JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematical Sciences
Vol. 80B, No. 2, April–June 1976

Inverting Sparse Matrices by Tree Partitioning*

D. R. Shier

Institute for Basic Standards, National Bureau of Standards, Washington, D.C. 20234

(August 16, 1974)

This paper studies the tree partitioning of a graph whose definition is based on the pattern of zero elements present in a given matrix. This partitioning then indicates a particularly advantageous strategy for employing block Gaussian elimination over a certain class of matrices. The strategy is exploited for matrix inversion, where it is especially appropriate for problems which require finding only selected submatrices of the inverse. A graph-theoretic algorithm is given for automatically generating tree partitionings for any matrix. Combinatorial properties of this procedure are also discussed.

Key words: Block Gaussian elimination; graph; inversion; partitioning; sparse matrix; tree.

1. Introduction

The solution of "sparse" systems of linear equations has received considerable attention in recent years [13][1], [15], [19], and justifiably so, since coefficient matrices having relatively few nonzero entries arise quite frequently from physical problems [1], [4], [8]. The object of this paper is to discuss a partitioning method for inverting such sparse matrices which can rather readily adapt to the given zero-nonzero structure of a particular sparse matrix. This work is motivated by the observation [12] that the solution of a linear system is particularly simple when the graph underlying the coefficient matrix is exactly a tree. The present approach shows how a natural tree-like representation can be obtained for *any* matrix and how this leads to an especially simple method for *inverting* the given matrix. In addition, a straightforward extension of the method allows one to solve, rather effectively, systems of linear equations.

Henceforth, our attention will be focused on the class K of square matrices all of whose principal submatrices are nonsingular. Inasmuch as the class K contains all positive definite symmetric matrices (which do arise quite often in physically meaningful contexts), the method to be described will be applicable to a number of important situations. It is not necessary that the coefficient matrix A be symmetric; in fact, nonsymmetric matrices for which $A + A^T$ is positive definite also belong to K, as well as the nonsymmetric M-matrices [6] (having nonpositive off-diagonal elements and all principal minors positive).

2. Tree Partitionings

Let the real $n \times n$ matrix A be in K. Eventually, a simultaneous permutation of the rows and columns of A will be produced so that the resulting matrix has an especially simple form with identifiable (and exploitable) blocks of zero entries. To do this, we first associate a finite undirected *graph* G_A with the matrix $A = (a_{ij})$. The *nodes* of G_A correspond to the rows/columns of A, and an *edge* (i, j) will join node i to node j whenever $|a_{ij}| + |a_{ji}| > 0$.

In any graph G, the sequence of edges $(e_1, e_2, \ldots, e_k)$ forms a *chain* whenever $k + 1$ nodes i_0, $i_1, \ldots, i_k$ can be identified so that e_j joins nodes i_{j-1} and i_j for $j = 1, \ldots, k$; a chain is *elementary*

AMS Subject Classification: 05C05, 15A09.

*This work was done while the author was a National Research Council-National Academy of Sciences Postdoctoral Research Associate at the National Bureau of Standards, Washington, D.C. 20234.

[1] Figures in brackets indicate the literature references at the end of this paper.

if it does not meet the same node twice. A chain which consists of at least three distinct edges and which joins a node to itself is termed a *cycle*. If every pair of distinct nodes in a graph is joined by a chain, then the graph is said to be *connected*. A *tree* is a connected graph which contains no cycles.

It is now supposed that the nodes N of the graph G_A are partitioned into $m \geq 2$ distinct sets of nodes $N_1, N_2, \ldots, N_m$ which exhibit a tree structure when viewed as an undirected graph T_A. Namely, the nodes of T_A are the sets $N_1, N_2, \ldots, N_m$ and an edge joins N_i with N_j $(i = j)$ in T_A whenever an edge exists in G_A between some node in N_i and another node in N_j. The graph T_A so constructed is assumed then to be a tree. It will be shown in section 4 how to construct for connected graphs such a *tree partitioning*. (If the graph G_A is not connected then clearly A is decomposable as a block diagonal matrix; in this case the inverse of A is readily computable from the inverses of the diagonal blocks, each of which corresponds to a connected graph.) As an example, the "snowflake" graph [14, p. 209] depicted in figure 1 has the tree partitioning given in figure 2.

