GRAPH PARTITIONING WITH MATRIX COEFFICIENTS FOR SYMMETRIC POSITIVE DEFINITE LINEAR 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 increasing the quality of the additive Schwarz preconditioning for symmetric positive definite linear systems. Numerical results for a set of test problems demonstrate a noticeable improvement in the robustness 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

1. Introduction.

Partitioning of a linear system for its parallel solution typically aims at satisfying the two standard objectives: *minimizing the communication volume* and *maintaining the load balance* among different processors. Both of these requirements are motivated by considerations of the efficiency of the parallel matrix-vector multiplications, which lie in the heart of the iterative solution methods. Once the partitioning is performed, the obtained partitions are further used for constructing parallel preconditioners—another crucial ingredient, contributing into the overall performance of the computational scheme. However, the quality of the resulting preconditioner may depend significantly on the given partition, which, while targeting the efficiency of the parallel matrix-vector multiplication, ignores the nature of the employed preconditioning strategy. The latter often leads to preconditioners of a poor quality, especially in the cases, where the coefficient matrices have entries with large variations in magnitudes.

In the current work, we suggest to remove the requirement on the communication volume and, instead, consider partitionings, which favor the quality of the resulting preconditioner. In particular, we focus on the additive Schwarz (AS) preconditioners, see, e.g., [14, 15], for symmetric positive definite (SPD) linear systems, and present a partitioning algorithm, which aims at optimizing the quality of the AS procedure by attempting to minimize the condition number of the preconditioned matrix, while maintaining the load balance.

The problem of partitioning of a linear system $Ax = b$ is commonly formulated in terms of the adjacency graph $G(A) = (V, E)$ of the coefficient matrix $A = (a_{ij})$.

*This work was supported in part by Iowa State University under the contract DE-AC02-07CH11358 with the U.S. Department of Energy and by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract number DE-AC02-05CH11231 and by the U.S. Department of Energy under the grant DE-FG-08ER25841. The first two authors also benefitted from resources from the Minnesota Supercomputing Institute

†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455, USA ({eugenev, saad}@cs.umn.edu).

‡ Ames Laboratory, Iowa State University, Ames, IA 50011 (masha@scl.ameslab.gov).

Here, $V = \{1, 2, \ldots, n\}$ is the set of vertices (nodes) corresponding to the equations/unknowns of the linear 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 is to partition $G(A)$ into s "nonoverlapping" subgraphs $G_k = (V_k, E_k)$, where $V_k \subseteq V$ and $E_k \subseteq E$, such that

$$\bigcup_{k=1,s} V_k = V, \qquad \bigcap_{k=1,s} V_k = \emptyset\,, \tag{1.1}$$

imposing the additional constraint that the edge cut between G_k is kept to a minimum, while the cardinalities of the vertex sets V_k are approximately the same, i.e., $|V_k| \approx n/s$. Equations and unknowns with numbers in V_k are then typically mapped to the same processor. The requirement on the small edge cut between G_k aims at reducing the cost of communications coming from the parallel matrix-vector multiplication. The condition $|V_k| \approx n/s$ attempts to ensure the load balancing.

The solution of the above-described graph partitioning problem is NP-complete. However, there exist a variety of heuristics, which have been successfully applied to the problem; see, e.g., [11] for an overview. Efficient implementations of relevant algorithms are delivered by a number of graph partitioning software packages, e.g., Chaco [8] and MeTiS [9]. We note that alternative approaches for partitioning of linear systems are known, e.g., based on bipartite graph [7] or hypergraph model [2], however, we do not consider them in this paper.

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} 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 [12].

Indeed, if one considers partitioning as an "early phase" of a preconditioning procedure (which, in the purely algebraic setting, is based solely on the knowledge of A), 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 this work, we address these three question for the case of an SPD linear system and a simple domain decomposition (DD) type preconditioner—the *nonoverlapping* AS. In particular, for a given A, the proposed approach is based on the idea of constructing a (bi)partition, which attempts to minimize an upper bound on the condition number of the preconditioned coefficient matrix over all possible balanced (bi)partitions. The resulting algorithm relies on the computation of eigenvectors corresponding to the smallest eigenvalues of generalized eigenvalue problems, which simultaneously involve the weighted and standard graph Laplacians. Although the formal discussion is focused on the case of the *nonoverlapping* AS, we show numerically that, in practice, adding several "layers" of neighboring nodes to the obtained sets V_k in (1.1) leads to decompositions of V, which provide a good quality of the *overlapping* AS preconditioning.

The paper is organized as following. In Section 2, we recall several known results concerning the block-diagonal preconditioning for SPD matrices. These results

motivate the new partitioning scheme, presented in Section 3. The relation of the introduced approach to the existing graph partitioning schemes is discussed in Section 4. Finally, in Section 5, we report on a few numerical examples.

2. Block-diagonal preconditioning. Let us consider A as a block 2-by-2 matrix, i.e.,

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

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, e.g., the the Preconditioned Conjugate Gradient method (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 $\kappa(T^{-1}A)$ of $T^{-1}A$; $\kappa(T^{-1}A) = \lambda_{max}(T^{-1}A)/\lambda_{min}(T^{-1}A)$ 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. In the subsequent sections, we will need this statement to justify the new partitioning algorithm.

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, $W_1 \cap W_2 = (\mathbf{0}\ \mathbf{0})^*$, it is readily seen that $0 \leq \gamma < 1$. Also we note that γ is the smallest possible constant satisfying the strengthened Cauchy-Schwarz-Bunyakowski inequality

$$|(w_1, Aw_2)| \leq \gamma (w_1, Aw_1)^{1/2}(w_2, Aw_2)^{1/2},$$

which motivates its name.

The CBS constant γ, defined by (2.3), turns to play an important role in estimating the condition number $\kappa(T^{-1}A)$ for the class of SPD matrices A and block-diagonal preconditioners T.

THEOREM 2.2 ([1], Chapter 9). *If $T_1 = A_{11}$ and $T_2 = A_{22}$ in (2.2), and A in (2.1) is SPD, then*

$$\kappa(T^{-1}A) \leq \frac{1+\gamma}{1-\gamma}\,.$$

The bound given by Theorem 2.2 is sharp. In what follows, we use the CBS constants to construct a new approach for graph partitioning.

3. Partitioning with 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 linear 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.

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. *$w = w_1 + \ldots + w_s$.*

In this section, we focus on the case, where sets (subdomains) $\{V_k\}_{k=1}^s$ are nonoverlapping, i.e., (1.1) holds. Algorithm 3.1 then gives a nonoverlapping AS preconditioner, which is a form of the block Jacobi iteration. Indeed, 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

$$T = P^T B P,\ B = \text{blockdiag}\,\{A_1 \ldots, A_s\}\ , \tag{3.1}$$

where $A_k = A(V_k, V_k)$. Thus, Algorithm 3.1 results in the block-diagonal, or block Jacobi, preconditioner, up to a permutation of its rows and columns.

In the following subsection we define an optimal *bipartitioning* for Algorithm 3.1 with two nonoverlapping subdomains.

3.1. Optimal bipartitions. Let us assume that $s = 2$ and consider a *bipartition* of $V = \{1, 2, \ldots, n\}$, where $I \equiv V_1$ and $J \equiv V_2$ are nonempty, such that

$$V = I \cup J, \qquad I \cap J = \emptyset\ . \tag{3.2}$$

The following theorem provides a relation between a given bipartition and the condition number of the preconditioned matrix.

THEOREM 3.1. *Let $\{I, J\}$ in (3.2) be a bipartition of V, where $|I| = m > 0$. Let T be the AS preconditioner for linear system $Ax = b$ with an SPD matrix A, given by Algorithm 3.1, with respect to the bipartition $\{I, J\}$. Then,*

$$\kappa(T^{-1}A) \leq \frac{1+\gamma_{IJ}}{1-\gamma_{IJ}}\ , \tag{3.3}$$

where

$$\gamma_{IJ} = \max_{u\in W_I, v\in W_J} \frac{|(u, Av)|}{(u, Au)^{1/2}(v, Av)^{1/2}} \ . \tag{3.4}$$

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

$$W_I = \{u \in \mathbb{R}^n \ : \ u(J) = \boldsymbol{0}\} \ , \ W_J = \{v \in \mathbb{R}^n \ : \ v(I) = \boldsymbol{0}\} \ . \tag{3.5}$$

Proof. According to (3.1), for the given bipartition $\{I, J\}$ in (3.2), the preconditioner T, constructed by Algorithm 3.1, is of the form

$$T = P^T B P, \ B = \begin{pmatrix} A_I & 0 \\ 0 & A_J \end{pmatrix} , \tag{3.6}$$

where $A_I = A(I, I)$, $A_J = A(J, J)$, and P is a permutation matrix corresponding to the reordering of V with respect to the partition $\{I, J\}$. In particular, for any x, the vector $y = Px$ is such that $y = (x(I)\ x(J))^T$, i.e., the entries of x with indices in I become the first m components of y, while the entries with indices in J get positions from $m+1$ through n.

