SIAM J. NUMER. ANAL.
Vol. 13, No. 2, April 1976

AN ALGORITHM FOR REDUCING THE BANDWIDTH AND PROFILE OF A SPARSE MATRIX*

NORMAN E. GIBBS, WILLIAM G. POOLE, JR. AND PAUL K. STOCKMEYER†

Abstract. A new algorithm for reducing the bandwidth and profile of a sparse matrix is described. Extensive testing on finite element matrices indicates that the algorithm typically produces bandwidth and profile which are comparable to those of the commonly-used reverse Cuthill–McKee algorithm, yet requires significantly less computation time.

1. Introduction. Let

$$Ax = b \tag{1.1}$$

be an $n \times n$ sparse nonsingular system of linear algebraic equations. We are concerned with the band and profile schemes of storage and decomposition for the solution of (1.1). A matrix is banded if all of the nonzero elements are clustered near the main diagonal. The bandwidth, β, of the matrix A is defined by

$$\beta = \max_{a_{ij} \neq 0} |i - j|. \tag{1.2}$$

To define the profile of A, first define $f_i = \min \{j : a_{ij} \neq 0\}$ for $i = 1, 2, \ldots, n$ (it is assumed that $a_{ii} \neq 0$). This locates the leftmost nonzero element in each row. Now define $\delta_i = i - f_i$. The profile is defined to be $\sum_{i=1}^{n} \delta_i$. In this paper, a new algorithm is presented which permutes A into PAP^T, which has a smaller bandwidth and profile than does A. Of course, reducing bandwidth and reducing profile are not equivalent although there is considerable correlation between the two ideas and the new algorithm is designed to reduce both. The algorithm can be applied to matrices with symmetric zero-nonzero structure, i.e., $a_{ij} \neq 0$ if and only if $a_{ji} \neq 0$.

Many bandwidth and profile reduction algorithms have been proposed [2], [1], [22], [9], [17], [16], [3], [14], [20], [7], [26], [6] although the reverse Cuthill–McKee algorithm, a modification by George [13] of the algorithm developed by Cuthill and McKee [9], is perhaps most commonly used. This paper presents an algorithm for reducing bandwidth and profile which appears to be superior to the reverse Cuthill–McKee algorithm. Test results (in § 6) indicate that the new algorithm yields bandwidth and profile which are comparable to those of the reverse Cuthill–McKee algorithm, yet is many times faster.

In § 2 the basic concepts of graph theory that are needed later are discussed, § 3 contains a description of the reverse Cuthill–McKee algorithm and § 4 describes the new algorithm. Section 5 illustrates the application of both algorithms to an example. Section 6 contains the test results of the two algorithms applied to several finite element matrices arising from structural engineering problems.

* Received by the editors July 15, 1974, and in final revised form June 5, 1975. This paper was prepared as a result of work performed under Office of Naval Research Contract N00014-73-A-0374-0001, NR044-459, and in part under NASA Grant NGR 47-102-001 while the second author was in residence at the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia.

† Department of Mathematics, College of William and Mary, Williamsburg, Virginia 23185.

2. Basic concepts from graph theory. Considerable insight often can be gained by using a graph representation of sparse matrices [5], [9], [21]. Of significance to this paper is the fact that permuting the rows and columns of a matrix corresponds to renumbering the vertices of a graph.

If V is a finite nonempty set and $E \subseteq \{\{a, b\}: a \neq b \text{ and } a, b \in V\}$ is a collection of unordered pairs of elements of V, then $G = \langle V, E \rangle$ is a finite undirected graph without loops or multiple edges, or more simply, a *graph*. Given a matrix $A = (a_{ij})$, we can define a graph $G = \langle V, E \rangle$ where V has n vertices, $\{v_1, v_2, \ldots, v_n\}$, and $\{v_i, v_j\} \in E$ if $a_{ij} \neq 0$ and $i \neq j$. The elements of $V = V(G)$ and $E = E(G)$ are called *vertices* and *edges*, respectively. If $\{v_1, v_2\} \in E$, then v_1 and v_2 are said to be *adjacent*. The *degree* of a vertex is the number of vertices adjacent to it.

A *path* of length t is a sequence of edges $\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{t-1}, v_t\}$ such that $v_i = v_j$ implies $i = j$. A graph G is *connected* if there is a path connecting each pair of vertices. If G is not connected, then it consists of two or more *connected components*, or maximal connected subgraphs. The *distance* between vertices v_1 and v_2 of a connected graph is the length of a shortest path from one to the other. A *diameter* of G is a shortest path connecting two vertices of maximal distance apart. The term diameter will also be used for the length of such a path.

In a more specialized vein, if G has n vertices, then a one-to-one map, f, from $V(G)$ onto the set $\{1, 2, \ldots, n\}$ is called a *numbering* of G. For each numbering f, we define $\beta_f(G)$, the *bandwidth of G relative to f*, by

$$\beta_f(G) = \max \{|f(v_1) - f(v_2)| : \{v_1, v_2\} \in E(G)\}. \tag{2.1}$$

The minimum of $\beta_f(G)$ over all numberings of G is called the *bandwidth of G* and denoted by $\beta(G)$.

An important concept in many bandwidth and profile reduction algorithms is that of level structure [3]. A *level structure*, $\mathsf{L}(G)$, of a graph G is a partition of the set $V(G)$ into levels $L_1, L_2, \ldots, L_k$ such that

1. all vertices adjacent to vertices in level L_1 are in either level L_1 or L_2,
2. all vertices adjacent to vertices in level L_k are in either level L_k or L_{k-1}, and
3. for $1 < i < k$, all vertices adjacent to vertices in level L_i are in either level L_{i-1}, L_i, or L_{i+1}.