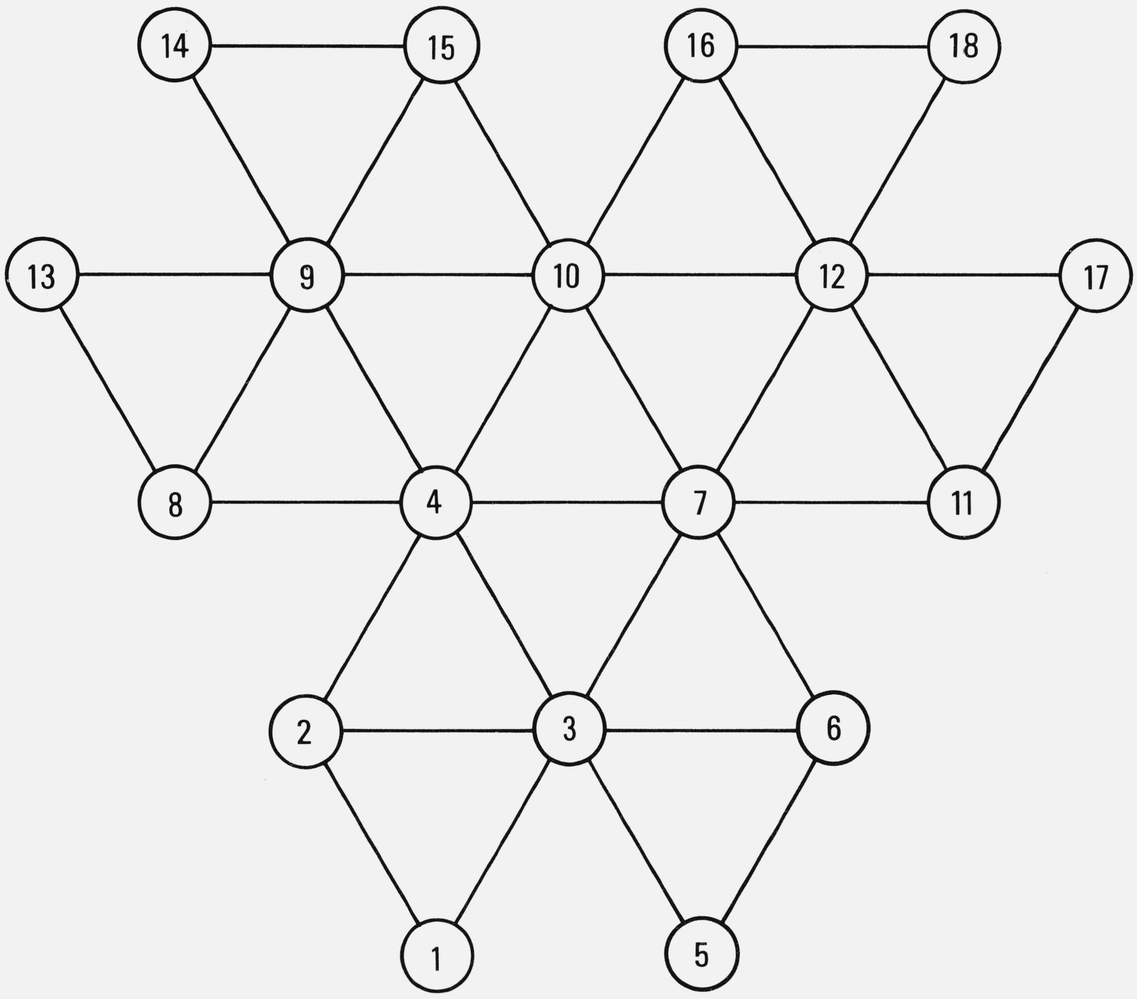

FIGURE 1. *An illustrative graph.*

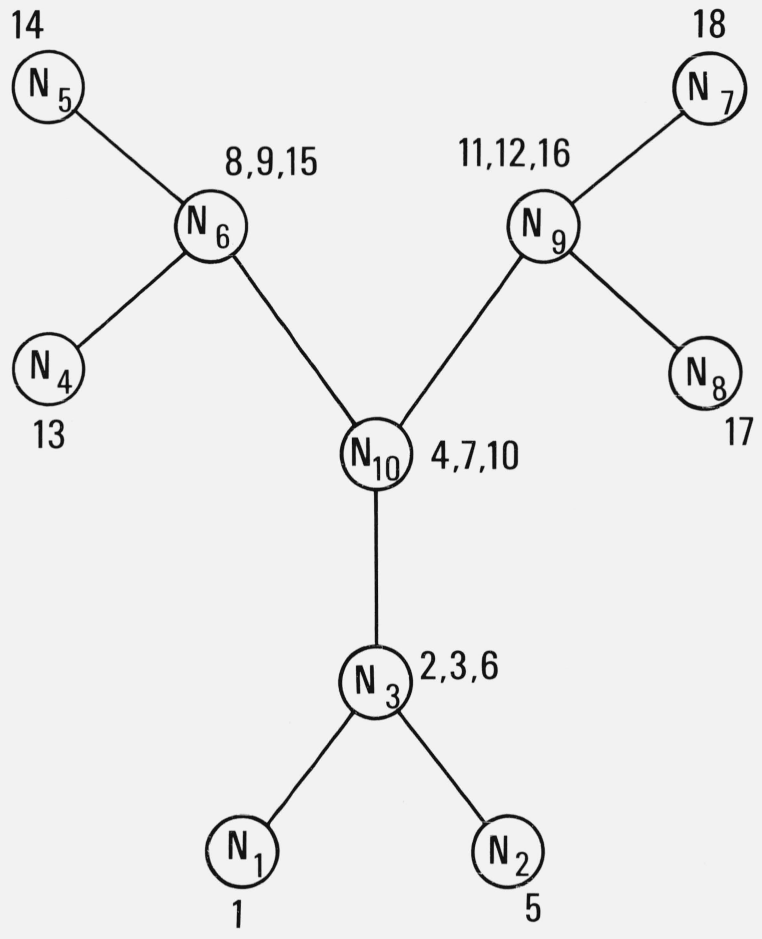

FIGURE 2. *A tree partitioning for the graph of figure 1 (the nodes comprising each node set are also shown).*

The concept of a tree partitioning has been previously applied [16] to shortest path calculations in large sparse networks. This concept can also be applied to the present situation of matrix inversion. The basic idea is that of decomposing the original problem into smaller subproblems, the solutions to which can be easily recombined to yield the information required from the original problem.

The present decomposition approach relies on a fundamental property of trees. This is, in each tree there always exists a *pendant* node—a node which is incident with precisely one edge. When such a pendant node i and its incident edge are deleted from a tree T, then a new tree $T - i$ is formed. This process of deleting a pendant node together with its incident edge can then be repeated anew using the tree $T - i$. Therefore, given the tree T_A, the node sets can be suitably relabelled so that N_1 is a pendant node of T_A, N_2 is a pendant node of $T_A - N_1, \ldots$, and N_{m-1} is a pendant node of $T_A - N_1 - \ldots - N_{m-2}$. Such a labelling of T_A is shown in figure 2. Moreover, for each index $i \neq m$ there is associated a unique index $r(i)$ such that $N_{r(i)}$ is the next node set after N_i on the unique elementary chain joining N_i and N_m in the tree T_A. Because of the assumed labelling induced by the pendant node sets of T_A, it is always the case that $r(i) > i$. Table 1 displays these indices for the tree of figure 2.

