SIAM J. MATRIX ANAL. APPL.
Vol. 11, No. 3, pp. 430–452, July 1990

007

PARTITIONING SPARSE MATRICES WITH EIGENVECTORS OF GRAPHS*

ALEX POTHEN†, HORST D. SIMON‡, AND KANG-PU LIOU§

Abstract. The problem of computing a small vertex separator in a graph arises in the context of computing a good ordering for the parallel factorization of sparse, symmetric matrices. An algebraic approach for computing vertex separators is considered in this paper. It is shown that lower bounds on separator sizes can be obtained in terms of the eigenvalues of the Laplacian matrix associated with a graph. The Laplacian eigenvectors of grid graphs can be computed from Kronecker products involving the eigenvectors of path graphs, and these eigenvectors can be used to compute good separators in grid graphs. A heuristic algorithm is designed to compute a vertex separator in a general graph by first computing an edge separator in the graph from an eigenvector of the Laplacian matrix, and then using a maximum matching in a subgraph to compute the vertex separator. Results on the quality of the separators computed by the spectral algorithm are presented, and these are compared with separators obtained from other algorithms for computing separators. Finally, the time required to compute the Laplacian eigenvector is reported, and the accuracy with which the eigenvector must be computed to obtain good separators is considered. The spectral algorithm has the advantage that it can be implemented on a medium-size multiprocessor in a straightforward manner.

Key words. graph partitioning, graph spectra, Laplacian matrix, ordering algorithms, parallel orderings, sparse matrix, vertex separator

AMS(MOS) subject classifications. 65F50, 65F05, 65F15, 68R10

1. Introduction.

In the solution of large, sparse, positive definite systems on parallel computers, it is necessary to compute an ordering of the matrix such that it can be factored efficiently in parallel. Several algorithms have been developed recently for computing good parallel orderings: for instance [39], [40]. For large problems, the storage required for the structure of the matrix may exceed the storage capacities of a single processor, and the ordering itself will need to be computed in parallel. One strategy to compute a good parallel ordering is to employ the divide-and-conquer paradigm: Find a set of vertices in the adjacency graph of the matrix, whose removal disconnects the graph into two nearly equal parts. Number the vertices in the separator last, and recursively number the vertices in the two parts by the same strategy. This strategy is employed in several algorithms which order sparse matrices for factorization; e.g., the Sparspak nested dissection algorithm [27].

In computing an ordering by the above approach, at each step, the following *partitioning problem* needs to be solved: Given an adjacency graph G of a sparse matrix, find a vertex separator S such that S has few vertices and S disconnects $G \setminus S$ into two parts A, B with nearly equal numbers of vertices. In the context of the ordering problem, since a separator S becomes a clique in the factor matrix (filled matrix), a small S controls the fill incurred by the ordering. The requirement that the parts A and B be

* Received by the editors August 15, 1989; accepted for publication (in revised form) December 27, 1989.

† Computer Science Department, Pennsylvania State University, Whitmore Lab, University Park, Pennsylvania 16802 (pothen@shire.sys.cs.psu.edu, na.pothen@na-net.stanford.edu.). A part of this work was done while the author was at the University of Wisconsin, Madison. The research of this author was supported by National Science Foundation grant CCR-8701723, National Science Foundation Equipment grant CCR-8705110, and Air Force Office of Scientific Research grant AFOSR-88-0161.

‡ Numerical Aerodynamic Simulation (NAS) Systems Division, Mail-Stop 258-5, NASA Ames Research Center, Moffett Field, California 94035 (simon@orville.nas.nasa.gov.). The work of this author was supported through National Aeronautics and Space Administration contract NAS2-12961.

§ Computer Science Department, Pennsylvania State University, Whitmore Lab, University Park, Pennsylvania 16802.

roughly equal is a simple way of maintaining load balance in parallel computation, since the submatrix represented by each part will be mapped to a subset of half the processors.

In this paper, we consider a spectral algorithm for solving the partitioning problem. We associate with the given sparse, symmetric matrix (and its adjacency graph), a matrix called the Laplacian matrix. We compute a particular eigenvector of the Laplacian matrix and use its components to initially partition the vertices into two sets A', B'. The set of edges joining A' and B' is an edge separator in the graph G. A vertex separator S is computed from the edge separator by a matching technique.

The use of spectral methods to compute edge separators in graphs was first considered by Donath and Hoffman [16], [17], and since then spectral methods for computing various graph parameters have been considered by several others. A discussion of some of this work is included in § 2.

The spectral algorithm for computing vertex separators considered in this paper has three features that distinguish it from previous algorithms that are worthy of comment.

First, previous algorithms for computing separators, such as the level-structure separator algorithm in Sparspak or the Kernighan–Lin algorithm make use of *local* information in the graph, viz. information about the neighbors of a vertex, to compute separators. The spectral method employs *global* information about the graph, since it computes a separator from eigenvector components. Thus the spectral method has the potential of finding separators in the graph that are qualitatively different from the separators obtained by previous approaches.

Second, we can view the spectral method as an approach in which a vertex in the graph makes a *continuous* choice, with a weight between $+1$ and -1, about which part in the initial partition it is going to belong to. All vertices with weights below the median weight form one part, and the rest, the other part. In the Kernighan–Lin method, each vertex makes a discrete choice (zero or one) to belong to one set. The weights in the spectral method can be used to move a few vertices from one part to the other, if a slightly different partition is desired in the course of the separator algorithm.

Third, the dominant computation in the spectral method is an eigenvector computation by a Lanczos or similar algorithm. This distinguishes the new algorithm from standard graph-theoretical algorithms computationally. Most of the computation is based on standard vector operations on floating point numbers. Because of its algebraic nature, the algorithm is parallelizable in a fairly straightforward manner on medium-grain multiprocessors used in scientific computing. Furthermore, since most of the computations are also vector floating point operations, this algorithm is well suited for vector supercomputers used for large scale scientific computing.

This paper is organized as follows. We include background material on the spectral properties of Laplacian matrices and their relevance to graph partitioning in § 2. We also review earlier work on computing edge separators from the eigenvectors of the adjacency matrix in this section. In § 3, we obtain lower bounds on the size of the smallest vertex separators of a graph in terms of the eigenvalues of the Laplacian matrix. Two different techniques for proving lower bounds are illustrated: One uses the Courant–Fischer–Poincaré minimax criterion, and the second employs an inequality from the proof of the Wielandt–Hoffman theorem. We then show that the spectra of rectangular and square grid graphs can be computed explicitly from the spectra of path graphs by employing suitable graph products and Kronecker products in § 4. We proceed to show how good edge and vertex separators in the grid graphs can be computed from the spectral information. In § 5, we describe our heuristic spectral algorithm to compute vertex separators in general graphs. The algorithm initially computes an edge separator, and then uses a maximum matching in a subgraph to compute the vertex separator. Results about the

quality of the separators computed by the algorithm are presented in § 6. In this section, we also compare the spectral separators with separators computed in the first step of the Sparspak nested dissection ordering algorithm and the Kernighan–Lin algorithm, as well as with results obtained recently by Liu [42] and Leiserson and Lewis [38]. The time required to compute the Laplacian eigenvectors with the Lanczos algorithm and the accuracy needed in the eigenvector to obtain good separators are addressed in § 7. The final § contains our conclusions and some directions for future work.

2. Background. Let $G = (V, E)$ be an undirected graph on $|V| = n$ vertices. The $n \times n$ adjacency matrix $A = A(G)$ has element $a_{v,w}$ equal to one if $(v, w) \in E$, and zero otherwise. By convention, $a_{v,v}$ is zero for all $v \in V$. The rows and columns of the matrices associated with a graph are indexed by the vertices of the graph, their order being arbitrary. Let $d(v)$ denote the degree of a matrix, and define D to be the $n \times n$ diagonal matrix with $d_{v,v} = d(v)$. The matrix $Q = Q(G) = D - A$ is the *Laplacian matrix* of G.

Let the edges of the graph G be directed arbitrarily, and let C denote the vertex-edge incidence matrix of the directed graph. The $|V| \times |E|$ matrix C has elements

$$c_{v,e} = \begin{cases} +1 & \text{if } v \text{ is the head of } e, \\ -1 & \text{if } v \text{ is the tail of } e, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that $Q(G) = CC^t$, and that Q is independent of the direction of the edges in C. Biggs [11] contains a good discussion of the techniques from algebraic graph theory that are used here.

The spectral properties of Q have been studied by several authors [4], [23]. Since

$$\underline{x}^t Q \underline{x} = \underline{x}^t CC^t \underline{x} = (C^t \underline{x})^t (C^t \underline{x}) = \sum_{(v,w) \in E} (x_v - x_w)^2,$$

Q is positive semidefinite. Let the eigenvalues of Q be ordered $\lambda_1 = 0 \leqq \lambda_2 \leqq \cdots \leqq \lambda_n$. An eigenvector corresponding to λ_1 is $\underline{e}$, the vector of all ones. The multiplicity of the zero eigenvalue is equal to the number of connected components of the graph. If G is connected, then the second smallest eigenvalue λ_2 is positive. We call an eigenvector $\underline{y}$ corresponding to λ_2 a *second eigenvector*.

Fiedler [23], [24] has studied the properties of the second eigenvalue λ_2 and a corresponding eigenvector $\underline{y}$. He calls λ_2 the *algebraic connectivity*, and relates it to the vertex and edge connectivities of a graph. He has also investigated the partitions of G generated by the components of the eigenvector $\underline{y}$. One of his results of interest in this paper can be rephrased as follows.

THEOREM 2.1. *Let G be a connected graph, and let $\underline{y}$ be an eigenvector corresponding to λ_2. For a real number $r \geqq 0$, define $V_1(r) = \{v \in V : y_v \geqq -r\}$. Then the subgraph induced by $V_1(r)$ is connected. Similarly, for a real number $r \leqq 0$, the subgraph induced by the set $V_2(r) = \{v \in V : y_v \leqq |r|\}$ is also connected.*

In both sets V_1 and V_2, it is necessary to include all vertices with zero components for the theorem to hold. The role played by these latter vertices in the connectedness of the two subgraphs has been investigated at greater length by Powers [54], [55].