We observe that the condition number of the matrix $T^{-1}A$ is the same as the condition number of the matrix $B^{-1}C$, where $C = PAP^T$ and B in (3.6). Indeed, since a unitary similarity transformation

$$P(T^{-1}A)P^T = P(P^T B^{-1} P A)P^T = B^{-1}(PAP^T) = B^{-1}C \ ,$$

preserves the eigenvalues of $T^{-1}A$, we have $\kappa(T^{-1}A) = \kappa(B^{-1}C)$, where $\kappa(\cdot) = \lambda_{max}(\cdot)/\lambda_{min}(\cdot)$.

The matrix C represents a symmetric permutation of A with respect to the given bipartition $\{I, J\}$, and, thus, can be written in the 2-by-2 block form,

$$C = PAP^T = \begin{pmatrix} A_I & A_{IJ} \\ A_{IJ}^* & A_J \end{pmatrix} , \tag{3.7}$$

where $A_I = A(I, I)$, $A_J = A(J, J)$, and $A_{IJ} = A(I, J)$. Since C is SPD and the preconditioner B in (3.6) is the block diagonal of C, we apply Theorem 2.2 to get the upper bound on the condition number $\kappa(B^{-1}C)$, and hence bound (3.3) on $\kappa(T^{-1}A)$, where, according to Definition 2.1, the CBS constant $\gamma \equiv \gamma_{IJ}$ is given by

$$\begin{aligned} \gamma_{IJ} &= \max_{w_1\in W_1, w_2\in W_2} \frac{|(w_1, Cw_2)|}{(w_1, Cw_1)^{1/2}(w_2, Cw_2)^{1/2}} \\ &= \max_{w_1\in W_1, w_2\in W_2} \frac{|(w_1, PAP^T w_2)|}{(w_1, PAP^T w_1)^{1/2}(w_2, PAP^T w_2)^{1/2}} \\ &= \max_{w_1\in W_1, w_2\in W_2} \frac{|(P^T w_1, AP^T w_2)|}{(P^T w_1, AP^T w_1)^{1/2}(P^T w_2, AP^T w_2)^{1/2}} \ . \end{aligned}$$

The matrix P^T defines the permutation that is the "reverse" of the one corresponding to P. Thus, the substitution $u = P^T w_1$ and $v = P^T w_2$ leads to expression (3.4)–(3.5) for γ_{IJ}, where the W_I and W_J contain vectors, which can have nonzero entries only at positions defined by I or J, respectively. ∎

While no reasonably simple expression for the condition number of $\kappa(T^{-1}A)$ is generally available, (3.3)–(3.5) provides a sharp upper bound. This suggests that instead of choosing the bipartition which directly minimizes $\kappa(T^{-1}A)$, we look for $\{I, J\}$, such that the *upper bound* in (3.3) is the smallest.

The function $f(x) = (1+x)/(1-x)$ is monotonically increasing. Therefore, the optimal value of $(1+\gamma_{IJ})/(1-\gamma_{IJ})$ in (3.3) is attained for a bipartition $\{I, J\}$ corresponding to the smallest value of the CBS constant γ_{IJ}. Since the targeted partitions are also required to be balanced, throughout this section, we request n to be even, which guarantees the existence of *fully balanced* bipartitions $\{I, J\}$ in (3.2), i.e., such that $|I| = |J| = n/2$. The latter, however, will not be a restriction for the practical algorithm described below. Thus, we suggest an optimal bipartition $\{I_{opt}, J_{opt}\}$ for the nonoverlapping AS preconditioner to be such that

$$\gamma_{opt} \equiv \gamma_{I_{opt}J_{opt}} = \min_{\substack{I,J\subset V=\{1,\ldots,n\}\,,\\ |I|=|J|=\frac{n}{2}, J=V\setminus I}} \max_{u\in W_I, v\in W_J} \frac{|(u,Av)|}{(u,Au)^{1/2}(v,Av)^{1/2}}\,, \tag{3.8}$$

where W_I and W_J are the subspaces defined in (3.5).

3.2. Approximating optimal bipartitions. In the previous subsection, we have shown that the CBS constant $0 \le \gamma_{IJ} < 1$ in (3.4) provides a reasonable quantification of the quality of the bipartition $\{I, J\}$ in terms of the nonoverlapping AS preconditioner with respect to the two subdomains. However, finding an optimal bipartition $\{I_{opt}, J_{opt}\}$ (possibly not unique) according to (3.8), represents a challenging task. Therefore, we further construct bipartitions which *attempt to approximate* the minimizer $\{I_{opt}, J_{opt}\}$ of (3.8), rather than determine it exactly. In particular, we use the following approach.

Let $f(I, J) = \gamma_{IJ}$, with γ_{IJ} in (3.4), be the objective function defined on all possible fully balanced bipartitions $\{I, J\}$ in (3.2). Let $g(I, J) = \tilde{\gamma}_{IJ}$ be some other (simpler) function, which behaves similarly to $f(I, J)$, i.e., the values of $\tilde{\gamma}_{IJ}$ and γ_{IJ} change compatibly with respect to different bipartitions $\{I, J\}$, and

$$f(\operatorname*{argmin}_{I,J\subset V} g(I,J)) \approx \gamma_{opt},\ |I| = |J| = \frac{n}{2},\ J = V\setminus I\,.$$

Then, instead of $f(I, J) = \gamma_{IJ}$, we attempt to minimize $g(I, J) = \tilde{\gamma}_{IJ}$. The constructed minimizer is used to define the bipartition for $Ax = b$ under the nonoverlapping AS preconditioning, given in Algorithm 3.1. Below, we suggest the choice of $g(I, J)$, and describe the resulting bipartitioning procedures.

Given a bipartition $\{I, J\}$ in (3.2), $|I| = |J| = n/2$, let us consider a set of pairs

$$S = \{(e_i, e_j)\ :\ i \in I, j \in J, a_{ij} \neq 0\}\,, \tag{3.9}$$

where $e_k \in \mathbb{R}^n$ denotes the unit vector with 1 at position k and zeros elsewhere. By (3.4), the computation of the CBS constant γ_{IJ} for this bipartition involves finding the maximum in $u \in W_I$ and $v \in W_J$ of the quantity

$$\frac{|(u,Av)|}{(u,Au)^{1/2}(v,Av)^{1/2}}\,. \tag{3.10}$$

Instead, we suggest to evaluate (3.10) on the pairs of the unit vectors $(e_i, e_j) \in S$ in (3.9), and then find the mean of the values which result from this "sampling", i.e., define $\tilde{\gamma}_{IJ} \leq \gamma_{IJ}$, such that

$$\tilde{\gamma}_{IJ} = \frac{1}{|S|} \sum_{i \in I, j \in J} \frac{|a_{ij}|}{\sqrt{a_{ii} a_{jj}}} \, .$$

We note that, in terms of the adjacency graph $G(A)$ of A, $|S|$ is equal to the edge cut between vertex sets I and J, i.e., $\mathrm{cut}(I, J) = |S|$. The expression above can be written as

$$\tilde{\gamma}_{IJ} = \frac{w(I, J)}{\mathrm{cut}(I, J)} \, , \tag{3.11}$$

where $w(I, J) = \sum_{i \in I, j \in J} w_{ij}$ is the weighted cut with $w_{ij} = \frac{|a_{ij}|}{\sqrt{a_{ii} a_{jj}}}$ denoting the weights assigned to the edges of $G(A)$.