TABLE 1. *The indices r(i) for the tree of figure 2.*

i	1	2	3	4	5	6	7	8	9
$r(i)$	3	3	10	6	6	10	9	9	10

3. The Matrix Decomposition Algorithm

In this section, a general decomposition algorithm for inverting matrices $A \in \mathrm{K}$ will be presented and justified. This algorithm is based on a tree partitioning $N = (N_1, N_2, \ldots, N_m)$ of the associated graph G_A. Let the matrix A be partitioned as $A = (A_{ij})$, where A_{ij} is the submatrix composed of entries a_{uv} where $u \in N_i$ and $v \in N_j$. Let A^{-1} be similarly partitioned as $A^{-1} = (X_{ij})$. It is assumed that the node sets have been suitably relabelled in accordance with the scheme described in section 2. Hence, the ith row of A, $i \neq m$, is given by

$$A_i = (A_{i1}, \ldots, A_{ii}, 0, \ldots, 0, A_{i,r(i)}, 0, \ldots, 0), \tag{3.1}$$

where the submatrices $A_{ik} = 0$ $(k < i)$ unless $r(k) = i$. Also, the mth row of A is given by

$$A_m = (A_{m1}, \ldots, A_{mm}), \tag{3.2}$$

where the submatrices $A_{mk} = 0$ $(k < m)$ unless $r(k) = m$. The proposed decomposition algorithm for obtaining A^{-1} from A is embodied in the following procedure.

PROCEDURE INVERT.

1. Set $B_{ii} = A_{ii}$, $i = 1, \ldots, m$.
2. For $i = 1, \ldots, m-1$ calculate

$$B_{i,r(i)} = -B_{ii}^{-1} A_{i,r(i)} \tag{3.3}$$

$$B_{r(i),i} = -A_{r(i),i} B_{ii}^{-1} \tag{3.4}$$

and perform the replacement $B_{r(i),r(i)} := B_{r(i),r(i)} + A_{r(i),i} B_{i,r(i)}$.

3. Compute $X_{mm} = B_{mm}^{-1}$ and for $i = m-1, \ldots, 1$

$$X_{i,r(i)} = B_{i,r(i)} X_{r(i),r(i)}$$

$$X_{r(i),i} = X_{r(i),r(i)} B_{r(i),i}$$

$$X_{ii} = B_{ii}^{-1} + B_{i,r(i)} X_{r(i),i}.$$

4. The remaining X_{ij} blocks can be calculated using

$$X_{ij} = B_{i,r(i)} X_{r(i),j} \qquad i < j$$

$$X_{ij} = X_{i,r(j)} B_{r(j),j} \qquad i > j.$$

It is noted that this procedure requires the inversion of submatrices B_{ii} as well as the multiplication and addition of appropriate submatrices B_{jk}. Of course, there is no reason why the submatrices B_{ii} cannot be inverted (if still sufficiently sparse) by further applying the decomposition algorithm to the subnetworks based on individual node sets N_i. Such an approach can be recursively applied until the sparsity of the B_{ii}'s has been exploited to its fullest.

The validity of the procedure will now be established. First, note that at the end of Step 2 of INVERT,

$$B_{ii} = A_{ii} + \sum_{r(k)=i} A_{ik} B_{ki}. \tag{3.5}$$

Since $A^{-1} = X = (X_{ij})$ is the solution to

$$A X = I, \tag{3.6}$$

then from (3.1) with $i \neq j$, $i < m$

$$\sum_{r(k)=i} A_{ik} X_{kj} + A_{ii} X_{ij} + A_{i,r(i)} X_{r(i),j} = 0. \tag{3.7}$$

It is now claimed that

$$X_{ij} = B_{i,r(i)} X_{r(i),j} \text{ if } i < j. \tag{3.8}$$

This assertion will be established inductively. Inasmuch as $i = r(k) > k$, equation (3.5) shows that $B_{11} = A_{11}$. Thus, using (3.7) with $j > 1$

$$A_{11}X_{1j} + A_{1,r(1)}X_{r(1),j} = 0,$$

whence from (3.3)

$$X_{1j} = -A_{11}^{-1}A_{1,r(1)}X_{r(1),j} = B_{1,r(1)}X_{r(1),j}.$$

Therefore, the assertion (3.8) holds for $i = 1$. Suppose the assertion is true for indices $k < i$, where $i < m$. Then from (3.7) with $i < j$

$$\sum_{r(k)=i} A_{ik}B_{ki}X_{ij} + A_{ii}X_{ij} + A_{i,r(i)}X_{r(i),j} = 0,$$

since $k < r(k) = i < j$ and so the inductive hypothesis can be applied. By (3.5) the above relation becomes

$$B_{ii}X_{ij} + A_{i,r(i)}X_{r(i),j} = 0,$$

whence

$$X_{ij} = -B_{ii}^{-1}A_{i,r(i)}X_{r(i),j} = B_{i,r(i)}X_{r(i),j},$$

using (3.3). Accordingly, the assertion (3.8) is established by induction.

