MAXIMUM-WEIGHT-BASIS PRECONDITIONERS

ERIK G. BOMAN, DORON CHEN, BRUCE HENDRICKSON, AND SIVAN TOLEDO

Abstract. This paper analyzes a novel method for constructing preconditioners for diagonally-dominant symmetric positive-definite matrices. The method discussed here is based on a simple idea: we construct M by simply dropping offdiagonal nonzeros from A and modifying the diagonal elements to maintain a certain row-sum property. The preconditioners are extensions of Vaidya's augmented maximum-spanning-tree preconditioners. The preconditioners presented here were also mentioned by Vaidya in an unpublished manuscript, but without a complete analysis.

The preconditioners that we present have only $O(n+t^2)$ nonzeros, where n is the dimension of the matrix and t is a parameter that one can choose. Their construction is efficient and guarantees that the condition number of the preconditioned system is $O(n^2/t^2)$ if the number of nonzeros per row in the matrix is bounded by a constant.

We have developed an efficient algorithm to construct these preconditioners and we have implemented it. We used our implementation to solve a simple model problem; we show the combinatorial structure of the preconditioners and we present encouraging convergence results.

1. Introduction

This paper analyzes a novel method for constructing preconditioners for diagonally-dominant symmetric matrices with positive diagonal entries. A good preconditioner should balance two conflicting objectives. It should approximate the matrix well and it should be easy to factor. For symmetric positive-definite matrices, a good approximation is one that results in clustered eigenvalues for the preconditioned system

Boman's and Hendrickson's work was funded by the Applied Mathematical Sciences, U.S. Department of Energy, Office of Energy Research and performed at Sandia National Labs, a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the U.S. DOE under contract number DE-AC-94AL85000. Chen and Toledo were supported by Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (grant number 572/00 and grant number 9060/99) and by the University Research Fund of Tel-Aviv University.

$M^{-1/2}AM^{-1/2}$, where M is the preconditioner and A is the matrix. The method discussed here is based on a simple idea: we construct M by simply dropping offdiagonal nonzeros from A and modifying the diagonal elements to maintain a certain row-sum property. The trick, of course, is to drop nonzeros in a way that makes M easier to factor than A but still clusters the eigenvalues.

The preconditioners that we analyze in this paper were proposed about a decade ago by Pravin Vaidya in an unpublished manuscript [19] which he presented in a scientific meeting. In that manuscript, Vaidya proposed the overall idea of dropping elements of A to form M, sketched a method for analyzing such preconditioners, and proposed a family of preconditioners. The preconditioners that he proposed are based on a construction called a *maximum-weight basis*, and are parametrized by a single parameter t that controls how many nonzeros are dropped. When the matrices are not only diagonally-dominant with positive diagonals and symmetric, but have only nonpositive offdiagonals, the maximum-weight basis corresponds to a maximum spanning tree of the graph of the matrix. This special case received quite a bit of attention. The theory required to analyze maximum-spanning-tree preconditioners was developed by Gremban et al. [9, 10] and by Bern et al. [4], and the performance of the preconditioners in practice was investigated by Chen and Toledo [6, 7] (Vaidya's manuscript contains no proofs and no experimental results). The more general case was never fully analyzed.

We note that Gremban [10] showed that any linear system with an n-by-n symmetric diagonally-dominant coefficient matrix can be solved by solving a related system with a $2n$-by-$2n$ symmetric diagonally dominant coefficient matrix with nonpositive off-diagonals. This transformation allows one to apply Vaidya's maximum-spanning-tree preconditioners to all diagonally-dominant symmetric matrices. This transformation has several drawbacks compared to the method presented in this paper. The larger $2n$-by-$2n$ matrices are likely to lead to slower solution times since vector-vector and matrix-vector operations take longer and the factor of the matrix is likely to fill more. Also, the transformation does not preserve graph properties that may be relevant to performance, such as planarity (but it does preserve vertex separators; a size s separator in the original matrix corresponds to a $2s$ separator in the larger matrix). A more detailed comparison of the two approaches is beyond the scope of this paper.

This paper analyzes maximum-weight-basis (MWB) preconditioners. It turns out that the analysis is quite complex, much more so than the special case of maximum spanning trees. We also present an efficient algorithm for constructing MWB preconditioners. The algorithm, too, is nontrivial and requires a sophisticated data structure to ensure its efficiency. Unlike previous analysis of preconditioners based on Vaidya's dropping idea [4, 9, 10, 12], our analysis is based not on graph embeddings, but on an algebraic generalization developed by Boman and Hendrickson [5]. We have implemented the preconditioner and the paper presents a numerical example that shows that it is effective in practice. This example is only meant to illustrate the structure of the preconditioner and its performance; it is not a thorough experimental study.

Vaidya's unpublished work has led to research in several directions. Some of the research provided proofs for Vaidya's claims. Gremban et al. proved some of the basic spectral lemmas [9, 10], Bern et al. proved a few more lemmas and analyzed maximum-spanning-tree preconditioners [4], Reif analyzed Vaidya's proposed recursive preconditioners, in which M is not factored completely [17], and this paper analyzes Vaidya's MWB preconditioners. Other research focused on applying Vaidya's analysis technique to other preconditioners. Guattery used the technique to analyze a class of incomplete-factorization preconditioners [12], and Bern et al. used the technique to analyze another class of incomplete-factorization preconditioners, as well as a simple multilevel preconditioner. Finally, some of the research uses Vaidya's techniques to design new preconditioners. Gremban et al. proposed a cheap-to-factor hierarchical preconditioner for diagonally-dominant positive-definite symmetric matrices [9, 10] and Howle and Vavasis extended Gremban's preconditioners to complex symmetric linear systems [13].

The rest of the paper is structured as follows. Section 2 presents technical tools that we use in the analysis of the preconditioners. In Section 3 we show that whenever we form a preconditioner for a diagonally-dominant symmetric matrix by symmetrically dropping nonzeros and modifying diagonal elements appropriately, the small eigenvalue of the preconditioned matrix is at least 1. Section 4 shows how to represent A as a sum of rank-1 matrices $A = \sum_{k=1}^{m} u_k u_k^T$, where each rank-1 matrix corresponds to one edge of G_A, the underlying graph of A. The vectors u_k, which we call scaled *edge vectors*, play a prominent role in this paper. The next section, Section 5, characterizes the structure of independent sets of edge vectors. Section 6 uses this characterization to prove that if the preconditioner M corresponds to an independent set of edge vectors that maximizes the sum of the scaling factors of the vectors, then the large eigenvalue of the preconditioned matrix is at most $4nm$, where n is the dimension of A and m is the number of nonzeros in the strictly upper triangular part of A. Finding the set of vectors that maximizes this sum is conceptually simple, since the vectors form a combinatorial structure known as a *matroid*. There exists a generic algorithm that finds a *maximum-weight basis* in any matroid, and it turns out that the set of vectors that we seek form such a basis. The generic algorithm requires an efficient subroutine for determining whether a given vector is dependent on an independent set of vectors. We present an efficient independence-testing algorithm in Section 7. Section 8 shows how to augment a MWB preconditioner with additional nonzeros from A, in a way that guarantees a reduction in the condition number. More specifically, we show how to add $O(t^2)$ nonzeros to M so that the total number of nonzeros is bounded by $O(n+t^2)$ and so that the condition number drops to $O(n^2/t^2)$, when the number of nonzeros per row in A is bounded by a constant. (Reif shows a construction that overcomes the need to bound the number of nonzeros per row [17].) Section 9 presents a numerical example. We present a simple class of diagonally-dominant symmetric matrices and construct augmented-MWB preconditioners for them. We graphically show the phases of the construction algorithm on a small matrix. We also show that augmented-MWB preconditioners perform well compared to modified incomplete Cholesky preconditioners on a large matrix. We conclude the paper with a summary of our results.

2. Background

This section presents technical tools that we use in the analysis of the preconditioners.

The number of iterations of the conjugate-gradients method for the solution of systems of linear equations $Ax = b$ is bounded above by the square root of the spectral condition number $\kappa(A)$ of A. (To reduce the norm of the residual by a constant factor.) The condition number is the ratio of the extreme eigenvalues of A, $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$. The Conjugate Gradient method can be used to solve consistent linear systems with a singular coefficient matrix A when a basis for the null space of A is known. In such cases, the number of iterations is proportional to square root of the ratio of the extreme positive eigenvalues. When a preconditioner M is used in the conjugate-gradients method, the number of iterations is proportional to the the square root of the ratio of the extreme finite generalized eigenvalues of the pair (A, M), defined below.

Definition 2.1. The number λ is a finite generalized eigenvalue of the matrix pencil (A, M) if there exists a vector x such that $Ax = \lambda Mx$

and $Mx \neq 0$. We denote the set of finite generalized eigenvalues by $\lambda_f(A, M)$.

