COMBINATORIAL PRECONDITIONERS

SIVAN TOLEDO AND HAIM AVRON

TEL-AVIV UNIVERSITY

1. Introduction

The Conjugate Gradient (CG) method is an iterative algorithm for solving linear systems of equations $Ax = b$, where A is symmetric and positive definite. The convergence of the method depends on the spectrum of A; when its eigenvalues are clustered, the method converges rapidly. In particular, CG converges to within a fixed tolerance in $O(\sqrt{\kappa})$, where $\kappa = \kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$ is the spectral condition number of A.

When the spectrum of A is not clustered, a *preconditioner* can accelerate convergence. The Preconditioned Conjugate Gradients (PCG) method applies the CG iteration to the linear system $(B^{-1/2}AB^{-1/2})(B^{1/2}x) = B^{-1/2}b$ using a clever transformation that only requires applications of A and B^{-1} in every iteration; B also needs to be symmetric positive definite. The convergence of PCG is determined by the spectrum of $(B^{-1/2}AB^{-1/2})$, which is the same as the spectrum of $B^{-1}A$. If a representation of B^{-1} can be constructed quickly and applied quickly, and if $B^{-1}A$ has a clustered spectrum, the method is very effective. There are also variants of PCG that require only one of A and B to be positive definite, and variants that allow them to be singular, under some technical conditions on their null spaces.

Combinatorial preconditioning is a technique that relies on graph algorithms to construct effective preconditioners. The simplest applications of combinatorial preconditioning target a class of matrices that are isomorphic to weighted undirected graph. The coefficient matrix A is viewed as its isomorphic graph G_A. A specialized graph algorithm constructs another graph G_B such that the isomorphic matrix B is a good preconditioner for A. The graph algorithm aims to achieve two goals: the inverse of B should be easy to apply, and the spectrum of $B^{-1}A$ should be clustered. It turns out that the spectrum of $B^{-1}A$ can be bounded in terms of properties of the graphs G_A and G_B; in particular, the quality of embeddings of G_A in G_B (and sometimes vice versa) plays a fundamental role in these spectral bounds.

This chapter focuses on explaining the relationship between the spectrum of $B^{-1}A$ and quantitative properties of embeddings of the two graphs. The last section surveys algorithms that construct combinatorial preconditioners.

We omit most proofs from this chapter; some are trivial, and the others appear in the paper cited in the statement of the theorem or lemma.

Date: January, 2010.

This research was supported in part by an IBM Faculty Partnership Award and by grant 1045/09 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities).

2. Symmetric Diagonally-Dominant Matrices and Graphs

We begin by exploring the structure of diagonally-dominant matrices and their relation to graphs.

2.1. Incidence Factorizations of Diagonally-Dominant Matrices.

Definition 2.1. A square matrix $A \in \mathbb{R}^{n\times n}$ is called *diagonally-dominant* if for every $i = 1, 2, \ldots n$ we have $A_{ii} \geq \sum_{i\neq j} |A_{ij}|$.

Symmetric diagonally dominant matrices have symmetric factorizations $A = UU^T$ such that each column of U has at most two nonzeros, and all nonzeros in each column have the same absolute values. We now establish a notation for such columns.

Definition 2.2. Let $1 \leq i, j \leq n$, $i \neq j$. The length-n *positive edge vector* denoted $\langle i, -j\rangle$ and the *negative edge vector* $\langle i, j\rangle$ are defined by

$$\langle i,-j\rangle_k = \begin{cases} +1 & k = i \\ -1 & k = j \\ 0 & \text{otherwise.} \end{cases} \quad \text{and} \quad \langle i,j\rangle_k = \begin{cases} +1 & k = i \\ +1 & k = j \\ 0 & \text{otherwise.} \end{cases}.$$

The reason for the assignment of signs to edge vectors will become apparent later. A *vertex vector* $\langle i\rangle$ is the unit vector

$$\langle i\rangle_k = \begin{cases} +1 & k = i \\ 0 & \text{otherwise.} \end{cases}$$

A symmetric diagonally dominant matrix can always be expressed as a sum of outer products of edge and vertex vectors, and therefore, as a symmetric product of a matrix whose columns are edge and vertex vectors.

Lemma 2.3. *([6]) Let $A \in \mathbb{R}^{n\times n}$ be a diagonally dominant symmetric matrix. We can decompose A as follows*

$$\begin{aligned} A &= \sum_{\substack{i<j \\ A_{ij}>0}} |A_{ij}| \langle i,j\rangle \langle i,j\rangle^T \\ &\quad + \sum_{\substack{i<j \\ A_{ij}<0}} |A_{ij}| \langle i,-j\rangle \langle i,-j\rangle^T \\ &\quad + \sum_{i=1}^{n} \left(A_{ii} - \sum_{\substack{j=1 \\ j\neq i}}^{n} |A_{ij}| \right) \langle i\rangle \langle i\rangle^T \end{aligned}$$

Matrix decompositions of this form play a prominent role in support theory, so we give them a name:

Definition 2.4. A matrix whose columns are scaled edge and vertex vectors (that is, vectors of the forms $c\langle i, -j\rangle$, $c\langle i, j\rangle$, and $c\langle i\rangle$) is called an *incidence matrix*. A factorization $A = UU^T$ where U is an incidence matrix is called an *incidence factorization*. An incidence factorization with no zero columns, with at most one vertex vector for each index i, with at most one edge vector for each index pair i, j, and whose positive edge vectors are all of the form $c\langle \min(i,j), -\max(i,j)\rangle$ is called a *canonical incidence factorization*.

Lemma 2.5. *Let $A \in \mathbb{R}^{n\times n}$ be a diagonally dominant symmetric matrix. Then A has an incidence factorization $A = UU^T$, and a unique canonical incidence factorization.*

2.2. **Graphs and Their Laplacians Matrices.** We now define the connection between undirected graphs and diagonally-dominant symmetric matrices.

Definition 2.6. Let $G = (\{1, 2, \ldots n\}, E, c, d)$ be a weighted undirected graph on the vertex set $\{1, 2, \ldots, n\}$ with no self loops and no parallel edges, and with weight functions $c : E \to \mathbb{R} \setminus \{0\}$ and $d : \{1, \ldots, n\} \to \mathbb{R}_+ \cup \{0\}$. That is, the edge set consists of unordered pairs of unequal integers (i, j) such that $1 \le i, j \le n$. The *Laplacian* of G is the matrix $A \in \mathbb{R}^{n\times n}$ such that

$$A_{ij} = \begin{cases} d(i) + \sum_{(i,k)\in E} |c(i,k)| & i = j \\ -c(i,j) & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

A vertex i such that $d(i) > 0$ is called a *strictly dominant* vertex. If $c = 1$, the graph is not considered weighted. If $c > 0$ and is not always 1, the graph is *weighted*. If some weights are negative, the graph is *signed*.