In particular, choosing $j = r(i)$ in (3.8) yields

$$X_{i,r(i)} = B_{i,r(i)}X_{r(i),r(i)} \qquad \text{if } i < m. \tag{3.9}$$

Furthermore, from (3.2) and (3.6)

$$\sum_{r(k)=m} A_{mk}X_{km} + A_{mm}X_{mm} = I$$

or, applying (3.9),

$$\sum_{r(k)=m} A_{mk}B_{km}X_{mm} + A_{mm}X_{mm} = I,$$

so that from (3.5), $B_{mm}X_{mm} = I$, whereupon

$$X_{mm} = B_{mm}^{-1}. \tag{3.10}$$

Finally, from (3.1) and (3.6) with $i \neq m$

$$\sum_{r(k)=i} A_{ik}X_{ki} + A_{ii}X_{ii} + A_{i,r(i)}X_{r(i),i} = I$$

or, using (3.8) and (3.5),

$$B_{ii}X_{ii} + A_{i,r(i)}X_{r(i),i} = I.$$

By virtue of (3.3),

$$\begin{aligned} X_{ii} &= B_{ii}^{-1} - B_{ii}^{-1}A_{i,r(i)}X_{r(i),i} \\ &= B_{ii}^{-1} + B_{i,r(i)}X_{r(i),i}. \end{aligned} \tag{3.11}$$

The counterparts to equations (3.8) and (3.9) can be established by using the relation $X\ A = I$ instead of (3.6), thus yielding

$$X_{ij} = X_{i,r(j)}B_{r(j),j} \qquad \text{if } i > j \tag{3.12}$$
$$X_{r(i),i} = X_{r(i),r(i)}B_{r(i),i} \qquad \text{if } i < m. \tag{3.13}$$

Together, eqs (3.8) — (3.13) serve to establish the validity of procedure INVERT, provided that the submatrices B_{ii} encountered for inversion are nonsingular. This property of the B_{ii}'s will next be shown to follow from the fact that $A \in$ K. First, given any tree partitioning of G_A, the matrix $A = (A_{ij}) \in$ K evidently satisfies the following condition:

Block Principal Submatrix (BPS) Condition. Given the matrix A partitioned into m^2 blocks by corresponding sets $N_1, N_2, \ldots, N_m$ of row and column indices, then any submatrix determined by deleting corresponding sets N_j (possibly none at all) is nonsingular.

It will now be shown that, under the transformation described by Step 2 of INVERT, this property is inherited by certain submatrices of the transformed matrix. Let $A = (A_{ij})$ be written as

$$A = M'_0 = \left[\begin{array}{c|ccccc} B_{11} & 0 & \cdots & A_{1,r(1)} & \cdots & 0 \\ \hline 0 & & & & & \\ \vdots & & & & & \\ A_{r(1),1} & & & M_1 & & \\ \vdots & & & & & \\ 0 & & & & & \end{array}\right].$$

After the transformation with $i = 1$ in Step 2 of INVERT, the matrix M_1 will be transformed into a new matrix M_1' which differs from M_1 only in that the diagonal block $B_{r(1),r(1)}$ has been replaced by

$$\begin{aligned} B'_{r(1),r(1)} &= B_{r(1),r(1)} + A_{r(1),1}B_{1,r(1)} \\ &= B_{r(1),r(1)} - A_{r(1),1}B_{11}^{-1}A_{1,r(1)}. \end{aligned} \tag{3.14}$$

Now it is easily verified that $M_1' = (M_0'/B_{11})$, the *Schur complement* [2] of B_{11} in M_0', defined by:

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Rightarrow (M/A) = D - CA^{-1}B.$$

Given that $A = M_0'$ is *BPS*, it is claimed that M_1' is also *BPS*. In fact, consider any principal submatrix S' determined by the blocks of M_1'. If the diagonal block $B'_{r(1),r(1)}$ does not occur in S', then S' is also a principal block submatrix of M_0', and hence is nonsingular. If S' does contain $B'_{r(1),r(1)}$, then consider the corresponding submatrix S of M_0' obtained by replacing $B'_{r(1),r(1)}$ by its pre-transformation value $B_{r(1),r(1)}$. Let T be the submatrix of M_0' defined by

$$T = \left[\begin{array}{c|ccccc} B_{11} & 0 & \cdots & A_{1,r(1)} & \cdots & 0 \\ \hline 0 & & & & & \\ \vdots & & & & & \\ A_{r(1),1} & & & S & & \\ \vdots & & & & & \\ 0 & & & & & \end{array}\right].$$

Since T and B_{11} are principal block submatrices of M_0' (which is *BPS*), then T and B_{11} are nonsingular. Moreover, by the determinantal formula of Schur [7]

$$\det T = \det B_{11} \cdot \det (T/B_{11}),$$

it follows that (T/B_{11}) is nonsingular as well. However, it is easily seen from (3.14) that $(T/B_{11}) = S'$ and so S' is nonsingular. Thus, M_1' is BPS.

More generally, the above argument can be repeated to show that if M_{k-1}' is BPS then the submatrix M_k defined by

$$M_{k-1}' = \left[\begin{array}{c|ccccc} B_{kk} & 0 & \cdots & A_{k,r(k)} & \cdots & 0 \\ \hline 0 & & & & & \\ \vdots & & & & & \\ A_{r(k),k} & & & M_k & & \\ \vdots & & & & & \\ 0 & & & & & \end{array}\right]$$

yields a transformed M_k' (after executing Step 2 of INVERT with $i = k$) which is also *BPS*. Given, then, that each M_k' is BPS we can conclude that in particular the diagonal block $B_{k+1,k+1}$ is nonsingular. This is precisely what is needed to ensure that at each iteration of Step 2, the matrix B_{ii}^{-1} is defined. Accordingly, the procedure INVERT is guaranteed to be well-defined and will then produce the matrix A^{-1} as required. It should be noted that if A satisfies the BPS condition for the given partition, then the INVERT procedure will remain valid, even if A is not a member of K.

In effect, procedure INVERT is a statement of block Gaussian elimination, which becomes especially simple and manageable in the context of a tree partitioning. It should be noted that in

Steps 2 and 3 the blocks of the inverse matrix (corresponding to nonzero blocks of the original matrix A) can be calculated without introducing any new "fill-ins"; this is just to say that the given zero blocks are preserved. Moreover, we see that the existence of a tree partitioning allows one to solve linear equations in very much the same way as inverting matrices. Rather than needing to perform submatrix inversions, we need instead to solve a linear subsystem of equations at each step.

