Computers & Structures Vol. 33, No. 2, pp. 411–419, 1989
Printed in Great Britain.

0045-7949/89 $3.00 + 0.00

A DIRECT COMPARISON OF THREE ALGORITHMS FOR REDUCING PROFILE AND WAVEFRONT

S. W. SLOAN and W. S. NG
Department of Civil Engineering and Surveying, University of Newcastle, N.S.W. 2308, Australia

(*Received* 22 *September* 1988)

Abstract—The effectiveness of the reverse Cuthill–McKee, Gibbs–King and Sloan ordering algorithms for reducing the profile and wavefront of a sparse matrix with a symmetric pattern of zeros is assessed by applying them to each of Everstine's 30 test problems and comparing the results with those obtained by Armstrong's simulated annealing algorithm. This last procedure, although much too slow for practical use, is believed to give profiles which are close to the minimum and thus provides useful benchmark results for comparing possible procedures. For the test problems of Everstine, the reverse Cuthill–McKee procedure furnishes profiles which, on average, are 31% above those produced by the simulated annealing method. In the worst case, the reverse Cuthill–McKee algorithm yields a profile which is 114% in excess of the corresponding simulated annealing profile. The Gibbs–King scheme is generally more reliable than the reverse Cuthill–McKee scheme, but still yields an average profile which is 22% above that produced by the simulated annealing algorithm. Its worst-case performance gives a profile which is 67% in excess of the simulated annealing profile. The Sloan algorithm appears to be a substantial improvement on both the reverse Cuthill–McKee and Gibbs–King methods when measured by its average and worst-case profiles which are, respectively, 10 and 24% above those of the simulated annealing method. The root-mean-square wavefront reductions for the three algorithms mimic those of the profile reductions except that, in general, they are slightly further from the optimum values. In addition to those for Everstine's cases, some test results for a set of rectangular grid problems are also given. These examples correspond to meshes of 2D linear and quadratic finite elements which are used frequently in practice. The relative performance of the various algorithms is generally similar to that for Everstine's test problems although the spread of results is reduced significantly.

INTRODUCTION

A large number of problems in engineering require the solution of a set of linear equations of the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a prescribed positive definite $N \times N$ matrix which is sparse and has a symmetric pattern of zeros, $\mathbf{b}$ is a prescribed vector of length N, and $\mathbf{x}$ is a vector of length N which is sought. Such equations arise in finite element computations, and in most problems involving networks in general, and are usually solved by some form of Gaussian elimination. As the matrix $\mathbf{A}$ is sparse, it is possible to devise very efficient solution procedures which exploit sparsity in various ways. The profile method, which has proved to be particularly effective in finite element analysis, discards all leading zeros in each row and column and only stores (and operates on) entries which lie within the profile of the matrix. For the profile method to be efficient it is necessary to label the equations, which in finite element grids is equivalent to labelling the nodes, so that few zeros follow the leading non-zero in each row and column. If we consider the lower triangle of $\mathbf{A}$, this may be viewed as labelling the equations to minimize the sum of the row bandwidths. More formally we may define the profile, P, according to

$$P = \sum_{i=1}^{N} b_i,$$

where the row bandwidth, b_i, is the difference between $i + 1$ and the column index of the first nonzero in row i. The success of the profile scheme hinges on the fact that Gaussian elimination (or any of its common variants) does not require any row or column interchanges to preserve numerical stability if the coefficient matrix $\mathbf{A}$ is positive definite. Any fill-in which occurs during the elimination process is thus confined to zeros inside the profile and this allows simple and efficient implementations to be developed [1–3]. Labelling a set of sparse matrix equations to minimize the profile is not a trivial task, especially when the equations are derived from large finite element meshes, and a number of automatic procedures have been proposed. The most successful of these are based on graph theory and include the reverse Cuthill–McKee algorithm [3, 4], the King algorithm [5], the Gibbs–King algorithm [6, 7] and the algorithm proposed by Sloan [8, 9].