Henceforth whenever we refer to an "eigenvalue" of a matrix pencil, we mean a finite generalized eigenvalue.

To bound the amount of work in the Preconditioned Conjugate Gradient method, we need to bound the finite eigenvalues of (A, M). We need to prove two bounds: an upper bound on $\max \lambda_f(A, M)$ and a lower bound on $\min \lambda_f(A, M)$. We will prove the upper bound directly and the lower bound by proving an upper bound on $\max \lambda_f(M, A) = 1/\min \lambda_f(A, M)$. We therefore only need to show how to prove upper bounds on the $\lambda_f(A, M)$, since the lower bound is proved in essentially the same way for the matrix pencil (M, A).

Definition 2.2. The *support* $\sigma(A, M)$ of a matrix pencil (A, M) is the smallest number τ such that $\tau M - A$ is positive semidefinite. If there is no such number, we take $\sigma(A, M) = \infty$.

The importance of support numbers stems from the following lemma, on which all of the analyses of Vaidya's preconditioners and follow-up preconditioners are based.

Lemma 2.3. *(Support Lemma* [10]*) If $\lambda \in \lambda_f(A, M)$ and M is positive semidefinite and* $\mathrm{null}(A) \subseteq \mathrm{null}(M)$*, then $\lambda \leq \sigma(A, M)$.*

The bound given in the Support Lemma is tight if $\sigma(A, M)$ is finite.

The support lemma allows us to bound the spectrum of a matrix pencil by showing that $\tau A - M$ is positive semidefinite. One way to do so is to split $\tau A - M$ into a sum of matrices and to show that each term is positive semidefinite. The next lemma, which was used implicitly by Vaidya and proved by Gremban [10], states this formally.

Lemma 2.4. *(Splitting Lemma) Let $Q = Q_1 + Q_2 + \cdots + Q_m$. If $Q_1, Q_2, \ldots, Q_m$ are all positive semidefinite, then Q is positive semidefinite.*

In previous research, the main tool that was used to prove that the terms of a splitting are positive semidefinite was the so-called Congestion-Dilation Lemma [4]. In this paper we use an algebraic generalization of the Congestion-Dilation Lemma, which is due to Boman and Hendrickson.

Lemma 2.5. *(Rank-1 Support Lemma* [5]*) if $u \in R^{n \times 1}$ is in the range of $V \in R^{n \times k}$, then $\sigma(uu^T, VV^T) = \min w^T w$ subject to $Vw = u$.*

Our analysis relies on the connection between matrices and graphs, which we now define formally.

Definition 2.6. The *underlying graph* $G_A = (V_A, E_A)$ of an n-by-n symmetric matrix $A = (a_{ij})$ is a weighted undirected graph whose vertex set is $V_A = \{1, 2, \ldots, n\}$ and whose edge set is $E_A = \{(i, j) : i \neq j \text{ and } a_{ij} \neq 0\}$. The *weight* of an edge (i, j) is $-a_{ij}$.

Although defining the weight of an edge to be the negative of a_{ij} may seem odd, it turns out to be useful in the context of this paper.

3. Bounding the Smallest Eigenvalue

This section analyzes a certain class of preconditioners for diagonally-dominant symmetric matrices. The graph G_M of the preconditioner M is a subgraph of the graph G_A of A, nonzero off-diagonals in M have the same values as in A, and diagonal elements in M are set up in a way that preserves a generalized row-sum property. For this class of preconditioners, we prove that the smallest eigenvalue of the pencil (A, M) is at least 1. That is, $\lambda_{max}(M, A) \leq 1$.

The preconditioners that we analyze must preserve the generalized row-sums that we define below.

Definition 3.1. The *row-weight* of row i of matrix A is $a_{ii} - \sum_{i \neq j} |a_{ij}|$.

We analyze preconditioners whose row weights equal the row weights of A. If the row weights are nonzero, we subtract from both A and M a diagonal matrix D so that the row weights in $A - D$ and $M - D$ are all zero. Clearly, if $(A - D) - (M - D)$ is positive semidefinite, then $A - M$ is positive semidefinite. Thus, we can assume without loss of generality that the row weights in A and M are zero.

The next lemma proves that the small eigenvalue of the pencil is at least 1.

Lemma 3.2. *If A is a diagonally-dominant matrix with positive diagonal entries and M is a preconditioner whose underlying graph is a subgraph of G_A, and whose row-weights are the same as A's, then $\lambda_{max}(M, A) \leq 1$.*

Proof. Lemma 2.3 shows that if $A - M$ is positive semidefinite, then $\lambda_{max}(M, A) \leq 1$. A and M have the same row weights,

$$a_{ii} - \sum_{i \neq j} |a_{ij}| = m_{ii} - \sum_{i \neq j} |m_{ij}|$$

so

$$a_{ii} - m_{ii} = \sum_{i \neq j} |a_{ij}| - \sum_{i \neq j} |m_{ij}| = \sum_{i \neq j} (|a_{ij}| - |m_{ij}|) \ .$$

The matrix M may contain zeros in positions where A contains non-zeros, but all of M's non-zeros are non-zeros in A (with the same values). Since

$$a_{ii} - m_{ii} = \sum_{m_{ij}=0} |a_{ij}| = \sum_{i \neq j} |a_{ij} - m_{ij}| \, ,$$

$A - M$ is diagonally dominant. Its diagonal elements are sums of absolute values, and hence are nonnegative. Such a matrix is positive semidefinite (see, for example,[3, Theorem 4.9]). □

We will be able to prove this lemma even more simply, once we prove the Congestion-Dilation Lemma for general undirected graphs (Lemma 6.3). Then we could simply state that each edge in M is supported by the equivalent edge A with congestion 1 and dilation 1.

4. Edge Vectors

We now turn our attention to the largest eigenvalue of (A, M). We propose a preconditioner M whose graph is a specific subgraph of G_A, which allows us to prove an $4nm$ upper bound on the eigenvalues of the pencil.

We use the Splitting Lemma and the Rank-1 Support Lemma to prove the upper bound. We split A into a sum of rank-1 matrices $A = \sum_{k=1}^{m} u_k u_k^T$, where each rank-1 matrix correspond to one edge of G_A. We split $4nmM$ trivially into $4nmM = \sum_{k=1}^{m} 4nM$. We then use the Rank-1 Support Lemma (Lemma 2.5) to show that $\sigma(u_k u_k^T, M) \leq 4n$, and hence, that $4nM - u_k u_k^T$ is positive semidefinite. This shows that each of the m terms in the splitting

$$4nmM - A = \sum_{k=1}^{m} \left(4nM - u_k u_k^T\right)$$

is positive semidefinite, and hence the entire sum.

We use Rank-1 Support Lemma to show that $\sigma(u_k u_k^T, M) \leq 4n$ by proving that there exist V and w such that $M = VV^T$, $Vw = u_k$, and the entries w_i of w satisfy $|w_i| \leq 2$.

We now show how to represent A as a sum of rank-1 matrices and how to represent M as $M = VV^T$. These representations rely on the following definitions of *edge vectors* and *vertex vectors*.

Definition 4.1. The edge vector $\langle ij \rangle$ of a nonzero entry $a_{ij} < 0$ in a matrix A has exactly two non-zeros, $\langle ij \rangle_{\min(i,j)} = 1$ and $\langle ij \rangle_{\max(i,j)} = -1$. The edge vector $\rangle ij \langle$ of a nonzero $a_{ij} > 0$ also has two non-zeros, $\rangle ij \langle_i = 1$ and $\rangle ij \langle_j = 1$. The vertex vector $\langle i \rangle$ of row and column i

of a matrix has exactly one nonzero, $\langle i \rangle_i = 1$. All of these vectors are n-by-1 column vectors, where n is the dimension of A.

The next lemma shows how to represent A as a sum of rank-1 matrices $u_k u_k^T$ where each u_k is an edge vector.