4. Finding Tree Partitionings

In this section, an algorithm will be described for obtaining tree partitions of an undirected graph $G = (N,E)$. Without loss of generality, it may be assumed that G is a connected graph having at least two nodes. Indeed, any graph which is not connected can be resolved into at least two maximally connected subgraphs[2] or *connected components*. Clearly, then, the tree partitioning problem can be studied and solved separately with regard to each of these *connected* components.

The basic idea of the algorithm can be most easily explained by means of an example. Consider, therefore, the connected graph of figure 1 for which $N = \{1, \ldots, 18\}$. The process is begun by selecting a subset N_1 of nodes from $C_1 = N$ with $\phi \subset N_1 \subset C_1$;[3] say, $N_1 = \{4, 7, 10\}$. It is then easy to compute

$$\Gamma(N_1) = \{j \in C_1 - N_1\colon \quad (i, j) \in E \text{ for some } i \in N_1\},$$

the set of all nodes in C_1 adjacent with (but not including) nodes in N_1. For this example, $\Gamma(N_1) = \{2, 3, 6, 8, 9, 11, 12, 15, 16\}$. It is also easy to find the connected components of the subgraph based on the nodes $C_1 - N_1$ and thus the node sets C_k constituting each of these connected components. Here, $C_2 = \{8, 9, 13, 14, 15\}$, $C_3 = \{11, 12, 16, 17, 18\}$ and $C_4 = \{1, 2, 3, 5, 6\}$. The node sets N_2, N_3, and N_4 are then chosen as $N_2 = C_2 \cap \Gamma(N_1) = \{8, 9, 15\}$, $N_3 = C_3 \cap \Gamma(N_1) = \{11, 12, 16\}$, and $N_4 = C_4 \cap \Gamma(N_1) = \{2, 3, 6\}$.

The next iteration of the process computes

$$\begin{aligned}\Gamma(N_2) &= \{j \in C_2 - N_2\colon \quad (i, j) \in E \text{ for some } i \in N_2\} \\ &= \{13, 14\}\end{aligned}$$

and determines the sets $C_5 = \{13\}$ and $C_6 = \{14\}$ associated with the two connected components for $C_2 - N_2$. The node sets N_5 and N_6 are given by $N_5 = C_5 \cap \Gamma(N_2) = \{13\}$, and $N_6 = C_6 \cap \Gamma(N_2) = \{14\}$. In similar fashion, subsequent iterations produce $N_7 = \{17\}$, $N_8 = \{18\}$, $N_9 = \{1\}$, $N_{10} = \{5\}$. The tree partitioning which reflects this disposition of node sets (apart from relabelling) is shown in figure 2.

A general statement of the tree partitioning procedure for connected graphs $G = (N,E)$ is provided by

```
PROCEDURE TREEPART.
    1. Let $N_1$ satisfy $\phi \subset N_1 \subset N$. Set $C_1 = N$, $p = 1$, $q = 1$.
    2. Compute $\Gamma(N_p) = \{j \in C_p - N_p\colon (i, j) \in E$ for some $i \in N_p\}$. If $\Gamma(N_p) = \phi$ then go to Step
    3. Otherwise,
       a. Find the node sets $C_{q+1}, \ldots, C_{q+r}$ constituting the connected components of the
          subgraph determined by $C_p - N_p$.
       b. Form $N_{q+i} = C_{q+i} \cap \Gamma(N_p)$ for $i = 1, \ldots, r$.
       c. Join node set $N_p$ by an edge to each of the node sets $N_{q+1}, \ldots, N_{q+r}$.
       d. Let $q := q + r$.
    3. Let $p := p + 1$. If $p \leq q$ then go to Step 2. Otherwise, terminate.
```

In the above algorithm, the actual edges of the tree formed from the node sets N_k are generated at

[2] A *subgraph* of a given undirected graph $G = (N,E)$ has for its nodes a subset $N_0 \subseteq N$ and contains those edges of E which join nodes in N_0.
[3] The notation $A \subset B$ means that A is a proper subset of B.

Step 2c. Because N_1 is a nonempty proper subset of N, the resulting tree partitioning has always at least two and at most $n = |N|$ constituent node sets. Generally speaking, the less the density of edges in G, the greater will be the number of node sets in a tree partition for G. (This statement will be given a more precise form in the theorem which appears later in this section.)

The fact that G is connected ensures that each node of G will appear in some node set N_k. Moreover, the connectivity of G is also enough to guarantee that each of the node sets generated in Step 2b is nonempty. Indeed, suppose that $N_p \neq \phi$ and, say, $N_{q+1} = \phi$. This means that no edges exist between nodes in C_{q+1} and nodes in N_p. Since the nodes of C_{q+1} form a connected component relative to the nodes of $C_p - N_p$, then it follows that C_{q+1} forms a connected component of C_p. Because $N_p \neq \phi$, then $C_{q+1} \subset C_p$ and thus one connected component properly contains another. This is a manifest contradiction, and so $N_p \neq \phi$ must imply that $N_{q+1} \neq \phi, \ldots, N_{q+r} \neq \phi$. In view of the fact that $N_1 \neq \phi$, this observation can be repeatedly applied to demonstrate that all node sets N_k are nonempty. Since the $N_{q+1}, \ldots, N_{q+r}$ defined in Step 2b are nonempty, the edges generated in Step 2c correctly reflect the adjacency relations between the node sets in G. In addition, the fact that no two distinct node sets N_{q+i}, N_{q+j} formed in Step 2b are connected by a chain in $C_p - N_p$ (else C_{q+i} and C_{q+j} would be connected in $C_p - N_p$) is enough to guarantee that the procedure always generates a tree graph for the node sets N_k. Accordingly, it has been verified that TREEPART does indeed produce a tree partitioning for every (nontrivial) connected graph G.