Another procedure for solving sparse matrix equations, which was originally developed for finite element analysis, is the frontal technique [10, 11]. This algorithm, in contrast to other variants of Gaussian elimination, does not assemble the matrix $\mathbf{A}$ fully before commencing the elimination step; instead, the assembly and elimination phases are conducted in tandem, with each variable being eliminated once its corresponding row and column in $\mathbf{A}$ are fully summed. Any equation that is fully or

partially summed before each elimination step is said to be active, and the total number of active equations is known as the wavefront (or frontwidth). More formally, we define row j to be active during the elimination of column i if $j \geq i$ and there exists $a_{jk} \neq 0$ with $k \leq i$. Thus the wavefront at the ith elimination step is simply the number of rows of the profile of **A**, ignoring those which have already been eliminated, which intersect column i. Letting f_i denote the number of equations which are active during the elimination of the variable x_i, it follows from the symmetric structure of **A** that

$$P = \sum_{i=1}^{N} f_i = \sum_{i=1}^{N} b_i.$$

To describe the efficiency of an ordering, it is convenient to define a quantity called the root-mean-square wavefront according to

$$F = \left(\frac{1}{N} \sum_{i=1}^{N} f_i^2 \right)^{1/2}.$$

Assuming that N and the average value of f_i are reasonably large, it may be shown that the total number of operations required for a profile or frontal elimination is $0(NF^2)$. Thus to minimize storage we wish to minimize the profile, whereas to minimize the computation time we wish to reduce the root-mean-square wavefront. Clearly, any algorithm which attempts to minimize the number of equations which are active during each step of the elimination will tend to reduce both the profile and root-mean-square wavefront.

To assess the merits of various algorithms for reducing profile and wavefront, it is generally necessary to resort to empirical testing on a standard set of problems. One such set of problems, collected by Everstine [12], has been assembled primarily for the purpose of comparing various ordering algorithms. These sparse matrices arise from 30 finite element grids and reflect a broad range of matrix structures that occur in practice. For most problems of practical interest it is not possible to deduce an ordering with the optimum profile and root-mean-square wavefront by theoretical or computational means (recall that a set of N equations has N factorial possible orderings). Recently, however, Armstrong [13] has published profiles and wavefronts for Everstine's examples which are considered to be close to the true minima. These were obtained using a simulated annealing algorithm which, although much too slow for practical purposes, provides useful benchmarks for comparing various other algorithms. Using Everstine's test problems and Armstrong's results for the simulated annealing method, we compare here the performance of the following ordering algorithms:

(1) the reverse Cuthill–McKee scheme as implemented by George and Liu [3],

(2) the Gibbs–King scheme as implemented by Lewis [7], and

(3) the Sloan scheme as implemented by Sloan [9].

All of these procedures are fully automatic, in the sense that they require no intervention from the user, and produce their labellings in a single pass. Each algorithm contains an iterative method for selecting the node at which the numbering is started, but these usually terminate quickly and result in the schemes being particularly fast. A detailed description of the reverse Cuthill–McKee algorithm, together with an efficient computer program written in FORTRAN 66, can be found in [3]. Except for a number of small changes which are made to incorporate features of FORTRAN 77, this program is used here. An efficient implementation of the Gibbs–King scheme, which was first developed by Gibbs [6], has been given by Lewis [7]. This code is distributed by the Association for Computing Machinery as algorithm 582 and is written in FORTRAN 66. The implementation of the algorithm developed by Sloan [8] is written in FORTRAN 77, and described in [9].