To each vertex $v \in V(G)$ there corresponds a particular level structure $\mathsf{L}_v(G)$ called the *level structure rooted at v*. Its levels are determined by

1. $L_1 = \{v\}$, and
2. for $i > 1$, L_i is the set of all those vertices adjacent to vertices of level L_{i-1} not yet assigned to a level.

In any level structure $\mathsf{L}(G)$, rooted or not, $w_i(\mathsf{L}) = |L_i|$ (the cardinality of the set L_i) is called the *width of level i*, and $w(\mathsf{L}) = \max \{w_i\}$ is the *width of the level structure* $\mathsf{L}(G)$. It is easily observed (see [9]) that for any level structure, L, a numbering f_{L} of G that assigns consecutive integers level by level, first to the vertices of level L_1, then to those of L_2, and so forth, yields a bandwidth, $\beta_{f_{\mathsf{L}}}$, satisfying

$$\beta_{f_{\mathsf{L}}} \leq 2w(\mathsf{L}) - 1. \tag{2.2}$$

If in addition the level structure L is rooted, then we also have

$$\beta_{f_L} \geq w(\mathsf{L}). \tag{2.3}$$

The *depth* of a level structure is k, the number of levels.

3. The reverse Cuthill–McKee algorithm. The bandwidth and profile reduction algorithm most widely used today is the reverse Cuthill–McKee algorithm. In order to provide a basis of comparison with the new algorithm of § 4, we now describe the reverse Cuthill–McKee algorithm in some detail. For both algorithms it is assumed that the graph is connected. If not, the connected components are determined and the algorithms applied to each component separately.

A. Generate the level structure rooted at each vertex of low degree, and compute its width. Normally, low degree here means less than or equal to $\max\{\min\{(d_{\max}+d_{\min})/2, d_{\text{median}}-1\}, d_{\min}\}$, although this can be controlled somewhat by parameters (see [10]).

B. For each rooted level structure of minimal width generated in step A, number the graph level by level with consecutive positive integers according to the following procedure:
   1. The root vertex is assigned the number 1. (If this is not the first component of the original graph the root vertex is assigned the smallest unassigned positive integer.)
   2. For each successive level, beginning with level 2, first number the vertices adjacent to the lowest numbered vertex of the preceding level, in order of increasing degree. Ties are broken arbitrarily. The remaining vertices adjacent to the next lowest numbered vertex of the preceding level are numbered next, again in order of increasing degree. Continue the process until all vertices of the current level are numbered, then begin again on the next level. The procedure terminates when the vertices of all levels have been numbered.

C. For each numbering f produced in step B.2, compute the corresponding bandwidth $\beta_f(G)$. Select the numbering which produces the smallest bandwidth.

D. The numbering is reversed by setting i to $n-i+1$, for $i=1, 2, \ldots, n$.

Step D was first suggested by George [13] after he observed that profile could frequently be further reduced by numbering the vertices in decreasing order from n to 1 rather than increasing from 1 to n. Recently it was proved that this modification can never increase the profile [24], and of course it has no effect on bandwidth.

This algorithm has several shortcomings. The first is that the algorithm is inefficient because of the time consumed performing an exhaustive search to find rooted level structures of minimal width. In the case that all vertices have the same degree, a level structure must be generated from every vertex of the graph. A second problem is that the graph is renumbered, and the corresponding bandwidth recomputed, for every level structure found of minimal width. A third problem is that the bandwidth obtained by a Cuthill–McKee numbering can never be less than the width of the rooted level structure used (see (2.3)), although the (minimum) bandwidth of a graph can be considerably less than the width of any rooted level structure. This is illustrated by the example in § 5.

In the next section we address the above three problems and present an alternative algorithm. The first two shortcomings are overcome by carefully selecting a starting vertex after generating only a relatively small number of level structures. The graph is renumbered, and corresponding bandwidth and profile computed, only once. The third problem is resolved by utilizing a more general type of level structure.

4. A new bandwidth and profile reduction algorithm. The description of the new algorithm is divided into three parts, each part addressing one of the three problems mentioned in the previous section. The new bandwidth and profile reduction algorithm is simply a combination of Algorithms I and II and III which follow.

4.1. Finding a starting vertex. In our work we have found that level structures of small width are usually among those of maximal depth. Clearly, increasing the number of levels always decreases the average number of vertices in each level, and tends to reduce the width of the level structure as well. Ideally, then, one would like to generate level structures rooted at endpoints of a diameter. Since there is no known efficient procedure that always finds such vertices, we employ the following algorithm to find the endpoints of a *pseudo-diameter*, that is, a pair of vertices that are at nearly maximal distance apart. For a large class of graphs, including all trees and all of the 19 test graphs arising from the problems discussed in § 6, the pseudo-diameter produced is actually a real diameter.

ALGORITHM I. *Finding endpoints of a pseudo-diameter.*

A. Pick an arbitrary vertex of minimal degree and call it v.

B. Generate a level structure L_v rooted at vertex v. Let S be the set of vertices which are in the last level of L_v (i.e., those vertices which are farthest away from v).

C. Generate level structures rooted at vertices $s \in S$ selected in order of increasing degree. If for some $s \in S$ the depth of L_s is greater than the depth of L_v, then set $v \leftarrow s$ and return to step B.