Lemma 2.7. *The Laplacians of the graphs defined in Definition 2.6 are symmetric and diagonally dominant. Furthermore, these graphs are isomorphic to symmetric diagonally-dominant matrices under this Laplacian mapping.*

We prefer to work with vertex weights rather than allowing self loops because edge and vertex vectors are algebraically different.

In algorithms, given an explicit representation of a diagonally-dominant matrix A, we can easily compute an explicit representation of an incidence factor U (including the canonical incidence factor if desired). Sparse matrices are often represented by a data structure that stores a compressed array of nonzero entries for each row or each column of the matrix. Each entry in a row (column) array stores the column index (row index) of the nonzero, and the value of the nonzero. From such a representation of A we can easily construct a sparse representation of U by columns. We traverse each row of A, creating a column of U for each nonzero in the upper (or lower) part of A. During the traversal, we can also compute all the $d(i)$'s. The conversion works even if only the upper or lower part of A is represented explicitly.

We can use the explicit representation of A as an implicit representation of U, with each off-diagonal nonzero of A representing an edge-vector column of U. If A has no strictly-dominant rows, that is all. If A has strictly dominant rows, we need to compute their weights using a one-pass traversal of A.

3. Support Theory

Support theory is a set of tools that aim to bound the generalized eigenvalues λ that satisfy $Ax = \lambda Bx$ from above and below. If B is nonsingular, these eigenvalues are also the eigenvalues of $B^{-1}A$, but the generalized representation allows us to derive bounds that also apply to singular matrices.

Definition 3.1. Let A and B be n-by-n complex matrices. We say that a scalar λ is a *finite generalized eigenvalue* of the matrix pencil (pair) (A, B) if there is a vector $v \neq 0$ such that $Av = \lambda Bv$ and $Bv \neq 0$. We say that ∞ is a *infinite generalized eigenvalue* of (A, B) if there exists a vector $v \neq 0$ such that $Bv = 0$ but $Av \neq 0$. Note that ∞ is an eigenvalue of (A, B) if and only if 0 is an eigenvalue of (B, A). The finite and infinite eigenvalues of a pencil are *determined eigenvalues* (the eigenvector uniquely determines the eigenvalue). If both $Av = Bv = 0$ for a vector $v \neq 0$, we say that v is an *indeterminate eigenvector*, because $Av = \lambda Bv$ for any scalar λ.

The tools of support theory rely on symmetric factorizations $A = UU^T$ and $B = VV^T$; this is why incidence factorizations are useful. In fact, the algebraic tools of support theory are particularly easy to apply when U and V are incidence matrices.

3.1. **From Generalized Eigenvalues to Singular Values.** If $A = UU^T$ then $\Lambda(A) = \Sigma^2(U^T)$, where $\Lambda(A)$ is the set of eigenvalues of A and $\Sigma(U^T)$ is the set of singular values of U^T, and Σ^2 is the set of the squares of the singular values. The following lemma extends this trivial result to generalized eigenvalues.

Lemma 3.2. *([2])Let $A = UU^T$ and $B = VV^T$ with $\mathrm{null}(B) = \mathrm{null}(A) = \mathbb{S}$. We have*

$$\Lambda(A, B) = \Sigma^2\left(V^+U\right)$$

and

$$\Lambda(A, B) = \Sigma^{-2}\left(U^+V\right) .$$

In these expressions, $\Sigma(\cdot)$ is the set of nonzero singular values of the matrix within the parentheses, Σ^ℓ denotes the same singular values to the ℓth power, and V^+ denotes the Moore-Penrose pseudoinverse of V.

The lemma characterizes all the generalized eigenvalues of the pair (A, B), but for large matrices, it is not particularly useful. Even if U and V are highly structured (e.g., they are incidence matrices), U^+ and V^+ are usually not structured and are expensive to compute. The next section shows that if we lower our expectations a bit and only try to bound $\Lambda$from above and below, then we do not need the pseudo-inverses.

3.2. **The Symmetric Support Lemma.** In the previous section we have seen that the singular values of V^+U provide complete information on the generalized eigenvalues of (A, B). If we denote $W_{\mathrm{opt}} = V^+U$, we have

$$\begin{aligned} VW_{\mathrm{opt}} &= VV^+U \\ &= U . \end{aligned}$$

It turns out that any W such that $VW = U$ provides some information on the generalized eigenvalues of (A, B).

Lemma 3.3. *([9]) Let $A = UU^T$ and let $B = VV^T$, and assume that $\mathrm{null}(B) \subseteq \mathrm{null}(A)$. Then*

$$\max\{\lambda \mid Ax = \lambda Bx, Bx \neq 0\} = \min\left\{\|W\|_2^2 \mid U = VW\right\} .$$

This lemma is fundamental to support theory and preconditioning, because it is often possible to prove that W such that $U = VW$ exists and to give a-priori bounds on its norm.

3.3. **Norm bounds.** The Symmetric Product Support Lemma bounds generalized eigenvalues in terms of the 2-norm of some matrix W such that $U = VW$. Even if we have a simple way to construct such a W, we still cannot easily derive a corresponding bound on the spectrum from the Symmetric-Support Lemma. The difficulty is that there is no simple closed form expression for the 2-norm of a matrix, since it is not related to the entries of W in a simple way. It is equivalent to the largest singular value, but this must usually be computed numerically.

Fortunately, there are simple (and also some not-so-simple) functions of the elements of the matrix that yield useful bounds on its 2-norm. The following bounds are standard and are well known and widely used (see [5, Fact 9.8.10.ix] and [5, Fact 9.8.15]).

Lemma 3.4. *The two norm of $W \in \mathbb{C}^{k\times m}$ is bounded by*

$$\|W\|_2^2 \leq \|W\|_F^2 = \sum_{i=1}^{k}\sum_{i=1}^{m} W_{ij}^2 , \tag{3.1}$$

$$\|W\|_2^2 \leq \|W\|_1\|W\|_\infty = \left(\max_{j=1}^{m}\sum_{i=1}^{k}|W_{ij}|\right)\left(\max_{i=1}^{k}\sum_{j=1}^{m}|W_{ij}|\right) . \tag{3.2}$$

The next two bounds are standard and well known; they follow directly from $\|WW^T\|_2 = \|W\|_2^2$ and from the fact that $\|S\|_1 = \|S\|_\infty$ for a symmetric S.

Lemma 3.5. *The two norm of $W \in \mathbb{C}^{k\times m}$ is bounded by*

$$\|W\|_2^2 \leq \|WW^T\|_1 = \|WW^T\|_\infty , \tag{3.3}$$

$$\|W\|_2^2 \leq \|W^TW\|_1 = \|W^TW\|_\infty . \tag{3.4}$$

The following bounds are more specialized. They all exploit the sparsity of W to obtains bounds that are usually tighter than the bounds given so far.