Overall, the results for Everstine's test problems indicate that the reverse Cuthill–McKee scheme is the most unreliable of the three algorithms. On one occasion it yields a profile which is 114% in excess of the corresponding simulated annealing profile. For the average case, it produces a profile which is 31% above that of the simulated annealing method. Similar statistics are also observed for the root-mean-square wavefronts. The major attraction of the reverse Cuthill–McKee algorithm, is its speed, as it is typically 140% faster than the Gibbs–King method and 50% faster than the Sloan scheme. Overall, for Everstine's collection of test problems, the Sloan procedure appears to be the most successful profile and wavefront reduction scheme with average and worst-case profiles which are, respectively, 10 and 24% above the simulated annealing profiles. The performance of the Gibbs–King method is between that of the reverse Cuthill–McKee and Sloan schemes.

Results for an additional set of problems, comprised of rectangular grids of 2D linear and quadratic finite elements, are also given. Meshes of these types occur frequently in practice and are thus useful additions to the library of test problems. The relative performances of the various algorithms for these examples are similar to those for Everstine's examples, although the spread of results is reduced considerably.

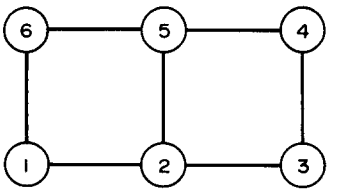
NOTATION

As first pointed out by Cuthill and McKee [4], the ordering of a sparse matrix for minimum profile or wavefront can be viewed as a labelling of an undirected graph. Elementary concepts from graph theory provide a concise means of describing order-

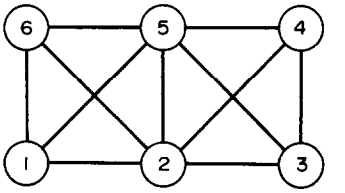
ing algorithms and they are usually discussed in this context. As it is not intended to describe each algorithm in detail, for this has been done elsewhere, this notation will not be covered in great depth. Nonetheless, it is useful to define some basic terms so that the various properties of the algorithms, and their implementations, can be discussed succinctly.

A graph consists of a set of members called nodes and a set of unordered pairs of nodes called edges. A graph satisfying this definition is said to be undirected, as each nodal pair defining the set of edges is unordered. Note that we exclude the occurrence of loops, which are edges connecting nodes to themselves, and multiple edges where pairs of nodes are connected by more than one edge. In this paper, each node is assumed to be labelled with the integers $1, 2, \ldots, N$. The degree of a node is the number of edges that are incident to it, and any pair of nodes are adjacent if they are connected by an edge. A path is defined by a sequence of edges such that consecutive edges share a common node. Two nodes are said to be connected if there is a path joining them, and the graph is said to be connected if each pair of nodes is connected.

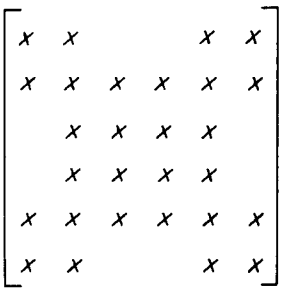
Any sparse matrix with a symmetric structure may be viewed as an undirected graph, as shown in Fig. 1. Each node in the graph corresponds to a row in the matrix, and the degree of the node gives the number of off-diagonal nonzeros in its row. As a pair of nodes i and j are joined by an edge only if $a_{ij} = a_{ji} \neq 0$, it follows that the number of nonzeros in the matrix is equal to $N + 2E$, where E is the total number of edges. (Note that for the 2D grid shown in Fig. 1, where each node is associated with two equations, the quantity $2E + N$ actually corresponds to the total number of nonzero 2×2 submatrices.) In finite element applications, the graph corresponding to a mesh is constructed using the rule that all nodes which share an element are connected by an edge (Fig. 1). As all finite element analyses require a list of the nodes, defining each element, to describe the mesh topology, the information needed to store the graph, namely, a list giving the nodes which are adjacent to each node, may be generated automatically. In the Cuthill–McKee [3] and Sloan [9] implementations, the graph is stored as an adjacency list which is accessed via a pointer vector. Letting ADJ and XADJ denote the adjacency list and pointer vector, respectively, the nodes adjacent to node I are found in ADJ(J), where J = XADJ(I), XADJ(I) + 1, ..., XADJ(I + 1) − 1. The degree of node I is given by XADJ(I + 1) − XADJ(I). Note that this data structure stores each edge twice for ease of access and requires $2E + N + 1$ words of integer memory. To illustrate this scheme, the adjacency list for the graph shown in Fig. 1 is stored as ADJ = {2, 6, 5, 1, 3, 6, 5, 4, 2, 5, 4, 5, 2, 3, 6, 1, 2, 3, 4, 1, 2, 5} and the corresponding pointer vector is XADJ = {1, 4, 9, 12, 15, 20, 23}. In the Lewis [7] implementation of the Gibbs–King scheme, the graph is stored