D. Let u be the vertex of S whose associated level structure has smallest width, with ties broken arbitrarily. The algorithm terminates with u and v the endpoints of a pseudo-diameter.

Although the number of iterations required to find a pseudo-diameter depends on arbitrary choices, none of the nineteen test problems required more than two.

4.2. Minimizing level width. In the process of finding a pseudo-diameter, Algorithm I constructs level structures L_u and L_v rooted at the endpoints u and v, respectively. It is possible to combine these two level structures into a new level structure whose width is usually less than that of either of the original ones, using the following algorithm.

ALGORITHM II. *Minimizing level width.*

A. Using the rooted level structures $\mathsf{L}_v = \{L_1, L_2, \ldots, L_k\}$ and $\mathsf{L}_u = \{M_1, M_2, \ldots, M_k\}$ obtained from Algorithm I, associate with each vertex w of G the ordered pair (i, j), called the associated level pair, where i is the index of the level in L_v that contains w, and $k+1-j$ is the index of the level in L_u that contains w. Thus the pair (i, j) is associated with a vertex w if and only if $w \in L_i \cap M_{k+1-j}$. Note that the pair $(1, 1)$ is associated with the vertex v, while the pair (k, k) is associated with u.

B. Assign the vertices of G to levels in a new level structure $\mathsf{L} = \{N_1, N_2, \ldots, N_k\}$ as follows:

1. If the associated level pair of a vertex w is of the form (i, i) then vertex w is placed in N_i. The vertex w and all edges incident to w are removed from the graph. If $V(G) = \emptyset$, stop.
2. The graph G now consists of a set of one or more disjoint connected components $C_1, C_2, \ldots, C_t$ ordered so that $|V(C_1)| \geqq |V(C_2)| \geqq \cdots \geqq |V(C_t)|$.
3. For each connected component C_i, $i = 1, 2, \ldots, t$, do the following:
   (a) Compute the vector $(n_1, n_2, \ldots, n_k)$ where $n_i = |N_i|$.
   (b) Compute the vectors $(h_1, h_2, \ldots, h_k)$ and $(l_1, l_2, \ldots, l_k)$ where $h_i = n_i +$ (the number of vertices which would be placed in N_i if the first element of the associated level pairs were used) and $l_i = n_i +$(the number of vertices which would be placed in N_i if the second element of the associated level pairs were used).
   (c) Find $h_0 = \max_i \{h_i : h_i - n_i > 0\}$ and $l_0 = \max_i \{l_i : l_i - n_i > 0\}$.
      (i) If $h_0 < l_0$, place all the vertices of the connected component in the levels indicated by the first elements of the associated level pairs.
      (ii) If $l_0 < h_0$, use the second elements of the level pairs to place the vertices in the levels.
      (iii) If $h_0 = l_0$, then use the elements of the level pairs which arise from the rotted level structure of smaller width. If the widths are equal, use the first elements.

The algorithm terminates when each vertex of G has been assigned a level in the level structure L.

4.3. Numbering. The numbering procedure is similar to that of the reverse Cuthill–McKee algorithm in that it assigns consecutive positive integers to the vertices of G level by level. A few modifications were necessary, however, since the level structures obtained by Algorithm II are of a more general type than the rooted ones used in the reverse Cuthill–McKee algorithm. When the resulting numbering is similar to that obtained by the (forward) Cuthill–McKee algorithm, profile can be further reduced by using the reverse numbering described in step D below.

ALGORITHM III. *Numbering.*

A. If the degree of u is less than the degree of v, then interchange u and v and reverse the level structure obtained in Algorithm II by setting N_i to N_{k-i+1}. (This insures that the numbering starts from the endpoint of lower degree.)

B. Assign consecutive positive integers to the vertices of level N_1 in the following order:

1. Assign the number 1 to the vertex v (if this is not the first component of the original graph, then assign the smallest unassigned positive integer to v).
2. Let w be the lowest numbered vertex of level N_1 which has unnumbered vertices in N_1 adjacent to it. Number the vertices of N_1 adjacent to w, in

order of increasing degree. Repeat this step until all vertices of N_1 adjacent to numbered vertices are themselves numbered.

3. If any unnumbered vertices remain in level N_1, number the one of minimal degree, then go to step B.2. Otherwise proceed to step C.

C. Number the vertices of level N_i, $i = 2, 3, \ldots, k$, as follows:

1. Let w be the lowest numbered vertex of level N_{i-1} that has unnumbered vertices of level N_i adjacent to it. Number the vertices of N_i adjacent to w in order of increasing degree. Repeat this step until all vertices of level N_i adjacent to vertices of level N_{i-1} are numbered.
2. Repeat steps B.2 and B.3, replacing 1 with i.

D. The numbering is reversed by setting i to $n-i+1$, for $i = 1, 2, \ldots, n$ if either of the two following conditions holds:

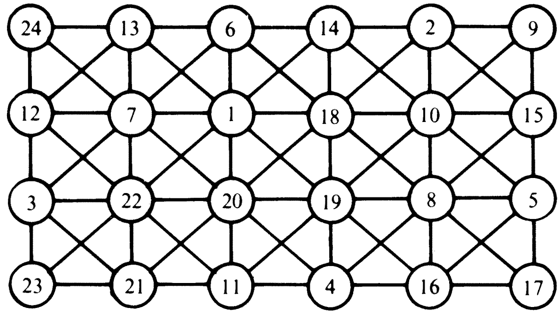
1. Step A interchanged vertices u and v and Algorithm II selected the second elements of the level pairs for component C_1.
2. Step A did not interchange vertices u and v and Algorithm II selected the first elements of the level pairs for component C_1.