Lemma 3.6. *([11]) The two norm of $W \in \mathbb{C}^{k\times m}$ is bounded by*

$$\|W\|_2^2 \leq \max_j \sum_{i:W_{i,j}\neq 0} \|W_{i,:}\|_2^2 = \max_j \sum_{i:W_{i,j}\neq 0} \sum_{c=1}^{m} W_{i,c}^2 , \tag{3.5}$$

$$\|W\|_2^2 \leq \max_i \sum_{j:W_{i,j}\neq 0} \|W_{:,j}\|_2^2 = \max_i \sum_{j:W_{i,j}\neq 0} \sum_{r=1}^{k} W_{r,j}^2 . \tag{3.6}$$

The bounds in this lemma are a refinement of the bound $\|W\|_2^2 \leq \|W\|_F^2$. The Frobenious norm, which bounds the two norm, sums the squares of *all* the elements of W. The bounds (3.5) and (3.6) sum only the squares in some of the rows or some of the columns, unless the matrix has a row or a columns with no zeros.

There are similar refinements of the bound $\|W\|_2^2 \leq \|W\|_1 \|W\|_\infty$.

Lemma 3.7. *([11]) The two norm of $W \in \mathbb{C}^{k\times m}$ is bounded by*

$$\|W\|_2^2 \leq \max_j \sum_{i:W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{c=1}^{m} |W_{i,c}| \right) , \tag{3.7}$$

$$\|W\|_2^2 \leq \max_i \sum_{j:W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{r=1}^{k} |W_{r,j}| \right) . \tag{3.8}$$

3.4. **Support numbers.** Support numbers generalize the notion of the maximal eigenvalue of a matrix pencil.

Definition 3.8. A matrix B *dominates* a matrix A if for any vector x we have $x^T(B-A)x \geq 0$. We denote domination by $B \succeq A$.

Definition 3.9. The *support number* for a matrix pencil (A, B) is

$$\sigma(A, B) = \min \{t | \tau B \succeq A, \text{ for all } \tau \geq t\} .$$

If B is symmetric positive definite and A is symmetric, then the support number is always finite, because $x^T Bx/x^T x$ is bounded from below by $\min \Lambda(B) > 0$ and $x^T Ax/x^T x$ is bounded from above by $\max \Lambda(A)$, which is finite. In other cases, there may not be any t satisfying the formula; in such cases, we say that $\sigma(A, B) = \infty$.

Example 3.10. Suppose that $x \in null(B)$ and that A is positive definite. Then for any $\tau > 0$ we have $x^T(\tau B - A)x = -x^T Ax < 0$. Therefore, $\sigma(A, B) = \infty$.

Example 3.11. If B is not positive semidefinite, then there is some x for which $x^T Bx < 0$. This implies that for any A and for any large enough τ, $x^T(\tau B - A)x < 0$. Therefore, $\sigma(A, B) = \infty$.

The next result, like the Symmetric Support Lemma, bounds generalized eigenvalues.

Theorem 3.12. *([9]) Let A and B be symmetric, let B also be positive semidefinite. If* $\mathrm{null}(B) \subseteq \mathrm{null}(A)$*, then*

$$\sigma(A, B) = \max\{\lambda | Ax = \lambda Bx, Bx \neq 0\} \ .$$

A primary motivation for support numbers is to bound (spectral) condition numbers. For symmetric matrices, the relation $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$ holds. Let $\kappa(A, B)$ denote the condition number of the matrix pencil (A, B), that is, $\kappa(B^{-1}A)$ when B is non-singular.

Theorem 3.13. *When A and B are symmetric positive definite, then $\kappa(A, B) = \sigma(A, B)\sigma(B, A)$.*

A common strategy in support theory is to bound condition numbers by bounding both $\sigma(A, B)$ and $\sigma(B, A)$. Typically, one direction is easy and the other is hard.

Applications usually do not solve singular systems. Nonetheless, it is often convenient to analyze preconditioners in the singular context. For example, finite element systems are often singular until boundary conditions are imposed, so we can build B from the singular part of A and then impose the same boundary-constraints on both matrices.

3.5. **Splitting.** Support numbers are convenient for algebraic manipulation. One of their most powerful properties is that they allow us to split complicated matrices into simpler pieces (matrices) and analyze these separately. Let $A = A_1 + A_2 + \cdots A_q$, and similarly, $B = B_1 + B_2 + \cdots + B_q$. We can then match up pairs (A_i, B_i) and consider the support number for each such pencil separately.

Lemma 3.14. *([17, Lemma 4.7]) Let $A = A_1 + A_2 + \cdots A_q$,and similarly, $B = B_1 + B_2 + \cdots + B_q$, where all A_i and B_i are symmetric and positive semidefinite. Then*

$$\sigma(A, B) \leq \max_i \sigma(A_i, B_i)$$

Proof. Let $\sigma = \sigma(A, B)$, let $\sigma_i = \sigma(A_i, B_i)$, and let $\sigma_{\max} = \max_i \sigma_i$. Then for any x

$$\begin{aligned} x^T(\sigma_{\max} B - A)x &= x^T \left(\sigma_{\max} \sum_i B_i - \sum_i A_i \right) x \\ &= \sum_i x^T \left(\sigma_{\max} B_i - A_i \right) x \\ &\geq \sum_i x^T \left(\sigma_i B_i - A_i \right) x \\ &\geq 0 \ . \end{aligned}$$

Therefore, $\sigma \leq \sigma_{\max}$. □

The splitting lemma is quite general, and can be used in many ways. In practice we want to break both A and B into simpler matrices that we know how to analyze. The term "simpler" can mean sparser, or lower rank, and so on. In order to get a good upper bound on the support number, the splitting must be chosen carefully. Poor splittings give poor bounds. Here is an example.

Example 3.15. Let $A = \begin{bmatrix} 3 & -2 \\ -2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = A_1 + A_2$, and $B = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = B_1 + B_2$. Then the Splitting Lemma says $\sigma(A, B) \leq \max\{\sigma(A_1, B_1), \sigma(A_2, B_2)\}$. It is easy to verify that $\sigma(A_1, B_1) = 2$ and that $\sigma(A_2, B_2) = 1$; hence $\sigma(A, B) \leq 2$. Note that B_1 can not support A_2, so correct pairing of the terms in A and B is essential. The exact support number is $\sigma(A, B) = \lambda_{\max}(A, B) = 1.557$.

4. Embeddings and Combinatorial Support Bounds

To bound $\sigma(A, B)$ using the Symmetric Support Lemma, we need to factor A and B into $A = UU^T$ and $B = VV^T$, and we need to find a W such that $U = VW$. We have seen that if A and B are diagonally dominant, then there is an almost trivial way to factor A and B such that U and V are about as sparse as A and B. But how do we find a W such that $U = VW$? In this chapter, we show that when A and B are weighted (but not signed) Laplacians, we can construct such W using an embedding of the edges of G_A into paths in G_B. Furthermore, when W is constructed from an embedding, the bounds on $\|W\|_2$ can be interpreted as combinatorial bounds on the quality of the embedding.

4.1. Defining W using Path Embeddings.

We start with the construction of a matrix W such that $U = VW$.

