLE: Parallel multilevel graph-partitioning software—An overview*, in Mesh Partitioning Techniques and Domain Decomposition Techniques, F. Magoules, ed., Civil-Comp Ltd., 2007, pp. 27–58.

[39] A. TOSELLI AND O. WIDLUND, *Domain Decomposition Methods—Algorithms and Theory*, Springer, 2005.