Thus, instead of the objective function $f(I, J) = \gamma_{IJ}$, which according to (3.8), results in optimal bipartitions, we suggest to minimize $g(I, J) = \tilde{\gamma}_{IJ}$ in (3.11), i.e., find the minimizer $\left\{ \tilde{I}_{opt}, \tilde{J}_{opt} \right\}$ of

$$\tilde{\gamma}_{opt} = \min_{\substack{I, J \subset V = \{1, \ldots, n\} \, , \\ |I| = |J| = \frac{n}{2}, J = V \setminus I}} \frac{w(I, J)}{\mathrm{cut}(I, J)} . \tag{3.12}$$

Minimization (3.12) represents the problem of bipartitioning of the graph $G(A)$, which has the prescribed edge weights $w_{ij} = \frac{|a_{ij}|}{\sqrt{a_{ii} a_{jj}}}$, with respect to the objective of minimizing the weighted cut normalized by the standard cut. Since, by our assumption A is SPD, the weights w_{ij} are well-defined. The solution of (3.12) is then expected to approximate the optimal partition $\{I_{opt}, J_{opt}\}$, i.e., the minimizer of problem (3.8), which leads to the nonoverlapping AS preconditioner of the optimal quality, in terms of minimizing the upper bound on the condition number of the preconditioned matrix.

Let us reformulate optimization problem (3.12) in terms of bilinear forms involving graph Laplacians. First, we introduce the n-dimensional indicator vector p with the components p_k such that

$$p_k = \begin{cases} 1, & k \in I \, , \\ -1, & k \in J \, . \end{cases} \tag{3.13}$$

Then, for the given bipartition $\{I, J\}$,

$$\begin{aligned} 4w(I, J) &= \sum_{(i,j) \in E} w_{ij} (p_i - p_j)^2 = \sum_{(i,j) \in E} w_{ij} (p_i^2 + p_j^2) - 2 \sum_{(i,j) \in E} w_{ij} p_i p_j \\ &= \sum_{i=1}^{n} d_w(i) p_i^2 - \sum_{i,j=1}^{n} w_{ij} p_i p_j \, , \end{aligned}$$

where $d_w(i) = \sum_{j \in N(i)} w_{ij}$ is the weighted degree of the vertex i; $N(i)$ denotes the set of vertices adjacent to the vertex i. The weighted cut $w(I, J)$ can then be written as a bilinear form evaluated at the indicator vector p,

$$4w(I, J) = p^T L_w p, \qquad L_w = D_w - W \, , \tag{3.14}$$

where $D_w = \mathrm{diag}(d_w(1), \ldots, d_w(n))$ is the weighted degree matrix, $W = (w_{ij})$ is the weighted adjacency matrix ($w_{ij} = w_{ji} = \frac{|a_{ij}|}{\sqrt{a_{ii}a_{jj}}}$, $w_{ii} = 0$) of $G(A)$, and L_w is the corresponding (weighted) graph Laplacian.

Similarly, for the same bipartition $\{I, J\}$, we repeat the above derivations with $w_{ij} \equiv 1$ to get the expression for the (unweighted) cut, i.e,

$$4\mathrm{cut}(I, J) = p^T L p, \qquad L = D - Q\ , \tag{3.15}$$

where D is the diagonal degree matrix, $Q = (q_{ij})$ is the adjacency matrix ($q_{ij} = q_{ji} = 1$, iff $(i, j) \in E$, and $q_{ij} = 0$ otherwise, $q_{ii} = 0$) of $G(A)$, and L is the standard graph Laplacian.

Using expressions (3.14)–(3.15), minimization problem (3.12) can be written as

$$\min_p \frac{p^T L_w p}{p^T L p}, \qquad p^T \mathbf{1} = 0\ , \tag{3.16}$$

where the minimum is searched over all possible indicator vectors p. The condition $p^T\mathbf{1} = 0$ imposes the requirement that $|I| = |J| = n/2$; $\mathbf{1} = (1\ 1\ \ldots\ 1)^T$ is n-vector of ones.

In order to find an approximate solution of (3.16), we relax the requirement on p to be the vectors of ± 1, and embed the problem into the real space. Thus, instead of (3.16), we attempt to find $v \in \mathbb{R}^n$, such that

$$\min_v \frac{v^T L_w v}{v^T L v}, \qquad v^T \mathbf{1} = 0\ , \tag{3.17}$$

where L_w and L are the weighted and the standard graph Laplacians of the adjacency graph of A, respectively.

Both L_w and L are symmetric positive semi-definite. We assume for simplicity that the adjacency graph $G(A)$ is connected, i.e., the nullspace of L is one-dimensional and is spanned by the vector $\mathbf{1}$. We also note that $\mathbf{1}$ is in the nullspace of L_w. Problem (3.17) then corresponds to the minimization of the generalized Rayleigh quotient $\frac{v^T L_w v}{v^T L v}$ on the subspace $\mathbf{1}^\perp$. Since L is SPD on $\mathbf{1}^\perp$, the minimum in (3.17) exists and is given by the smallest eigenvalue of the symmetric generalized eigenvalue problem on $\mathbf{1}^\perp$,

$$L_w v = \lambda L v, \qquad v \in \mathbf{1}^\perp\ . \tag{3.18}$$

The minimizer, i.e., the eigenvector corresponding to the smallest eigenvalue of (3.18), can be viewed as an approximation to the minimizer of the discrete problem (3.16) from the real vector space. The solution of (3.18) is delivered by eigensolvers, which can be applied to the problem, preferably, without factorizing the matrix L, and which can be configured to perform iterations on the subspace $\mathbf{1}^\perp$. In our numerical tests in Section 5, we use the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method; see [10].

A number of possible approaches can be used to define an indicator vector and, hence, the related partition, based on the given eigenvector v corresponding to the smallest eigenvalue of problem (3.18). For example, if the strict requirement on load balancing is enforced, i.e., $|I| = |J| = n/2$ (or $|I| = |J| \pm 1$ if, in practice, n is odd), then the set I is formed by assigning the indices of $n/2$ smallest components of v. The indices of the remaining components form the set J.

In general, however, the requirement on the full load balancing may be restrictive. For example, a slight imbalance in the cardinalities of I and J may lead to a significant improvement in the preconditioner quality. In such cases, ignoring the explicit requirement on the load balancing, one can, e.g., consider all the *negative* components of v as approximations to -1, and assign their indices to the set I. The *nonnegative* components are then considered as approximations to 1, and their indices are put in J. Thus, the sets I and J can be defined as

$$I = \{i \ : \ v_i < 0\}\,, \ J = \{i \ : \ v_i \geq 0\}\,.$$

Similarly, I and J can be formed as

$$I = \{i \ : \ v_i < v_q\}\,, \ J = \{i \ : \ v_i \geq v_q\} \ ,$$

where v_q is a component of v with the median value. A known generalization of this approach is to consider, say, l "candidate" partitions $\{I_j, J_j\}$, such that

$$I_j = \{i \ : \ v_i < \bar{v}_j\}\,, \ J_j = \{i \ : \ v_i \geq \bar{v}_j\}\,, \tag{3.19}$$

where the values $\bar{v}_j$ are some chosen components of the vector $\bar{v} = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})$, e.g., $j = 1, t, 2t, \ldots, lt$; $t = \lfloor \frac{n}{l} \rfloor$. The vector $\bar{v}$ is obtained by sorting the eigenvector v in the ascending order, and determines a linear search order. All l partitions $\{I_j, J_j\}$ are used to evaluate (3.11). The resulting bipartition $\{I, J\}$ is chosen to be the one, which delivers the smallest value of (3.11), i.e., corresponds to the minimizer of (3.12) over the l "candidate" bipartitions in (3.19).

In the partitioning algorithm below, we introduce a parameter *loadBalance*, which controls the sizes of the resulting sets I and J. In particular, the parameter defines a smallest possible size, say $m \leq n/2$, of the sets I and J, so that the indices of the m smallest and m largest components of the eigenvector v are moved into the sets I and J, respectively. The indices of the rest $(n - 2m)$ components are distributed among I and J similarly to (3.19), with $j = m+1, m+t, \ldots, m+lt$, where $t = \lfloor \frac{n-2m}{l} \rfloor$ and $l \leq n - 2m$ is a given parameter.