It is interesting to note that when applied to a graph which is itself a tree this procedure yields the original tree back again whenever N_1 is chosen to consist of a single node. In general, different choices for the initial node set N_1 will lead to different tree partitionings. Thus, TREEPART allows one to obtain a whole range of different tree partition structures for the same graph G. In order to obtain a "good" decomposition of the original network (e.g., one containing a large number of node sets), a reasonable heuristic might be to choose an N_1 which consists of relatively few nodes and which itself forms a highly connected subgraph. It often seems desirable to choose N_1 so that $N - N_1$ has several connected components.

As noted above, the number m of node sets in a tree partitioning is at least 2 and at most n. A more precise upper bound on the number of node sets is given by the following result.

THEOREM: *If* $G = (N,E)$ *where* $|N| = n$, $|E| = k$ *then the number* m *of node sets in a tree partitioning of* G *satisfies*

$$m \leq \left[\frac{3 + \sqrt{4n^2 - 4n - 8k + 1}}{2} \right],$$

where [ ] *is the greatest integer function. Moreover, this upper bound is actually tight; that is, there exist graphs with* n *nodes and* k *edges for which equality obtains.*

First, it will be convenient to prove the following lemma[4].

LEMMA: *For a given tree structure* T *with node sets* $N_1, N_2, \ldots, N_m$, *the maximum number of edges possible in any underlying graph* G *is achieved when* $|N_i| = 1$ *for* $i \neq p$ *and* $|N_p| = n - m + 1$, *where* N_p *is a node set of maximal degree in* T.

PROOF: Given the tree structure T, regard the quantities $n_i = |N_i|$ as integer variables constrained by

$$n_i \geq 1, \sum_{i=1}^{m} n_i = n. \tag{4.1}$$

[4] A referee has pointed out a simplified proof of the theorem which does not use this lemma. However, because the lemma refers to a more general problem and is of independent interest, we include its statement and proof.

For given n_i's, the maximum number of edges in the underlying graph G occurs when each N_i corresponds to a complete subgraph, and when for each $(N_i, N_j) \in T$ all of the possible $n_i n_j$ edges between the corresponding subgraphs of G are actually present in G. This maximum number is given by

$$F(\mathbf{n}) = F(n_1, \ldots, n_m) = \frac{1}{2} \sum_{i=1}^{m} n_i (n_i - 1) + \sum \{ n_i n_j : \quad (N_i, N_j) \in T \}.$$

We therefore address the problem of maximizing $F(\mathbf{n})$ subject to (4.1). Let $\mathbf{n}$ be an arbitrary assignment of weights to the nodes satisfying (4.1). It will be shown that there exists an assignment to the nodes of weights $\mathbf{n}^0$ having the form

$$\begin{aligned} n_i &= 1, \; i \neq p \\ n_p &= n - m + 1, \end{aligned} \tag{4.2}$$

with N_p a node set in T of maximal degree, such that $F(\mathbf{n}) \leq F(\mathbf{n}^0)$.

In order to show this, consider the following procedure which performs a reassignment for some current assignment $\mathbf{n}$ relative to the edge $(N_i, N_j) \in T$.

PROCEDURE REASSIGN (i, j).

1. Let w_i be the sum of weights n_s for all node sets N_s adjacent to node set N_i $(s \neq j)$ and let w_j be the sum of weights n_s for all node sets N_s adjacent to node set N_j $(s \neq i)$. Define $r = n_i + n_j$.
2. If $w_i > w_j$ then set $n_i' = r - 1$, $n_j' = 1$. Otherwise, set $n_i' = 1$, $n_j' = r - 1$.

It is clear that for this new assignment $\mathbf{n}'$, where only n_i and n_j have been changed, equation (4.1) is satisfied. Moreover, it will now be shown that $F(\mathbf{n}) \leq F(\mathbf{n}')$. Indeed, if $I = n_i$ and $J = n_j$ then

$$F(\mathbf{n}) = \alpha + \frac{1}{2}\{I(I - 1) + J(J - 1)\} + IJ + Iw_i + Jw_j,$$

where α is independent of I and J. Or, setting $I + J = r$,

$$F(\mathbf{n}) = \alpha + \frac{1}{2} r(r - 1) + Iw_i + (r - I)w_j = \beta + I(w_i - w_j),$$

where β is independent of I, J. Thus, if $w_i > w_j$ it is clearly advantageous, in the sense of increasing $F(\mathbf{n})$, to make I as large as is feasible; namely, $I = r - 1$. Similarly, if $w_i \leq w_j$ then setting $J = r - 1$ will increase $F(\mathbf{n})$ as much as possible. Since this is precisely what is being done in REASSIGN, one has $F(\mathbf{n}) \leq F(\mathbf{n}')$.

Suppose that procedure REASSIGN is performed so long as there are adjacent nodes N_i and N_j for which $n_i > 1$, $n_j > 1$. Consider then the set P of node sets N_i for which $n_i > 1$. Note that the lemma certainly holds when $P = \phi$. If $|P| \geq 2$, it will be demonstrated that a further reassignment of node weights n_i can be made which does not decrease $F(\mathbf{n})$ but which will reduce $|P|$.