A corollary to this result is that if $y_v \neq 0$ for all $v \in V$, then each of the subgraphs induced by $P = \{v \in V : y_v > 0\}$ and $N = \{v \in V : y_v < 0\}$ is a connected subgraph of G.

The eigenvectors of the adjacency matrix corresponding to its algebraically largest eigenvalues have also been used to partition graphs. It is of interest to ask if a similar theorem holds for an eigenvector corresponding to the second largest eigenvalue of the adjacency matrix.

Let $\underline{x}$, $\underline{y}$ denote eigenvectors corresponding to the algebraically largest and second largest eigenvalues, respectively, of the adjacency matrix of G. By the Perron–Frobenius theory, it is known that all components of $\underline{x}$ are positive. Fiedler's theorem states that if α is a nonnegative number, then the subgraph induced by

$$V_1 = \{v \in V : y_v + \alpha x_v \geqq 0\}$$

is connected. Similarly, if α is a nonpositive number, then the subgraph induced by $V_2 = \{v \in V : y_v - |\alpha| x_v \leqq 0\}$ is also connected.

Alon [1] and Mohar [44] have studied the relationship of the second Laplacian eigenvalue to the *isoperimetric number*, $i(G)$. If U is a subset of the vertices of the graph G, and δU denotes the set of edges with one endpoint in U and the other in $V \setminus U$, then

$$i(G) = \min_{|U| \leqq n/2} \frac{|\delta U|}{|U|}.$$

Clearly $i(G)$ is related to the problem of computing good edge separators.

Alon, Galil, and Milman [2], [3] have related the second Laplacian eigenvalue to the expansion properties of graphs. The relationship of the Laplacian spectrum to several other graph properties has been considered by several authors; two recent survey articles are by Mohar [45] and Bien [10].

Spectral methods for computing edge separators have been considered by several researchers: Donath and Hoffman [16], [17], Barnes [7], [8], Barnes and Hoffman [9], Boppana [12]. An algorithm for coloring a graph by employing the eigenvectors of the adjacency matrix has been considered by Aspvall and Gilbert [6] and a spectral algorithm for finding a pseudoperipheral node has been described by Grimes, Pierce, and Simon [33]. A spectral algorithm for envelope reduction is considered in [53].

Algorithms that make use of flows in networks to compute separators have been designed by Bui et al. [13], and Leighton and Rao [37]. The former describes a bisection algorithm with good average-case behavior for degree-regular random graphs, and the latter describes an approximation algorithm for minimum quotient edge separators.

3. Lower bounds. We obtain lower bounds on the sizes of vertex separators in terms of the eigenvalues of the Laplacian matrix $Q(G)$ in this section. The lower bounds hold for *any* vertex separator in the graph; in particular, these bounds apply to a smallest separator in the graph. We assume that the graph G is connected.

Let $G = (V, E)$ denote a graph on $|V| = n$ vertices, and let A be a subset of its vertices. Denote by $\rho(v, A)$ the distance of a vertex v from A, i.e., the fewest number of edges in a shortest path from v to a vertex in A. Let S denote the set of vertices which are at a distance of less than $\rho \geqq 2$ from A, and not belonging to A. Hence

$$S = \{v \in V \setminus A : \rho(v, A) < \rho\}.$$

Define $B = V \setminus (A \cup S)$; if $B \neq \varnothing$, then the distance between A and B, $\rho(A, B) = \rho$. If $\rho > 2$, the set S is a *wide separator* that separates A from B. If $\rho = 2$, we get the commonly used notion of separators. Wide separators were first used in sparse matrix algorithms by Gilbert and Schreiber [28].

Let E_A denote the set of edges with both endpoints in A, and E_{AS} denote the set of edges with one endpoint in A, and the other in S. The sets E_B, E_S, and E_{BS} are defined similarly. In the following, it will be convenient to work with the fractional sizes $a \equiv |A|/n$, $b \equiv |B|/n$, and $s \equiv |S|/n$. The degree of a vertex v will be denoted by $d(v)$, and Δ will denote the maximum degree of vertices in G.

The first result is a lower bound on the size of a wide separator separating any pair of vertex disjoint sets A and B that are at a distance ρ from each other. As will be described later, it generalizes a result of Alon, Galil, and Milman [2].

THEOREM 3.1. *Let A, B be disjoint subsets of vertices of G that are at a distance $\rho \geqq 2$ from each other. Let S denote the set of vertices not belonging to A that are at a distance less than ρ from A. Then*

$$s^2 + \beta s - \rho^2 a(1-a) \geqq 0, \quad \text{where } \beta = (\Delta/\lambda_2) + \rho^2 a - 1.$$

Proof. Let $\underline{e}$, $\underline{0}$ be the vector of all ones and all zeros, respectively. The Courant–Fischer–Poincaré minimax principle states that

$$\lambda_2 = \min_{\substack{\underline{x} \neq \underline{0} \\ \underline{e}^t \underline{x} = 0}} \frac{\underline{x}^t Q \underline{x}}{\underline{x}^t \underline{x}} = \min_{\substack{\underline{x} \neq \underline{0} \\ \underline{e}^t \underline{x} = 0}} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_{i=1}^{n} x_i^2}.$$

Using the Lagrange identity in the above equation, Fiedler [24] derived the following inequality, which is valid for all real n-vectors.

$$n \sum_{(i,j) \in E} (x_i - x_j)^2 \geqq \lambda_2 \sum_{\substack{i,j \in V \\ i<j}} (x_i - x_j)^2. \tag{1}$$

We prove the result by making an appropriate choice of $\underline{x}$ in the above inequality.

Choose the vth component of $\underline{x}$ to be $x_v = 1 - (2/\rho) \min \{\rho, \rho(v, A)\}$. If $v \in A$, then $x_v = 1$; if $v \in B$, then $x_v = -1$; and if $v \in S$, then $-1 + (2/\rho) \leqq x_v \leqq 1 - (2/\rho)$. Also, if v, w are adjacent vertices, then $|x_v - x_w| \leqq 2/\rho$.

The left-hand side of (1) has nonzero contributions from three terms, and it can be bounded from above as follows:

$$\begin{aligned} \sum_{(i,j) \in E} (x_i - x_j)^2 &= \left(\sum_{\substack{(i,j) \in E \\ i \in A, j \in S}} + \sum_{\substack{(i,j) \in E \\ i \in B, j \in S}} + \sum_{\substack{(i,j) \in E \\ i \in S, j \in S}} \right) (x_i - x_j)^2 \\ &\leqq \frac{4}{\rho^2} (|E_{AS}| + |E_{BS}| + |E_S|) \qquad (2) \\ &\leqq \frac{4}{\rho^2} ns\Delta. \end{aligned}$$

Similarly, nonzero contributions to the right-hand side of (1) also come from three terms, and we obtain a lower bound as shown:

$$\begin{aligned} \sum_{\substack{i,j \in V \\ i<j}} (x_i - x_j)^2 &= \left(\sum_{i \in A, j \in S} + \sum_{i \in A, j \in B} + \sum_{i \in B, j \in S} + \sum_{\substack{i \in S, j \in S \\ i<j}} \right) (x_i - x_j)^2 \\ &\geqq \left(\sum_{i \in A, j \in S} + \sum_{i \in A, j \in B} + \sum_{i \in B, j \in S} \right) (x_i - x_j)^2 \\ &\geqq \left(1 - \left(1 - \frac{2}{\rho}\right)\right)^2 n^2 as + (1 - (-1))^2 n^2 ab + \left(-1 - \left(-1 + \frac{2}{\rho}\right)\right)^2 n^2 bs \qquad (3) \\ &= \frac{4n^2}{\rho^2} ((a+b)s + \rho^2 a(1-a-s)) \\ &= \frac{4n^2}{\rho^2} ((1-s)s + \rho^2 a(1-a-s)). \end{aligned}$$

Using inequalities (2) and (3) in Fiedler's inequality (1), and canceling common terms, we obtain

$$s\Delta \geqq \lambda_2((1-s)s+\rho^2 a(1-a-s)).$$

This last inequality yields the desired result after some rearrangement. □

Fiedler [23] showed that $\lambda_2 \leqq (n/(n-1))\min\{d(v): v \in V\}$. Mohar [44] proved that for all graphs except the complete graphs K_n, $\Delta \geqq \lambda_2$. Thus for all graphs except the complete graphs, the ratio $\Delta/\lambda_2 \geqq 1$, and β is a positive number. Indeed, the ratio Δ/λ_2, and hence β, is much larger than one, for all the adjacency graphs of sparse matrices that we have computed partitions.

COROLLARY 3.2. *If* $\beta \geqq \rho$, *then*

$$s \geqq \frac{\rho^2 a(1-a)}{\beta} = \frac{\rho^2 a(1-a)}{(\Delta/\lambda_2)+\rho^2 a-1}.$$

Proof. Let s_1, s_2 be the roots of the quadratic equation corresponding to the inequality in Theorem 3.1, with $s_1 \leqq s_2$. Then $s \geqq s_2$, and

$$s_2 = \frac{1}{2}(-\beta+(\beta^2+4\rho^2 a(1-a))^{1/2}).$$

If $\beta \geqq 2\rho(a(1-a))^{1/2}$, then expanding the right-hand side in power series yields the result.

It remains to verify the condition of the corollary. Since $(a(1-a))^{1/2}$ has its maximum value $\frac{1}{2}$ when $0 \leqq a \leqq 1$, the power series expansion is valid when $\beta \geqq \rho$. □

The corollary exhibits the dependence of vertex separator sizes on λ_2: the smaller the second eigenvalue, the larger the ratio Δ/λ_2, and the smaller the lower bound on the vertex separator size. The corollary also shows the dependence of the lower bound on the distance ρ and the fractional size of the set A.

The common situation of a separator corresponds to $\rho = 2$. In this case, the quadratic inequality becomes $s^2 + \beta s - 4a(1-a) \geqq 0$, with $\beta = (\Delta/\lambda_2) + 4a - 1$. After some simplification, it can be seen that the inequality in Theorem 2.1 of Alon, Galil, and Milman [2] is equivalent to the above inequality. In this case, when $\beta \geqq 2$, we obtain the lower bound

$$s \geqq \frac{4a(1-a)}{(\Delta/\lambda_2)+4a-1}.$$

Mohar [43, Lem. 2.4] has obtained a lower bound on vertex separators in terms of λ_n and λ_2. Lower bounds on edge separators can also be obtained by this technique.