5. An example. In this section we demonstrate the application of the two algorithms of § 3 and 4 by an example.
Let

$$
(5.1)\quad A = \begin{bmatrix}
x&&&&&x&x&&&&&&x&x&&&&x&x&x&&x&& \\
&x&&&&&&&x&x&&&&x&x&&&x&&&&&& \\
&&x&&&&x&&&&&x&&&&&&&&&x&x&x& \\
&&&x&&&&x&&&x&&&&&x&&&x&x&&&& \\
&&&&x&&&x&&x&&&&&x&x&x&&&&&&& \\
x&&&&&x&x&&&&&&x&x&&&&x&&&&&& \\
x&&x&&&x&x&&&&&x&x&&&&&&&x&&x&&x \\
&&&x&x&&&x&&x&&&&&x&x&x&x&x&&&&& \\
&x&&&&&&&x&x&&&&&x&&&&&&&&& \\
&x&&&x&&&x&x&x&&&&x&x&&&x&x&&&&& \\
&&&x&&&&&&&x&&&&&&&&x&x&x&x&& \\
&&x&&&&x&&&&&x&x&&&&&&&&&x&&x \\
x&&&&&x&x&&&&&x&x&&&&&&&&&&&x \\
x&x&&&&x&&&&x&&&&x&&&&x&&&&&& \\
&x&&&x&&&x&x&x&&&&&x&&&&&&&&& \\
&&&x&x&&&x&&&&&&&&x&x&&x&&&&& \\
&&&&x&&&x&&&&&&&&x&x&&&&&&& \\
x&x&&&&x&&x&&x&&&&x&&&&x&x&x&&&& \\
x&&&x&&&&x&&x&x&&&&&x&&x&x&x&&&& \\
x&&&x&&&x&&&&x&&&&&&&x&x&x&x&x&& \\
&&x&&&&&&&&x&&&&&&&&&x&x&x&x& \\
x&&x&&&&x&&&&x&x&&&&&&&&x&x&x&x& \\
&&x&&&&&&&&&&&&&&&&&&x&x&x& \\
&&&&&&x&&&&&x&x&&&&&&&&&&&x
\end{bmatrix},
$$

where x denotes the location of the nonzero elements. The associated numbered graph is

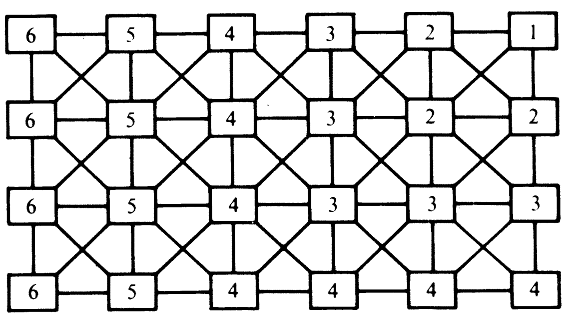
(5.2)



.

Whenever the vertices are represented by circles, the integers contained in the circles refer to the appropriate numbering of the vertices. Whenever the vertices are represented by rectangles, the integers contained in the rectangles refer to the appropriate levels generated by the algorithms. Because $d_{\text{median}} = 5$ and $(d_{\max} + d_{\min})/2 = 5.5$, the Cuthill–McKee algorithm generates level structures rooted at the four vertices numbered 9, 17, 23 and 24. For vertex 9, the following structure is obtained:

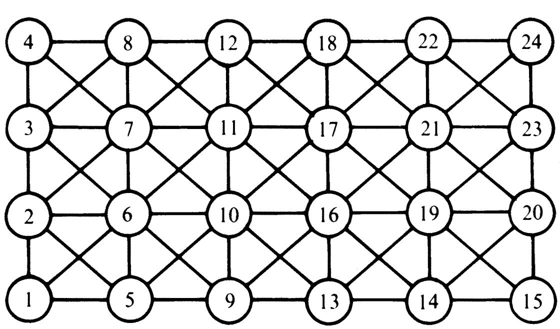
(5.3)



.

The other three level structures have a similar form. Using level structure (5.3), we obtain the reverse numbering

(5.4)



.

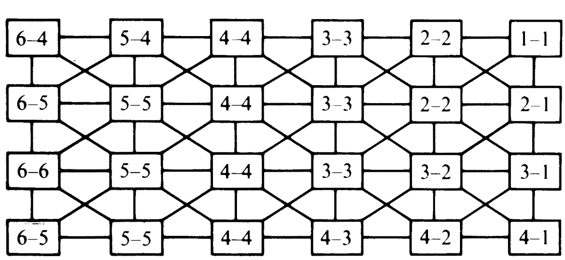
The level width and the bandwidth are both 7 for all four level structures and the resulting matrix is