Lemma 4.1. *Let $(i_1, i_2, \ldots, i_\ell)$ be a sequence of integers between 1 and n, such that $i_j \neq i_{j+1}$ for $j = 1, \ldots \ell - 1$. Then*

$$\langle i_1, -i_\ell \rangle = \sum_{j=1}^{\ell-1} \langle i_j, -i_{j+1} \rangle \ ,$$

where all the edge vectors are length n.

To see why this lemma is important, consider the role of a column of W. Suppose that the columns of U and V are all positive edge vectors. Denote column c of U by

$$U_{:,c} = \langle \min(i_1, i_\ell), -\max(i_1, i_\ell) \rangle = (-1)^{i_1 > i_\ell} \langle i_1, -i_\ell \rangle \ ,$$

where the $(-1)^{i_1 > i_\ell}$ evaluates to -1 if $i_1 > i_\ell$ and to 1 otherwise. This column corresponds to the edge (i_1, i_ℓ) in G_{UU^T}. Now let $(i_1, i_2, \ldots, i_\ell)$ be a simple path in G_{VV^T} (a simple path is a sequence of vertices $(i_1, i_2, \ldots, i_\ell)$ such that (i_j, i_{j+1}) is an edge in the graph for $1 \leq j < \ell$ and such that any vertex appears at most once on the path). If $U = VW$, then

$$U_{:,c} = VW_{:,c} = \sum_{r=1}^{k} V_{:,r} W_{r,c} \ .$$

Let $r_1, r_2, \ldots, r_{\ell-1}$ be the columns of V that corresponds to the edges of the path $(i_1, i_2, \ldots, i_\ell)$, in order. That is, $V_{:,r_1} = \langle \min(i_1, i_2), -\max(i_1, i_2) \rangle$, $V_{:,r_2} = \langle \min(i_2, i_3), -\max(i_2, i_3) \rangle$, and

so on. By the lemma,

$$\begin{aligned} U_{:,c} &= (-1)^{i_1 > i_\ell} \langle i_1, -i_\ell \rangle \\ &= (-1)^{i_1 > i_\ell} \sum_{j=1}^{\ell-1} \langle i_j, -i_{j+1} \rangle \\ &= (-1)^{i_1 > i_\ell} \sum_{j=1}^{\ell-1} (-1)^{i_j > i_{j+1}} V_{:,r_j} \ . \end{aligned}$$

It follows that if we define $W_{:,c}$ to be

$$W_{r,c} = \begin{cases} (-1)^{i_1 > i_\ell} (-1)^{i_j > i_{j+1}} & r = r_j \text{ for some } 1 \le j < \ell \\ 0 & \text{otherwise,} \end{cases}$$

then we have $U_{:,c} = VW_{:,c}$. We can construct all the columns of W in this way, so that W satisfies $U = VW$.

A path of edge vectors that ends in a vertex vector supports the vertex vector associated with the first vertex of the path.

Lemma 4.2. *Let $(i_1, i_2, \ldots, i_\ell)$ be a sequence of integers between 1 and n, such that $i_j \neq i_{j+1}$ for $j = 1, \ldots \ell - 1$. Then*

$$\langle i_1 \rangle = \langle i_\ell \rangle + \sum_{j=1}^{\ell-1} \langle i_j, -i_{j+1} \rangle \ ,$$

where all the edge and vertex vectors are length n.

The following theorem generalizes these ideas to scaled positive edge vectors and to scaled vertex vectors. The theorem also states how to construct all the columns of W. The theorem summarizes results in [9, 4].

Theorem 4.3. *Let A and B be weighted (unsigned) Laplacians and let U and V be their canonical incidence factors. Let π be a path embedding of the edges and strictly-dominant vertices of G_A into G_B, such that for an edge (i_1, i_ℓ) in G_A, $i_1 < i_\ell$, we have*

$$\pi(i_1, i_\ell) = (i_1, i_2, \ldots, i_\ell)$$

for some simple path $(i_1, i_2, \ldots, i_\ell)$ in G_B, and such that for a strictly-dominant i_1 in G_A,

$$\pi(i_1) = (i_1, i_2, \ldots, i_\ell)$$

for some simple path $(i_1, i_2, \ldots, i_\ell)$ in G_B that ends in a strictly-dominant vertex i_ℓ in G_B. Denote by $c_V(i_j, i_{j+1})$ the index of the column of V that is a scaling of $\langle i_j, -i_{j+1} \rangle$. That is,

$$V_{:,c_V(i_j,i_{j+1})} = \sqrt{-B_{i_j,i_{j+1}}} \, \langle \min(i_j, i_{j+1}), -\max(i_j, i_{j+1}) \rangle \ .$$

Similarly, denote by $c_V(i_j)$ the index of the column of V that is a scaling of $\langle i_j \rangle$,

$$V_{:,c_V(i_j)} = \sqrt{B_{i_j,i_j} - \sum_{\substack{i_k=1 \\ i_k \neq i_j}}^{n} \left| B_{i_k,i_j} \right|} \langle i_j \rangle \ ,$$

and similarly for U.

We define a matrix W as follows. For a column index $c_U(i_1, i_\ell)$ with $i_1 < i_\ell$ we define

$$W_{r,c_U(i_1,i_\ell)} = \begin{cases} (-1)^{i_j > i_{j+1}} \sqrt{A_{i_1,i_\ell}/B_{i_j,i_{j+1}}} & \text{if } r = c_V(i_j, i_{j+1}) \text{ for} \\ & \text{some edge } (i_j, i_{j+1}) \text{ in } \pi(i_1, i_\ell) \\ 0 & \text{otherwise.} \end{cases}$$

For a column index $c_U(i_1)$, we define

$$W_{r,c_U(i_1)} = \begin{cases} \sqrt{\frac{A_{i_1,i_1} - \sum_{j \neq i_1} \left| A_{i_1,j} \right|}{B_{i_\ell,i_\ell} - \sum_{j \neq i_\ell} \left| B_{i_\ell,j} \right|}} & \text{if } r = c_V(i_\ell) \\ (-1)^{i_j > i_{j+1}} \sqrt{\frac{A_{i_1,i_1} - \sum_{k \neq i_1} \left| A_{i_1,k} \right|}{\left| B_{i_j,i_{j+1}} \right|}} & \text{if } r = c_V(i_j, i_{j+1}) \text{ for} \\ & \text{some edge } (i_j, i_{j+1}) \text{ in } \pi(i_1) \\ 0 & \text{otherwise.} \end{cases}$$

Then $U = VW$.