A grid of 4-noded quadrilaterals



The corresponding graph



The structure of the corresponding matrix (X = nonzero entry)

Fig. 1. Graph representation of a sparse matrix with a symmetric structure.

using a similar data structure. The adjacency list is named CONNEC and accessed by the vectors RSTART (which points to the start of each set of neighbours) and DEGREE (which lists the degree of each node). The nodes adjacent to node I are found in CONNEC(J), where J = RSTART(I), RSTART(I)+1, ..., RSTART(I)+DEGREE(I)−1, and the total storage requirement is $2E + 2N$ integer words of memory. Including the storage needed to define the structure of the graph, the reverse Cuthill–McKee [3] and Sloan [9] algorithms require, respectively, $2E + 4N + 2$ and $2E + 5N + 2$ words of memory to complete the labelling. The Lewis [7] implementation of the Gibbs–King scheme has an average requirement of approximately $2E + 7N$ storage locations, and a worst-case requirement of $2E + 9N + 3$ storage locations. Note that all three algorithms have the ability to label disconnected graphs, a feature which is needed to run Everstine's test problems.

RESULTS FOR EVERSTINE'S TEST PROBLEMS

To assess the performance of the reverse Cuthill–McKee, Gibbs–King and Sloan algorithms directly, results are given for the set of test problems collected by Everstine [12]. The profiles and root-mean-square wavefronts produced are compared with those obtained by Armstrong [13], who employed a simulated annealing scheme. As discussed by Armstrong, it is likely that the simulated annealing profiles are within a few per cent of the optimum, as they are always less than (or on rare occasions equal to) those obtained by any other procedure. Moreover, where the optimum orderings can be deduced by inspection, these are reproduced (either exactly or very closely) by the simulated annealing scheme.

The profiles produced by the various algorithms for Everstine's test problems are shown in Table 1. The reverse Cuthill–McKee implementation clearly performs less favourably than the other two schemes and, on average, gives a profile which is 31% in excess of the simulated annealing profile. The Gibbs–King and Sloan algorithms yield average profiles which are, respectively, 22 and 10% above the profiles produced by the simulated annealing method. In their worst performances, the reverse Cuthill–McKee and Gibbs–King schemes yield results which are, respectively, 114 and 76% above those produced by simulated annealing. On another three occasions, these two algorithms give profiles which are at least 50% above the optimum. The Sloan procedure performs considerably better in this regard, with a worst-case profile which is 24% larger than the simulated annealing profile. Of the three algorithms tested, the Sloan scheme yields the lowest profile on 24 occasions, whereas the Gibbs–King scheme gives the lowest profile on four occasions. The reverse Cuthill–McKee procedure does not achieve the lowest profile for any of the cases. The worst performance of the Sloan scheme, relative to that of Gibbs–King, results in a 4% increase in profile (problem 24). Conversely, the worst relative performance of the Gibbs–King algorithms results in a 62% increase in profile (problem 17). The worst performance of the reverse Cuthill–McKee algorithm, relative