$$
(5.5)\qquad
\left[\begin{array}{cccccccccccccccccccccccc}
x & x & & & x & x & & & & & & & & & & & & & & & & & & \\
x & x & x & & x & x & x & & & & & & & & & & & & & & & & & \\
 & x & x & x & & x & x & x & & & & & & & & & & & & & & & & \\
 & & x & x & & & x & x & & & & & & & & & & & & & & & & \\
x & x & & & x & x & & & x & x & & & & & & & & & & & & & & \\
x & x & x & & x & x & x & & x & x & x & & & & & & & & & & & & & \\
 & x & x & x & & x & x & x & & x & x & x & & & & & & & & & & & & \\
 & & x & x & & & x & x & & & x & x & & & & & & & & & & & & \\
 & & & & x & x & & & x & x & & & x & & & x & & & & & & & & \\
 & & & & x & x & x & & x & x & x & & x & & & x & x & & & & & & & \\
 & & & & & x & x & x & & x & x & x & & & & x & x & x & & & & & & \\
 & & & & & & x & x & & & x & x & & & & & x & x & & & & & & \\
 & & & & & & & & x & x & & & x & x & & x & & & x & & & & & \\
 & & & & & & & & & & & & x & x & x & x & & & x & x & & & & \\
 & & & & & & & & & & & & & x & x & & & & x & x & & & & \\
 & & & & & & & & x & x & x & & x & x & & x & x & & x & & x & & & \\
 & & & & & & & & & x & x & x & & & & x & x & x & x & & x & x & & \\
 & & & & & & & & & & x & x & & & & & x & x & & & x & x & & \\
 & & & & & & & & & & & & x & x & x & x & x & & x & x & x & & x & \\
 & & & & & & & & & & & & & x & x & & & & x & x & x & & x & \\
 & & & & & & & & & & & & & & & x & x & x & x & x & x & x & x & x \\
 & & & & & & & & & & & & & & & & x & x & & & x & x & x & x \\
 & & & & & & & & & & & & & & & & & & x & x & x & x & x & x \\
 & & & & & & & & & & & & & & & & & & & & x & x & x & x
\end{array}\right]
$$

whose profile is 97.

The algorithm of § 4 applied to the graph of (5.2) will also choose vertex 9 as a starting vertex, generating the same level structure as in (5.3). However this algorithm now differs from reverse Cuthill–McKee by noticing that the level structure rooted at vertex 3 gives a smaller level width of 6. We now have two level structures, one rooted at vertex $v = 9$ and one at $u = 3$. The associated level pairs are

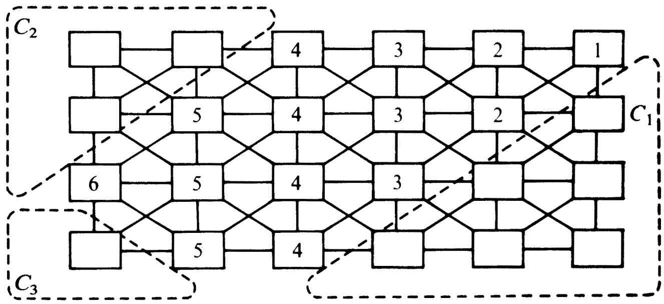
(5.6)



.

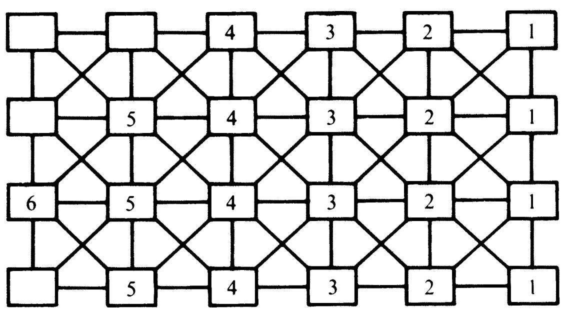
Algorithm II now assigns vertices with level pair (i, i) to level N_i. The disjoint connected components are

(5.7)



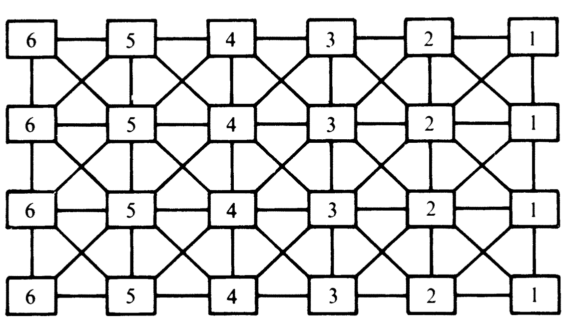
For components C_1, $(n_1, n_2, \ldots, n_6) = (1, 2, 3, 4, 3, 1)$, $(h_1, h_2, \ldots, h_6) = (1, 3, 5, 7, 3, 1)$ and $(l_1, l_2, \ldots, l_6) = (4, 4, 4, 4, 3, 1)$. It follows that $h_0 = 7$ and $l_0 = 4$, and thus we use the second elements in the level pairs, yielding the partially completed level assignments

(5.8)



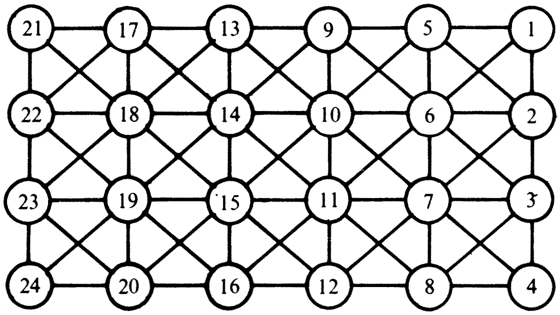
After assigning levels to vertices in components C_2 and C_3 we get the following level structure:

(5.9)