A second lower bound. We now obtain a lower bound that exhibits another factor influencing the size of vertex separators. The technique used is derived from the Wielandt–Hoffman theorem, and has been previously used by Donath and Hoffman [17] to obtain lower bounds on edge separators.

Let S be a vertex separator that separates the graph G into two sets A and B, with $|A| \geqq |B| \geqq |S|$. Let $d(v)$ denote the degree of a vertex v, and let $i(v)$ denote the "internal" degree of v, i.e., the number of edges incident on v with the other endpoint in the same set as v.

Recall that the eigenvalues of Q are ordered as $\lambda_1 = 0 < \lambda_2 \leqq \lambda_3 \cdots \leqq \lambda_n$. Let the $n \times n$ matrix $J = \operatorname{diag}(J_a, J_b, J_c)$, where J_a is the $na \times na$ matrix of all ones, and J_b, J_c are similarly defined. The eigenvalues of J are $\mu_1 = na \geqq \mu_2 = nb \geqq \mu_3 = ns > \mu_4 = \cdots = \mu_n = 0$.

THEOREM 3.3. *Let S be a vertex separator that divides a graph G into two parts, A, B, with $|A| \geq |B| \geq |S|$. Then*

$$s \geq \frac{(1-a)\lambda_2}{2\Delta - (\lambda_3 - \lambda_2)}.$$

Proof. From the proof of the Wielandt–Hoffman theorem [34] (see also [17]),

$$\text{trace}\,(QJ) \geq \sum_{i=1}^{n} \lambda_i \mu_i. \tag{4}$$

We now compute both sides of the above inequality.

The right-hand side is

$$\sum_{i=1}^{n} \lambda_i \mu_i = na \cdot 0 + nb \cdot \lambda_2 + ns \cdot \lambda_3 = n(1-a-s)\lambda_2 + ns\lambda_3.$$

To evaluate the left-hand side, we partition the symmetric matrix Q to conform to J:

$$Q = \begin{pmatrix} Q_{aa} & 0 & Q_{as} \\ 0 & Q_{bb} & Q_{bs} \\ Q_{as}^t & Q_{bs}^t & Q_{ss} \end{pmatrix}.$$

$$\begin{aligned} \text{trace}\,(QJ) &= \text{trace}\,(Q_{aa}J_a) + \text{trace}\,(Q_{bb}J_b) + \text{trace}\,(Q_{ss}J_s) \\ &= \left(\sum_{v \in A} + \sum_{v \in B} + \sum_{v \in S} \right) d(v) - i(v) \\ &= 2(|E| - |E_A| - |E_B| - |E_S|) \qquad (5) \\ &\leq 2(|E| - |E_A| - |E_B|) \\ &\leq 2ns\Delta. \end{aligned}$$

Substituting the inequalities (3) and (5) in (4), we obtain

$$2ns\Delta \geq n(1-a-s)\lambda_2 + ns\lambda_3.$$

This yields the final result after some rearrangement. □

This last lower bound on a vertex separator size shows as before that the magnitude of λ_2 influences the lower bound; it also shows that the "gap" between λ_3 and λ_2 has an effect.

A word of caution is in order about these lower bounds. These bounds should be considered the same way one treats an upper bound on the error in an a priori roundoff error analysis [58]. The lower bounds obtained are not likely to be tight, except for particular classes of graphs. They do illustrate, however, that a large λ_2, with an accompanying small Δ/λ_2, will result in large sizes for the best separators in a graph.

4. Partitions of grid graphs. In this section we show that the second eigenvector of the Laplacian matrix can be used to find good vertex separators in grid graphs, which are model problems in sparse matrix computations. The separators obtained are identical to the separators used by George [26] at the first step in a nested dissection ordering of grid graphs.

To compute separators by this technique, we need to first compute the eigenvectors of grids. The Laplacian spectra of grid graphs can be explicitly computed in terms of the

Laplacian spectra of path graphs. Some of this material is well known in spectral graph theory [15], but such treatments consider only eigenvalues and not eigenvectors. Further, the nine-point grid needs to be modified before its spectrum can be explicitly computed. The techniques used are quite general, and can be used to compute the spectra of several other classes of graphs which can be expressed in terms of graph products of simpler graphs.

The path graph. Let P_n denote the path graph on n vertices. We assume in the following discussion that $n \geqq 2$ is even. We number the vertices of the path from 1 to n in the natural order from left to right.

The Laplacian matrix of P_n is tridiagonal, and hence its spectrum is easily computed. Let $\phi_n \equiv \pi/n$. We denote the elements of a vector $\underline{x}$ by writing its ith component as (x_i).

LEMMA 4.1. *The Laplacian spectrum of P_n is*

$$\lambda_{k,n} = 4 \sin^2\left(\frac{1}{2}(k-1)\phi_n\right),$$

$$\underline{x}_{k,n} = (\cos((i-1/2)(k-1)\phi_n)), \quad \text{for } k = 1, \cdots, n,\ i = 1, \cdots, n. \quad \square$$

As k ranges from 1 to n, the angle $\frac{1}{2}(k-1)\phi_n$ varies from zero to $\pi/2$; hence the eigenvalues are ordered as $\lambda_{1,n} \leqq \lambda_{2,n} \cdots \leqq \lambda_{n,n}$. Note that $\lambda_{1,n} = 0$, $\underline{x}_{1,n} = \underline{1}$, and $\lambda_{2,n} = 4 \sin^2(\phi_n/2)$, and $\underline{x}_{2,n} = (\cos((i - \frac{1}{2})\phi_n))$. The components of $\underline{x}_{2,n}$ plotted against the vertices of P_{30} decrease monotonically from left to right.

Let x_l denote the median ($n/2$th largest) component of the second eigenvector, and partition the vertices of the path into two sets, one set consisting of all vertices with components less than or equal to the median component, and the other consisting of all vertices with components larger than the median component. This partitions the path into subsets of vertices of equal size, one consisting of the vertices with positive eigenvector components, and the other consisting of vertices with negative components.

Graph products. We can compute the spectra of grid graphs from the spectra of the path graph. We require the concepts of graph products and the Kronecker products of matrices. One notation for graph products is from Cvetkovic, Doob, and Sachs [15], and a good discussion of Kronecker products may be found in Fiedler [25].

For $i = 1, 2$, let $G_i = (V_i, E_i)$ be graphs. The *Cartesian sum* $G_1 + G_2$ is the graph $(V_1 \times V_2, E)$, where vertices (i_1, j_1) and (i_2, j_2) are joined by an edge if either $i_1 = i_2$ and $\{j_1, j_2\}$ is an edge in G_2, or $j_1 = j_2$ and $\{i_1, i_2\}$ is an edge in G_1. The *Cartesian product* $G_1 \cdot G_2$ is the graph $(V_1 \times V_2, F)$, where vertices (i_1, j_1) and (i_2, j_2) are joined by an edge if $\{i_1, i_2\}$ is an edge in G_1 and $\{j_1, j_2\}$ is an edge in G_2. The *strong sum* $G_1 \oplus G_2$ is the graph $(V_1 \times V_2, E \cup F)$; thus it contains the edges in both the Cartesian sum and the Cartesian product.

It is easy to verify that the Cartesian sum $P_n + P_m$ is the five-point $m \times n$ grid graph, and that the strong sum $P_n \oplus P_m$ is the nine-point $m \times n$ grid graph.

Since the grid graphs can be obtained from appropriate graph products of the path graph, the Laplacian matrices of the grid graphs can be obtained from Kronecker products involving the Laplacian matrices of the path graph. If C is a $p \times q$ matrix, and D is $r \times s$, recall that the *Kronecker product* $C \otimes D$ is the $pr \times qs$ matrix with each element d_{ij} of D replaced by the submatrix (Cd_{ij}).

The five-point grid. We consider the $m \times n$ five-point grid, and without loss of generality consider $m \leqq n$. Initially we consider the case when n is even, and $m < n$. At the end of this section, we discuss how the results are modified when n is odd, or $m = n$. We draw the $m \times n$ grid with n vertices in each row and m vertices in each column.

Let Q denote the Laplacian matrix of the five-point $m \times n$ grid graph, R_n denote the Laplacian matrix of the path graph on n vertices, and I_n be the identity matrix of order n. Recall that $\lambda_{k,n}$, $\underline{x}_{k,n}$ denotes the kth eigenpair (when eigenvalues are listed in increasing order) of the path graph with n vertices. The following result is well-known; we include a proof for completeness, and because we wish to indicate how a similar result is obtained for the Laplacian spectrum of a modified nine-point grid.

THEOREM 4.2. *The Laplacian spectrum of the $m \times n$ five-point grid is*

$$\mu_{k,l} = \lambda_{k,n} + \lambda_{l,m},$$

$$\underline{y}_{k,l} = \underline{x}_{k,n} \otimes \underline{x}_{l,m}, \qquad k = 1, \cdots, n, \; l = 1, \cdots, m.$$

Proof. It is easy to verify that the Laplacian matrix of the five-point grid can be expressed in terms of the Laplacian matrix of the path graph as $Q = R_n \otimes I_m + I_n \otimes R_m$. The first term in the sum creates m copies of the path on n vertices, and the second term adds the "vertical" edges, which join neighboring vertices in each column of the grid.