3.3. The recursive bipartitioning with matrix coefficients. Let $\{I, J\}$ be the bipartition of the set $V = \{1, \ldots, n\}$ resulting from the approach, based on the eigenvalue problem (3.18), discussed in the previous subsection. A natural way to construct further partitions is to apply the bipartitioning process separately to I and J, then to the resulting partitions and so on, until all the computed subpartitions are sufficiently small. The obtained procedure is similar to the well-known Recursive Spectral Bisection (RSB) algorithm, see, e.g., [11], which is based on computing the Fiedler vector [5, 6], i.e., the eigenvector corresponding to the smallest eigenvalue of

$$Lv = \lambda v, \qquad v \in \mathbf{1}^{\perp} \ . \tag{3.20}$$

Each of the subgraphs associated with the sets I and J, which are determined by (3.18), can be, and often is, disconnected. This constitutes an important difference between bipartitions delivered by eigenvalue problems (3.18) and (3.20). At the same time, the assumption on the connectedness is crucial for eigenvalue problem (3.18) to be well-posed for the given graph. Indeed, if a graph has more then one (trivial) connected component, then the dimension of the nullspace of the corresponding graph Laplacian L is larger than one, i.e., the nullspace of L is no longer spanned only by the vector $\mathbf{1}$. In this case, L becomes symmetric positive *semidefinite* on $\mathbf{1}^{\perp}$, and

hence (3.18) does not lead to a correct symmetric eigenvalue problem on $\mathbf{1}^{\perp}$. The latter complication can be simply avoided by searching for the connected components in the subgraphs corresponding to I and J, and further treating the detected components as separate subpartitions in the suggested recursive procedure. Therefore, it is important to realize that the bipartitioning step corresponding to problem (3.18) may result in more than two connected components.

Finally, we note that if the weights $w_{ij} = \frac{|a_{ij}|}{\sqrt{a_{ii}a_{jj}}}$ take a single nonzero value, i.e., are the same for all edges, then the weighted and the standard graph Laplacians L_w and L represent the multiples of each other. This happens for matrices A with a very special, "regular", behavior of entries, e.g., as for the discrete Laplace operator with constant (diffusion) coefficients. In such cases, the eigenvector v, which is used to define the bipartition, corresponds to the only nonzero eigenvalue of (3.18) of multiplicity $n-1$. This observation implies that *any* bipartition can be expected, and, hence, the results of the approach based on (3.18) are highly uncertain. In these situations, we simply replace (3.18), e.g., by eigenvalue problem (3.20), or use any other (bi)partitioning scheme, which targets to satisfy the standard criteria of minimizing the edge cut and maintaining the load balance.

Let us now summarize the discussion into the following partitioning algorithm.

ALGORITHM 3.2 (CBSPartition($G(A)$, A)). *Input:* $G(A)$, A. *Output: Partition* $\{V_i\}$ *of* $\{1, 2, \ldots, n\}$.

1. *Assign weights* $w_{ij} = \frac{|a_{ij}|}{\sqrt{a_{ii}a_{jj}}}$ *to the edges of* $G(A)$. *If* w_{ij} *are the same for all edges, then construct the bipartition* $\{I, J\}$ *using a standard approach, e.g., based on the Fiedler vector, and go to step* 6 *with* $V_1 \equiv I$ *and* $V_2 \equiv J$.
2. *Construct graph Laplacians* $L_w = D_w - W$ *and* $L = D - Q$.
3. *Find the eigenpair corresponding to the smallest eigenvalue of problem (3.18).*
4. *Define the bipartition* $\{I, J\}$ *based on the computed eigenvector. The sizes of* I *and* J *are controlled by the parameter* loadBalance.
5. *Find connected components* $\{G_i \equiv (V_i, E_i)\}$ *in the subgraphs of* $G(A)$ *corresponding to the vertex sets* I *and* J.
6. *For all* G_i *with* $|V_i| >$ maxSize, *apply CBSPartition(*G_i,$A(V_i, V_i)$*). If all* $|V_i| \leq$ maxSize, *then return* $\{V_i\}$.

The parameters *loadBalance* and *maxSize* in Algorithm 3.2 are provided by the user. The connected components can be detected by the standard algorithms based on the breadth-first search (BFS) or the depth-first search (DFS); see, e.g., [3]. Note that every weight w_{ij} assigned to the edge (i, j) is the same at every level of the recursion in Algorithm 3.2, and, in practice, is assigned only once, i.e., when constructing the adjacency graph $G(A)$ of the whole matrix A.

4. Relation to existing graph partitioning techniques. In the previous section, we have used the idea of minimizing the CBS constant γ_{IJ} to define the edge weights for the matrix adjacency graph and obtain objective (3.12) for graph partitioning, which aims at increasing the quality of the AS preconditioner in Algorithm 3.1. Below, we show that similar considerations can lead to the problems of graph partitioning with well-known objectives, such as the (weighted) cut minimization and the Min-max cut [4].

4.1. Relation to the weighted cut minimization (MINcut). Given a partition $\{I, J\}$ in (3.2), $|I| = |J| = n/2$, let us consider the set

$$S = \{(e_i, e_j) \ : \ i \in I, j \in J\} \ , \tag{4.1}$$

where $e_k \in \mathbb{R}^n$ denotes the unit vector with 1 at position k and zeros elsewhere.

Unlike (3.9), the set S in (4.1) contains *all* pairs (e_i, e_j) with $i \in I$ and $j \in J$, i.e., including those, which correspond to $a_{ij} = 0$. Thus, instead of maximizing (3.10) to compute the CBS constant γ_{IJ} in (3.4) for the given bipartition $\{I, J\}$, we sample (3.10) at all $(n/2)^2$ pairs in (4.1) and then find the mean value. In other words, we define the quantity $\bar{\gamma}_{IJ} \le \gamma_{IJ}$, such that

$$\bar{\gamma}_{IJ} = \frac{4}{n^2} \sum_{i \in I, j \in J} \frac{|a_{ij}|}{\sqrt{a_{ii} a_{jj}}} \equiv \frac{4}{n^2} w(I, J) \ , \tag{4.2}$$

where $w(I, J)$ is the weighted cut, and $w_{ij} = \frac{|a_{ij}|}{\sqrt{a_{ii} a_{jj}}}$ are the weights assigned to the edges of the adjacency graph of A. Thus, following the pattern of the previous subsection, instead of the objective function $f(I, J) = \gamma_{IJ}$, which gives an optimal bipartition, we suggest to minimize $g(I, J) = \bar{\gamma}_{IJ}$ in (4.2), i.e., find such a bipartition $\left\{\bar{I}_{opt}, \bar{J}_{opt}\right\}$ that

$$\bar{\gamma}_{opt} = \frac{4}{n^2} \min_{\substack{I, J \subset V = \{1, \ldots, n\}, \\ |I| = |J| = \frac{n}{2}, J = V \setminus I}} w(I, J) \ . \tag{4.3}$$

It is readily seen that (4.3) represents the well-known problem of graph partitioning, which aims at finding equal-sized vertex sets I and J with the minimal (weighted) edge cut. In particular, repeating the derivations in Subsection 3.2 for this problem, we can define the bipartition $\{I, J\}$ according to the components of the eigenvector corresponding to the smallest eigenvalue of

$$L_w v = \lambda v, \qquad v \in \mathbf{1}^{\perp} \ , \tag{4.4}$$

where L_w is the *weighted* graph Laplacian. The recursive application of this procedure leads to the partitioning scheme, which is similar to the standard RSB algorithm with the difference that now edge weights $w_{ij} = \frac{a_{ij}}{\sqrt{a_{ii} a_{jj}}}$ are encapsulated into the graph Laplacian.

4.2. Relation to the Min-max cut (Mcut). Let us further assume that the matrix A is diagonally scaled, i.e.,

$$A \equiv FAF, \ F = \operatorname{diag}\{1/\sqrt{a_{11}}, \ldots, 1/\sqrt{a_{nn}}\} \ . \tag{4.5}$$

In this case, the diagonal entries of A are all equal to 1, and the off-diagonal elements are less than one. In particular, we note that the weights w_{ij} defined in the previous sections simply coincide with the entries of the scaled matrix.