The numbering of the vertices yields the graph

(5.10)



and the matrix

(5.11)

$$\left[\begin{array}{cccccccccccccccccccccccc}
x & x & & & x & x & & & & & & & & & & & & & & & & & & \\
x & x & x & & x & x & x & & & & & & & & & & & & & & & & & \\
& x & x & x & & x & x & x & & & & & & & & & & & & & & & & \\
& & x & x & & & x & x & & & & & & & & & & & & & & & & \\
x & x & & & x & x & & & x & x & & & & & & & & & & & & & & \\
x & x & x & & x & x & x & & x & x & x & & & & & & & & & & & & & \\
& x & x & x & & x & x & x & & x & x & x & & & & & & & & & & & & \\
& & x & x & & & x & x & & & x & x & & & & & & & & & & & & \\
& & & & x & x & & & x & x & & & x & x & & & & & & & & & & \\
& & & & x & x & x & & x & x & x & & x & x & x & & & & & & & & & \\
& & & & & x & x & x & & x & x & x & & x & x & x & & & & & & & & \\
& & & & & & x & x & & & x & x & & & x & x & & & & & & & & \\
& & & & & & & & x & x & & & x & x & & & x & x & & & & & & \\
& & & & & & & & x & x & x & & x & x & x & & x & x & x & & & & & \\
& & & & & & & & & x & x & x & & x & x & x & & x & x & x & & & & \\
& & & & & & & & & & x & x & & & x & x & & & x & x & & & & \\
& & & & & & & & & & & & x & x & & & x & x & & & x & x & & \\
& & & & & & & & & & & & x & x & x & & x & x & x & & x & x & x & \\
& & & & & & & & & & & & & x & x & x & & x & x & x & & x & x & x \\
& & & & & & & & & & & & & & x & x & & & x & x & & & x & x \\
& & & & & & & & & & & & & & & & x & x & & & x & x & & \\
& & & & & & & & & & & & & & & & x & x & x & & x & x & x & \\
& & & & & & & & & & & & & & & & & x & x & x & & x & x & x \\
& & & & & & & & & & & & & & & & & & x & x & & & x & x
\end{array}\right]$$

which has bandwidth 5. It is well known that this cannot be further reduced. The profile produced by this algorithm is 98.

6. Description of test results. Every bandwidth or profile reduction algorithm currently in use is heuristic in the sense that one cannot make absolute a priori statements concerning the performance of the algorithm—the performance is data dependent. The standard way to evaluate such an algorithm is to test it on several examples—in some sense "typical", if possible—and compare the results with other algorithms or some "standard" algorithm. We have chosen this method in order to evaluate our algorithm.

For our test matrices we have chosen 19 sparse matrices which were accumulated over a period of several years by E. H. Cuthill and G. C. Everstine of the Naval Ship Research and Development Center (NSRDC). Several of these appear in [11]. These matrices arise in the solution of various differential equations and variational problems in structural engineering when the finite element method is used. One- and two-dimensional elements (triangles and quadrilaterals) were used. The problems include such diverse applications as aircraft structures, liquid nitrogen gas tanks, propeller blades and submarines.

For our "standard" algorithm we have chosen the reverse Cuthill–McKee algorithm of § 3. This is probably the most commonly used bandwidth and profile reduction algorithm [11, p. 47], [8], [13, p. 101] and is included in several structural engineering software packages [11, p. 47]. The particular implementation is a FORTRAN IV program which was given to us by E. H. Cuthill and G. C. Everstine of NSRDC. The algorithm described in § 4 has been implemented by the authors, also in FORTRAN IV. Tests on the 19 matrices were run on the IBM 360 model 50 computer at the College of William and Mary. The interval timer was used in order to minimize the side effects of operating in a multiprogramming environment. It is felt that the comparative times of the two programs fairly represent their performances.

Table 1 presents the results of the tests. Figures 1, 2 and 3 graphically display the test results with respect to bandwidth, profile and execution time, respectively. The results indicate that the new algorithm typically gave a bandwidth comparable to that of the reverse Cuthill–McKee algorithm. The new algorithm actually gave a slightly smaller bandwidth on average primarily because of examples 17 and 18. Furthermore the profiles produced by the new algorithm were usually slightly smaller than the profiles obtained using the reverse Cuthill–McKee algorithm.

7. Conclusions. We feel that the new algorithm is a viable choice when one is selecting a bandwidth or profile reduction algorithm. This conclusion is based on our experience with using both it and the reverse Cuthill–McKee algorithm on the abovementioned 19 examples and many other test cases. The bandwidth and profiles produced by the new algorithm are comparable and the new algorithm requires significantly less execution time. Also our FORTRAN program required no more storage than did the NSRDC implementation of the Cuthill–McKee algorithm [10].

As stated in § 3, there are primarily three reasons for explaining why the new algorithm is an improvement over the reverse Cuthill–McKee algorithm. Because of the method for finding a pseudo-diameter, relatively few vertices must be examined as potential starting vertices for the numbering. For the test problems of

TABLE 1
Test results

Case	N	β	β_c	β_n	P	P_c
1	68	45	5	7	598	236
2	90	85	9	7	1,020	575
3	92	80	14	13	2,127	739
4	130	126	19	18	3,615	1,562
5	159	19	11	12	1,046	983
6	174	16	14	13	1,569	1,615
7	185	168	30	29	7,534	3,664
8	220	166	13	12	8,532	1,809
9	263	262	19	19	2,681	2,337
10	263	30	13	14	2,040	2,023
11	310	302	14	14	23,357	2,725
12	312	262	33	37	18,076	5,812
13	346	216	43	46	16,435	7,180
14	360	344	33	34	29,790	6,001
15	436	173	34	33	7,913	8,181
16	512	399	28	29	35,837	4,838
17	555	480	110	91	56,322	29,904
18	861	833	79	71	100,560	45,961
19	918	840	46	49	124,607	21,479
Totals		4,846	567	548	443,659	147,624