Lemma 4.2. *If A is symmetric and has zero row-weights $a_{ii} = \sum_{i \neq j} |a_{ij}|$, then we can split A into*

$$
\begin{aligned}
A &= \sum_{\substack{a_{ij} < 0 \\ i < j}} |a_{ij}| \, \langle ij \rangle \, \langle ij \rangle^T + \sum_{\substack{a_{ij} > 0 \\ i < j}} a_{ij} \, \rangle ij \langle \, \rangle ij \langle^T \\
&= \sum_{\substack{a_{ij} < 0 \\ i < j}} \left(\sqrt{|a_{ij}|} \, \langle ij \rangle \right) \left(\sqrt{|a_{ij}|} \, \langle ij \rangle \right)^T + \sum_{\substack{a_{ij} > 0 \\ i < j}} \left(\sqrt{a_{ij}} \, \rangle ij \langle \right) \left(\sqrt{a_{ij}} \, \rangle ij \langle \right)^T
\end{aligned}
$$

Proof. Each term in the sums contributes to exactly two off-diagonal non-zeros, a_{ij} and a_{ji}, and to two diagonal elements a_{ii} and a_{jj}. Furthermore, each off-diagonal nonzero in A receives contributions from exactly one term in the sums. It is easy to see that the contributions sum up to exactly the correct values. □

The preconditioner M can be written as a sum of rank-1 matrices that correspond to edge vectors. The rank-1 matrices whose sum is M are a subset of the rank-1 matrices whose sum is A,

$$
M = \sum_{\substack{(i,j) \in E_M \\ a_{ij} < 0 \\ i < j}} |a_{ij}| \, \langle ij \rangle \, \langle ij \rangle^T + \sum_{\substack{(i,j) \in E_M \\ a_{ij} > 0 \\ i < j}} a_{ij} \, \rangle ij \langle \, \rangle ij \langle^T .
$$

We define V to be the matrix whose columns are $\sqrt{|a_{ij}|} \, \langle ij \rangle$ and $\sqrt{a_{ij}} \, \rangle ij \langle$ for $i < j$ and $(i,j) \in E_M$. We have $M = VV^T$. The preconditioner M that we construct satisfies the conditions of the next lemma. Once we show that it does indeed satisfy the conditions, the lemma proves the $4nm$ condition-number upper bound.

Lemma 4.3. *Let $A = UU^T$ and let $M = VV^T$, where U is n-by-m and V consists of the first ℓ columns of U. If for every column u_k of U we have $u_k = Vw_k$ for some w_k with entries whose absolute values are smaller than or equal to 2, then $\sigma(A, M) \leq 4mn$.*

Proof. We use the splitting lemma to split $A = \sum_{k=1}^m u_k u_k^T$ and $mM = \sum_{k=1}^m M$ and show that $4nM - u_k u_k^T$ is positive semidefinite. This is true because by the Rank-1 Support Lemma, $\sigma(u_k u_k^T, M) \leq w_k^T w_k \leq 4n$. $\square$

5. The Combinatorial Structure of a Maximum-Weight Basis

Given a set of scaled edge vectors $u_k = \sqrt{|a_{ij}|}\,\langle ij\rangle$ (or $u_k = \sqrt{a_{ij}}\,\rangle ij\langle$) and a weight α_k for each vector u_k, we wish to find a *maximum-weight basis* for the u_k. This basis should consist of a subset of the u_k's and should maximize the sum of the weights of the u_k's in the basis. This section analyses the structure of the maximum-weight basis. We note that Grossman et al. [11] analyzed the structure of the maximum-weight basis in the special case in which all the off-diagonal nonzeros of A are positive. They also described an efficient algorithm for constructing maximum-weight bases for this special case.

We begin by showing a simple property of maximum-weight bases.

Lemma 5.1. *Let $u_1, \ldots, u_\ell$ be a maximum-weight basis for the vectors $u_1, \ldots, u_m$ with weights $\alpha_1, \ldots, \alpha_m$ (that is, we assume without loss of generality that the basis consists of the first ℓ vectors). Let $u_k = \beta_1 u_1 + \cdots + \beta_\ell u_\ell$. If $\beta_i \neq 0$ then $\alpha_i \geq \alpha_k$.*

Proof. Suppose for contradiction that for some i, $\alpha_i < \alpha_k$ and $\beta_i \neq 0$. We show that if we remove u_i from the basis and insert u_k, we end up with another basis with a larger sum of weights. We have

$$u_i = \frac{1}{\beta_i}\left(u_k - \beta_1 u_1 - \cdots - \beta_{i-1}u_{i-1} - \beta_{i+1}u_{i+1} - \cdots - \beta_l u_l\right) .$$

Therefore, the new subset is also spanning. The sum of weights is larger than in the supposedly maximum-weight basis, a contradiction. $\square$

Our next task is more involved. We show that a combinatorial property of a graph ensures that its edge vectors are linearly independent. We need the following definitions.

Definition 5.2. The *sign of an edge* (i, j) in the graph G_A of a symmetric matrix A is the opposite of the sign of a_{ij}. (That is, the sign is positive if $a_{ij} < 0$.) The *sign of a path* in G_A is negative if it contains an odd number of negative edges; otherwise the path is positive.[1]

We can now state the combinatorial property that guarantees linear independence of edge vectors.

[1] Defining the sign of an edge (i, j) to be the opposite of the sign of a_{ij} is somewhat arbitrary, but it allows us to view the sign of a path as the product of the signs of the edges along the path.

Theorem 5.3. *The edge vectors of an undirected graph G_A are linearly independent if and only if each connected component contains no positive cycles and at most one negative cycle.*

We shall prove the theorem later using three technical lemmas that characterize various ways of spanning an edge vector.

The following lemma shows how to span an edge vector using vectors of edges along a *simple path* between the original edge's endpoints. In this paper we use the term *simple path* to stand for a path in which each *edge* appears only once.

Lemma 5.4. *The edge vectors of a simple positive path between vertices i and j span the edge vector $\langle ij\rangle$. The coefficients of the linear combination are all either 1 or -1. The edge vectors of a simple negative path between vertices i and j span the edge vector $\rangle ij\langle$. The coefficients of the linear combination are all either 1 or -1.*

Proof. We prove the lemma by induction on the length of the simple path. The claim is clearly true for paths of length 1. Suppose that the lemma is true for paths of length ℓ. Suppose that there is a path of length $\ell+1$ between i and k such that the vertex just before k in the path is j. By induction, the edges of the path from i to j span $\langle ij\rangle$ if that prefix of the path is positive, or $\rangle ij\langle$ otherwise. There are now four cases. If the edge (j,k) is positive and so is the prefix of the path, then either $\langle ij\rangle+\langle jk\rangle$, $\langle ij\rangle-\langle jk\rangle$, $-\langle ij\rangle+\langle jk\rangle$, $-\langle ij\rangle-\langle jk\rangle$ is equal to $\langle ik\rangle$ (the others are $-\langle ik\rangle$, $\langle ik\rangle-2\langle j\rangle$, and $-\langle ik\rangle+2\langle j\rangle$). The second case occurs when (j,k) is positive but the prefix of the path is negative, the third and fourth when (j,k) is negative and the prefix is either positive or negative. Their analysis is similar and is omitted. $\square$

Lemma 5.5. *The edge vectors of a negative cycle that contains vertex i and of a simple path between i and j, where the edges of the path are disjoint from the cycle, span the vertex vector $\langle j\rangle$. The coefficients of the linear combination are ± 1 for the edges of the path and $\pm 1/2$ for the edges of the cycle.*

Proof. Let (i,k) be an edge in the cycle. If (i,k) is positive, then the path from i to k along the cycle must be negative, since the entire cycle is negative. Lemma 5.4 shows that $\rangle ik\langle$ is a linear combination of the edge vectors along this negative path, with coefficients either 1 or -1. Since $\langle ik\rangle+\rangle ik\langle=2\langle i\rangle$ (if $i<k$; otherwise $-\langle ik\rangle+\rangle ik\langle=2\langle i\rangle$), $\langle i\rangle$ is a linear combination of the edges of the cycle. The coefficients are either $\frac{1}{2}$ or $\pm\frac{1}{2}$. If (i,k) is negative, the rest of the cycle is positive, and a similar argument shows that the cycle spans $\langle i\rangle$. Since the cycle

spans $\langle i \rangle$ with coefficients $\pm\frac{1}{2}$ and the path from i to j spans either $\langle ij \rangle$ or $\rangle ij \langle$ with coefficients ± 1, the cycle and path together span $\langle j \rangle$ with the desired coefficients. □

Lemma 5.6. *The edge vectors of a connected component that contains a negative cycle span the edge vectors $\langle ij \rangle$ and $\rangle ij \langle$, for any two vertices i and j in the component. The coefficients are all ± 1, ± 2 or 0.*

Proof. Suppose that there is a simple path from i to j that contains cycle edges. Then we can construct another simple path from i to j, in which we will replace the cycle edges in the first path with all the other cycle edges. These two simple paths have opposing signs. Therefore, by Lemma 5.4, one path spans $\langle ij \rangle$ and the other spans $\rangle ij \langle$, both with coefficients ± 1.

Now suppose that there is no simple path from i to j that contains cycle edges. Let k be the first vertex that is both on the path from i to the cycle and on the path from j to the cycle. Such a vertex must exist, otherwise there is a simple path between i and j that contains cycle edges, a contradiction of our supposition. The vertex k may, however, be one of i and j. The sign of the path between i and j is determined by the sign of the paths between i and k and between k and j. The vectors $\langle ik \rangle$ and $2\langle k \rangle$ span $\rangle ik \langle$ with coefficients ± 1, which means that the path from i to j and the path from k to the cycle and the cycle span both $\langle ij \rangle$ and $\rangle ij \langle$ with the desired coefficients. □

We are now in position to prove Theorem 5.3,

Proof. ($\Rightarrow$) Suppose to the contrary that there is a positive cycle in G_A. Let e be an edge in that cycle. Then the path between e's endpoints along the cycle has the same sign as e's. Lemma 5.4 shows that the vector corresponding to e is a linear combination of the vectors of the edges along the path. Therefore, the vectors are linearly dependent.