Given a partition $\{I, J\}$ in (3.2) of $V = \{1, \ldots, n\}$, let us consider the set of pairs

$$S = S_1 \cup S_2 \ , \tag{4.6}$$

where

$$S_1 = \{(e_i, v_i) \ : \ i \in I\} \, , \ S_2 = \{(u_j, e_j) \ : \ j \in J\} \, ,$$

and $e_k \in \mathbb{R}^n$ denotes the unit vector with 1 at position k and zeros elsewhere. The vectors v_i in S_1 are defined to have components $v_i(k)$, such that

$$v_i(k) = \begin{cases} 0, & k \in I, \\ \operatorname{sign}(a_{ik}), & k \in J. \end{cases} \tag{4.7}$$

Similarly, the vectors u_j in S_2 are such that

$$u_j(k) = \begin{cases} \operatorname{sign}(a_{jk}), & k \in I, \\ 0, & k \in J. \end{cases} \tag{4.8}$$

Now, following the approach exploited throughout this paper, instead of maximizing (3.10) to compute the CBS constant γ_{IJ} in (3.4) for the given bipartition $\{I, J\}$, we sample (3.10) at all n pairs in (4.6) and find the mean value. In particular, recalling that the diagonal entries of A are equal to 1 after the diagonal scaling, we get

$$\frac{1}{n}\left(\sum_{i\in I}\frac{\sum_{j\in J}|a_{ij}|}{\left[\sum_{k,l\in J}a_{kl}\operatorname{sign}(a_{il})\operatorname{sign}(a_{ik})\right]^{1/2}}+\sum_{j\in J}\frac{\sum_{i\in I}|a_{ji}|}{\left[\sum_{k,l\in I}a_{kl}\operatorname{sign}(a_{jl})\operatorname{sign}(a_{jk})\right]^{1/2}}\right)\geq$$

$$\frac{1}{n}\left(\sum_{i\in I}\frac{\sum_{j\in J}|a_{ij}|}{\left[\sum_{k,l\in J}|a_{kl}|\right]^{1/2}}+\sum_{j\in J}\frac{\sum_{i\in I}|a_{ji}|}{\left[\sum_{k,l\in I}|a_{kl}|\right]^{1/2}}\right)\geq\frac{1}{n}\left(\frac{\sum_{i\in I,j\in J}|a_{ij}|}{\sum_{k,l\in J}|a_{kl}|}+\frac{\sum_{j\in J,i\in I}|a_{ji}|}{\sum_{k,l\in I}|a_{kl}|}\right),$$

and define the quantity $\hat{\gamma}_{IJ} \leq \gamma_{IJ}$, such that

$$\hat{\gamma}_{IJ} = \frac{1}{n}\left(\frac{w(I,J)}{w(J)}+\frac{w(I,J)}{w(I)}\right), \tag{4.9}$$

where $w(I,J) = \sum_{i\in I, j\in J}|a_{ij}| = \sum_{j\in J, i\in I}|a_{ji}|$ is the weighted cut between sets I and J with edge weights $w_{ij} = a_{ij}$; $w(I) = \sum_{k,l\in I}|a_{kl}|$ and $w(J) = \sum_{k,l\in J}|a_{kl}|$. Thus, instead of the objective function $f(I,J) = \gamma_{IJ}$, which gives an optimal bipartition, one can attempt to minimize $g(I,J) = \hat{\gamma}_{IJ}$ in (4.9), i.e., find the minimizer $\left\{\hat{I}_{opt}, \hat{J}_{opt}\right\}$ of

$$\hat{\gamma}_{opt} = \frac{1}{n}\min_{\substack{I,J\subset V=\{1,\ldots,n\},\\ J=V\setminus I}}\left(\frac{w(I,J)}{w(J)}+\frac{w(I,J)}{w(I)}\right). \tag{4.10}$$

Minimization (4.10) represents the problem of finding the so-called Min-max cut (Mcut); see [4]. We note that the explicit requirement $|I| = |J| = n/2$ has been dropped in (4.10), however, the Mcut is known to target balanced partitions. The corresponding algorithm, which attempts to construct $\{I, J\}$ satisfying (4.10), is based on finding the eigenpair corresponding to the smallest eigenvalue of the problem

$$L_w v = \lambda D_w v, \qquad v \in \mathbf{1}^{\perp_{D_w}}, \tag{4.11}$$

where L_w is the weighted graph Laplacian and D_w is the diagonal weighted degree matrix. The recursive application of this procedure delivers the actual partitioning scheme. We remark that eigenvalue problem (4.11) is also used to construct the normalized cuts (Ncuts), introduced in [13].

5. Numerical results. In this section, we present the results of graph partitioning with respect to the new objective (3.12), and demonstrate the effects of the proposed partitioning strategy on the quality of preconditioning. In our numerical experiments, we apply Algorithm 3.2 (CBSPartition), introduced in Subsection 3.3, to a number of test problems with SPD coefficient matrices. The resulting partitions (subdomains) are passed as an input to the AS preconditioner in Algorithm 3.1, which is used to accelerate the convergence of the PCG method. We refer to this solution scheme as PCG–AS.

In order to assess the quality of the constructed AS preconditioners, we consider the iteration counts of PCG–AS for different partitioning schemes. In particular, we compare PCG–AS with partitions resulting from Algorithm 3.2 versus PCG–AS with the standard partitioning based on the RSB algorithm. We also provide the comparisons for PCG–AS with partitioning schemes based on the (weighted) MINcut and Mcut objectives, discussed in Section 4. Although, throughout the paper, the formal discussion has been concerned only with the case of the nonoverlapping AS procedure, in some of our numerical examples, we skip this theoretical limitation and expand the obtained partitions with several "layers" of neighboring nodes. This allows considering the effects of the partitioning strategies on the quality of the *overlapping* AS preconditioners.

In all the tests, the right-hand side and the initial guess vectors are randomly chosen. We apply PCG–AS to the diagonally scaled linear systems, so that the corresponding coefficient matrices have 1 on the diagonal; see (4.5). In this case, the weights w_{ij}, assigned to the edges of the adjacency graph, are equal to the entries of the scaled matrix. For all partitioning schemes, the parameter *loadBalance*, which controls the load balancing, is set to 0.8; $loadBalance \leq \min\{|I|/|J|, |J|/|I|\} \leq 1$.

The partitioning algorithms that we consider in the current paper are based on finding the eigenvectors of certain eigenvalue problems, i.e., represent the *spectral* partitioning techniques. We recall that Algorithm 3.2 computes an eigenpair of problem (3.18). The RSB algorithm targets the Fiedler vector in (3.20). The approaches based on the (weighted) MINcut and Mcut use the eigenvectors of problems (4.4) and (4.11), respectively. In our numerical examples, as an underlying eigensolver, we use the LOBPCG method, which allows to handle generalized symmetric eigenvalue problems, such as (3.18), without any factorizations of the matrices L_w and L, and can be easily configured to perform iterations on the subspace $\mathbf{1}^{\perp}$.

The LOBPCG algorithm is a form of a (block) three-term recurrence, which performs the local minimization of the Rayleigh quotient; see [10] for more details. The method is known to be practical for large-scale eigenvalue computations, if a good preconditioner is provided to accelerate the convergence to the desired eigenpairs. For all LOBPCG runs, we construct preconditioners using incomplete Cholesky (IC) factors with the drop tolerance 10^{-3} of matrices $L_w + \sigma I$ (for problems (3.18), (4.4), and (4.11)) and $L + \sigma I$ (for problem (3.20)). The parameter σ is assigned with a small value, $\sigma = 0.1$ in our examples, to ensure that the IC procedure is correctly applied to the SPD matrices. For problems (3.20), (4.4), and (4.11), we remove the orthogonality constraints on v, and perform the block iteration, with the block size 2. The solution is then given by the eigenpair corresponding to the second smallest

eigenvalue. For problem (3.18), however, we run a single vector iteration, choosing the initial guess from $\mathbf{1}^\perp$ and ensure that the residuals are projected back to $\mathbf{1}^\perp$ after the IC preconditioning. In all the tests, the LOBPCG residual norm tolerance is set to 10^{-4}.

As a model problem we choose the 2D diffusion equation

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

with zero Dirichlet boundary conditions. The functions $a(x,y)$ and $b(x,y)$ are piecewise constants with jumps in the specified subregions. For all tests, we use the standard 5-point FD discretization on 22-by-22 uniform grid to obtain a (diagonally scaled) linear systems of size $n=20^2$. We consider two geometries for the location of jumps in coefficients $a(x,y)$ and $b(x,y)$. In the first case, the jumps occur in the subregion $0.25<x,y<0.75$ of the domain. The second case is more complex, with jumps located in the "checkerboard" fashion ("5-by-5 black-white checkerboard").

In the following example, we assume that *both* coefficients $a(x,y)$ and $b(x,y)$ have jumps in the subregion $0.25<x,y<0.75$ of the problem's domain, such that

$$a(x,y)=b(x,y)=\begin{cases}100, & 0.25<x,y<0.75\\ 1, & \text{otherwise}\ .\end{cases} \tag{5.1}$$

Figure 5.1 shows the result of a single step, i.e., the bipartitioning, performed by Algorithm 3.2 (top left), the standard RSB approach (top right), as well as the weighted MINcut (bottom left) and the Mcut (bottom right) algorithms. We observe that, unlike the rest, the partition resulting from Algorithm 3.2 does not perform "cuts" within the jump region. In fact, this is consistent with the well-established computational experience, which suggests that the subdomains with different physics phenomena should be "segmented" out of the original domain.