Proof. For scaled edge-vector columns in U we have

$$\begin{aligned} VW_{:,c_U(i_1,i_\ell)} &= \sum_r V_{:,r} W_{r,c_U(i_1,i_\ell)} \\ &= \sum_{\substack{r = c_V(i_j,i_{j+1}) \\ \text{for some edge} \\ (i_j,i_{j+1}) \text{ in } \pi(i_1,i_\ell)}} V_{:,r} W_{r,c_U(i_1,i_\ell)} \\ &= \sum_{j=1}^{\ell-1} \sqrt{\left| B_{i_j,i_{j+1}} \right|} \left\langle \min(i_j, i_{j+1}), -\max(i_j, i_{j+1}) \right\rangle (-1)^{i_j > i_{j+1}} \sqrt{\frac{A_{i_1,i_\ell}}{B_{i_j,i_{j+1}}}} \\ &= \sqrt{|A_{i_1,i_\ell}|} \sum_{j=1}^{\ell-1} \langle i_j, -i_{j+1} \rangle \\ &= U_{:,c_U(i_1,i_\ell)} \ . \end{aligned}$$

For scaled vertex-vector columns in U we have

$$\begin{aligned}
VW_{:,c_U(i_1)} &= \sum_r V_{:,r} W_{r,c_U(i_1)} \\
&= V_{:,c_V(i_\ell)} W_{c_V(i_\ell),c_U(i_1,i_\ell)} + \sum_{\substack{r=c_V(i_j,i_{j+1}) \\ \text{for some edge} \\ (i_j,i_{j+1}) \text{ in } \pi(i_1)}} V_{:,r} W_{r,c_U(i_1)} \\
&= \sqrt{B_{i_\ell,i_\ell} - \sum_{j\neq i_\ell} |B_{i_k,i_\ell}|}\, \langle i_\ell \rangle \sqrt{\frac{A_{i_1,i_1} - \sum_{j\neq i_1} |A_{i_1,j}|}{B_{i_\ell,i_\ell} - \sum_{j\neq i_\ell} |B_{i_\ell,j}|}} \\
&\quad + \sum_{j=1}^{\ell-1} \sqrt{\left|B_{i_j,i_{j+1}}\right|}\, \langle \min(i_j, i_{j+1}), -\max(i_j, i_{j+1}) \rangle \\
&\qquad \cdot (-1)^{i_j > i_{j+1}} \sqrt{\frac{A_{i_1,i_1} - \sum_{j\neq i_1} |A_{i_1,j}|}{\left|B_{i_j,i_{j+1}}\right|}} \\
&= \sqrt{A_{i_1,i_1} - \sum_{j\neq i_1} |A_{i_1,j}|}\, \langle i_\ell \rangle + \sqrt{A_{i_1,i_1} - \sum_{j\neq i_1} |A_{i_1,j}|} \sum_{j=1}^{\ell-1} \langle i_j, -i_{j+1} \rangle \\
&= \sqrt{A_{i_1,i_1} - \sum_{j\neq i_1} |A_{i_1,j}|}\, \langle i_1 \rangle \\
&= U_{:,c_U(i_1)} \, .
\end{aligned}$$

□

The generalization of this theorem to signed Laplacians is more complex, because a path from i_1 to i_ℓ supports an edge (i_1, i_ℓ) only if the parity of positive edges in the path and in the edge is the same. In addition, a cycle with an odd number of positive edges spans all the vertex vectors of the path. For details, see [6].

Theorem 4.3 plays a fundamental role in many applications of support theory. A path embedding π that can be used to construct W exists if and only if the graphs of A and B have related in a specific way, which the next lemma specifies.

Lemma 4.4. *Let $A = UU^T$ and $B = VV^T$ be weighted (but not signed) Laplacians with arbitrary symmetric-product factorizations. The following conditions are necessary for the equation $U = VW$ to hold for some matrix W (by Theorem 4.3, these conditions are also sufficient).*

1. *For each edge (i, j) in G_A, either i and j are in the same connected component in G_B, or the two components of G_B that contain i and j both include a strictly-dominant vertex.*
2. *For each strictly-dominant vertex i in G_A, the component of G_B that contains i includes a strictly-dominant vertex.*

Proof. Suppose for contradiction that one of the conditions is not satisfied, but that there is a W that satisfies $U = VW$. Without loss of generality, we assume that the vertices are ordered such that vertices that belong to a connected component in G_B are consecutive. Under that assumption,

$$V = \begin{bmatrix} V_1 & & & \\ & V_2 & & \\ & & \ddots & \\ & & & V_k \end{bmatrix},$$

and

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_k \end{bmatrix} = \begin{bmatrix} V_1 V_1^T & & & \\ & V_2 V_2^T & & \\ & & \ddots & \\ & & & V_k V_k^T \end{bmatrix}.$$

The blocks of V are possibly rectangular, whereas the nonzero blocks of B are all diagonal and square.

We now prove the necessity of the first condition. Suppose for some edge (i, j) in G_A, i and j belong to different connected components of G_B (without loss of generality, to the first two components), and that one of the components (w.l.o.g. the first) does not have a strictly-dominant vertex. Because this component does not have a strictly-dominant vertex, the row sums in $V_1 V_1^T$ are exactly zero. Therefore, $V_1 V_1^T \vec{1} = \vec{0}$, so V_1 must be rank deficient.

Since (i, j) is in G_A, the vector $\langle i, -j \rangle$ is in the column space of the canonical incidence factor of A, and therefore in the column space of any U such that $A = UU^T$. If $U = VW$, then the vector $\langle i, -j \rangle$ must also be in the column space of V, so for some x

$$\langle i, -j \rangle = Vx = \begin{bmatrix} V_1 & & & \\ & V_2 & & \\ & & \ddots & \\ & & & V_k \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} V_1 x_1 \\ V_2 x_2 \\ \vdots \\ V_k x_k \end{bmatrix}.$$

Therefore, $V_1 x_1$ is a vertex vector. By Theorem 4.3, if V_1 spans a vertex vector, it spans all the vertex vectors associated with the vertices of the connected component. This implies that V_1 is full rank, a contradiction.

The necessity of the second condition follows from a similar argument. Suppose that vertex i is strictly dominant in G_A and that it belongs to a connected component in G_B (w.l.o.g. the first) that does not have a vertex that is strictly dominant in G_B. This implies that for some y

$$\langle i \rangle = Vy = \begin{bmatrix} V_1 & & & \\ & V_2 & & \\ & & \ddots & \\ & & & V_k \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} V_1 y_1 \\ V_2 y_2 \\ \vdots \\ V_k y_k \end{bmatrix}.$$

Again $V_1 y_1$ is a vertex vector, so V_1 must be full rank, but it cannot be full rank because $V_1 V_1^T$ has zero row sums. □

Not every W such that $U = VW$ corresponds to a path embedding, even if U and V are the canonical incidence factors of A and B.In particular, a column of W can correspond to a linear combination of multiple paths. Also, even if W does correspond to a path embedding, the paths are not necessarily simple. A linear combination of scaled positive edge vectors that correspond to a simple cycle can be identically zero, so the coefficients of such linear combinations can be added to W without affecting the product VW. However, it seems that adding cycles to a path embedding cannot reduce the 2-norm of W, so cycles are unlikely to improve support bounds.

4.2. **Combinatorial Support Bounds.** To bound $\sigma(A, B)$ using the Symmetric Support Lemma, we factor A into $A = UU^T$, B into $B = VV^T$, find a matrix W such that $U = VW$, and bound the 2-norm of W from above. We have seen how to factor A and B (if they are weighted Laplacians) and how to construct an appropriate W from an embedding of G_B in G_A. We now show how to use combinatorial metrics of the path embeddings to bound $\|W\|_2$.