Suppose to the contrary that a connected component contains two simple negative cycles. Let us choose a vertex i in the following way: if the two cycles contain common vertices, then we choose i to be one of those vertices. Otherwise we choose i to be one of the vertices on a path connecting the two cycles. Lemma 5.5 shows that $\langle i \rangle$ is a linear combination of the vectors corresponding to the edges along any of the paths from i to itself traveling through a negative cycle. Since $\langle i \rangle$ could be represented as two different linear combinations of the edge vectors, the vectors are linearly dependent.

($\Leftarrow$) Let $G = (V, E)$ be a graph, where each connected component contains no positive simple cycles, and at most one negative simple cycle. Suppose to the contrary that the vectors corresponding to the edges are linearly dependent. Therefore, there exists a subgraph $G^* = (V, E^*) \subset (V, E)$ and coefficients $\alpha_{ij} \neq 0$, such that

$$\sum_{\substack{(i,j) \in E^* \\ (i,j) \text{ is positive} \\ i<j}} \alpha_{ij} \langle ij \rangle + \sum_{\substack{(i,j) \in E^* \\ (i,j) \text{ is negative} \\ i<j}} \alpha_{ij} \rangle ij \langle = 0 \ .$$

The subgraph G^* cannot contain any leaves. If i is a leaf, only one edge vector contains a nonzero in position i, so this nonzero cannot be canceled out by the other edge vectors in G^*. Also, G^* is a subgraph of G, so each connected component contains no more than one simple cycle. Therefore, G^* is a union of distinct simple negative cycles. By Lemma 5.5, each simple negative cycle of length n_0 spans the n_0-dimensional subspace of the n_0 corresponding vertex vectors, and therefore they are linearly independent. No vertex appears in more than one cycle, so the entire set of vectors is linearly independent, a contradiction. □

Theorem 5.3 provides a characterization of independent edge vectors. The next theorem characterizes the combinatorial structure of any basis of a connected graph.

Theorem 5.7. *Let $G_A = (V, E)$ be a connected graph, let E' be a set of edges corresponding to a basis of the edge vectors of E, and let $G_B = (V, E')$. Then G_B is either a spanning tree of G_A or it is a spanning set of 1-trees (graphs consisting of a tree plus one edge whose endpoints are in the tree[2]).*

Proof. Suppose for contradiction that G_B contains two or more connected components, one of which is a tree. Since G_A is connected, there is an edge $e \in E$ that connects the tree to another component. The edge vector corresponding to e is linearly independent of the edge vectors of G_B, because even after we add e to G_B, no component contains a positive cycle or more than one negative cycle. This contradicts the assumption that the edges of G_B form a basis. Therefore, if G_B contains more than one component, all the components must be 1-trees (and in particular, their cycles must be negative). □

This theorem makes it clear exactly when a spanning tree forms a basis for a set of edge vectors:

[2]In the literature, there are at least two definitions of the term *k-trees* [2, 16]; our definition of 1-trees is *not* a special case of any of those definitions.

Lemma 5.8. *The edges of a spanning tree of a connected graph G_A form a basis for the edge vectors of G_A if and only if G_A has no negative cycles.*

6. The Condition Number of MWB Preconditioners

The characterization of linearly-independent sets of edge vectors that Theorem 5.3 provides will prove useful in the next section, where we use it to efficiently find a maximum-weight basis. Our remaining task in this section is to complete the analysis of the upper bound on the condition number. The next lemma provides the last technical tool that we need.

Lemma 6.1. *Let $u_1, \ldots, u_\ell$ be a maximum-weight basis for a set of m scaled edge vectors $u_k = \sqrt{|a_{ij}|}\,\langle ij\rangle$ (or $u_k = \sqrt{a_{ij}}\,\rangle ij\langle$) with weights $\sqrt{|a_{ij}|}$. Let $u_k = w_1 u_1 + \cdots + w_\ell u_\ell$. Then $w_i \leq 2$.*

Proof. Let $u_k = \sqrt{|a_{ij}|}\,\langle ij\rangle$ (or $u_k = \sqrt{a_{ij}}\,\rangle ij\langle$) be the scaled edge vector we want to support. Let $e = (i, j)$ and let G_M be the graph underlying the maximum-weight basis.

We first show how the edge vectors of the edges in the maximum-weight basis support $\langle ij\rangle$ (or $\rangle ij\langle$). This analysis splits into three cases depending on the connected components that i and j belong to. We then show how the maximum-weight basis itself supports u_k.

If i and j are in the same connected component in the maximum-weight basis and that component has no cycles, then the path between i and j must have the same sign as e's, or else e could have been added to the basis. By Lemma 5.4, the vector $\langle ij\rangle$ (or $\rangle ij\langle$) is a linear combination of the edge vectors of the edges in the maximum-weight basis with coefficients ± 1 or 0.

If i and j are in the same connected component in the G_M, and that component has a negative cycle, then by Lemma 5.6 vector $\langle ij\rangle$ (or $\rangle ij\langle$) is a linear combination of the edges in the maximum-weight basis (without scaling), with coefficients ± 1,± 2 or 0.

If i and j are in two separate connected components, then these two components must both include a negative cycle, or else e could have been added to the basis. By Lemma 5.5, the vector $\langle ij\rangle$ (or $\rangle ij\langle$) is a linear combination of the edges in the maximum-weight basis (without scaling), with coefficients $\pm\frac{1}{2}$, ± 1 or 0.

In all three cases, the $\langle ij\rangle$ (or $\rangle ij\langle$) is a linear combination of the unscaled vectors of the edges in the maximum-weight basis, with coefficients whose absolute values are smaller than or equal to 2. Therefore,

$$w_r = \gamma_r \frac{\sqrt{|a_{ij}|}}{\sqrt{|b_r|}} \, ,$$

where $\gamma_r \le 2$ and where the b_r's are the weights of the edges in the MWB. By Lemma 5.1, $\frac{\sqrt{|a_{ij}|}}{\sqrt{|b_r|}} \le 1$ for $1 \le r \le \ell$. It follows that for $1 \le r \le \ell$, $w_r = \gamma_r \frac{\sqrt{|a_{ij}|}}{\sqrt{|b_r|}} \le 2 \cdot 1 = 2$. □

This concludes the analysis of the condition number of a maximum-weight basis, since we can now apply Lemma 4.3 to prove the upper bound on the spectrum. The lower bound has already been established in Lemma 3.2. We have, therefore, proven the following theorem.

Theorem 6.2. *The condition-number of a matrix pencil (A, M) where A is symmetric, diagonally dominant with positive diagonals and M is a maximum-weight basis preconditioner is bounded by $4mn$.*

As a side effect of our analysis, we can now formulate and prove a generalized Congestion-Dilation Lemma. We essentially use the same technique that Boman and Hendrickson used to prove the original Congestion-Dilation Lemma [5].

Lemma 6.3. *Let $e = (i, j)$ be an edge of weight a. Let u_0 be the scaled vector representing e. Let $V = [u_1, u_2, \ldots, u_\ell]$ be scaled edge vectors $u_k = \sqrt{|b_k|}\,\langle ij\rangle$ (or $u_k = \sqrt{b_k}\,\rangle ij\langle$), corresponding to edges that support e in one of the following ways: either by a simple path whose sign is the same as e's, or by two negative cycles and two paths from each of e's endpoints to the cycles, or by a path from e's endpoints through a negative cycle. Then $\sigma(uu^T, VV^T) \le \frac{4a}{\min\{b_k\}}\ell$.*

Furthermore, in the first two cases the support $\sigma(uu^T, VV^T)$ is bounded by $\frac{a}{\min\{b_k\}}\ell$.