We show that $\mu_{k,l}$, $\underline{y}_{k,l}$ is an eigenpair of Q.

$$\begin{aligned} Q\underline{x}_{k,n} \otimes \underline{x}_{l,m} &= (R_n \otimes I_m)(\underline{x}_{k,n} \otimes \underline{x}_{l,m}) + (I_n \otimes R_m)(\underline{x}_{k,n} \otimes \underline{x}_{l,m}) \\ &= (R_n \underline{x}_{k,n}) \otimes (I_m \underline{x}_{l,m}) + (I_n \underline{x}_{k,n}) \otimes (R_m \underline{x}_{l,m}) \\ &= \lambda_{k,n} \underline{x}_{k,n} \otimes \underline{x}_{l,m} + \underline{x}_{k,n} \otimes \lambda_{l,m} \underline{x}_{l,m} \\ &= (\lambda_{k,n} + \lambda_{l,m}) \underline{x}_{k,n} \otimes \underline{x}_{l,m}. \end{aligned}$$

The transformation from the first line to the second line uses the associativity of the Kronecker product. □

The smallest eigenvalue $\mu_{1,1} = \lambda_{1,n} + \lambda_{1,m}$ is zero. The next smallest eigenvalue is $\mu_{2,1} = 4 \sin^2 (\phi_n/2)$, and the corresponding eigenvector is

$$\underline{y}_{2,1} = \underline{x}_{2,n} \otimes \underline{x}_{1,m} = \left(\cos \left(i - \frac{1}{2} \right) \phi_n \right) \otimes \underline{1}.$$

The components of $\underline{y}_{2,1}$ are constant along each column of m vertices, and the components decrease from left to right across a row. Columns numbered 1 to $n/2$ have positive components, and the rest of the columns have negative components. The components of this eigenvector of the $m \times n$ five-point grid are plotted in Fig. 1.

These results show that the second eigenvector of the grid can be used to compute good edge separators and vertex separators. Let $\underline{y}$ denote this eigenvector in the following discussion, and let y_l denote the median component ($(mn/2)$th largest component out of mn). Let y_v denote the eigenvector component corresponding to vertex v.

COROLLARY 4.3. *Let V denote the set of vertices of the five-point $m \times n$ grid ($m < n$, n even), and let V be partitioned by its second eigenvector as follows:*

$$A' = \{ v : y_v \leqq y_l \}, \qquad B' = V \backslash A.$$

If E' denotes the set of edges joining A' to B', then E' is an edge separator of size m which separates the grid into two parts each with $(mn/2)$ vertices. Further, if S denotes the set of endpoints of E' which belong to B', then S is a vertex separator of size m which separates the grid into two parts of $(mn/2)$ and $m((n/2) - 1)$ vertices.

The corollary follows from noting that A' consists of vertices in the columns 1 to $n/2$ of the grid, and B' is the remaining set of columns. The edge separator E' consists of the m edges of the grid which join vertices in column $n/2$ to column $(n/2) + 1$. Finally, the vertex separator S consists of vertices in column $(n/2) + 1$. Note that the vertex separator is the same as the separator at the first step of a nested dissection ordering

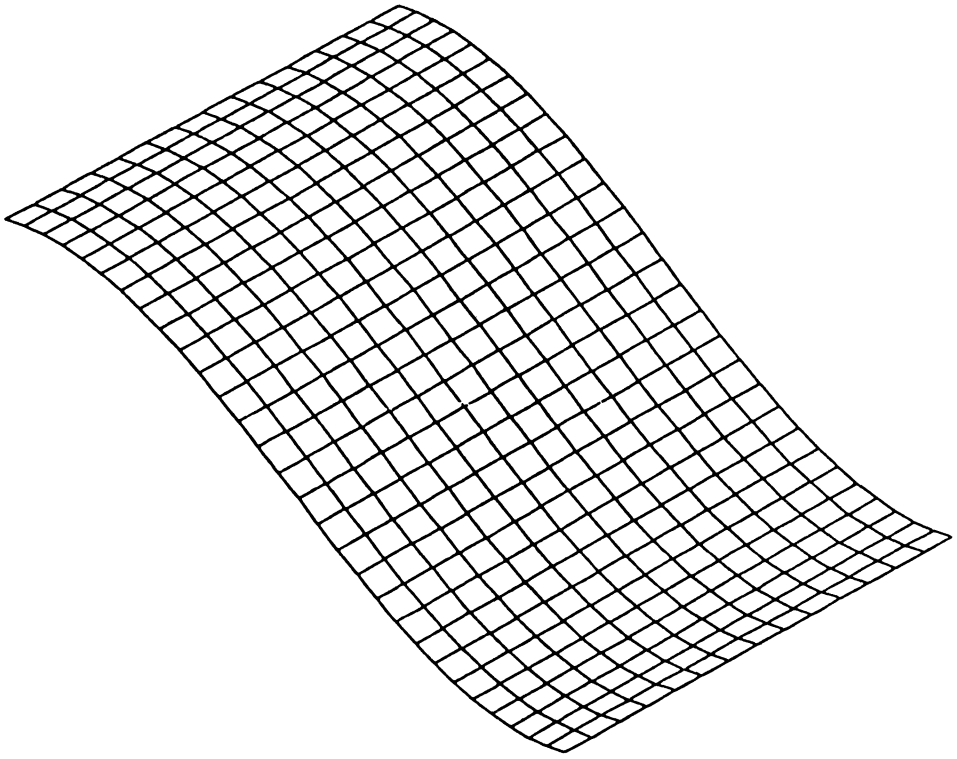

FIG. 1. *The second Laplacian eigenvector of the five-point grid.*

described by George [26]. Buser [14] has shown that the edge separator E' yields the optimal isoperimetric number for grid graphs.

We now consider the case when n is odd or $m = n$. When n is odd, the only difference is that vertices in the middle column ($((n+1)/2$th column) have eigenvector components equal to zero. Columns numbered less than the middle column have positive components, and columns numbered higher have negative components. The middle column can be chosen as a vertex separator. The second case corresponds to a square grid, $m = n$. Then $\mu_{2,1} = \mu_{1,2}$, and the second smallest eigenvalue of Q has geometric multiplicity two. The two linearly independent eigenvectors obtained by the graph product approach are $\underline{y}_{2,1} = \underline{x}_{2,n} \otimes \underline{x}_{1,n}$, and $\underline{y}_{1,2} = \underline{x}_{1,n} \otimes \underline{x}_{2,n}$. The eigenvector $\underline{y}_{2,1}$ has components as described earlier for the rectangular case. The eigenvector $\underline{y}_{1,2}$ has components constant across each row, and decreasing from bottom to top along each column. From these two independent eigenvectors, we obtain a middle column and a middle row as the vertex separators.

Note that when the Lanczos algorithm is used to compute an eigenvector corresponding to the second eigenvalue of the square grid, the eigenvector obtained will be some linear combination of the two eigenvectors $\underline{y}_{1,2}$ and $\underline{y}_{2,1}$. This will lead to a larger vertex separator than the ones above. We report computational results on separators of square grids obtained from the Lanczos algorithm in § 6.

The nine-point grid. Let Q' denote the Laplacian matrix of the nine-point grid, and let D_n be the $n \times n$ diagonal degree matrix of the n-vertex path. As before, let R_n denote the Laplacian matrix of the n-vertex path, and I_n the identity matrix of order n. It is again not difficult to verify that $Q' = R_n \otimes I_m + I_n \otimes R_m + R_n \otimes D_m + D_n \otimes R_m -$

$R_n \otimes R_m$. Unfortunately, the spectrum of Q' cannot be expressed in terms of the spectra of the path graphs, as for the five-point grid.

However, we can first embed the nine-point grid graph in a modified grid, whose Laplacian spectrum is computable in terms of the spectra of the path graphs, and then partition the modified grid. We use the partition of the modified grid to partition the nine-point grid.

The necessary modification to the nine-point grid is as follows. Replace each boundary edge of the $m \times n$ grid by *two edges* joining the same endpoints. Let Q denote the Laplacian of the resulting multigraph.

THEOREM 4.4. *The spectrum of Q is*

$$\mu_{k,l} = 3(\lambda_{k,n} + \lambda_{l,m}) - \lambda_{k,n}\lambda_{l,m},$$

$$\underline{y}_{k,l} = \underline{x}_{k,n} \otimes \underline{x}_{l,m}, \quad \text{for } k = 1, \cdots, n, \ l = 1, \cdots, m.$$

Proof. It is easy to show that $Q = 3(R_n \otimes I_m + I_n \otimes R_m) - R_n \otimes R_m$. A direct computation, as in Theorem 4.2, shows that $\mu_{k,l}, \underline{y}_{k,l}$ is an eigenpair of Q. □

Note that the eigenvectors of the modified nine-point grid are the same as the eigenvectors of the five-point grid, and hence the partitions of the modified nine-point grid are exactly the same as those of the five-point grid.

Finally, we remark that the adjacency spectra of the grids can also be explicitly computed in terms of the adjacency spectra of the path graphs.

5. A spectral partitioning algorithm. In this section we describe an algorithm for finding a vertex separator of a graph by means of its Laplacian matrix. Recall that we require the separator to partition the graph into two parts with nearly equal numbers of vertices in each part, and also that the size of the vertex separator be small.

The algorithm uses a second eigenvector of the Laplacian matrix to compute the partition. We compute x_l, the median value of the components of the eigenvector. Let A' be the set of vertices whose components are less than or equal to x_l, and let B' be the remaining set of vertices. If there is a single vertex with the component corresponding to x_l, then A' and B' differ in size by at most one. If there are several vertices with components equal to x_l, arbitrarily assign such vertices to A' or B' to make these sets differ in size by at most one.

This initial partition of G gives an edge separator in the graph. Let A_1 denote the vertices in A' that are adjacent to some vertex in B', and similarly let B_1 be the set of vertices in B' that are adjacent to some vertex in A'. Let E_1 be the set of edges of G with one endpoint in A_1 and the other in B_1. Then E_1 is an edge separator of G. Note that the subgraph $H = (A_1, B_1, E_1)$ is bipartite.

We require a vertex separator of G, which can be obtained from the edge separator E_1 by several methods. The simplest method is to choose the smaller of the two endpoint sets A_1 and B_1. Gilbert and Zmijewski [29] have computed vertex separators from edge separators in this manner in the context of a parallel Kernighan–Lin algorithm. However, there is a way to choose a *smallest* vertex separator, which can be computed from the given edge separator E'.

The idea is to choose a set S consisting of some vertices from *both* sets of endpoints A_1 and B_1, such that every edge in E_1 is incident on at least one of the vertices in S. The set S is a vertex separator in the graph G, since the removal of these vertices causes the deletion of all edges incident on them, and this latter set of edges contains the edge separator E_1. The set S is a *vertex cover* (*cover*) of the bipartite graph H.