Table 1. Results for profile reduction on Everstine's test problems

Problem	N	P_o	P_{rcm}	P_{gk}	P_{slo}	P_{arm}	$\frac{P_{rcm}}{P_{arm}}$	$\frac{P_{gk}}{P_{arm}}$	$\frac{P_{slo}}{P_{arm}}$
1	59	464	314	314	294	273	1.15	1.15	1.08
2	66	640	217	193	193	193	1.12	1.00	1.00
3	72	244	244†	244†	244†	219	1.11	1.11	1.11
4	87	2336	696	682	525	515	1.35	1.32	1.02
5	162	2806	1641	1579	1554	1272	1.29	1.24	1.22
6	193	7953	5505	4609	4618	4409	1.25	1.05	1.05
7	198	5817	1417	1313	1297	1287	1.10	1.02	1.01
8	209	9712	3819	4032	3316	2693	1.42	1.50	1.23
9	221	10,131	2225	2154	2052	1848	1.20	1.17	1.11
10	234	1999	1539	1349	1089	1016	1.51	1.33	1.07
11	245	4179	4179†	3813	2676	2161	1.93	1.76	1.24
12	307	8132	8132†	8132†	7550	6535	1.24	1.24	1.16
13	310	3006	3006†	3006†	2982	2940	1.02	1.02	1.01
14	346	9054	8034	7681	6726	6136	1.31	1.25	1.10
15	361	5445	5075	5060	5062	4992	1.02	1.01	1.01
16	419	40,145	8649	8073	7255	6512	1.33	1.24	1.11
17	492	34,282	7067	5513	3412	3304	2.14	1.67	1.03
18	503	36,417	15,319	15,042	14,436	11,958	1.28	1.26	1.21
19	512	6530	5319	4970	4680	4384	1.21	1.13	1.07
20	592	29,397	11,440	10,925	10,073	9417	1.21	1.16	1.07
21	607	30,615	17,868	14,760	14,412	13,065	1.37	1.13	1.10
22	758	23,871	8580	8175	7598	7123	1.20	1.15	1.07
23	869	20,397	19,293	15,728	14,909	13,207	1.46	1.19	1.13
24	878	26,933	22,391	19,696	20,537	17,835	1.26	1.10	1.15
25	918	109,273	23,105	20,498	17,236	15,949	1.45	1.29	1.08
26	992	263,298	38,128	34,068	33,928	32,528	1.17	1.05	1.04
27	1005	122,075	43,068	40,141	36,396	32,513	1.32	1.23	1.10
28	1007	26,793	24,703	22,465	22,669	19,913	1.24	1.13	1.14
29	1242	111,430	50,052	52,952	36,822	33,098	1.51	1.60	1.11
30	2680	590,543	105,663	99,271	89,686	84,900	1.24	1.17	1.06
						Average	1.31	1.22	1.10

N, Number of nodes.
P_o, Original profile.
P_{rcm}, Profile for reverse Cuthill–McKee.
P_{gk}, Profile for Gibbs–King.
P_{slo}, Profile for Sloan.
P_{arm}, Profile for Armstrong.
†No improvement.

to that of Sloan, results in a 107% increase in profile (problem 17).

The results for the root-mean-square wavefronts are very similar to those for the profiles, and are shown in Table 2. Again, the reverse Cuthill–McKee scheme performs less favourably than the Gibbs–King or Sloan schemes, and gives an average root-mean-square wavefront which is 35% above that of the simulated annealing algorithm. In the worst case, it results in a root-mean-square wavefront which is 124% in excess of the simulated annealing root-mean-square wavefront. The Gibbs–King procedure gives average and worst-case results which are, respectively, 24 and 80% above those produced by simulated annealing. The Sloan scheme gives the most consistent reductions in root-mean-square wavefront. Its average and worst-case results are, respectively, 11 and 27% above those given by simulated annealing and, out of the three algorithms tested, it gives the lowest root-mean-square wavefront on 24 occasions. The Gibbs–King procedure furnishes the best result for four cases and the reverse Cuthill–McKee scheme never achieves the best result. The worst performance of the Sloan scheme, relative to that of Gibbs–King, results in a 7% increase in root-mean-square wavefront (problem 24). Conversely, the worst performances of the Gibbs–King and reverse Cuthill–McKee schemes, relative to that of Sloan, result in increases of 67 and 115% respectively (problem 17).