Proof. By Lemmas 5.4, 5.5 and 5.6, e's vector is a linear combination of the vectors in V, with all the coefficients c_k either ± 2, ± 1 or $\pm\frac{1}{2}$. Let the linear combination coefficients be $(c_1, c_2, \ldots, c_\ell)$. Let $w = (c_1\frac{\sqrt{a}}{\sqrt{b_1}}, c_2\frac{\sqrt{a}}{\sqrt{b_2}}, \ldots, c_\ell\frac{\sqrt{a}}{\sqrt{b_\ell}})^T$. Then $u = Vw$. Therefore:

$$\sigma(uu^T, VV^T) \le ww^T = \sum_{k=1}^{\ell} c_k^2 \frac{a}{b_k} \le \sum_{k=1}^{\ell} 4\frac{a}{b_k} \le \sum_{k=1}^{\ell} \frac{4a}{\min\{b_k\}} = \frac{4a}{\min\{b_k\}}\ell$$

In fact, if the support of u_k is done by one of first two ways, then the coefficients of the linear combination are all either ± 1 or $\pm\frac{1}{2}$, and we have

$$\sigma(uu^T, VV^T) \le ww^T = \sum_{k=1}^{\ell} c_k^2 \frac{a}{b_k} \le \sum_{k=1}^{\ell} 1 \cdot \frac{a}{b_k} \le \sum_{k=1}^{\ell} \frac{a}{\min\{b_k\}} = \frac{a}{\min\{b_k\}} \ell$$

□

As in the analysis of the Congestion-Dilation Lemma for M-matrices, we interpret ℓ to be the dilation and $\frac{a}{\min\{b_k\}}$ to be the congestion.

As we have mentioned, we can use this generalized Congestion-Dilation Lemma to provide another proof of Lemma 3.2: each edge in M is supported by the equivalent edge in A with congestion 1 and dilation 1.

7. An Efficient Algorithm for Constructing Maximum-Weight Bases

It turns out that finding a maximum-weight basis for a set of scaled edge vectors is an instance of a well-studied problem [8, Section 17.4]. A set of m scaled edge vectors $u_k = \sqrt{|a_{ij}|}\,\langle ij\rangle$ (or $u_k = \sqrt{a_{ij}}\,\rangle ij\langle)$ with weights $\sqrt{|a_{ij}|}$ and the collection of linearly-independent subsets define a combinatorial structure called a *matroid*. There exists a generic greedy algorithm for finding a so-called *maximal independent set* in a matroid. In our matroid, a maximal independent set is the maximum-weight basis that we wish to construct.

The generic maximum-weight basis algorithm works by sorting the elements of the matroid (the scaled edge vectors) by weight and trying to add them to the basis, starting from the heaviest. The next vector to be considered is added to the independent set if it is linearly independent of the vectors already in the set.

To apply the generic algorithm, we must provide a routine that tests whether an edge vector is linearly dependent on the vectors already in the set. Using a rank-revealing factorization, such as the singular-value decomposition (SVD) is too expensive. The vectors are highly structured, so we can test for linear independence more efficiently.

The algorithm that we use for testing independence relies on the characterization of independent sets that Theorem 5.3 provides. We maintain a data structure that allows us to quickly test whether we can add a new edge vector to the basis. More specifically, we test whether the new edge closes a positive cycle or a second negative cycle in the underlying graph. If so, it is linearly dependent on the edges already in the basis.

The data structure that we use is a forest of shallow rooted trees that represent connected components in the underlying graph. We augment this data structure, which is sometimes referred to as a *union-find* data structure, with labels that allow us to quickly determine the sign of paths in the graph. The basic union-find data structure was apparently first used by McIlroy and Morris (see [1, page 169]); The data structure and its complexity analysis are presented in several textbooks, such as [1] and [8]. Tarjan proposed the augmentation technique that we use [18].

The forest is represented by an array π of length n, where n is the size of the graph. Each rooted tree in the forest represents a connected component of the graph, although the topology of the trees has nothing to do with the topology of the graph. The parent of vertex i is $\pi[i]$. If i is the root of a tree, $\pi[i] = i$. We also maintain an array r; if i is a root then $r[i]$ is the height of the tree rooted at i; r is undefined otherwise. The two arrays π and r are part of the standard implementation of union-find data structures.

Algorithm 1 Finding the representative vertex of a connected component (the root of the tree) with path compression. The algorithm is an augmented version of the standard union-find procedure that also maintains the sign of paths in the graph when the tree is compressed. Indentation denotes block structure.

```
vertex AugmentedFindSet(vertex i)
  temporary vertex j
  if (i ≠ π[i])
      j ← AugmentedFindSet(π[i])
      s[i] ← s[i] xor s[π[i]]
      π[i] ← j
  return π[i]
```

We augment the union-find data structure with two additional bit arrays, s and c. The value $s[i]$ represents the sign of the path in the graph between i and $\pi[i]$ (0 for positive and 1 for negative); it is only defined if the connected component is a tree. The value $c[i]$ is defined only for roots and specifies whether the connected component has a cycle.

Algorithm 2 Unifies i's and j's trees, using an edge whose sign is ℓ. Returns the root of the united tree.

vertex AugmentedUnion(vertex i,vertex j,boolean edgesign)
  temporary vertices ρ_i,ρ_j // *representatives of i and j*
  $\rho_i \leftarrow \pi[i]$
  $\rho_j \leftarrow \pi[j]$
  if $(r[\rho_i] > r[\rho_j]$)
    $\pi[\rho_j] \leftarrow \rho_i$
    $s[\rho_j] \leftarrow s[i] \,\mathrm{xor}\, s[j] \,\mathrm{xor}$ edgesign
    return ρ_i
  else
    $\pi[\rho_i] \leftarrow \rho_j$
    $s[\rho_i] \leftarrow s[i] \,\mathrm{xor}\, s[j] \,\mathrm{xor}$ edgesign
    if $(r[\rho_i] = r[\rho_j])$
      $r[\rho_j] \leftarrow r[\rho_j] + 1$
    return ρ_j

Our algorithm is presented in Algorithms 1, 2, 3 and 4. Algorithm 4 is an instance of the generic greedy maximal-independent-set algorithm, applied to our case. Algorithm 3 tests for independence; it uses Algorithms 1 and 2 as subroutines.

The correctness of the algorithm relies on the correctness of the generic greedy algorithm, the correctness of the union-find data structure, on Theorem 5.3, and on the correct maintenance of the arrays s and c. The correct maintenance of c is trivial. The correct maintenance of s is more challenging to prove. We start with a simple technical lemma.

Lemma 7.1. *If i, j, and k are vertices in a connected component of G_A that contains no cycles and*

$$s[i,j] = \begin{cases} 0 & \text{if the path in } G_A \text{ between } i \text{ and } j \text{ is positive} \\ 1 & \text{otherwise,} \end{cases}$$

then $s[i,k] = s[i,j] \,\mathrm{xor}\, s[j,k]$.

Proof. Since the connected component is a tree, the paths from i to j and from k to j must meet at some vertex x; from x to j the two paths are identical (x and j may be the same vertex). The simple path from i to k is in fact concatenation of the path from i to x to the path from

Algorithm 3 Given two vertices i and j and the weight w of the edge connecting them, this algorithm adds (i, j) to the basis if and only if the edge is independent of the current independent set.

```
AddEdgeIfIndependent (vertex i, vertex j, real w)
  temporary vertices ρ_i, ρ_j, unionroot
  temporary boolean edgesign
  edgesign ← (w > 0)
  ρ_i ← AugmentedFindSet(i)
  ρ_j ← AugmentedFindSet(j)
  if (ρ_i ≠ ρ_j)
      // i and j are in different connected components
      if ((c[ρ_i] = 0)or(c[ρ_j] = 0))
          // one of the connected components does not contain a
cycle
          AddEdgeToBasis(i, j, w)
          unionroot ← AugmentedUnion(i, j,edgesign)
          c[unionroot] ← c[ρ_i] or c[ρ_j]
  else
      // i and j are in the same connected component
      if ((edgesign ≠ s[i] xor s[j]) and(c[ρ_1]==0))
          // the connected component does not contain a cycle, and
          // adding (i, j) does not close a positive cycle
          AddEdgeToBasis(i, j, w)
          c[ρ_i] ← 1
```

Algorithm 4 This is the generic greedy algorithm to find a maximal-independent set of a matroid.

```
GreedyMaximumWeight()
  Sort(edges by absolute value of weight)
  foreach (e=(i,j) an edge of weight w)
    if (EdgeIsIndependent(i, j, w))
      AddEdgeToBasis(e)
```

x to k. Therefore,

$$\begin{aligned} s[i,j] \operatorname{xor} s[j,k] &= (s[i,x] \operatorname{xor} s[x,j]) \operatorname{xor} (s[j,x] \operatorname{xor} s[x,k]) \\ &= s[i,x] \operatorname{xor} (s[x,j] \operatorname{xor} s[j,x]) \operatorname{xor} s[x,k] \\ &= s[i,x] \operatorname{xor} s[x,k] \\ &= s[i,k] \ . \end{aligned}$$

□

The next three lemmas show that the algorithm does, indeed, maintain s correctly.