A cover of smallest cardinality is a *minimum cover*. A minimum cover S of the graph H is a smallest vertex separator of G corresponding to the edge separator E_1. It is

well known [36], [47] that a minimum cover of a bipartite graph H can be computed by finding a maximum matching, since these are dual concepts.

In general, S will consist of vertices from both A_1 and B_1. Let A_s and B_s denote the vertices of S that belong to A_1 and B_1, respectively. Then S separates G into two subgraphs with vertex sets $A = A' \setminus A_s$, $B = B' \setminus B_s$. Usually the structure of H permits some freedom in the choice of the sets A_s and B_s; only the sum $|A_s| + |B_s|$ is invariant. This freedom can be used to make the two sets A and B less unequal in size. The sets A_s and B_s may be computed from a canonical decomposition of bipartite graphs called the Dulmage–Mendelsohn decomposition, which is induced by a maximum matching. An implementation of this decomposition is described in [52].

The *Spectral Partitioning Algorithm* is summarized in Fig. 2.

Complexity of the algorithm. In finite precision arithmetic, how accurately must the components of a Laplacian eigenvector be computed to ensure that the vertices are correctly partitioned with respect to the median component? Since the eigenvector components are algebraic numbers, it follows from a discussion in Aspvall and Gilbert [6] that only a polynomial number of bits are needed to order the components of a second eigenvector correctly. In theory, this can be computed in polynomial time by any algorithm that is at least linearly convergent.

In practice, we will have to be content with eigenvector components that are accurate to a fixed number of digits. Since the Lanczos algorithm is an iterative algorithm, the number of Lanczos steps required to approximately compute a second eigenvector will depend on the accuracy desired in the eigenvector. In exact arithmetic, the distribution of the eigenvalues of the Laplacian matrix Q is the primary factor which influences the number of steps required to approximate a second eigenvector (§ 12.4, Parlett [48, § 12.4]. We will assume that the number of iterations of the Lanczos algorithm required to compute a second eigenvector to a small number of digits (say, four) is bounded by a constant. Our experiments in § 7 indicate that this is a reasonable assumption. Each iteration of the Lanczos algorithm costs $O(e)$ flops, and by our assumption, a second eigenvector can also be approximated to a few digits in $O(e)$ flops.

The median component of the eigenvector can be obtained by an algorithm that selects the kth element out of n. This can be done in $O(n)$ time in the worst case by a well-known algorithm of Blum, Floyd, Pratt, Rivest, and Tarjan. This algorithm finds the desired element by repeatedly partitioning a subarray with respect to a pivot element, without sorting the array.

1. Compute the eigenvector x_2 and the median value x_l of its components;
2. Partition the vertices of G into two sets:
   $A' = \{\text{vertices with } x_v \leqq x_l\}$;
   $B' = V \setminus A$;
   If $|A'| - |B'| > 1$, move enough vertices with components equal to x_l from A' to B' to make this difference at most one;
3. Let A_1 be the set of vertices in A' adjacent to some vertex in B';
   Let B_1 be the set of vertices in B' adjacent to some vertex in A';
   Compute $H = (A_1, B_1, E_1)$, the bipartite subgraph induced by the vertex sets A_1, B_1;
4. Find a minimum vertex cover S of H by a maximum matching;
   Let $S = A_s \cup B_s$, where $A_s \subseteq A_1, B_s \subseteq B_1$;
   S is the desired vertex separator, and separates G into subgraphs with vertex sets $A = A' \setminus A_s$, $B = B' \setminus B_s$.

FIG. 2. *The Spectral Partitioning Algorithm.*

The partition into the sets A and B can be done in $O(n)$ time. The bipartite graph H can be generated in $O(e)$ time, by examining the adjacency list of each vertex at most once. Let m be the smaller of $|A'|$ and $|B'|$, and let $e' \equiv |E'|$. A maximum matching and a minimum cover S can be obtained in $O(\sqrt{me'}) = O(\sqrt{ne})$ time by an algorithm of Hopcroft and Karp. Thus the worst-case time complexity of the *Spectral Partitioning Algorithm* is $O(\sqrt{ne})$.

Some comment is necessary about the above analysis. In practice, the matching is obtained quite fast. Several matching algorithms have been efficiently implemented in [18], [20], [52], and these algorithms exhibit $O(n + e)$ time complexity in practice. Also, we used a less sophisticated median-finding algorithm, which is $O(n)$ in the average-case, and $O(n^2)$ in the worst-case. In practice, the dominant step in the *Spectral Partitioning Algorithm* is the computation of a second eigenvector by the Lanczos algorithm.

6. Results. In this section, we report computational results obtained from the *Spectral Partitioning Algorithm* and provide comparisons with several other separator algorithms: a modified level-structure separator algorithm implemented in Sparspak, the *Kernighan–Lin algorithm*, the *Fiduccia–Mattheyses algorithm* as implemented by Leiserson and Lewis [38], and the separator algorithm of Liu [42] based on the Multiple Minimum Degree algorithm. We implemented the spectral algorithm, the modified Sparspak separator algorithm, and the Kernighan–Lin algorithm; results for the last two algorithms were obtained from Lewis (personal communication) and Liu's paper [42]. Several sparse matrices from the Boeing–Harwell collection [19] and five- and nine-point grids are partitioned using these algorithms.

Our primary goal in this paper is to establish that the spectral algorithm computes separators that compare favorably with separators computed by previous algorithms. Thus in this section, we report statistics about the quality of the separators computed by the various algorithms. In the next § 7, we report the time required to compute the second Laplacian eigenvector (the dominant computation in the spectral algorithm) for a few representative problems.

In current work, we are implementing a parallel Lanczos algorithm for computing the second eigenvector *in parallel*. This algorithm will be used to compute the separators in parallel. The parallel separator algorithm will then be used to recursively find separators and thereby to compute, in parallel, orderings appropriate for parallel factorizations.

Arioli and Duff [5] have reported results on generating bordered block triangular forms of unsymmetric matrices by finding separators in a directed graph associated with the matrix. Their goal was to use the bordered block triangular form for the parallel solution of large, sparse systems of equations.

The spectral algorithm. We computed vertex and edge separators using the *Spectral Partitioning Algorithm* from the second Laplacian eigenvector. The Lanczos algorithm was terminated either when the approximate eigenvector satisfied the eigenvalue equation to a residual of 10^{-6} or when 300 Lanczos steps were performed. The partitions obtained with the *Spectral Partitioning Algorithm* are tabulated in Table 1. In this table, we list the edge separator first and the vertex separator next, since the former is computed first, and the latter is computed from the former. The edge separator E_1 separates the graph into two parts A' and B'. The sizes of these sets are shown in the first group of three columns in the table. We show two vertex separators obtained from E_1: the first vertex separator is chosen to be the smaller endpoint set of E_1; in the table, this set is denoted A_1. The second vertex separator S includes subsets of vertices from both endpoint sets, and is computed by means of a maximum matching to be a minimum vertex cover of the bipartite graph induced by E_1.

TABLE 1
Partitions using median component of the second Laplacian eigenvector.

				Vertex separators					
	Edge separator			Endpoint set			Matching		
Key	$\|E_1\|$	$\|A'\|$	$\|B'\|$	$\|A_1\|$	$\|A'\| - \|A_1\|$	$\|B'\|$	$\|S\|$	$\|A\|$	$\|B\|$
BCSPWR09	34	862	861	22	840	861	20	857	846
BCSPWR10	44	2,650	2,650	35	2,615	2,650	31	2,623	2,646
BCSSTK13	3,585	1,002	1,001	295	707	1,001	236	862	905
CAN 1072	165	536	536	53	483	536	33	525	514
DWT 2680	85	1,340	1,340	29	1,311	1,340	28	1,313	1,339
JAGMESH	50	468	468	26	442	468	26	442	468
LSHP3466	121	1,733	1,733	61	1,672	1,733	61	1,672	1,733
NASA1824	740	912	912	103	809	912	102	839	883
NASA2146	934	1,073	1,073	96	977	1,073	74	1,036	1,036
NASA4704	1,324	2,352	2,352	185	2,167	2,352	172	2,266	2,266
GRD61.101.5	61	3,111	3,050	61	3,050	3,050	61	3,050	3,050
GRD61.101.9	181	3,111	3,050	61	3,050	3,050	61	3,050	3,050
GRD80.80.5	80	3,200	3,200	80	3,120	3,200	80	3,120	3,200
GRD80.80.9	238	3,200	3,200	80	3,120	3,200	80	3,120	3,200

For six of the Boeing–Harwell problems, the matching method computes vertex separators that are almost the same size as the smaller endpoint set. However, on the CAN 1072 problem, the separator from the matching method is almost 40 percent smaller. On the average problem in this test set, matching finds a separator that is about 11 percent smaller than the separator obtained from the endpoint set. Further, since there are two choices for the minimum cover, a good choice also makes the two part sizes less different. Thus the use of matching techniques seems to be recommended in this context.

The edge separators obtained are small relative to the total number of edges in each graph, except for the BCSSTK13 problem, which has a high average degree. For all problems, except two, the vertex separators obtained are also relatively small (fractional separator size $s < 0.04$) in comparison to the parts generated by the separators. The exceptions are BCSSTK13 and NASA1824. Both these problems have large second eigenvalue λ_2. For BCSSTK13, $\lambda_2 \approx 0.65$; in contrast, for the 80×80 nine-point grid, which has good separators, $\lambda_2 \approx 4.6 \times 10^{-3}$.

For the grid graphs, good vertex separators can be computed by explicitly computing the second eigenvector by the methods in § 4. Here, we investigate the partitions obtained by the spectral algorithm with the eigenvector computed by the Lanczos algorithm. We partitioned the 61×101 grids initially into two sets with 3050 (50 columns) and 3111 (51 columns) vertices. The edge separator obtained joins vertices in the fiftieth column to vertices in the fifty-first column. The vertex separator computed is the middle (fifty-first) column.