Similarly, we apply the bipartitioning step of all four approaches to the model problem, where the jump occurs only in the coefficient $a(x,y)$, i.e.,

$$a(x,y)=\begin{cases}100, & 0.25<x,y<0.75\\ 1, & \text{otherwise}\end{cases}\ ,\quad b(x,y)=1\ . \tag{5.2}$$

The resulting bipartitions are illustrated in Figure 5.2. We note that the partitions given by Algorithm 3.2 (top left), the weighted MINcut (bottom left) and the Mcut (bottom right) algorithms do not discard the mesh edges in the y-direction within the jump region. The border between the subdomains is of a "smoother" shape for the MINcut and Mcut, which is preferable in terms of minimizing the communication volume. However, as suggested by the numerical experiments below, the partitions based on Algorithm 3.2 typically guarantee a smaller number of steps of the iterative scheme as the number of the desired subdomains becomes larger. We also remark that, although independent of the matrix coefficients, the partitions resulting from the standard RSB approach may not be unique, see Figures 5.1 and 5.2 (top right), if the targeted eigenvalue has multiplicity greater than one.

In Figure 5.3, we plot the components of the eigenvectors, corresponding to the smallest eigenvalues of (3.18), at the grid points. According to Algorithm 3.2, such eigenvectors are used to construct the bipartition. Indeed, the components of the eigenvectors are well-separated, which allows to easily determine the partitions and detect the "weak" connection between the grid nodes to be discarded in order to obtain the resulting bipartition.

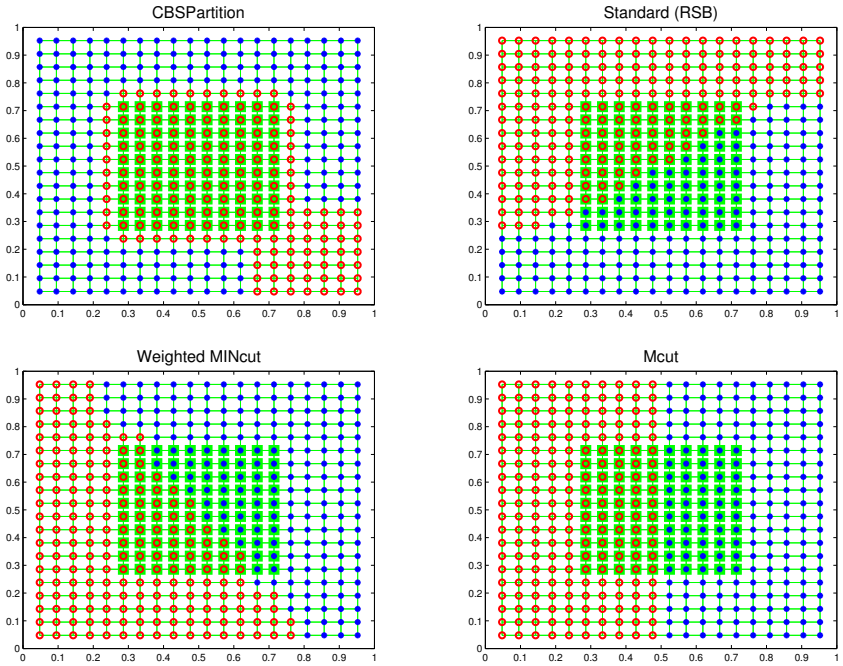

FIG. 5.1. *Bipartitions for the model problem in the case of the jump in both $a(x,y)$ and $b(x,y)$. The shaded bolder nodes (green for the color plot) correspond to the "jump region" $(0.25, 0.75) \times (0.25, 75)$. The nodes in the partitions I and J are marked with "$\circ$" and "$\bullet$", respectively.*

Figure 5.4 demonstrates the effects of partitioning schemes on the quality of the nonoverlapping AS preconditioners. In Figure 5.4 (left), we consider the PCG–AS runs for the model problem with coefficients in (5.1), where the AS preconditioners are constructed with respect to the partitions delivered by different algorithms. The parameter *maxSize*, which determines the largest possible size of a single subdomain, is set to 190 in this example. In Figure 5.4 (right), we apply PCG–AS for the model problem with coefficients in (5.2). Here, the parameter *maxSize*, is set to 50. This corresponds to a more realistic situation, where each subdomain is significantly smaller than the original domain.

The results in Figure 5.4 suggest that PCG–AS with the partitioning scheme in Algorithm 3.2 requires a noticeably smaller number of iterations to get the solution. We remark that the quality of the AS preconditioning with respect to the partitions delivered by Algorithm 3.2 may often depend on the value of the parameter *maxSize*. For example, in the case of the model problem with coefficients in (5.1), a small value of *maxSize* forces Algorithm 3.2 to perform the partitioning inside the jump region, i.e., inside the subdomain marked with "o"'s in Figure 5.1 (top left). This, clearly, can lead to less remarkable gains in the use of Algorithm 3.2 compared to the other partitioning approaches.

In the following pair of examples, we assume the "checkerboard" pattern for the jump regions ("5-by-5 black-white checkerboard"). First, we consider jumps in both $a(x,y)$ and $b(x,y)$ in "black" positions, i.e., $a(x,y) = b(x,y) = 100$ in "black", and $a(x,y) = b(x,y) = 1$ in "white". The corresponding bipartitions resulting from different partitioning schemes are given in Figure 5.5. Similarly, Figure 5.6 demonstrates

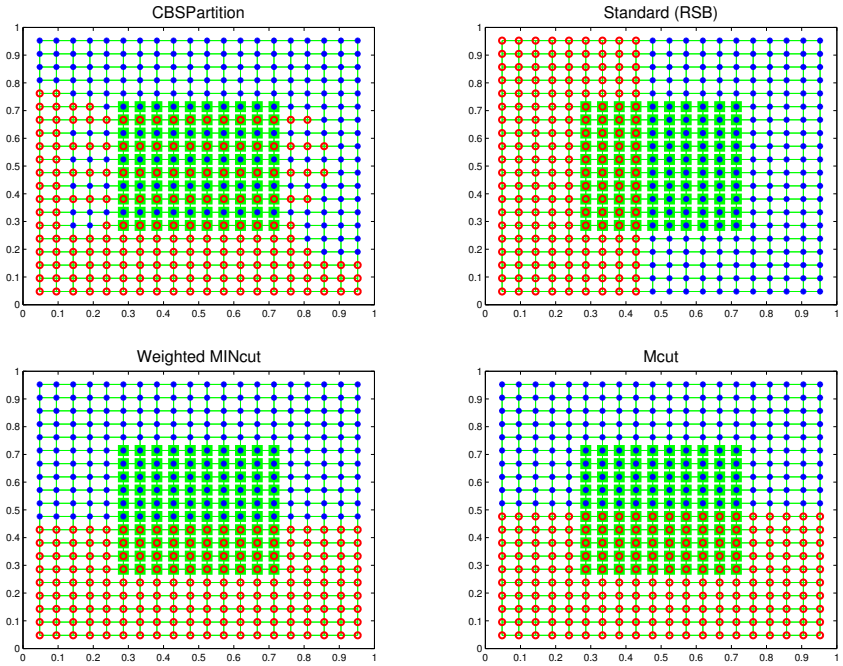

FIG. 5.2. *Bipartitions for the model problem in the case of the jump only in $a(x,y)$. The shaded bolder nodes (green for the color plot) correspond to the "jump region" $(0.25,0.75)\times(0.25,75)$. The nodes in the partitions I and J are marked with "$\circ$" and "$\bullet$", respectively.*

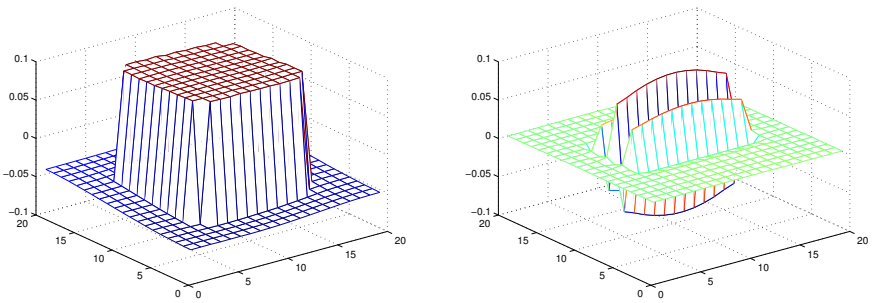

FIG. 5.3. *The "mesh" plot of components of the eigenvector v corresponding to the smallest eigenvalue of (3.18). The left figure corresponds to the case of the jump in both $a(x,y)$ and $b(x,y)$. The right figure corresponds to the case of the jump only in $a(x,y)$. In both cases, the "jump region" is $(0.25,0.75)\times(0.25,75)$.*

the bipartitions corresponding to the second case, with jumps only in $a(x,y)$, i.e., $a(x,y)=100$ in "black", and $a(x,y)=1$ in "white", $b(x,y)=1$.