Lemma 7.2. AUGMENTEDFINDSET *preserves the correctness of s. That is, if the array s is correct before the call of* AUGMENTEDFINDSET, *then it is correct after the subroutine returns.*

Proof. AUGMENTEDFINDSET changes the values of π and c along the path from a vertex i to the root. We prove the correctness by induction on the distance from the root. If i is the root, the algorithm returns immediately, so the claim holds. Suppose the lemma is correct for all the vertices between vertex i and the root. By Lemma 7.1, we have that $s[i, \mathrm{root}] = s[i, \pi[i]] \operatorname{xor} s[\pi[i], \mathrm{root}]$. The parent of $\pi[i]$ the recursive call is the root, and the parent of i when the subroutine returns is also the root, so correctness is maintained. □

Lemma 7.3. *If the arguments i and j to* AUGMENTEDUNION *are immediate children of roots and if s is correct before the call, then s is maintained correctly by* AUGMENTEDUNION.

Proof. The only change in the array π is $\pi[\rho_j] = \rho_i$ or $\pi[\rho_i] = \rho_j$. By Lemma 7.1 $s[\rho_i, \rho_j] = s[\rho_i, i] \operatorname{xor} s[i, j] \operatorname{xor} s[j, \rho_j]$. By the hypothesis of the lemma, $\rho_i = \pi[i]$ *and* $\rho_j = \pi[j]$. *Since* $s[\rho_i, i] = s[i]$, $s[j, \rho_j] = s[j]$ *and* $s[i, j] = \ell$, *the lemma is correct.* □

Lemma 7.4. ADDEDGEIFINDEPENDENT *is correct.*

Proof. If i and j are in different connected components and both contain cycles, the routine returns without adding (i, j) to the basis. If they are in different components and at most one contains a cycle, the routine adds (i, j).

If i and j are in different components, then $s[i] \operatorname{xor} s[j]$ is the sign of the path between them, since both are children of the same root. In that case, the routine adds the edge if and only if the sign of the edge is different from the sign of the path (i.e., the edge closes a negative cycle) and there is no cycle in the component.

The correct maintenance of s follows from the fact that we call AUGMENTEDUNION only when the arguments are children of roots. □

The complexity analysis of the algorithm is simple. It shows that the running time of the algorithm is dominated by sorting the edges. The total cost of the calls to ADDEDGEIFINDEPENDENT is essentially linear in m. The proof is essentially identical to Tarjan's analysis of augmented union-find data structures in [18]. Note that we do not claim that this algorithm is optimal; in particular, there may be a way to avoid sorting the edges.

Theorem 7.5. GREEDYMAXIMUMWEIGHT *runs in* $O(m \lg m + m\alpha(m,n)) = O(m \lg m)$, *where* α *is the inverse of Ackermann's function.*

Proof. Sorting the edges takes $O(m \lg m)$ time.

We make m calls to ADDEDGEIFINDEPENDENT. Each call makes two calls to AUGMENTEDFINDSET and at most one to AUGMENTEDUNION. The other costs in ADDEDGEIFINDEPENDENT are $O(1)$. Since AUGMENTEDFINDSET and AUGMENTEDUNION are $O(1)$ modifications to the corresponding standard union-find routines, the total cost of all the calls is $O(m\alpha(m,n))$. □

We can now bound the total amount of work required to solve linear systems using MWB preconditioners.

Theorem 7.6. *The total amount of work to solve a symmetric diagonally-dominant linear system with positive diagonals using the conjugate-gradients method with a MWB preconditioner is*

$$O(m \lg m + n + n\sqrt{mn})$$

where n *is the size of the system and* m *is the number of offdiagonal entries in the strictly upper (or lower) part of the coefficient matrix.*

Proof. Constructing the preconditioner costs $O(m \lg m + m\alpha(m,n)) = O(m \lg m)$.

Factoring the preconditioner costs $O(n)$ work. We eliminate first non-cycle vertices using the minimum-degree algorithm. The vertices that we eliminate always have degree 1 so their elimination costs $O(1)$ and generates no fill. We then eliminate the vertices that belong to cycles. Each such elimination generates one fill edge (except for the last 3 vertices in each cycle that generate no fill), so they also cost $O(1)$ to eliminate.

Finally, the preconditioner has $O(n)$ nonzeros, so each conjugate gradients iteration costs $O(n+m)$ work. Since the condition number is bounded by $4mn$, the number of iterations is bounded by $O(\sqrt{mn})$. □

8. AUGMENTED MAXIMUM-WEIGHT BASES

Our next step is to apply Vaidya's augmentation strategy to maximum-weight basis preconditioners. We construct the preconditioner in the following way:

(1) We first find the maximum-weight basis G_C of the graph G_A. We call this basis the *core basis*.
(2) We partition each connected component of the basis into connected subgraphs whose sizes are between n/t and $((d+1)n/t)+$

1, where d is the maximum degree of vertices in the MWB. Components whose initial size is at most n/t need not be partitioned at all.

(3) In each subgraph, we complete the set of edges induced by the core basis to a MWB of the subgraph.

(4) For each pair of subgraphs, we complete the set of edges induced by the core basis on the pair to a MWB of the pair. This final step is equivalent to the step of adding the heaviest edge between each pair of subgraphs in Vaidya's MST preconditioners.

We show that the support of this preconditioner, like that of Vaidya's preconditioner for M-matrices, is $O(n^2/t^2)$.

To partition a connected component of the basis, we remove one cycle edge (if it has a cycle) and use an algorithm called TreePartition, to decompose the remaining tree into connected subtrees. TreePartition, which is shown in Figure 8.1, decomposes recursively a rooted tree T into a set of connected subtrees. We denote the maximum number of children in the tree by d. The theorem below states that the number of vertices in each subtree is between n/t and $(dn/t)+1$, except for the subtree containing the root, which might be smaller. This algorithm is taken from [6, 7], where the correctness theorem is proved. The paper [6, 7] also discusses how certain details in TreePartition affect the sizes of subgraphs in practice.

Theorem 8.1. TreePartition(i) *splits the subtree rooted at i into connected subtrees. The size of each subtree is between n/t and $(dn/t)+1$, except perhaps for the subtree that contains i, which may be smaller (but not larger).*

The implementation uses a global array s of n integers, where s_i is initialized before the first call to TreePartition to be the number of vertices in the subtree rooted at i. The initial call to TreePartition passes the root of T as an argument. Partitioning a MWB is now straightforward.

Lemma 8.2. *Each connected component of a maximum-weight basis whose size is at least n/t can be partitioned into connected subgraphs whose sizes are between n/t and $((d+1)n/t)+1$.*

Proof. Each connected component contains at most one cycle. We can remove one edge from the cycle, and be left with a tree. We choose an arbitrary root for the tree. Running TreePartition on the tree partitions it into subgraphs whose sizes are between n/t and $(dn/t)+1$, except for the subgraph containing the root which may be smaller. If

```
TreePartition(vertex i)
    # comment: s_i = number of vertices in the subtree rooted at i
    s_i ← 1
    for each child j of i
        if (s_j > n/t + 1)
            TreePartition(j)
        if (s_j ≥ n/t)
            form a new subtree rooted at j
            disconnect j from i
        else
            s_i ← s_i + s_j
```

Figure 8.1. The algorithm that we use to decompose the maximum spanning tree. The code splits the tree T, which is stored in a global data structure. The code uses a global integer array s.

the size of the root's subtree is smaller than n/t, we connect it to one of the adjacent subtrees (whose size is no greater than $(dn/t)+1$). Therefore the basis is partitioned into connected subgraphs whose sizes are between n/t and $((d+1)n/t)+1$. □

In addition to partitioning large connected components, we also bundle together small components into subgraphs whose sizes are between n/t and $2(n/t)$, except perhaps for one subgraph which might be smaller.

Our next task is to show that we can always complete a subset of the core maximum-weight basis induced by a subgraph to a maximum-weight basis of the subgraph. We prove this using a more general lemma that applies to any matroid in which dependence is linear dependence.

Lemma 8.3. *Let S be a group of vectors with a maximum-weight basis $M \subseteq S$, and let $S' \subset S$. Then $M \cap S'$ can be completed to form a maximum-weight basis of S'.*

Proof. We assume without loss of generality that the basis M was constructed using the generic greedy algorithm. We prove the lemma by showing that the greedy algorithm, when executed on S', finds a basis M' such that $(M \cap S') \subseteq M'$. More specifically, we run the greedy algorithm on S' using the ordering of vectors that is induced by the ordering that was used to construct M.

Let us assume to the contrary that $M \cap S'$ cannot be completed into a maximum-weight basis of S'. We consider a run of the generic greedy algorithm to form a basis for S'. We start with an empty set

M', and add vectors one by one to the set, by weight. We add the next vector if it is linearly independent of the vectors already in M'. For the assumption to hold, we must eventually come upon a vector $s \in M \cap S'$ that we cannot add to M'.