In the square grids, the second eigenvalue has geometric multiplicity two, and there are two linearly independent eigenvectors. The eigenvectors in § 4, $\underline{y}_{2,1}$ and $\underline{y}_{1,2}$, obtained by the Kronecker products of the Laplacian eigenvectors of the path, can be used to compute two sets of edge separators. One edge separator joins vertices in the fortieth column to vertices in the forty-first column, and the other joins vertices in the fortieth row to the forty-first row. In general, the Lanczos algorithm will compute a linear combination of the two eigenvectors described above, leading to a different (and large) edge separator. However, for the starting vector we used, the Lanczos algorithm converged to

the eigenvector $y_{1,2}$, and the latter edge separator was computed. (The choice of the start vector is described in § 7.)

We now compare the quality of the separators computed by the spectral algorithm with separators computed from several other algorithms.

The modified level-structure separator algorithm. The separator routine in Sparspak, FNDSEP, finds a pseudoperipheral vertex in the graph, and generates a level structure from it. It then chooses the median level in the level structure as the vertex separator. However, this choice may separate the graph into widely disparate parts. We modified this routine such that the vertex separator is chosen to be the smallest level k such that the first k levels together contain more than half the vertices. A vertex separator is obtained by removing from the vertices in level k those vertices that are not adjacent to any vertex in level $k+1$. By the construction of the level structure, the removed vertices are adjacent to vertices in level $k-1$, and hence these are added to the part containing vertices in the first $k-1$ levels. The other part has vertices in levels $k+1$ and higher. We can also obtain two edge separators using the level structure from the set of edges joining the vertex separator to the two parts A and B.

Statistics about the edge and vertex separators computed by this technique are shown in Table 2. In this table, the vertex separator is listed first and then the edge separator since the former is computed first and the latter is obtained from the former.

The *Spectral Partitioning Algorithm* computes smaller vertex separators than the Sparspak separator algorithm; on the average problem in the Boeing–Harwell test set, the spectral vertex separator is about half the size of the Sparspak vertex separator. The spectral algorithm also succeeds in keeping the part sizes less disparate than the latter algorithm. The average difference in the part sizes is about 7 percent for the Sparspak separator, but there are problems for which this difference is greater than 20 percent.

For most problems, the spectral algorithm also finds smaller edge separators in the graph than the Sparspak level-structure separator algorithm. There are a few problems where the best edge separator obtained by the latter algorithm is smaller than that obtained by the spectral algorithm, but the former edge separators separate the graph into parts with widely differing sizes. In the spectral algorithm, equal part sizes can be obtained by

TABLE 2
Partitions from automated nested dissection.

	Vertex separator			Edge separators					
Key	$\|S\|$	$\|A\|$	$\|B\|$	$\|E_1\|$	$\|A\|$	$\|B \cup S\|$	$\|E_2\|$	$\|A \cup S\|$	$\|B\|$
BCSPWR09	68	762	893	80	762	961	130	830	893
BCSPWR10	169	2,421	2,710	209	2,421	2,879	317	2,590	2,710
BCSSTK13	302	764	937	3,035	764	1,239	4,792	1,066	937
CAN 1072	64	478	530	108	478	594	342	542	530
DWT 2680	28	1,327	1,325	84	1,327	1,353	84	1,355	1,325
JAGMESH	26	455	455	50	455	481	50	481	455
LSHP3466	59	1,711	1,696	118	1,711	1,755	116	1,770	1,696
NASA1824	137	839	848	910	839	985	1,347	976	848
NASA2146	131	1,008	1,007	1,473	1,008	1,138	1,569	1,139	1,007
NASA4704	296	2,245	2,163	2,134	2,245	2,459	2,424	2,541	2,163
GRD61.101.5	61	3,050	3,050	121	3,050	3,111	121	3,111	3,050
GRD61.101.9	111	3,025	3,025	327	3,025	3,131	333	3,131	3,025
GRD80.80.5	80	3,160	3,160	158	3,160	3,240	158	3,240	3,160
GRD80.80.9	113	3,136	3,151	333	3,136	3,264	339	3,249	3,151

partitioning with respect to the median eigenvector component; any other choice of part sizes can also be obtained by partitioning with respect to the appropriate component. Since edge separators are computed in the Sparspak algorithm by means of a level structure, part sizes cannot be controlled as effectively.

The Kernighan–Lin algorithm. The *Kernighan–Lin algorithm* is a heuristic algorithm for computing small edge separators. We investigated the use of this algorithm separately and in conjunction with the *Spectral Partitioning Algorithm*, to compute edge and vertex separators.

The Kernighan–Lin algorithm begins with an initial partition of the graph into two subsets A', B', which differ in their sizes by at most one. At each iteration, the algorithm chooses two subsets of equal size to swap between A and B, thereby reducing the number of edges that join A to B. We refer the reader to Kernighan and Lin [35], or Gilbert and Zmijewski [29] for a detailed description of how the algorithm chooses the subsets to be swapped. The algorithm terminates when it is no longer possible to decrease the size of the edge separator by swapping subsets. In our implementation, each iteration could require $O(n^3)$ time, though in practice, often the running time is $O(n^2 \log n)$, the time required for n sorts.

One initial partition we could use is the edge partition obtained from the *Spectral Partitioning Algorithm*, and a second choice is to use a randomly computed initial partition. We consider the four graphs with the largest edge separators from Table 1, and report the sizes of the edge and vertex separators obtained with the Kernighan–Lin algorithm in Table 3. An edge separator was computed first, and then a vertex separator was obtained as before by matching methods. The column labeled "SP" corresponds to the output of the spectral algorithm, "SP, KL" corresponds to the Kernighan–Lin algorithm with initial partition from the spectral algorithm, and "KL" corresponds to the Kernighan–Lin algorithm with a random initial partition.

The application of the Kernighan–Lin algorithm with the spectral partition as input succeeds in reducing the sizes of the edge separator considerably for two of the four problems. Thus if one is primarily concerned with small edge separators, applying the Kernighan–Lin algorithm to the partition produced by spectral algorithm could be

TABLE 3

Partitions from the Kernighan–Lin algorithm. The first table describes the edge separators, and the second, vertex separators.

			$\lvert E_1 \rvert$		
Key	$\lvert A' \rvert$	$\lvert B' \rvert$	SP	SP, KL	KL
BCSSTK13	1,002	1,001	3,585	2,880	3,550
NASA1824	912	912	740	739	739
NASA2146	1,073	1,073	934	870	870
NASA4704	2,352	2,352	1,324	1,313	1,525

	SP			SP, KL			KL		
Key	$\lvert S \rvert$	$\lvert A \rvert$	$\lvert B \rvert$	$\lvert S \rvert$	$\lvert A \rvert$	$\lvert B \rvert$	$\lvert S \rvert$	$\lvert A \rvert$	$\lvert B \rvert$
BCSSTK13	236	862	905	250	870	883	284	772	947
NASA1824	103	839	883	102	830	892	102	830	892
NASA2146	74	1,036	1,036	74	1,036	1,036	74	1,036	1,036
NASA4704	172	2,266	2,266	172	2,266	2,266	204	2,163	2,337

worthwhile. However, the size of the vertex separator is not improved. For two of the problems, the size remains the same; for a third, it decreases by one, and the size increases for a fourth problem. Also, for two of the four problems, the spectral algorithm by itself finds better vertex separators than those obtained by the Kernighan–Lin algorithm alone.

Gilbert and Zmijewski [29] have observed that the quality of the partition found by the Kernighan–Lin algorithm strongly depends on the quality of the initial partition. They show for a grid graph that it is possible to choose a bad initial partition for the Kernighan–Lin algorithm such that the algorithm will not find a minimum edge separator.

Edge separators obtained from the Kernighan–Lin algorithm with initial spectral partition are better than those obtained from the application of the Kernighan–Lin algorithm with random initial partitions for two of the four problems. Use of the initial partition from spectral algorithm also helps the Kernighan–Lin algorithm to converge faster. On these four problems, the Kernighan–Lin algorithm ran on the average about 3.2 times faster when the spectral partition was used. Thus the spectral algorithm could be used to generate initial partitions of high quality for the Kernighan–Lin algorithm.

The Leiserson–Lewis and Liu algorithms. In [38] Leiserson and Lewis have used the *Fiduccia–Mattheyses algorithm* [22] to compute vertex separators and then to order sparse matrices. Liu [42] uses the Multiple Minimum Degree ordering algorithm to compute vertex separators, and then improves the separator (by decreasing its size and making the parts less unequal) by a matching technique. He uses his separator algorithm in [41] to compute a good ordering for parallel factorization. In both implementations sparse matrices from the Boeing–Harwell collection are used, so we are able to give a direct comparison of the first level vertex separator. The data in Table 4 are obtained directly from Liu's report [42] and from Lewis (personal communication). In both cases we have added small disconnected components, which were created by the vertex separators, to the smaller of the two sets $|A|$ or $|B|$.

The results in Table 4 show that the Leiserson–Lewis implementation and Liu's algorithm find separators which are smaller than the spectral separators for the two power network problems. The reason for this seems to be that for these problems, the spectral algorithm computes a partition from an eigenvector that has converged to fewer than two correct digits. The accuracy of the computed eigenvectors is discussed in greater detail in § 7. For the other four problems, the Leiserson–Lewis and the spectral separators are almost the same size. Liu's algorithm finds a larger separator than the spectral algorithm for the BCSSTK13 problem. The Leiserson–Lewis algorithm does a good job of keeping the part size roughly equal. There is greater difference between the part sizes in Liu's algorithm. However, neither the Leiserson–Lewis algorithm nor Liu's algorithm offers any easy prospect for a parallel implementation. A factor which cannot be evaluated in

TABLE 4
Vertex separators from the Leiserson–Lewis and the Liu algorithms.

	Leiserson–Lewis			Liu		
Key	$\|S\|$	$\|A\|$	$\|B\|$	$\|S\|$	$\|A\|$	$\|B\|$
BCSPWR09	7	858	854	8	1,026	689
BCSPWR10	18	2,641	2,634	19	2,661	2,620
BCSSTK13	242	892	869	298	941	764
CAN1072	34	522	516	38	665	368
DWT2680	28	1,339	1,313	26	1,369	1,283
LSHP3466	57	1,708	1,701	61	1,727	1,678

this comparison is the relative execution time of the algorithms, since these algorithms were implemented on different computers.