Case	P_n	T_c	T_n	T_c/T_n	
1	269	6.63	.60	11.06	
2	579	5.20	1.33	3.90	
3	736	5.67	.88	6.42	
4	1,588	3.30	1.55	2.13	
5	971	3.80	1.88	2.02	
6	1,466	8.73	2.07	4.23	
7	3,610	13.98	3.08	4.54	
8	1,868	113.10	3.35	33.76	
9	2,346	57.48	5.45	10.55	
10	2,001	24.57	3.87	6.35	
11	2,726	34.98	4.28	8.17	
12	5,548	28.08	4.88	5.75	
13	7,650	39.18	6.12	6.41	
14	6,364	25.50	4.25	6.00	
15	7,844	34.93	9.17	3.81	
16	4,669	36.70	19.32	1.90	
17	28,976	62.27	7.28	8.55	
18	45,525	183.68	13.93	13.18	
19	20,369	178.60	17.92	9.97	
Totals	145,105	866.38	111.21	7.83	(average)

N — order of matrix
β — bandwidth of matrix
β_c — bandwidth after reverse Cuthill–McKee algorithm
β_n — bandwidth after new algorithm
P — profile of matrix
P_c — profile after reverse Cuthill–McKee algorithm
P_n — profile after new algorithm
T_c — time (in seconds) for reverse Cuthill–McKee algorithm
T_n — time (in seconds) for new algorithm

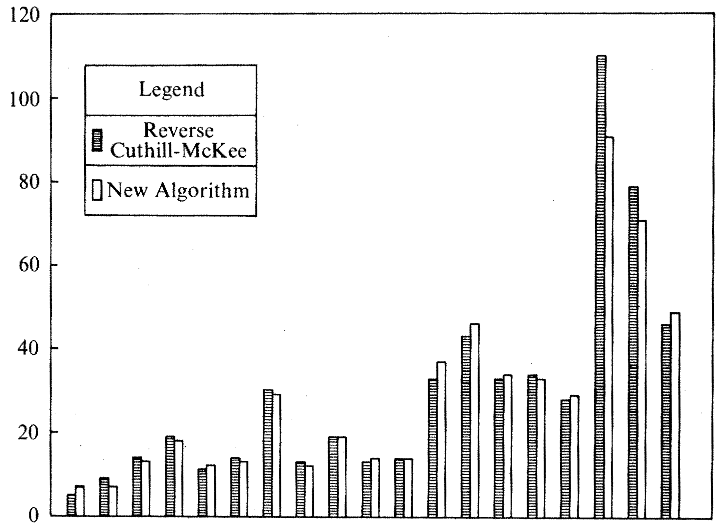


FIG. 1. *Bandwidth for* 19 *examples*

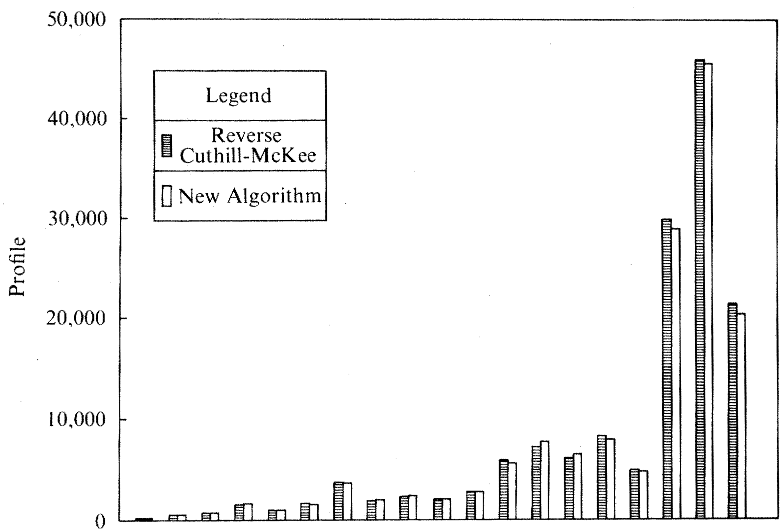


FIG. 2. *Profile for* 19 *examples*

Table 1, the reverse Cuthill–KcKee algorithm typically generated between 10 and 20 times as many level structures as the new algorithm. In our implementation of the new algorithm, however, the generation of each rooted level structure requires more time than for the reverse Cuthill–McKee algorithm due to the retention of additional leveling information utilized later in the algorithm. Secondly, the graph is renumbered, and corresponding bandwidth computed, only

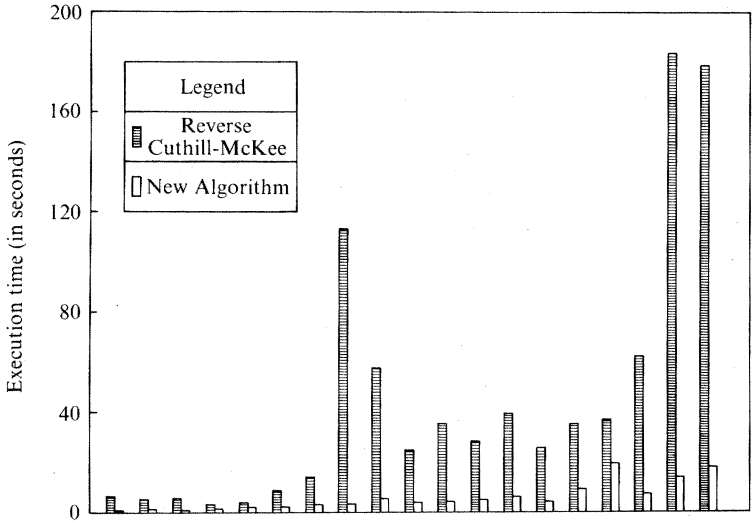


FIG. 3. *Execution time for* 19 *examples*

once. Because of these two reasons, the program implementing the new algorithm never required more time and, on the average, the NSRDC implementation of the reverse Cuthill–McKee algorithm required about 7 to 8 times as long. Finally, the more general level structure permits the case where the bandwidth is smaller than the width of any rooted level structure.