Let $s \in M \cap S'$ be the first vector that we cannot add to M'. This vector must be a linear combination of the vectors already in M'. These vectors are either vectors in M or vectors that were rejected when M was constructed but accepted when M' was constructed. Vectors that were rejected from M must have been linear combinations of preceding vectors which were accepted into M. Therefore, s is linearly dependent of vectors in M that precede it in the weight ordering. Hence, s would not have been accepted into M, so $s \notin M \cap S'$, a contradiction. □

We apply this lemma in the following way. The basis M corresponds to G_C, the core MWB of G_A, and S' corresponds to the set of edges induced by a subgraph of G_A.

This lemma allows us to bound the number of edges that are needed to complete a basis for a subgraph or a pair of subgraphs. The edge subset G_C, when restricted to the edges induced by the k vertices of a connected subgraph of G_C, contains at least $k-1$ edges. Therefore, we need to add at most one edge to complete the induced subset to a maximum-weight basis of the subgraph. When the maximum-weight basis is induced upon a pair of connected subgraphs, it contains at least $2(k-1)$ edges. Therefore, for each pair, at most two edges need to be added. If the subgraph of G_C consists of a bundle of several small connected components, the induced subset is already a maximum-weight basis. Similarly for a pair of such bundles. To a pair consisting of one bundle of small components and one large connected component, we may need to add at most one edge to complete a basis.

Theorem 8.4. *An augmented maximum-weight basis preconditioner M supports a symmetric diagonally-dominant matrix A whose diagonal entries are positive with support number bounded by*

$$O\left(\frac{d^3 n^2}{t^2}\right) ,$$

where $d+1$ is the maximum number of nonzeros per row in A, and where t is the parameter used to partition the core basis. Furthermore, the graph of the preconditioner has $n + O(t^2)$ edges.

Proof. We split A into $A = \sum_{k=1}^{m} A_k = \sum_{k=1}^{m} u_k u_k^T$, a sum of rank-1 matrices corresponding to edges in G_A. We split the preconditioner M into a sum of $k+1$ matrices, $M = R + \sum_{k=1}^{m} M_k$, where M_k supports A_k and R is symmetric positive semidefinite. For an edge k whose endpoints are in a single subgraph of G_C, M_k corresponds to a scaling of the MWB of that subgraph. For an edge whose endpoints are in two distinct subgraphs of G_C, M_k corresponds to a scaling of the MWB of the pair of subgraphs. An edge k' in G_M supports at most $d(((d+1)n/t)+1)$ edges. Consider an endpoint i of k'. This vertex is in some subgraph of G_C, and k' supports only edges with at least one endpoint in this subgraph. There are at most $d(((d+1)n/t)+1$ edges incident to vertices in a subgraph, hence the bound.

Therefore, we scale each maximum-weight basis by a factor of $d(((d+1)n/t)+1$ to form M_k. This ensures that R remains positive semidefinite.

By Lemma 6.3, M_k support A_k with support number

$$4 \cdot d \left(\frac{(d+1)n}{t} + 1 \right) \cdot 2 \left(\frac{(d+1)n}{t} + 1 \right) = O\left(\frac{d^3 n^2}{t^2} \right) .$$

We now bound the number of edges in the preconditioner. There are at most n edges in the core basis. There are $O(t)$ subgraphs of the core basis, and for each one and each pair we add at most two edges to the preconditioner. This proves the bound on the number of edges in the G_M. □

We now analyze the amount of work required for the entire solution process.

Theorem 8.5. *The total amount of work to solve a symmetric diagonally-dominant linear system with positive diagonals and with a bounded number of nonzeros per row using the conjugate-gradients method with an augmented MWB preconditioner is*

$$O(n \lg n + (n + t^6) + (n + t^4) \cdot n/t) = O(t^6 + n^2/t + nt^3)$$

where n is the size of the system and t is the number of subgraphs that the MWB is partitioned into.The asymptotically optimal value for t is $t = \Theta(n^{0.25})$, which yields $O(n^{1.75})$ work.

Proof. Constructing the preconditioner costs $O(n \lg n)$. We first build the core basis. We then split the basis into subgraphs and complete the basis for each subgraph. The total cost of all the completions is $O(n)$ since we process each edge by weight and determine whether it completes a subgraph basis in $O(\alpha(n, n))$ amortized work. Next, we complete the bases for all pairs in a similar way: we go over the edges in weight order and determine whether each one can be added to a pair basis. The cost per edge is the same as in the previous case.

The rest of the proof follows almost exactly the proof of Lemma 4.1 in [4]. The only difference is that the elimination of a cycle edge may introduce one fill edge, but this does not affect the analysis. □

If G_A is planar, we can show a better work bound. If the graph is planar and the MWB is a tree, Bern et al. [4] show that the total amount of work to solve a linear system is $O(n^{1.2})$. Their bound relies on two observations:

(1) Since each subgraph in the partitioning of the core basis is connected, we add at most $O(t)$ edges to the preconditioner to form basis for the pairs (versus $O(t^2)$ in the general case). This is true since the subgraphs themselves can be contracted to form a planar graph with $O(t)$ vertices.
(2) The augmented-MST basis without the edges that complete the pair bases forms a forest. We eliminate all the degree-1 and -2 vertices that are not adjacent to these pair edges. This introduces only $O(n)$ fill edges, costs only $O(n)$ work, and preserves planarity. The remaining forest has at most $O(t)$ vertices (those adjacent to pair edges). Thus, after eliminating degree-1 and -2 vertices, we are left with a planar graph with $O(t)$ vertices. Using nested dissection, we eliminate the rest of the vertices, which costs $O(t^{1.5})$ work and generates $O(t \lg t)$ fill elements.

This analysis remains valid for MWB as long as the decomposition of the core basis into subgraphs generates at most $O(t)$ connected subgraphs. If, on the other hand, the decomposition generates subgraphs with several connected components (this happens when the core basis includes many small 1-trees which our algorithm bundles into larger subgraphs), the analysis breaks down. Specifically, we may need to add $O(t^2)$ edges to support all the pairs of subgraphs, as in the general case. We conjecture that there may be a clever way to bundle small 1-trees into subgraphs that guarantees that only $O(t)$ edges are added to support all pairs of subgraphs, which would imply that we can solve such systems in $O(n^{1.2})$ total work.

9. A Numerical Example

This section presents a numerical example using a class of matrices that arise from a finite-differences discretization of the partial-differential equation

$$c_x \frac{\partial^2 u}{\partial x^2} - c_y \frac{\partial^2 u}{\partial y^2} + cu = g \ ,$$

where c, c_x, and c_y are positive constants and g is an arbitrary function. We discretized the equation with periodic boundary conditions using a 5-point stencil on a square domain. The graph of the matrix is shown in Figure 9.1. We selected c so that the matrix has zero row weights (except for one row where we increased the row sums to obtain a nonsingular matrix). We used right-hand sides with uniform random elements in $[0, 1]$ for the discretized equations.

We have implemented the augmented-MWB construction algorithm and have used it to solve these linear systems. The code is an extension of the augmented maximum-spanning-tree preconditioning code described in [6, 7].

Note that when the mesh has an even number of points in the y direction, the graph of the coefficient matrix has only positive cycles. Therefore, in these cases the bases reduce to spanning trees, as stated in Lemma 5.8.

Figure 9.1 shows the graph of A for a discretization on an 11-by-11 mesh, the graph of the maximum-weight basis, the partitioning of the basis into two subgraphs of roughly half the size, and the edges that augment the MWB.

Figure 9.2 describes the convergence of augmented-MWB (AMWB) and modified-incomplete-Cholesky (MICC) preconditioners on this model problem with a mesh size of 1001-by-1001. We used METIS [14, 15] to order the AMWB preconditioners to reduce fill. We do not preorder the matrix prior to the incomplete factorization, so it remains ordered using the natural row-by-row ordering of the mesh. The figure shows convergence for isotropic problems, where $c_x = c_y = 1$, and for anisotropic problems, where $c_x = 1$ and $c_y = 100$ or vice versa.

The graphs indicate that AMWB are more effective than MICC preconditioners on this problem. They converge at about the same rate when the direction of weak influences coincide with the order of incomplete elimination (y-direction anisotropy). The AMWB preconditioners maintain their performance when the direction of weak influences changes, but the MICC preconditioners deteriorate. The AMWB preconditioners are also more effective on isotropic problems.