7. Convergence. The dominant computation in the *Spectral Partitioning Algorithm* is the computation of the second eigenvector of the Laplacian matrix by the Lanczos algorithm. Since the Lanczos algorithm is an iterative algorithm, the number of iterations and the time required to compute this eigenvector is dependent on the number of correct digits needed in the eigenvector components. In this section, we describe the details of an implementation of the Lanczos algorithm for computing this eigenvector, and study how the quality of computed separators depends on the accuracy in the second eigenvector.

The Lanczos algorithm. The most efficient algorithm for computing a few eigenvalues and eigenvectors of large, sparse symmetric matrices is the Lanczos algorithm. Since the Lanczos algorithm is discussed extensively in the textbook literature [30], [48], we do not include a detailed description of the standard algorithm here. The convergence of the Lanczos algorithm depends critically on the distribution of the eigenvalues of the underlying matrix. Usually the extreme eigenpairs, i.e., the largest and smallest, are found first. However it is also known that for operators such as the discrete Laplacian for a grid problem, or more generally for positive definite finite element matrices which are approximations to elliptic operators, the Lanczos algorithm converges in most cases to the extreme right, i.e., the very large eigenvalues, before delivering good approximations to the eigenvalues close to zero. This behavior can be explained with the so-called Kaniel–Paige–Saad theory (see [48]). When computing the smallest positive eigenvalue of the Laplacian matrix Q, one faces exactly the same situation: the Lanczos algorithm delivers very good approximations to the large eigenvalues before converging to the desired second smallest eigenvalue. Thus the Lanczos algorithm potentially requires long runs before it computes an approximation to the second eigenpair.

A potential modification which can be incorporated in the Lanczos algorithm for faster computation of the second eigenvector would be to apply the shifted and inverted operator, i.e., to consider the eigenvalue problem

$$(Q-\sigma I)^{-1}\underline{u}=\mu\underline{u}.$$

This is a standard technique in finite element applications [31], and it has been used very successfully in a variety of implementations of the Lanczos algorithm [21], [32], [49], [56]. In the situation here, a shift σ chosen near zero would result in rapid convergence to the eigenvalue λ_2. This approach cannot be taken here, since it requires the factorization of the matrix $Q-\sigma I$, which is a large sparse symmetric matrix with the same sparsity structure as M. Our original goal, however, is to find an efficient reordering of M, so to be able to factor it efficiently. Hence the "shift and invert" approach would require us to factor a matrix closely related to M, and thus cannot be considered in this application.

Reorthogonalization has also been used in the Lanczos algorithm to improve both its reliability and computational efficiency [49], [50], [57]. However, in this application we do not require reorthogonalization techniques in their full generality. Only a limited amount of reorthogonalization is necessary for the computation of the second eigenpair. No reorthogonalizations are performed at the right end of the spectrum, with respect to the large eigenvalues, since there is no interest in the accurate computation of eigenvalues at this end. Also it is unlikely that preserving orthogonality at the right end will have any impact on the convergence of the Lanczos algorithm towards the second smallest eigenvalue, which is at the left end of the spectrum. The first eigenvector $\underline{x}_1$ of Q is $\underline{e}$, the vector of all ones, and this vector can be used for reorthogonalization at the left end of

the spectrum. At each step we explicitly orthogonalize the current Lanczos vector against $\underline{e}$. This is effectively a deflation of the problem and now the eigenpair λ_2, $\underline{x}_2$ will be computed as the first eigenpair at the left end of the spectrum.

Another important consideration for the Lanczos algorithm is the choice of a starting vector. In the absence of any other information, a random starting vector is appropriate. However, many practical matrix problems are presented already in an ordering relevant to the formulation of the problem, sometimes even in an ordering which is close to a good band or envelope ordering. In this case it is desirable to transmit this ordering information to the Lanczos algorithm. This was accomplished by setting the starting vector in the Lanczos algorithm to $\underline{r}$, with $\underline{r}_i = i - (n + 1)/2$. This choice also makes the starting vector orthogonal to $\underline{e}$. In most cases this resulted in faster convergence to the second eigenvector.

Finally, another point needs to be mentioned. Considering the simple structure of the Laplacian matrix Q, and the seeming simplicity of the task of computing just one eigenpair at the left end of the spectrum, one might be inclined to avoid the complexities of the Lanczos algorithm and attempt to solve this problem with a simple shifted power method with a deflation procedure analogous to the one described above. This was tried as a first attempt at the computation of a second eigenvector, but with very poor results. The power method converged exceedingly slowly, in many cases exhibiting the phenomenon of *misconvergence* [51]. This meant that the power method settled down at an eigenvalue of Q, which was not the Fiedler value, and whose eigenvector correspondingly delivered a very poor reordering. The results here support the claims of [51] that even in the simplest cases the Lanczos algorithm is the method of choice, when computing eigenvalues of large, sparse, symmetric matrices.

Figure 3 contains a description of the specialized Lanczos algorithm for computing the second Laplacian eigenvector. In this algorithm, we have assumed that the Laplacian $Q(G)$ is irreducible, or equivalently that the graph G is connected. Many of the sparse matrices from the Boeing–Harwell collection have disconnected adjacency graphs. If a graph has k connected components, the first k eigenvectors correspond to the multiple eigenvalue zero, and the k + 1th eigenvector is used to partition the graph. A simple modification to the above algorithm can be used to compute this eigenvector.

Convergence and quality of separators. We now present our results on the number of iterations and the time required by the Lanczos algorithm as the second eigenvector is computed to a set of different tolerances. The tolerance criterion, *tol*, is the 2-norm of the residual vector $Q\underline{u} - \lambda\underline{u}$, where λ, $\underline{u}$ are the computed quantities at the current step in the algorithm. We also study the quality of the vertex separators obtained from these approximate eigenvectors.

We report results for a few representative problems from the Boeing–Harwell collection and for two grid problems in Table 5. The iteration numbers reported are multiples of twelve, since we checked for convergence in the Lanczos algorithm by an eigendecomposition of the tridiagonal matrix only after every twelve iterations. Times are in seconds

1. Given the sparsity structure of a matrix M, form the Laplacian matrix Q.
2. Pick a starting vector $\underline{r}$, with $r_i = i - (n + 1)/2$.
3. Carry out a Lanczos iteration with the matrix Q and starting vector $\underline{r}$. At each step orthogonalize the Lanczos vector against the vector $\underline{e}$. Stop when a second eigenvector has been determined to sufficient accuracy.

FIG. 3. *The Lanczos algorithm for computing the second Laplacian eigenvector.*

TABLE 5
Convergence results. Times are in seconds on a Cray Y-MP. *A blank entry in the separator column indicates that the separator is unchanged from the row above it.*

Key	*tol*	Iterns	Time	$\|S\|$	$\|A\|$	$\|B\|$
NASA4704	10^{-1}	24	0.27	172	2,266	2,266
	10^{-2}	60	0.65			
	10^{-3}	72	0.80			
	10^{-4}	96	1.10			
	10^{-5}	108	1.30			
	10^{-6}	120	1.50			
BCSSTK13	10^{-1}	36	0.23	236	905	862
	10^{-2}	36	0.23			
	10^{-3}	48	0.30			
	10^{-4}	60	0.39			
	10^{-5}	72	0.49			
	10^{-6}	84	0.60			
BCSPWR10	10^{-1}	24	0.24	171	2,619	2,510
	10^{-2}	84	0.92	72	2,642	2,586
	10^{-3}	252	7.20	34	2,643	2,623
	10^{-4}	300	11.90	31	2,646	2,623
GRD61.101.5	10^{-2}	12	0.15	101	3,050	3,010
	10^{-3}	36	0.42	61	3,050	3,050
	10^{-4}	96	1.26			
	10^{-5}	108	1.47			
	10^{-6}	120	1.67			
GRD61.101.9	10^{-1}	12	0.16	101	3,030	3,030
	10^{-2}	24	0.30	61	3,050	3,050
	10^{-3}	108	1.53			
	10^{-4}	120	1.76			
	10^{-5}	144	2.38			
	10^{-6}	156	2.70			

on a Cray Y-MP, using our vectorized Lanczos code. For each value of *tol*, we report the size of the vertex separator and the corresponding part sizes computed by the *Spectral Partitioning Algorithm*. Blank entries in the separator columns mean that the separator computed is the same as the one obtained with the previous tolerance.

For most of the problems that we have computational results, it is only necessary to compute the second eigenvector to a tolerance of about 10^{-2}, to obtain the best separator obtained by the spectral algorithm. This accuracy requires only a modest number of Lanczos iterations, and can be obtained reasonably fast. One class of notable exceptions is the power network problems, illustrated by BCSPWR10 in the table. For these problems, the average degree of a vertex is small (about 1.5 for BCSPWR10), and the diameter of the graph is large; hence computing eigenvector components (which represent global information about the graph) is relatively slow. A large number of iterations are thus necessary to compute the second eigenvector accurately. In the BCSPWR10 problem, after 300 iterations, the norm of the residual in the eigenvalue equation was about 10^{-4}. In this problem, the vertex separator decreases in size as the eigenvector becomes more accurate.

8. Conclusions. We have considered an algebraic approach for computing vertex separators and have shown that the eigenvalues of the Laplacian matrix can be used to obtain lower bounds on the sizes of the separators. We have described a heuristic algorithm for computing vertex separators from the second eigenvector of the Laplacian. Thus the

spectral algorithm uses global information about the graph to compute separators. It is enough to compute the eigenvector to low accuracy to obtain good separators for most problems. Our results show that the spectral separators compare quite favorably with separators computed by previous algorithms. The spectral algorithm has an advantage over these algorithms in that its dominant computation is an eigenvector computation (which involves mainly dense and sparse vector operations), and is fairly straightforward to compute efficiently on medium-size multiprocessors used in scientific computing. For previous algorithms, it is either not clear how to implement them in parallel or the amount of parallelism is not high. Since the spectral algorithm involves mainly floating point computations, we expect it to be attractive over primarily combinatorial algorithms on machines like the Cray Y-MP, where floating point arithmetic is considerably faster than integer arithmetic.