In order to assess the relative efficiency of the reverse Cuthill–McKee, Gibbs–King and Sloan algorithms, the times required to complete the orderings were measured and are shown in Table 3. These statistics are for a VAX 11/780 running under VMS and were determined using the internal clock of the machine. Note that all the programs were compiled with full optimization on the DEC FORTRAN compiler. As can be seen from these results the reverse Cuthill–McKee algorithm is, on average, 140% faster than the Gibbs–King algorithm and 50% faster than the Sloan algorithm. All the schemes, however, are easily fast enough for practical use as the time required to order the equations would, in most instances, be only a small fraction of the time required to solve them.

Table 2. Results for root-mean-square wavefront reduction on Everstine's test problems

Problem	N	F_o	F_{rcm}	F_{gk}	F_{slo}	F_{arm}	$\frac{F_{rcm}}{F_{arm}}$	$\frac{F_{gk}}{F_{arm}}$	$\frac{F_{slo}}{F_{arm}}$
1	59	8.22	5.45	5.53	5.15	4.74	1.15	1.17	1.09
2	66	11.01	3.34	2.94	2.94	2.94	1.14	1.00	1.00
3	72	3.46	3.46†	3.46†	3.46†	3.12	1.11	1.11	1.11
4	87	29.38	8.50	8.24	6.30	6.16	1.38	1.34	1.02
5	162	18.96	10.53	10.09	9.93	7.97	1.32	1.27	1.25
6	193	43.84	30.48	24.86	24.73	23.70	1.29	1.05	1.04
7	198	30.90	7.64	6.95	6.88	6.83	1.12	1.02	1.01
8	209	50.32	19.23	20.40	16.77	13.33	1.44	1.53	1.26
9	221	50.39	10.52	10.18	9.71	8.64	1.22	1.18	1.12
10	234	9.35	7.30	6.29	4.95	4.59	1.59	1.37	1.08
11	245	18.48	18.48†	16.64	11.76	9.23	2.00	1.80	1.27
12	307	27.36	27.36†	27.36†	25.65	22.33	1.23	1.23	1.15
13	310	9.85	9.85†	9.85†	9.76	9.62	1.02	1.02	1.01
14	346	27.15	24.43	23.29	20.23	18.42	1.33	1.26	1.10
15	361	15.38	14.28	14.23	14.24	14.08	1.01	1.01	1.01
16	419	107.07	21.39	19.96	17.80	15.96	1.34	1.25	1.12
17	492	79.51	15.39	11.99	7.13	6.88	2.24	1.74	1.04
18	503	78.60	32.82	32.22	30.71	24.93	1.32	1.29	1.23
19	512	14.55	13.16	11.76	11.44	10.30	1.28	1.14	1.11
20	592	55.18	20.63	19.70	18.32	16.65	1.24	1.18	1.10
21	607	55.43	34.13	27.25	26.71	24.32	1.40	1.12	1.10
22	758	37.95	12.80	12.01	10.92	10.05	1.27	1.20	1.09
23	869	25.02	24.19	19.87	18.94	15.63	1.55	1.27	1.21
24	878	31.92	26.62	22.60	24.22	20.98	1.27	1.08	1.15
25	918	131.14	26.74	23.39	19.52	18.02	1.48	1.30	1.08
26	992	302.00	40.12	34.66	34.92	33.51	1.20	1.03	1.04
27	1005	137.66	48.87	44.80	40.47	33.79	1.45	1.33	1.20
28	1007	26.93	25.43	22.63	23.00	20.26	1.26	1.12	1.14
29	1242	105.20	45.29	46.26	30.42	27.30	1.55	1.69	1.11
30	2680	234.42	40.58	37.93	34.14	32.27	1.26	1.18	1.06
						Average	1.35	1.24	1.11

F_o, Original root-mean-square wavefront.
F_{rcm}, Root-mean-square wavefront for reverse Cuthill–McKee.
F_{gk}, Root-mean-square wavefront for Gibbs–King.
F_{slo}, Root-mean-square wavefront for Sloan.
F_{arm}, Root-mean-square wavefront for Armstrong.
† No improvement.