Acknowledgments. The authors wish to thank E. H. Cuthill and G. C. Everstine of NSRDC for the use of their programs and their 19 test matrices which they have accumulated over several years from many structures problems. The authors also wish to acknowledge the assistance of H. L. Crane, Jr. who did most of the programming for this project.

REFERENCES

[1] F. A. AKYUZ AND S. UTKU, *An automatic relabeling scheme for bandwidth minimization of stiffness matrices*, J. Amer. Inst. Aeronaut. Astronaut, 6 (1968), pp. 728–730.

[2] G. G. ALWAY AND D. W. MARTIN, *An algorithm for reducing the bandwidth of a matrix of symmetric configuration*, Comput. J., 8 (1965), pp. 264–272.

[3] I. ARANY, W. F. SMYTH AND L. SZODA, *An improved method for reducing the bandwidth of sparse symmetric matrices*, Proc. IFIP Conference, North–Holland, Amsterdam, 1971, pp. 1246–1250.

[4] JAMES R. BUNCH, *Analysis of sparse elimination*, this Journal, 11 (1974), pp. 847–873.

[5] ——— *Complexity of sparse elimination*, Complexity of Sequential and Parallel Numerical Algorithms, J. F. Traub, ed., Academic Press, New York, 1973.

[6] K. Y. CHENG, *Minimizing the bandwidth of sparse symmetric matrices*, Computing, 11 (1973), pp. 103–110.

[7] R. J. COLLINS, *Bandwidth reduction by automatic renumbering*, Internat. J. Numer. Meth. Engrg., 6 (1973), pp. 345–356.

[8] ELIZABETH CUTHILL, *Several strategies for reducing the bandwidth of matrices*, Sparse Matrices and Their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York, 1972.

[9] ELIZABETH CUTHILL AND J. MCKEE, *Reducing the bandwidth of sparse symmetric matrices*, Proc. ACM National Conference, Association for Computing Machinery, New York, 1969, pp. 157–172.

[10] G. C. EVERSTINE, *The BANDIT Computer Program for the reduction of matrix bandwidth for NASTRAN*, 3827, Naval Ship Research and Development Center, Washington, D. C., 1972.

[11] *First NSRDC-NASTRAN Colloquium Proceedings*, Naval Ship Research and Development Center, Washington, D. C., 1970.

[12] GEORGE FORSYTHE AND CLEVE B. MOLER, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1967.

[13] ALAN GEORGE, *Computer implementation of the finite element method*, STAN-CS-71-208, Computer Science Dept., Stanford Univ., Stanford, Calif., 1971.

[14] H. R. GROOMS, *Algorithm for matrix bandwidth reduction*, Amer. Soc. Civil Enginrg., J. Struct. Div. 98, ST1 (1972), pp. 203–214.

[15] A. JENNINGS, *A compact storage scheme for the solution of symmetric linear simultaneous equations*, Comput. J., 9 (1966), pp. 281–285.

[16] I. P. KING, *An automatic reordering scheme for simultaneous equations derived from network systems*, Internat. J. Numer. Meth. Engrg., 2 (1970), pp. 523–533.

[17] R. LEVY, *Resequencing of the structural stiffness matrix to improve computational efficiency*, Jet Propulsion Laboratory Tech. Rev., 1 (1971), pp. 61–70.

[18] R. S. MARTIN, C. REINSCH AND J. H. WILKINSON, *The QR algorithm for band symmetric matrices*, Numer. Math., 16 (1970), pp. 85–92.

[19] J. K. REID, ed., *Large Sparse Sets of Linear Equations*, Academic Press, London, 1971.

[20] ERNEST ROBERTS, JR., *Relabeling of finite-element meshes using a random process*, TM X-2660, National Aeronautics and Space Administration, Lewis Research Center, Cleveland, Ohio, 1972.

[21] DONALD J. ROSE AND RALPH A. WILLOUGHBY, eds., *Sparse Matrices and Their Applications*, Plenum Press, New York, 1972.

[22] R. ROSEN, *Matrix bandwidth minimization*, Proc. ACM National Conference, Brandon Systems Press, Princeton, N.J.; 1968, pp. 585–595.

[23] H. R. SCHWARZ, *Tridiagonalization of a symmetric band matrix*, Numer. Math., 12 (1968), pp. 231–241.

[24] ANDREW H. SHERMAN AND WAI-HUNG LIU, *Comparative analysis of the Cuthill–McKee and the reverse Cuthill–McKee ordering algorithms for sparse matrices*, to appear.

[25] REGINALD P. TEWARSON, *Sparse Matrices*, Academic Press, New York, 1973.

[26] P. T. R. WANG, *Bandwidth minimization, reducibility, decomposition, and triangularization of sparse matrices*, Ph.D. dissertation, Ohio State University, Columbus, 1973.

[27] R. A. WILLOUGHBY, ed., *Sparse Matrix Proceedings*, RA1, IBM Watson Research Center, Yorktown Heights, N. Y., 1969.

[28] DAVID M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.