The computation of good separators is useful in many divide-and-conquer algorithms. Several of the new parallel algorithms that have been reported to date make use of divide and conquer, and hence the spectral separator algorithm will have applications in parallel algorithm design. The spectral algorithm may also be useful in VLSI layout problems, since good edge separators are needed in this context.

But our immediate intent was to use the spectral separator algorithm to compute good orderings for parallel sparse factorizations. More work remains to be done in order to accomplish this goal. First, we intend to compute and study the quality of orderings obtained by the recursive application of the spectral algorithm. Second, we are working on the fast sequential and parallel computation of the second Laplacian eigenvector. The latter algorithm will enable us to compute the separators (and thereby orderings for parallel factorizations) in parallel. We are investigating the Lanczos algorithm and the generalized Davidson's algorithm of Morgan and Scott [46] in this regard.

Finally, much remains to be understood about the theoretical underpinnings of the spectral separator algorithm. It will be useful to obtain results on the quality of the partitions computed by the Laplacian eigenvector components. It will also be helpful to identify classes of graphs that are partitioned well by the spectral algorithm.

REFERENCES

[1] N. ALON, *Eigenvalues and expanders*, Combinatorica, 6 (1986), pp. 83–96.

[2] N. ALON, Z. GALIL, AND V. D. MILMAN, *Better expanders and superconcentrators*, J. Algorithms, 8 (1987), pp. 337–347.

[3] N. ALON AND V. D. MILMAN, λ_1, *isoperimetric inequalities for graphs, and superconcentrators*, J. Comb. Theory, Series B, 38 (1985), pp. 73–88.

[4] W. N. ANDERSON AND T. D. MORLEY, *Eigenvalues of the Laplacian of a graph*, Linear and Multilinear Algebra, 18 (1985), pp. 141–145. (Originally published as University of Maryland Tech. Report TR-71-45, 1971).

[5] M. ARIOLI AND I. S. DUFF, *Experiments in tearing large sparse systems*, Tech. Report CSS 217, Computer Science and Systems Division, Harwell Lab, Harwell, UK, January 1988.

[6] B. ASPVALL AND J. R. GILBERT, *Graph coloring using eigenvalue decomposition*, SIAM J. Algebraic Discrete Methods, 5 (1984), pp. 526–538.

[7] E. R. BARNES, *An algorithm for partitioning the nodes of a graph*, SIAM J. Algebraic Discrete Methods, 3 (1982), pp. 541–550.

[8] ———, *Partitioning the nodes of a graph*, in Graph Theory with Applications to Algorithms and Computer Science, Y. Alavi, G. Chartrand, L. Lesniak, D. R. Lick, and C. E. Wall, eds., John Wiley, 1985, pp. 57–72.

[9] E. R. BARNES AND A. J. HOFFMAN, *Partitioning, spectra, and linear programming*, in Progress in Combinatorial Optimization, W. E. Pulleyblank, ed., Academic Press, New York, 1984, pp. 13–25.

[10] F. BIEN, *Constructions of telephone networks by group representations*, Notices Amer. Math. Soc., 36 (1989), pp. 5–22.

[11] N. L. BIGGS, *Algebraic Graph Theory*, Cambridge University Press, Cambridge, 1974.
[12] R. B. BOPPANA, *Eigenvalues and graph bisection: an average case analysis*, in 28th Annual Symposium on Foundations of Computer Science, 1987, pp. 280–285.
[13] T. BUI, S. CHAUDHURI, T. LEIGHTON, AND M. SIPSER, *Graph bisection algorithms with good average-case behavior*, in 25th Annual Symposium on Foundations of Computer Science, 1984, pp. 181–192.
[14] P. BUSER, *On the bipartition of graphs*, Discrete Appl. Math., 9 (1984), pp. 105–109.
[15] D. M. CVETKOVIC, M. DOOB, AND H. SACHS, *Spectra of Graphs*, Academic Press, New York, 1980.
[16] W. E. DONATH AND A. J. HOFFMAN, *Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices*, IBM Technical Disclosure Bulletin, 15 (1972), pp. 938–944.
[17] ———, *Lower bounds for the partitioning of graphs*, IBM J. Res. Develop., 17 (1973), pp. 420–425.
[18] I. S. DUFF, *On algorithms for obtaining a maximum transversal*, ACM Trans. Math. Software, 7 (1981), pp. 315–330.
[19] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.
[20] I. S. DUFF AND T. WIBERG, *Implementations of $O(n^{1/2}\tau)$ assignment algorithms*, ACM Trans. Math. Software, 14 (1988), pp. 267–287.
[21] T. ERICSSON AND A. RUHE, *The spectral transformation Lanczos method*, Math. Comp., 34 (1980), pp. 1251–1268.
[22] C. FIDUCCIA AND R. MATTHEYSES, *A linear time heuristic for improving network partitions*, in ACM–IEEE 19th Design Automation Conference, Las Vegas, NV, IEEE Press, 1982, pp. 175–181.
[23] M. FIEDLER, *Algebraic connectivity of graphs*, Czechoslovak Math. J., 23 (1973), pp. 298–305.
[24] ———, *A property of eigenvectors of non-negative symmetric matrices and its application to graph theory*, Czechoslovak Math. J., 25 (1975), pp. 619–633.
[25] ———, *Special Matrices and their Applications in Numerical Mathematics*, Martinus Nijhoff Publishers, Dodrecht, The Netherlands, 1986.
[26] J. A. GEORGE, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 15 (1978), pp. 1053–1069.
[27] J. A. GEORGE AND J. W-H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.
[28] J. R. GILBERT, *Separator theorems and sparse Gaussian elimination*, Ph.D. thesis, Stanford University, Stanford, CA, 1981.
[29] J. R. GILBERT AND E. ZMIJEWSKI, *A parallel graph partitioning algorithm for a message passing multiprocessor*, Internat. J. Parallel Programming, 16 (1987), pp. 427–449.
[30] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, Second edition, 1989.
[31] R. G. GRIMES, J. G. LEWIS, AND H. D. SIMON, *Eigenvalue problems and algorithms in structural engineering*, in Large Scale Eigenvalue Problems, J. Cullum and R. Willoughby, eds., North-Holland, Amsterdam, 1986, pp. 81–93.
[32] ———, *The implementation of the block Lanczos algorithm with reorthogonalization methods*, Tech. Report ETA-TR-91, Boeing Computer Services, Seattle, WA, 1988.
[33] R. G. GRIMES, D. J. PIERCE, AND H. D. SIMON, *A new algorithm for finding a pseudoperipheral node in a graph*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 323–335.
[34] A. J. HOFFMAN AND H. W. WIELANDT, *The variation of the spectrum of a normal matrix*, Duke Math. J., 20 (1953), pp. 37–39.
[35] B. W. KERNIGHAN AND S. LIN, *An efficient heuristic procedure for partitioning graphs*, The Bell System Tech. J., 49 (1970), pp. 291–307.
[36] E. L. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, 1976.
[37] T. LEIGHTON AND S. RAO, *An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms*, in 29th Annual Symposium on Foundations of Computer Science, 1988, pp. 422–431.
[38] C. E. LEISERSON AND J. G. LEWIS, *Orderings for parallel sparse symmetric factorization*, Third SIAM Conference on Parallel Processing for Scientific Computing, 1987.
[39] J. G. LEWIS, B. W. PEYTON, AND A. POTHEN, *A fast algorithm for reordering sparse matrices for parallel factorization*, SIAM J. Sci. Statist. Comput., 6 (1989), pp. 1146–1173.
[40] J. W-H. LIU, *Reordering sparse matrices for parallel elimination*, Tech. Report 87-01, Computer Science, York University, North York, Ontario, Canada, 1987; Parallel Computing, to appear.
[41] ———, *The minimum degree ordering with constraints*, Tech. Report CS-88-02, Computer Science, York University, North York, Ontario, Canada, 1988.
[42] ———, *A graph partitioning algorithm by node separators*, ACM Trans. Math. Software, 1989, to appear.

[43] B. MOHAR, *Eigenvalues, diameter, and mean distance in graphs*, Preprint Series Dept. Math. No. 259, University E. K. of Ljubljana, Jadranska 19, 61111 Ljubljana, Yugoslavia, April 1988.
[44] ———, *Isoperimetric numbers of graphs*, J. Combin. Theory, Ser. B, 1988.
[45] ———, *The Laplacian spectrum of graphs*, in Sixth International Conference on Theory and Applications of Graphs, Kalamazoo, MI, 1988.
[46] R. B. MORGAN AND D. S. SCOTT, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 817–825.
[47] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.
[48] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, 1980.
[49] B. N. PARLETT, B. NOUR-OMID, AND Z. A. LIU, *How to maintain semi-orthogonality among Lanczos vectors*, Tech. Report PAM-420, Center for Pure and Applied Math., University of California, Berkeley, CA, 1988.
[50] B. N. PARLETT AND D. S. SCOTT, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 217–238.
[51] B. N. PARLETT, H. D. SIMON, AND L. STRINGER, *Estimating the largest eigenvalue with the Lanczos algorithm*, Math. Comp., 38 (1982), pp. 153–165.
[52] A. POTHEN AND C-J. FAN, *Computing the block triangular form of a sparse matrix*. Tech. Report CS-88-51, Computer Science, Penn State, December 1988; ACM Trans. Math. Software, 1990, to appear.
[53] A. POTHEN AND H. D. SIMON, *A parallel iterative algorithm for envelope reduction in sparse matrices*, in preparation.
[54] D. L. POWERS, *Structure of a matrix according to its second eigenvector*, in Current Trends in Matrix Theory, F. Uhlig and R. Grone, eds., Elsevier, New York, 1987, pp. 261–266.
[55] ———, *Graph partitioning by eigenvectors*, Linear Algebra Appl., 101 (1988), pp. 121–133.
[56] D. S. SCOTT, *Block Lanczos software for symmetric eigenvalue problems*, Tech. Report ORNL/CSD 48, Oak Ridge National Lab, Oak Ridge, TN, 1979.
[57] H. D. SIMON, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–136.
[58] J. H. WILKINSON, *Modern error analysis*, SIAM Rev., 13 (1971), pp. 548–568.