Bounding the 2-norm directly is hard, because the 2-norm is not related in a simple way to the entries of W. But the 2-norm can be bounded using simpler norms, such as the Frobenius norm, the infinity norm, and the 1-norm (see section 3.3). These simpler norms have natural and useful combinatorial interpretations when W represents a path embedding.

To keep the notation and the definition simple, we now assume that A and B are weighted Laplacians with zero row sums. We will show later how to deal with positive row sums. We also assume that W corresponds to a path embedding π. The following definitions provide a combinatorial interpretation of these bounds.

Definition 4.5. The *weighted dilation* of an edge of G_A in an path embedding π of G_A into G_B is

$$\text{dilation}_\pi(i_1, i_2) = \sum_{\substack{(j_1,,j_2) \\ (j_1,,j_2)\in\pi(i_1,i_2)}} \sqrt{\frac{A_{i_1,i_2}}{B_{j_1,j_2}}}\ .$$

The *weighted congestion* of an edge of G_B is

$$\text{congestion}_\pi(j_1, j_2) = \sum_{\substack{(i_1,,i_2) \\ (j_1,,j_2)\in\pi(i_1,i_2)}} \sqrt{\frac{A_{i_1,i_2}}{B_{j_1,j_2}}}\ .$$

The weighted stretch of an edge of G_A is

$$\text{stretch}_\pi(i_1, i_2) = \sum_{\substack{(j_1,,j_2) \\ (j_1,,j_2)\in\pi(i_1,i_2)}} \frac{A_{i_1,i_2}}{B_{j_1,j_2}}\ .$$

The weighted crowding of an edge in G_B is

$$\text{crowding}_\pi(j_1, j_2) = \sum_{\substack{(i_1,,i_2) \\ (j_1,,j_2)\in\pi(i_1,i_2)}} \frac{A_{i_1,i_2}}{B_{j_1,j_2}}\ .$$

Note that stretch is a summation of the squares of the quantities that constitute dilation, and similarly for crowding and congestion. Papers in the support-preconditioning literature are not consistent about these terms, so check the definitions carefully when you consult a paper that deals with congestion, dilation, and similar terms.

Lemma 4.6. *([9, 28]) Let A and B be weighted Laplacians with zero row sums, and let π be a path embedding of G_A into G_B. Then*

$$\begin{aligned}
\sigma(A,B) &\leq \sum_{(i_1,i_2)\in G_A} stretch_\pi(i_1,i_2) \\
\sigma(A,B) &\leq \sum_{(j_1,j_2)\in G_B} crowding_\pi(j_1,j_2) \\
\sigma(A,B) &\leq \left(\max_{(i_1,i_2)\in G_A} dilation_\pi(i_1,i_2)\right) \\
&\quad \cdot \left(\max_{(j_1,j_2)\in G_B} congestion_\pi(j_1,j_2)\right) .
\end{aligned}$$

We now describe one way to deal with matrices with some positive row sums. The matrix B is a preconditioner. In many applications, the matrix B is not given, but rather constructed. One simple way to deal with positive row sums in A is to define$\pi(i_1) = (i_1)$. That is, vertex vectors in the canonical incidence factor of A are mapped into the same vertex vectors in the incidence factor of B. In other words, we construct B to have exactly the same row sums as A. With such a construction, the rows of W that correspond to vertex vectors in U are columns of the identity.

Lemma 4.7. *Let A and B be weighted Laplacians with the same row sums, let π be a path embedding of G_A into G_B, and let ℓ be the number of rows with positive row sums in A and B. Then*

$$\begin{aligned}
\sigma(A,B) &\leq \ell + \sum_{(i_1,i_2)\in G_A} stretch_\pi(i_1,i_2) \\
\sigma(A,B) &\leq \left(\max\left\{1, \max_{(i_1,i_2)\in G_A} dilation_\pi(i_1,i_2)\right\}\right) \\
&\quad \cdot \left(\max\left\{1, \max_{(j_1,j_2)\in G_B} congestion_\pi(j_1,j_2)\right\}\right) .
\end{aligned}$$

Proof. Under the hypothesis of the lemma, the rows and columns of W can be permuted into a block matrix

$$W = \begin{pmatrix} W_Z & 0 \\ 0 & I_{\ell\times\ell} \end{pmatrix} ,$$

where W_Z represents the path embedding of the edges of G_A into paths in G_B. The bounds follow from the structure of W and from the proof of the previous lemma. □

The sparse bounds on the 2-norm of a matrix lead to tighter combinatorial bounds.

Lemma 4.8. *Let A and B be weighted Laplacians with zero row sums, and let π be a path embedding of G_A into G_B. Then*

$$\begin{aligned}\sigma(A,B) &\leq \max_{(j_1,j_2)\in G_B} \sum_{\substack{(i_1,i_2)\in G_A \\ (j_1,j_2)\in\pi(i_1,i_2)}} stretch_\pi(i_1,i_2)\ ,\\ \sigma(A,B) &\leq \max_{(i_1,i_2)\in G_A} \sum_{\substack{(j_1,j_2)\in G_A \\ (j_1,j_2)\in\pi(i_1,i_2)}} crowding_\pi(j_1,j_2)\ .\end{aligned}$$

We can derive similar bounds for the other sparse 2-norm bounds.

4.3. **Subset Preconditioners.** To obtain a bound on $\kappa(A,B)$, we need a bound on both $\sigma(A,B)$ and $\sigma(B,A)$. But in one common case, bounding $\sigma(B,A)$ is trivial. Many support preconditioners construct G_B to be a subgraph of G_A, with the same weights. That is, V is constructed to have a subset of the columns in U. If we denote by $\bar{V}$ the set of columns of U that are *not* in V, we have

$$\begin{aligned}B &= VV^T\\ A &= UU^T\\ &= VV^T + \bar{V}\bar{V}^T\\ &= B + \bar{V}\bar{V}^T\ .\end{aligned}$$

This immediately implies $x^TAx \geq x^TBx$ for any x, so $\lambda_{\min}(A,B) \geq 1$.

4.4. **Combinatorial Trace Bounds.** The Preconditioned Conjugate Gradients (PCG) algorithm requires $\Theta(\sqrt{\kappa(A,B)})$ iterations only when the generalized eigenvalues are distributed poorly between $\lambda_{\min}(A,B)$ and $\lambda_{\max}(A,B)$. For example, if there are only two distinct eigenvalues, PCG converges in two iterations. It turns out that a bound on $trace(A,B) = \sum \lambda_i(A,B)$ also yields a bound on the number of iterations, and in some cases this bound is sharper than the $O(\sqrt{\kappa(A,B)})$ bound.

Lemma 4.9. *([29]) The Preconditioned Conjugate Algorithm converges to within a fixed tolerance in*

$$O\left(\sqrt[3]{\frac{trace(A,B)}{\lambda_{\min}(A,B)}}\right)$$

iterations.