Table 3. Timing statistics for ordering of Everstine's test problems

Problem	N	T_{rcm}	T_{gk}	T_{slo}	$\frac{T_{gk}}{T_{rcm}}$	$\frac{T_{slo}}{T_{rcm}}$
1	59	0.02	0.03	0.02	1.5	1.0
2	66	0.02	0.04	0.02	2.0	1.0
3	72	0.01	0.02	0.01	2.0	1.0
4	87	0.02	0.05	0.03	2.5	1.5
5	162	0.07	0.11	0.07	1.6	1.0
6	193	0.14	0.40	0.22	2.9	1.6
7	198	0.07	0.20	0.09	2.9	1.3
8	209	0.08	0.18	0.12	2.3	1.5
9	221	0.09	0.17	0.11	1.9	1.2
10	234	0.06	0.18	0.07	3.0	1.2
11	245	0.08	0.21	0.14	2.6	1.8
12	307	0.14	0.28	0.20	2.0	1.4
13	310	0.11	0.23	0.17	2.1	1.5
14	346	0.17	0.34	0.20	2.0	1.2
15	361	0.13	0.24	0.16	1.8	1.2
16	419	0.19	0.36	0.26	1.9	1.4
17	492	0.18	0.36	0.24	2.0	1.3
18	503	0.29	0.49	0.40	1.7	1.4
19	512	0.21	0.69	0.35	3.3	1.7
20	592	0.29	0.55	0.38	1.9	1.3
21	607	0.24	0.66	0.42	2.8	1.8
22	758	0.34	0.68	0.44	2.0	1.3
23	869	0.33	1.03	0.48	3.1	1.5
24	878	0.34	1.18	0.58	3.5	1.7
25	918	0.34	1.02	0.61	3.0	1.8
26	992	0.69	2.60	1.09	3.8	1.6
27	1005	0.46	0.94	0.81	2.0	1.8
28	1007	0.39	1.30	0.71	3.3	1.8
29	1242	0.50	1.79	0.98	3.6	2.0
30	2680	1.30	2.72	2.41	2.1	1.9
				Average	2.4	1.5

T_{rcm}, Time for reverse Cuthill–McKee.
T_{gk}, Time for Gibbs–King.
T_{slo}, Time for Sloan.
All times in CPU sec for VAX11/780 operating under VMS with the FORTRAN optimizing compiler.

better profiles can be found by the automatic schemes.

Table 4 shows the results for various rectangular meshes which are comprised of three-noded triangles. The Gibbs–King and Sloan algorithms give identical results for all the cases considered and improve upon the initial labelling substantially. For the square mesh, where $I = J$, these algorithms reduce the profile and root-mean-square wavefront by roughly 20%. These reductions diminish as the aspect ratio (J/I) increases, however, and for $J/I = 5$ are roughly equal to 4%. Except for square meshes, the reverse Cuthill–McKee algorithm gives profiles and root-mean-square wavefronts which are roughly 5% above those of Gibbs–King and Sloan. As with Everstine's problems, the reverse Cuthill–McKee scheme is generally the fastest of the three schemes.

The results for the rectangular meshes comprised of four-noded quadrilaterals, shown in Table 5, indicate that none of the three algorithms is able to improve on the initial labelling. With regard to speed, the Gibbs–King scheme is markedly slower than the other two algorithms.