We recall that the bipartitioning step of Algorithm 3.2 may deliver two *disconnected* subdomains, i.e., two subgraphs which possibly contain more than one connected component. Each of these connected components is then processed separately in the recursive partitioning procedure. This explains the presence of more than two subdomains in (top left) Figures 5.5 and 5.6—the nodes of each connected component,

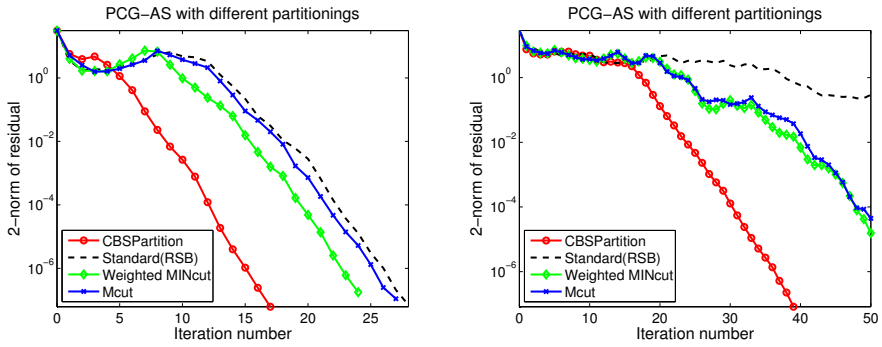

FIG. 5.4. *The convergence of PCG–AS with different partitions for the model problem with the "jump region" located in* $(0.25, 0.75) \times (0.25, 75)$*;. The left figure corresponds to the jump in both* $a(x, y)$ *and* $b(x, y)$*;* maxSize $= 190$*. The right figure corresponds to the jump only in* $a(x, y)$*;* maxSize $= 50$*. In both cases,* loadBalance $= 0.8$*;* $n = 400$*.*

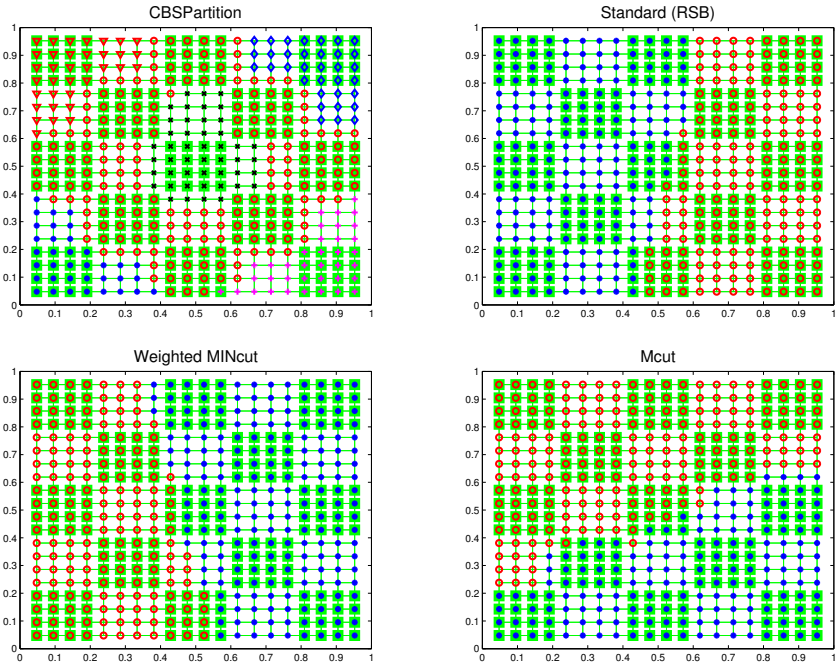

FIG. 5.5. *Bipartitions for the model problem in the case of the jump in both* $a(x, y)$ *and* $b(x, y)$*. The shaded bolder nodes (green for the color plot) correspond to the "jump regions" located in the "*5*-by-*5 *checkerboard" fashion. In the top right, bottom left, and bottom right figures, the nodes in the partitions* I *and* J *are marked with "*$\circ$*" and "*$\bullet$*", respectively. The top left figure exhibits the* 6 *connected components resulting from a single (bipartitioning) step of Algorithm 3.2. The nodes in each of the connected components are marked with "*$\circ$*", "*$\bullet$*", "*$\diamond$*", "*$\triangledown$*", "*$\times$*", and "*$+$*", respectively.*

resulting from a single step of Algorithm 3.2, are plotted as separate regions. We also note that the recursive bipartitioning given by Algorithm 3.2 may generate a number of small connected subdomains of sizes much smaller than the value of the *maxSize* parameter. Such subdomains should be treated with care when being assigned to parallel processors.

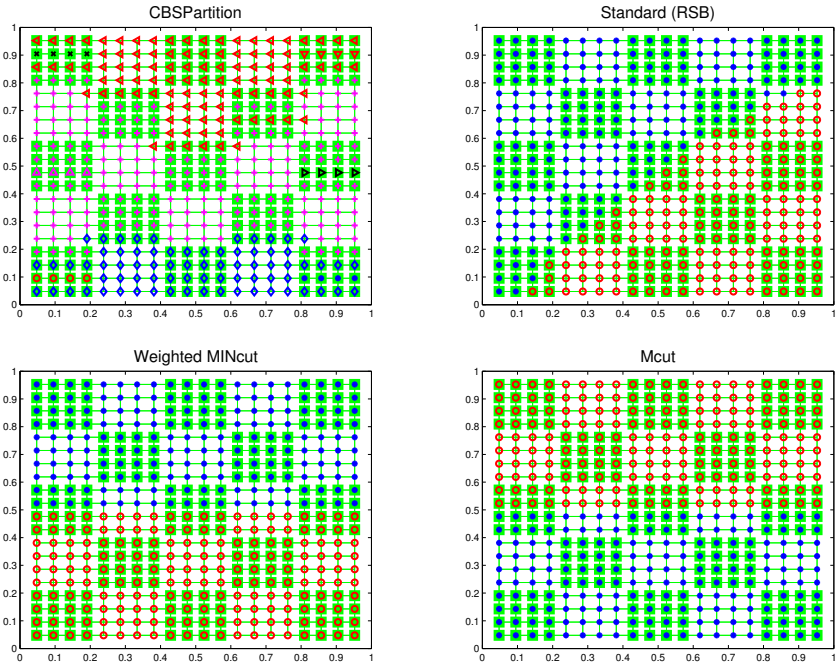

FIG. 5.6. *Bipartition for the model problem in the case of the jump only in $a(x, y)$. The shaded bolder nodes (green for the color plot) correspond to the "jump regions" located in the "5-by-5 checkerboard" fashion. In the top right, bottom left, and bottom right figures, the nodes in the partitions I and J are marked with "$\circ$" and "$\bullet$", respectively. The top left figure exhibits the 9 connected components resulting from a single (bipartitioning) step of Algorithm 3.2. The nodes belonging to the three largest connected components are marked with "$\diamond$", "$+$", and "$\triangleleft$". The nodes in the remaining components are marked with "$\circ$", "$\bullet$", "$\triangle$","$\triangledown$", "$\triangleright$", "$\times$", and "$+$", respectively.*

In Figure 5.7, we plot the components of the eigenvectors corresponding to the smallest eigenvalues of (3.18) at the grid points for the "checkerboard" example. It is possible to see that, as in the previous examples, both eigenvectors attempt to capture the discontinuities in the coefficients of the model problem.

In Figure 5.8, we compare the convergence behavior of PCG–AS with different partitioning schemes. Figure 5.8 (left) corresponds to the case of the jumps in both $a(x, y)$ and $b(x, y)$ in "black" positions. We observe that for this relatively complex geometry of the jump locations, all partitioning schemes which use the information on the matrix coefficients result in the AS preconditioners of a better quality. In this example, the number of PCG–AS iterations is typically similar for the partitioning techniques in Algorithm 3.2, the weighted MINcut, and Mcut. Figure 5.8 (right) demonstrates the runs of PCG–AS applied to the model problem with the jump only in $a(x, y)$ in "black" positions. In this case, the iterative scheme with partitions resulting from Algorithm 3.2 gives the fastest convergence. In both, Figure 5.8 (left) and Figure 5.8 (right), the *maxSize* parameter has been set to 50.