A variation of the Symmetric Support Lemma bounds the trace, and this leads to a combinatorial support bound.

Lemma 4.10. *Let $A = UU^T \in \mathbb{R}^{n\times n}$ and let $B = VV^T \in \mathbb{R}^{n\times n}$, and assume that $null(B) = null(A)$. Then*

$$trace(A,B) = \min\left\{\|W\|_F^2 \mid U = VW\right\}\ .$$

Proof. Let W be a matrix such that $U = VW$. For every $x \notin \mathrm{n}ull(B)$ we can write

$$\begin{aligned}\frac{x^TAx}{x^TBx} &= \frac{x^TVWW^TV^Tx}{x^TVV^Tx} \\ &= \frac{y^TWW^Ty}{y^Ty},\end{aligned}$$

where $y = V^Tx$. Let $S \subseteq \mathbb{R}^n$ be a subspace orthogonal to $\mathrm{n}ull(B)$ of dimesion k, and define $T_S = \left\{V^Tx \mid x \in S\right\}$. Because S is orthogonal to $\mathrm{n}ull(B)$ we have $\dim(T_S) = k$. The sets $\left\{x^TAx/x^TBx \mid x \in S\right\}$ and $\left\{y^TWW^Ty \mid y \in T_S\right\}$ are identical so their minimums are equal as well. The group of subspaces $\{T_S \mid \dim(S) = k,\ S \perp \mathrm{n}ull(B)\}$ is a subset of all subspaces of dimension k, thererfore

$$\begin{aligned}\max_{\substack{\dim(S)=k \\ S \perp \mathrm{n}ull(B)}} \min_{\substack{x \in S \\ x \neq 0}} \frac{x^TUU^Tx}{x^TVV^Tx} &= \max_{\substack{\dim(S)=k \\ S \perp \mathrm{n}ull(B)}} \min_{\substack{y \in T_S \\ y \neq 0}} \frac{y^TWW^Ty}{y^Ty} \\ &\leq \max_{\dim(T)=k} \min_{\substack{y \in T \\ y \neq 0}} \frac{y^TWW^Ty}{y^Ty}.\end{aligned}$$

According to the Courant-Fischer Minimax Theorem,

$$\lambda_{n-k+1}(WW^T) = \max_{\dim(T)=k} \min_{\substack{y \in T \\ y \neq 0}} \frac{y^TWW^Ty}{y^Ty}$$

and by the generalization of Courant-Fischer in [3],

$$\lambda_{n-k+1}(UU^T, VV^T) = \max_{\substack{\dim(S)=k \\ V \perp \mathrm{n}ull(B)}} \min_{x \in S} \frac{x^TUU^Tx}{x^TVV^Tx}.$$

Therefore, for $k = 1, \ldots, rank(B)$ we have $\lambda_{n-k+1}(A, B) \leq \lambda_{n-k+1}(WW^T)$ so $trace(A, B) \leq trace(WW^T) = \|W\|_F^2$. This shows that $trace(A, B) \leq \min\left\{\|W\|_F^2 \mid U = VW\right\}$.

According to Lemma 3.2 the minimum is attainable at $W = V^+U$. □

To the best of our knowledge, Lemma 4.10 is new. It was inspired by a specialized bound on the trace from [29]. The next theorem generalizes the result form [29]. The proof is trivial given Definition 4.5 and Lemma 4.10

Theorem 4.11. *Let A and B be weighted Laplacians with the same row sums, let π be a path embedding of G_A into G_B. Then*

$$\begin{aligned} trace(A,B) &\leq \sum_{(i_1,i_2)\in G_A} stretch_\pi(i_1,i_2)\ , \\ trace(A,B) &\leq \sum_{(j_1,j_2)\in G_B} crowding_\pi(j_1,j_2)\ . \end{aligned}$$

5. Combinatorial Preconditioners

The earliest graph algorithm to construct a preconditioner was proposed by Vaidya [31] (see also [4]). He proposed to use a so-called augmented maximum spanning tree of a weighted Laplacian as a preconditioner. This is a subset preconditioner that drops some of the edges in G_A while maintaining the weights of the remaining edges. When B^{-1} is applied using a sparse Cholesky factorization, this construction leads to a total solution time of $O(n^{7/4})$ for Laplacians with a bounded degree and $O(n^{6/5})$ when the graph is planar. For regular unweighted meshes in 2 and 3 dimensions, special constructions are even more effective [20]. Vaidya also proposed to use recursion to apply B^{-1}, but without a rigorous analysis; in this scheme, sparse Gaussian elimination steps are performed on B as long as the reduced matrix has a row with only two nonzeros. At that point, the reduced system is solved by constructing a graph preconditioner, and so on. Vaidya's preconditioners are quite effective in practice [12]. A generalization to complex matrices proved effective in handling a certain kind of ill conditioning [19].

Vaidya's bounds used a congestion-dilation product. The research on subset graph preconditioners continued with an observation that the sum of the stretch can also yield a spectral bound, and that so-called *low-stretch trees* would give better worst-case bounds than the maximum-spanning trees that Vaidya used [8]. Constructing low-stretch trees is more complicated than constructing maximum spanning trees. When the utility of low-stretch trees was discovered, one algorithm for constructing them was known [1]; better algorithms were discovered later [15]. Low-stretch trees can also be augmented, and this leads to preconditioners that can solve any Laplacian system with m nonzeros in $O(m^{4/3})$, up to some additional polylogarithmic factors [26, 29]. By employing recursion, the theoretical running time can be reduced to close to linear in m [27]. An experimental comparison of simplified versions of these sophisticated algorithms to Vaidya's algorithm did not yield a conclusive result [30]. Heuristic subset graph preconditioners have also been proposed [16].

Gremban and Miller proposed a class of combinatorial preconditioners called *support-tree* preconditioners [18, 17]. Their algorithms construct a weighted tree whose leaves are the vertices of G_A. Therefore, the graph G_T of the preconditioner T has additional vertices. They show that applying T^{-1} to extensions of residuals is equivalent to using a preconditioner B that is the Schur complement of T with respect to the original vertices. Bounding the condition number $\kappa(A,B)$ is more difficult than bounding the condition number of subset preconditioners, because the Schur complement is dense. More effective versions of this strategy have been proposed later [23, 21, 22].

Efforts to generalize these constructions to matrices that are not weighted Laplacians followed several paths. Gremban showed how to transform a linear system whose coefficient matrix is a signed Laplacian to a linear system of twice the size whose matrix is a weighted Laplacian. The coefficient matrix is a 2-by-2 block matrix with diagonal blocks with the same sparsity pattern as the original matrix A and with identity off-diagonal blocks. A different approach is to extend Vaidya's construction to signed graphs [6]. The class of symmetric matrices with a symmetric factorization $A = UU^T$ where columns of U have at most 2 nonzeros contains not only signed graphs, but also gain graphs, which are not diagonally dominant [7]; it turns out that these matrices can be scaled to diagonal dominance, which allows graph preconditioners to be applied to them [14].