This example is meant to illustrate the construction of MWB preconditioners and to demonstrate that they can be effective. We do not claim that this example establishes that MWB are effective and efficient in practice. To do so would require a more detailed study that examines several classes of problems, several performance metrics

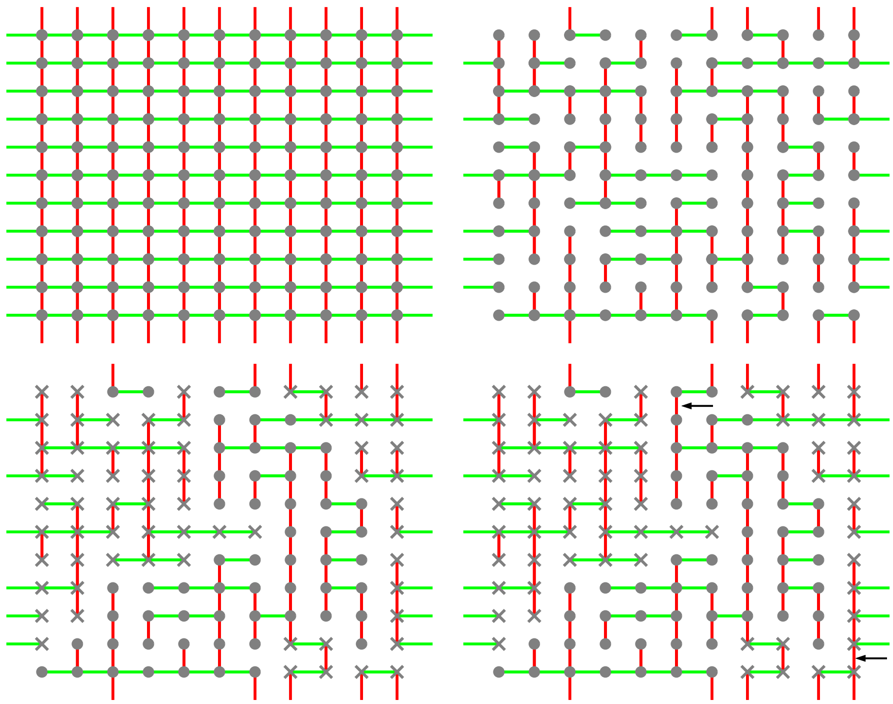

FIGURE 9.1. The graph of A for an 11-by-11 mesh (top left), the maximum-weight basis (top right), the partitioning into subgraphs (bottom left), and the complete preconditioner (bottom right). Red lines denote negative edges and green lines denote positive ones. The edges at the boundaries of the mesh are wraparound edges that connect a vertex at the top boundary to a vertex in the bottom with the same left-right displacement, and similarly for the left and right boundary vertices. In the two bottom graphs, $\times$'s denote the vertices of one subgraph and filled circles the vertices of the other subgraph (the basis was split into 2 subgraphs in this case). The two edges marked by black arrows in the preconditioner were added to the maximum-weight basis to complete the bases of the two subgraphs. Since there are only two subgraphs, no edges were required to complete the base of the pair. Note that the maximum-weight basis has a negative cycle in this case.

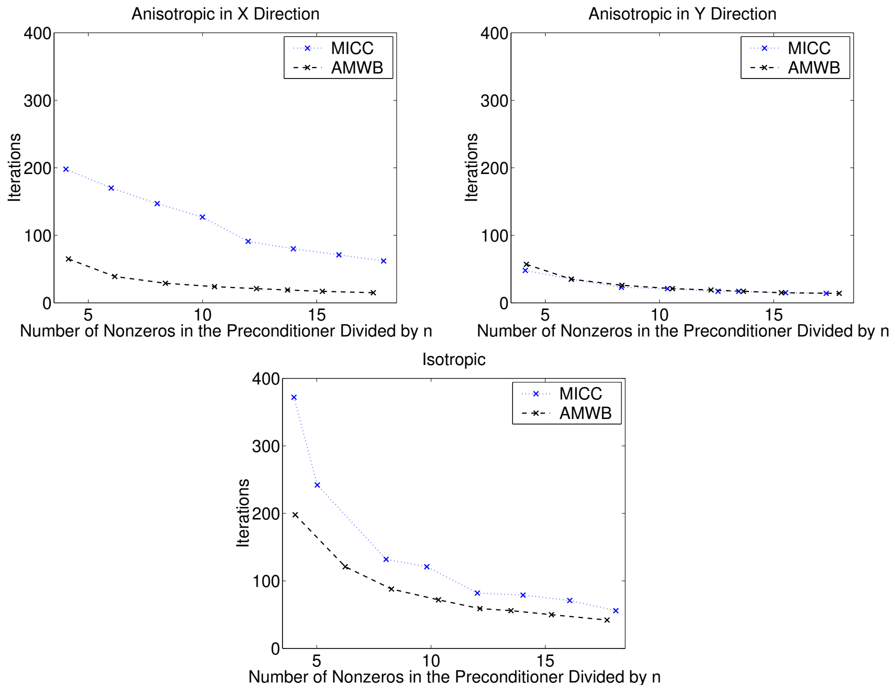

FIGURE 9.2. The convergence of augmented MWB versus the convergence of modified incomplete Cholesky (MICC) preconditioners for the model problem of this section. The mesh size was 1001-by-1001. The graphs show the number of iterations required to reduce the 2-norm of the residual by a factor of 10^8 in a conjugate-gradients solver as a function of the fill in the preconditioners.

(both convergence and running times), and perhaps several ordering algorithms.

10. Conclusions

This paper presents maximum-weight-basis preconditioners for diagonally-dominant positive-definite symmetric matrices. The theory presented here proves Vaidya's claims concerning these preconditioners, which he proposed about ten years ago. The theory presented here extends considerably our ability to analyze preconditioners by means of splittings and support bounds.

Much of the appeal of augmented-MWB preconditioners and other preconditioners proposed or inspired by Vaidya stems from the fact that their performance can be analyzed rigorously. Theorems 7.6 and 8.5

provide bounds on the amount of work required to solve linear systems using MWB and augmented-MWB preconditioners. These bounds apply to discretizations with unstructured grids and to problems with inhomogeneous (variable) coefficients. Provable bounds for incomplete-factorization preconditioners are typically weaker.

We have implemented the algorithm presented in this paper. We used this implementation to solve a model problem. Our limited experiments show that augmented-MWB preconditioners are effective.

Acknowledgements. Thanks to Haim Kaplan for pointing us to Tarjan's paper on augmented union-find data structures.

References

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] Baruch Awerbuch, Yossi Azar, Avrim Blum, and Santosh Vempala. New approximation guarantees for minimum-weight k-trees and prize collecting salesmen. *SIAM Journal on Computing*, 28:254–262, 1998.

[3] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1994.

[4] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. Technical report, School of Computer Science, Tel-Aviv University, 2001. Submitted for publication.

[5] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. Technical Report SAND-01-1794J, Sandia National Labs, March 2001. Submitted for publication. A preliminary version was presented at Preconditioning 2001, Tahoe, Californa in April 29–May 1, 2001.

[6] Doron Chen. Analysis, implementation, and evaluation of Vaidya's preconditioners. Master's thesis, School of Computer Science, Tel-Aviv University, 2001.

[7] Doron Chen and Sivan Toledo. Implementation and evaluation of Vaidya's preconditioners. Submitted for publication. Available online at `http://www.tau.ac.il/~stoledo/pubs.html`. A preliminary version was presented at Preconditioning 2001, Tahoe, Californa in April 29–May 1, 2001., February 2001.

[8] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[9] K.D. Gremban, G.L. Miller, and M. Zagha. Performance evaluation of a parallel preconditioner. In *9th International Parallel Processing Symposium*, pages 65–69, Santa Barbara, April 1995. IEEE.

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

[11] Jerrold W. Grossman, Devadatta M. Kulkarni, and Irwin E. Schochetman. Algebraic graph theory without orientation. *Linear Algebra and its Applications*, 212:289–307, 1994.

[12] S. Guattery. Graph embedding techniques for bounding condition number of incomplete-factor preconditioners. Technical Report 97-47, ICASE, NASA Langley Research Center, 1997.

[13] V. Howle and S. Vavasis. An iterative method for solving complex-symmetric systems arising in electric power modeling. Technical Report TR 00-5, Cornell Computational Optimization Project, Cornell University, 2000. Available online at `http://www.orie.cornell.edu/~ccop/reports.html`.

[14] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998.

[15] George Karypis and Vipin Kumar. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *Journal of Parallel and Distributed Computing*, 48:71–85, 1998.

[16] Lata Narayanan and Naomi Nishimura. Interval routing on k-trees. *Journal of Algorithms*, 26:325–369, 1998.

[17] J. H. Reif. Efficient approximate solution of sparse linear systems. *Computers and Mathematics with Applications*, 36:37–58, 1998. See also the errata in Volume 38, page 141, 1999. Both available online at `www.cs.duke.edu/~reif`.

[18] Robert Endre Tarjan. Applications of path compression on balanced trees. *Journal of the ACM*, 26:668–689, 1979.

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

Erik Boman and Bruce Hendrickson: Sandia National Labs, Albuquerque, NM 87185-1111, USA. Doron Chen and Sivan Toledo: School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel.

E-mail address: {`eboman,bah`}`@cs.sandia.gov`, {`mycroft,stoledo`}`@tau.ac.il`

URL: `http://www.tau.ac.il/~stoledo`