Finally, we apply the partitioning schemes, discussed in this paper, to a set of test problems from the University of Florida Sparse Matrix Collection. In particular, we consider ill-conditioned SPD matrices arising in structural engineering and computational fluid dynamics. In Tables 5.1 and 5.2, we report the numbers of iterations (averaged after 3-4 sample runs) required by PCG–AS to reach the tolerance

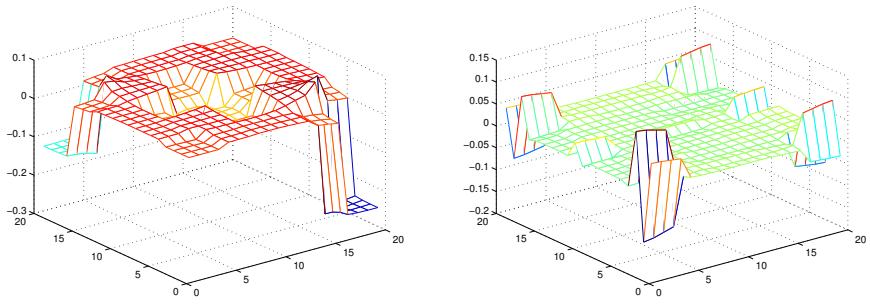

FIG. 5.7. *The "mesh" plot of components of the eigenvector v corresponding to the smallest eigenvalue of (3.18). The left figure corresponds to the case of the jump in both $a(x,y)$ and $b(x,y)$. The right figure corresponds to the case of the jump only in $a(x,y)$. In both cases, the "jump regions" are located in the "5-by-5 checkerboard" fashion.*

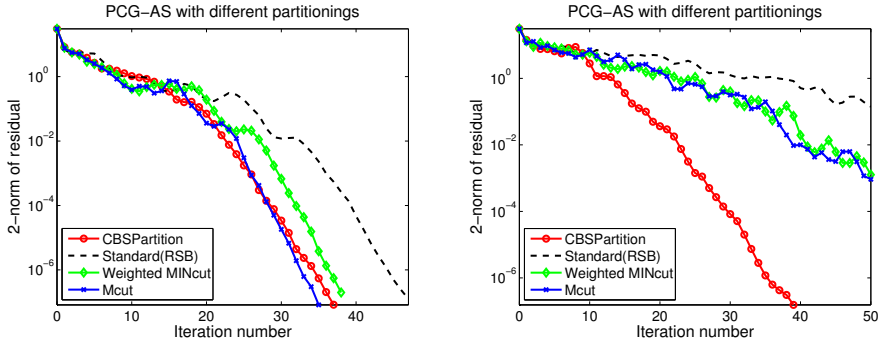

FIG. 5.8. *The convergence of PCG–AS with different partitions for the model problem with the "jump regions" located in the "5-by-5 checkerboard" fashion. The left figure corresponds to the jump in both $a(x,y)$ and $b(x,y)$. The right figure corresponds to the jump only in $a(x,y)$. In both cases,* maxSize $= 50$ *and* loadBalance $= 0.8$*; $n = 400$.*

10^{-8} in the residual norm (relative to the norm of the right-hand side vectors) for *nonoverlapping* and *overlapping* AS procedures, respectively.

In all the tests, we choose the right-hand side to be random of a unit norm. The PCG–AS runs are then applied to the diagonally scaled linear system. For each test, we set the parameter *maxSize* to $\lfloor \frac{n}{20} \rfloor$, where n is the problem size. The size of the overlap for Table 5.2 is set to 2. As has been previously mentioned, the construction of the overlapping subdomains is based on the (nonoverlapping) partitions resulting from the partitioning schemes under consideration, which are expanded by several "layers" of neighboring nodes. The numerical results in the tables suggest that the use of matrix coefficients has, in general, a positive effect on the convergence speed of the iterative method. The partitioning scheme given by Algorithm 3.2 typically provides the lowest number of PCG–AS iterations to reach the desired tolerance for both, overlapping and nonoverlapping, AS preconditioning procedures.

6. Concluding remarks. In the present paper, we have shown that using matrix coefficients for graph partitioning allows to achieve a noticeable decrease in the number of iterations performed by an iterative scheme. For a class of SPD matrices and AS preconditioners, we have suggested an approach for assigning weights to the

TABLE 5.1
*Iteration numbers of the **nonoverlapping** PCG–AS with different partitioning schemes applied to test problems from the University of Florida Sparse Matrix Collection.*

		PCG iterations (nonoverlapping AS)			
Matrix	n	CBSPart	MINcut	Mcut	Standard
BCSSTK13	2003	663	683	636	889
BCSSTK14	1806	147	204	187	290
BCSSTK15	3948	245	265	283	337

TABLE 5.2
*Iteration numbers of the **overlapping** PCG–AS with different partitioning schemes applied to test problems from the University of Florida Sparse Matrix Collection.*

		PCG iterations (overlapping AS)			
Matrix	n	CBSPart	MINcut	Mcut	Standard
BCSSTK13	2003	111	135	128	142
BCSSTK14	1806	49	52	58	54
BCSSTK15	3948	85	98	107	106
BCSSTK27	1224	29	62	54	61
ex3	1821	76	109	100	119
ex10hs	2548	49	60	60	66
ex15	6867	78	115	117	124
ex33	1733	44	107	68	104

edges of the adjacency graph and formulated a new partitioning objective, which aims at approximately minimizing the CBS constant. The resulting partitioning algorithm is based on computing the eigenpairs corresponding to the smallest eigenvalues of the sequence of generalized eigenvalue problems, which involve both weighted and standard graph Laplacians. In particular, this means that the proposed technique inherits all specificities of spectral partitioning algorithms, such as good quality of partitions, on the one hand, and the computational expenses related to finding eigenvectors, on the other hand. Thus, in order to obtain highly efficient graph partitioning schemes, it is important to study all aspects of the occurring eigencomputations, such as, e.g., preconditioning, the use of alternative eigenvalue solvers, possible ways to replace the eigenvalue problems by linear systems. Other approaches for satisfying the suggested partitioning objective may be delivered, e.g., by (multilevel) combinatorial graph partitioning techniques or by certain extensions of greedy algorithms. We note that methods for reaching the new partitioning objective may be combined with the communications minimizing techniques.

As one could conclude from this work, it is likely that different choices of iterative methods and preconditioning strategies may require different schemes for graph partitioning with matrix coefficients. In the current paper, we have considered the case of PCG with the AS preconditioning. Exploring the partitioning for other forms of parallel preconditioning (e.g., incomplete factorizations and multigrid) is a natural continuation of the research in this direction. Constructing partitioning algorithms with matrix coefficients for nonsymmetric problems is also of a particular interest.

REFERENCES

[1] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, New York, NY, 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, 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*, Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM), 107–114. IEEE Computer Society, Washington (2001).
[5] M. Fiedler, *Algebraic connectivity of graphs*, Czech. Math. J., 23 (1973), pp. 298–305.
[6] M. Fiedler, *A property of eigenvectors of non-negative symmetric matrices and its application to graph theory*, Czech. Math. J., 25 (1975), pp. 619–633.
[7] B. Hendrickson and T. G. Kolda, *Graph partitioning models for parallel computing*, Parallel Computing, 26 (2000), pp. 519–1534.
[8] B. Hendrickson and R. Leland, *The Chaco User's Guide: Version 2.0*, 1994.
[9] G. Karypis and V. Kumar, *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0*, University of Minnesota, Minneapolis, MN, 2009.
[10] A. V. Knyazev, *Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.
[11] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.
[12] Y. Saad and M. Sosonkina, *Non-standard parallel solution strategies for distributed sparse linear systems*, in Parallel Computation: Proc. of ACPC'99, A. U. P. Zinterhof, M. Vajtersic, ed., Lecture Notes in Computer Science, Berlin, 1999, Springer-Verlag.
[13] J. Shi and J. Malik, *Normalized Cuts and Image Segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 22 (2000), pp. 888–905.
[14] B. Smith, P. E. Bjørstad and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.
[15] A. Toselli and O. Widlund, *Domain Decomposition Methods – Algorithms and Theory*, Springer, 2005.