The matrices that arise in finite-element discretization of elliptic partial differential equations (PDEs) are positive semi-definite, but in general they are not diagonally dominant. However, when the PDE is scalar (e.g., describes a problem in electrostatics), the matrices can sometimes be approximated by diagonally dominant matrices. In this scheme, the coefficient matrix A is first approximated by a diagonally-dominant matrix D, and then G_D is used to construct the graph G_B of the preconditioner B. For large matrices of this class, the first step is expensive, but because finite-element matrices have a natural representation as a sum of very sparse matrices, the diagonally-dominant approximation can be constructed for each term in the sum separately. There are at least three ways to construct these approximations: during the finite-element discretization process [10], algebraically [2], and geometrically [32]. A slightly modified construction that can accommodate terms that do not have a close diagonally-dominant approximation works well in practice [2].

Another approach for constructing combinatorial preconditioners to finite element problems is to rely on a graph that describes the relations between neighboring elements. This graph is the dual of the finite-element mesh; elements in the mesh are the vertices of the graph. Once the graph is constructed, it can be sparsified much like subset preconditioners. This approach, which is applicable to vector problems like linear elasticity, was proposed in [24]; this paper also showed how to construct the dual graph algebraically and how to construct the finite-element problem that corresponds to the sparsified dual graph. The first effective preconditioner of this class was proposed in [13]. It is not yet known how to weigh the edges of the dual graph effectively, which limits the applicability of this method. However, in applications where there is no need to weigh the edges, the method is effective [25].

References

[1] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24:78–100, 1995.

[2] Haim Avron, Doron Chen, Gil Shklarski, and Sivan Toledo. Combinatorial preconditioners for scalar elliptic finite-element problems. *SIAM Journal on Matrix Analysis and Applications*, 31(2):694–720, 2009.

[3] Haim Avron, Esmond Ng, and Sivan Toledo. Using perturbed QR factorizations to solve linear least-squares problems. *SIAM Journal on Matrix Analysis and Applications*, 31(2):674–693, 2009.

[4] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27:930–951, 2006.

[5] Denis S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas with Applications to Linear Systems Theory*. Princeton University Press, 2005.

[6] Erik G. Boman, Doron Chen, Bruce Hendrickson, and Sivan Toledo. Maximum-weight-basis preconditioners. *Numerical Linear Algebra with Applications*, 11:695–721, 2004.

[7] Erik G. Boman, Doron Chen, Ojas Parekh, and Sivan Toledo. On the factor-width and symmetric H-matrices. *Numerical Linear Algebra with Applications*, 405:239–248, 2005.

[8] Erik G. Boman and Bruce Hendrickson. On spanning tree preconditioners. Unpublished manuscript, Sandia National Laboratories, 2001.

[9] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, 25(3):694–717, 2003.

[10] Erik G. Boman, Bruce Hendrickson, and Stephen Vavasis. Solving elliptic finite element systems in near-linear time with support preconditioners. *SIAM Journal on Numerical Analysis*, 46(6):3264–3284, 2008.

[11] Doron Chen, John R. Gilbert, and Sivan Toledo. Obtaining bounds on the two norm of a matrix from the splitting lemma. *Electronic Transactions on Numerical Analysis*, 21:28–46, 2005.

[12] Doron Chen and Sivan Toledo. Vaidya's preconditioners: Implementation and experimental study. *Electronic Transactions on Numerical Analysis*, 16:30–49, 2003.

[13] Samuel I. Daitch and Daniel A. Spielman. Support-graph preconditioners for 2-dimensional trusses. Mar 2007.

[14] Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *STOC '08: Proceedings of the 40th annual ACM Symposium on Theory of Computing*, pages 451–460, New York, NY, USA, 2008. ACM.

[15] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *Proceedings of the 37th annual ACM symposium on Theory of computing (STOC)*, pages 494–503, Baltimore, MD, 2005. ACM Press.

[16] A. Frangioni and C. Gentile. New preconditioners for KKT systems of network flow problems. *SIAM Journal on Optimization*, 14:894–913, 2004.

[17] Keith D. Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, School of Computer Science, Carnegie Mellon University, October 1996. Available as Technical Report CMU-CS-96-123.

[18] Keith D. Gremban, Gary L. Miller, and Marco Zagha. Performance evaluation of a new parallel preconditioner. In *Proceedings of the 9th International Parallel Processing Symposium*, pages 65–69. IEEE Computer Society, 1995. A longer version is available as Technical Report CMU-CS-94-205, Carnegie-Mellon University.

[19] Victoria E. Howle and Stephen A. Vavasis. An iterative method for solving complex-symmetric systems arising in electrical power modeling. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1150–1178, 2005.

[20] Anil Joshi. *Topics in Optimization and Sparse Linear Systems*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1997.

[21] Ioannis Koutis and Gary L. Miller. A linear work, $O(n^{1/6})$ time, parallel algorithm for solving planar Laplacians. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1002–1011. SIAM, 2007.

[22] Ioannis Koutis and Gary L. Miller. Graph partitioning into isolated, high conductance clusters: theory, computation and applications to preconditioning. In *SPAA '08: Proceedings of the twentieth annual Symposium on Parallelism in Algorithms and Architectures*, pages 137–145, New York, NY, USA, 2008. ACM.

[23] Bruce M. Maggs, Gary L. Miller, Ojas Parekh, R. Ravi, and Shan Leung Maverick Woo. Finding effective support-tree preconditioners. In *SPAA '05: Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 176–185. ACM Pres, 2005.

[24] Gil Shklarski and Sivan Toledo. Rigidity in finite-element matrices: Sufficient conditions for the rigidity of structures and substructures. *SIAM Journal on Matrix Analysis and Applications*, 30(1):7–40, 2008.

[25] Gil Shklarski and Sivan Toledo. Computing the null space of finite element problems. *Computer Methods in Applied Mechanics and Engineering*, 198(37-40):3084–3095, August 2009.

[26] Daniel A. Spielman and Shang-Hua Teng. Solving sparse, symmetric, diagonally-dominant linear systems in time $0(m^{1.31})$. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 416–427, October 2003.

[27] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90, New York, NY, USA, 2004. ACM Press.

[28] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. Unpublished manuscript available online at http://arxiv.org/abs/cs/0607105, 2009.

[29] Daniel A. Spielman and Jaeoh Woo. A note on preconditioning by low-stretch spanning trees. Mar 2009.

[30] Uri Unger. An experimental evaluation of combinatorial preconditioners. Master's thesis, Tel-Aviv University, July 2007.

[31] Pravin M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. Unpublished manuscript. A talk based on this manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computations, Minneapolis, October 1991.

[32] Meiqiu Wang and Vivek Sarin. Parallel support graph preconditioners. In Yves Robert, Manish Parashar, Ramamurthy Badrinath, and Viktor K. Prasanna, editors, *High Performance Computing - HiPC 2006*, volume 4297, chapter 39, pages 387–398. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.