Assume, then, that N_i and N_j are in P. Thus $(N_i, N_j) \notin T$ since otherwise procedure REASSIGN could have been employed. In fact, each node set N_k adjacent with N_i must have weight $n_k = 1$, and similarly for N_j. Let d_i be the degree of node set N_i, and d_j the degree of N_j. Also, set $I = n_i$, $J = n_j$ and $r = I + J$. It is claimed that $F(\mathbf{n})$ cannot be decreased by performing the reassignment

$$\begin{aligned} n_i &= 1, n_j = r - 1 \qquad \text{if } d_j \geq d_i \\ n_j &= 1, n_i = r - 1 \qquad \text{if } d_j < d_i. \end{aligned} \tag{4.3}$$

Indeed,

$$F(\mathbf{n}) = \alpha + \frac{1}{2}r(r-1) - IJ + Id_i + Jd_j$$

$$= \alpha + \frac{1}{2}r(r-1) - I(r-I) + Id_i + (r-I)d_j$$

$$= \beta + I^2 + (d_i - d_j - r)I,$$

where α, β are independent of I, J. Because $g(I) = I^2 + (d_i - d_j - r)I$ is a convex function of I, it is maximized over the interval $[1, r-1]$ at one of the endpoints. It is easy to verify that

$$g(1) \geq g(r-1) \quad \Longleftrightarrow \quad (d_j - d_i)(r-2) \geq 0 \quad \Longleftrightarrow \quad d_j \geq d_i,$$

since $r = I + J > 2$. Therefore, the reassignment given in (4.3) cannot decrease the value of $F(\mathbf{n})$. In other words, given node sets N_i, $N_j \in P$, one of the two sets can be made to have weight 1 without decreasing F, and so can be removed from P. By continuing this procedure, P eventually reduces to a single node set with weight $w > 1$. Finally, by using REASSIGN and (4.3) as necessary with respect to the node set N_p of weight $w > 1$ and each node set of weight 1, it will be assured that the resulting N_p is in fact a node of maximal degree d_p in T. Thus, the assignment which is ultimately produced is of the form (4.2). Since the value $F(\mathbf{n})$ is never decreased throughout the process, we indeed have $F(\mathbf{n}) \leq F(\mathbf{n}^0)$. Since the original assignment $\mathbf{n}$ was arbitrary and since all assignments $\mathbf{n}^0$ in (4.2) have the same value $F(\mathbf{n}^0)$, the lemma is proved.

By virtue of this lemma, the number k of edges in an undirected graph G with a tree partitioning of n nodes into m node sets satisfies

$$k \leq F(\mathbf{n}^0) = \frac{1}{2}(n-m+1)(n-m) + d_p(n-m+1) + (m-1-d_p),$$

since a tree on q nodes has precisely $q-1$ edges. Thus,

$$k \leq \frac{1}{2}(n-m+1)(n-m) + d_p(n-m) + (m-1)$$

$$= \alpha(n, m) + d_p(n-m),$$

with $\alpha(n, m)$ independent of d_p. Accordingly, the quantity $\{\alpha(n, m) + d_p(n-m)\}$ is maximized by choosing d_p as large as possible for fixed n, m: namely, $d_p = m-1$ (all other $d_i = 1$). In any event, then,

$$\mathrm{k} \leq \frac{1}{2}(n-m+1)(n-m) + (m-1)(n-m+1)$$

$$= \frac{1}{2}(n^2 - n - 2 + 3m - m^2),$$

whence

$$h(m) \equiv m^2 - 3m - (n^2 - n - 2k - 2) \leq 0.$$

Now the roots m_1 and m_2 of $h(m) = 0$ are $m_1 = \frac{1}{2}(3 - \sqrt{D})$ and $m_2 = \frac{1}{2}(3 + \sqrt{D})$, where $D = 9 + 4(n^2 - n - 2k - 2)$. Because $k \leq \frac{1}{2}n(n - 1)$ then $D \geq 1$ whence $m_1 \leq 1$ and $m_2 \geq 2$. It follows that $h(m) \leq 0$ for all m with $1 \leq m \leq m_2$. Inasmuch as m must be a positive integer, then

$$m \leq [m_2] = \left[\frac{3 + \sqrt{4n^2 - 4n - 8k + 1}}{2}\right].$$

Therefore, the first part of the theorem is established. To prove the second half, consider the graph $\hat{G} = \hat{G}(n, m)$ which consists of $n - m + 1$ "central" nodes, every two of which are joined by an edge, and $m - 1$ "satellite" nodes, each of which is joined by an edge to every central node. The number of edges in $\hat{G}$ is thus

$$k = \frac{1}{2}(n - m + 1)(n - m) + (m - 1)(n - m + 1)$$

and the resulting $m_2 = \frac{1}{2}(3 + \sqrt{(2m - 3)^2}) = m$. Moreover, this graph on n nodes does admit of a tree partitioning with m node sets: namely, choose $m - 1$ node sets each containing a single satellite node, and an mth node set containing all central nodes. Accordingly, the second part of the theorem is verified.

It is worthwhile to note that when the underlying graph is a tree ($k = n - 1$), the upper bound provided by the theorem is exactly n. Here again the upper bound is tight since a tree on n nodes admits of a tree partitioning into n node sets (just let each N_i contain a single node). When the underlying graph is *complete* (that is, every pair of distinct nodes is joined by an edge) then a tree partitioning can have at most two node sets. In fact, the upper bound provided in the theorem for the situation $k = \frac{1}{2}n(n - 1)$ is seen to be 2 also. As a final illustration, the upper bound on m is calculated to be 17 for the graph depicted in figure 1 ($n = 18$, $k = 33$); the tree partitioning of this graph shown in figure 2 contains 10 node sets.

5. Computational Remarks

The principal virtue of the tree decomposition algorithm given in section 3 is that computations need only be performed on arrays which are significantly smaller than the original matrix. Accordingly, much larger matrices than could normally be accommodated in core can be inverted. Moreover, the form of partitioning employed seems to readily adapt itself to the particular zero-nonzero structure of the matrix being studied. Of course, the tree partitioning concept is even more appropriate when repeated matrix inversions are to be made for a sequence of matrices differing only in that nonzero entries are reestimated or varied parametrically [4], [9]. Indeed, for a fixed structure of sparseness the underlying graph remains the same and so the tree partitioning can be found once and for all. In addition, it should be pointed out that TREEPART is at worst an $0(mk)$ algorithm since no edge of the graph need be scanned more than m times. In practice, the amount of computational labor required to find a tree partitioning is really quite modest. For example, when the underlying graph has $k = 0(n)$, as would often be the case for sparse matrices (e.g., resulting from the rectangular and triangular lattices which arise in numerical solution of partial differential equations), the theorem of section 4 shows that $mk \leq 0(n^2)$; accordingly, TREEPART is *at worst* an $0(n^2)$ algorithm for these sparse graphs.

Moreover, the tree decomposition approach is able to exploit effectively the sparsity of the original matrix and thus reduce the amount of computation required for matrix inversion. Suppose, for example, that each of the node sets N_i has $|N_i| = r$. Then the use of INVERT to find the inverse of A requires $0(m^2r^3) = 0(n^2 \cdot \frac{n}{m})$ operations; thus for a given number of nodes, the computational effort decreases as the number of node sets increases. Without exploiting sparsity, standard methods for finding A^{-1} necessitate $0(m^3r^3)$ operations. Furthermore, the decomposition procedure allows the user selectivity in calculating the submatrices X_{ij} of A^{-1}. Indeed, INVERT only *requires* the calculation of the $3m - 2$ submatrices X_{ij} which correspond to nonzero A_{ij} in the original matrix A. The remaining $(m - 1)(m - 2)$ submatrices can be calculated, *if desired,* during Step 4 of the algorithm. Thus, the present decomposition approach would be especially appropriate for problems which require finding only certain submatrices of A^{-1}; such a situation arises when one is interested in finding the variances of estimated coefficients in multiple linear regression, since the required variances are derived from the diagonal entries of a matrix $(X^TX)^{-1}$.

Several procedures have been described for transforming a given matrix into one with a particular partitioned block structure [3], [10], [11], [17], [18, Ch. 3]. The use of a tree partitioning seems sufficiently flexible to deal with a wide range of possible partitions. Fiedler [5] discusses a type of partitioning more general than that given here; however, the solution method indicated in [5] appears to involve too many submatrix inversions to be practically advantageous. Another approach to inverting matrices using graph-theoretic concepts has been described by Harary [10]. Such a method unfortunately is of little use when the matrix A is irreducible. A generalization of Harary's method is presented in [3]. In addition, Mayoh [11] and Steward [17] have discussed techniques for permuting the rows and columns of A so that the resulting matrix has a particularly simple form. The computational requirements of such techniques are, however, difficult to assess.

6. References

[1] Churchill, M. E. A sparse matrix procedure for power systems analysis programs, in Large Sparse Sets of Linear Equations, J. K. Reid, Ed., (Academic Press, London, 1971), pp. 127–138.

[2] Cottle, R. W. Manifestations of the Schur complement, Linear Algebra and Appl. **8** (1974), pp. 189–211.

[3] Dulmage, A. L. and Mendelsohn, N. S. On the inversion of sparse matrices, Math. Comp. **16** (1962), pp. 494–496.

[4] Erisman, A. M. Decomposition methods using sparse matrix techniques with application to certain electrical network problems, in Decomposition of Large-Scale Problems, D. M. Himmelblau, Ed., (North-Holland Publishing Co., Amsterdam, 1973), pp. 69–80.

[5] Fiedler, M. On inverting partitioned matrices, Czechoslovak Math. J. **13** (1963), pp. 574–586.

[6] Fiedler, M. and Pták, V., On matrices with non-positive off-diagonal elements and positive principal minors, Czechoslovak Math. J. **12** (1962), pp. 382–400.

[7] Gantmacher, F. R., The Theory of Matrices, Vol. I, (Chelsea Publishing Co., New York, 1960).

[8] George, A., Nested dissection of a regular finite element mesh, SIAM J. Numer. Anal. **10** (1973), pp. 345–363.

[9] Gustavson, F. G., Liniger, W. and Willoughby, R., Symbolic generation of an optimal Crout algorithm for sparse systems of linear equations, J. Assoc. Comput. Mach. **17** (1970), pp. 87–109.

[10] Harary, F., A graph theoretic approach to matrix inversion by partitioning, Numer. Math. **4** (1962), pp. 128–135.

[11] Mayoh, B. H., A graph technique for inverting certain matrices, Math. Comp. **19** (1965), pp. 644–646.

[12] Parter, S., The use of linear graphs in Gauss elimination, SIAM Rev. **3** (1961), pp. 119–130.

[13] Reid, J. K., Ed., Large Sparse Sets of Linear Equations, (Academic Press, London, 1971).

[14] Rose, D. J., A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, in Graph Theory and Computing, R. C. Read, Ed., (Academic Press, New York, 1972), pp. 183–217.

[15] Rose, D. J. and Willoughby, R. A., Eds., Sparse Matrices and Their Applications, (Plenum Press, New York, 1972).

[16] Shier, D. R., A decomposition algorithm for optimality problems in tree-structured networks, Discrete Math. **6** (1973), pp. 175–189.

[17] Steward, D. V., Partitioning and tearing systems of equations, J. Soc. Indust. Appl. Math. Ser. B Numer. Anal. **2** (1965), pp. 345–365.

[18] Tewarson, R. P., Sparse Matrices, (Academic Press, New York, 1973).

[19] Willoughby, R. A., Ed., Sparse Matrix Proceedings, IBM Research Report RA1, Yorktown Heights, New York, 1969.

(Paper 80B2–439)