The results for the rectangular meshes of six-noded triangles are shown in Table 6. In all cases, the initial top-to-bottom column-by-column numbering is shown to be far from optimal. In contrast to the examples considered so far, the reverse Cuthill–McKee algorithm provides the best reductions in profile and wavefront. For square meshes, the reductions in profile and root-mean-square wavefront are roughly 42%, but decrease with increasing aspect ratio to roughly 31% for $J/I = 5$. The performance of the reverse Cuthill–McKee and Sloan algorithms is similar for square grids, but the former yields results which are roughly 6% less than the latter for meshes

RECTANGULAR GRID PROBLEMS

The 2D rectangular grid problems considered in this section are shown in Fig. 2. Each mesh is comprised of I rows and J columns of grid squares, with each grid square consisting of one of the following:

(1) two three-noded triangles,
(2) one four-noded quadrilateral,
(3) two six-noded triangles,
(4) one eight-noded quadrilateral.

Mesh types (1) and (2) are comprised of linear elements, whereas mesh types (3) and (4) are comprised of quadratic elements. All four of these grid types are used frequently in 2D finite element computations. The graphs corresponding to these meshes are constructed by connecting all nodes which share an element. Initially, the nodes in each graph are labelled across the shortest 'dimension' in a top-to-bottom column-by-column fashion. This type of numbering would be expected to give reasonably good profiles, and it is of interest to see whether

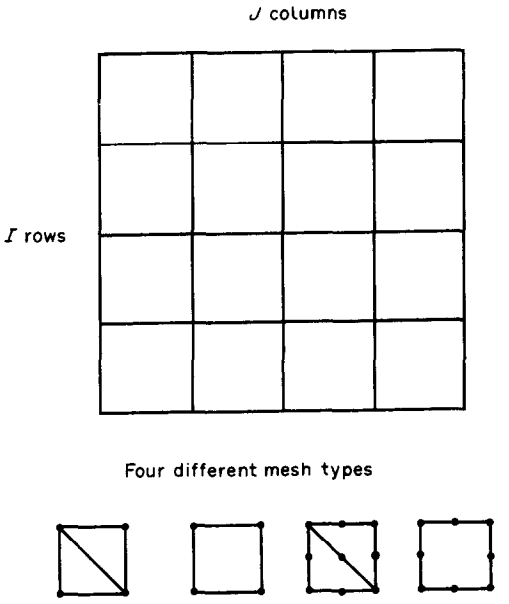


Fig. 2. Two-dimensional rectangular grid problems.

Table 4. Results for rectangular grid problems with the three-noded triangle

$I \times J$	10 × 10	10 × 20	10 × 30	10 × 40	10 × 50
N	121	231	341	451	561
E	320	630	940	1250	1560
P_o	1341	2661	3981	5301	6621
F_o	11.34	11.66	11.77	11.83	11.86
P_{rcm}	1056	2476	3896	5301†	6621†
P_{gk}	1056	2376	3696	5016	6336
P_{slo}	1056	2376	3696	5016	6336
F_{rcm}	9.13	11.09	11.71	11.83†	11.86†
F_{gk}	9.13	10.59	11.07	11.30	11.44
F_{slo}	9.13	10.59	11.07	11.30	11.44
T_{rcm}	0.05	0.09	0.12	0.16	0.18
T_{gk}	0.08	0.14	0.23	0.27	0.33
T_{slo}	0.05	0.09	0.14	0.19	0.23
P_{rcm}/P_{slo}	1.00	1.04	1.05	1.06	1.05
P_{gk}/P_{slo}	1.00	1.00	1.00	1.00	1.00
T_{gk}/T_{rcm}	1.6	1.6	1.9	1.7	1.8
T_{slo}/T_{rcm}	1.0	1.0	1.2	1.2	1.3

I, Number of rows of grid squares.
J, Number of columns of grid squares.
E Number of edges.
$(\ )_{rcm}$, Results for reverse Cuthill–McKee.
$(\ )_{gk}$, Results for Gibbs–King.
$(\ )_{slo}$, Results for Sloan.
† No improvement.
All times in CPU sec for VAX11/780 operating under VMS with the FORTRAN optimizing compiler.

Table 5. Results for rectangular grid problems with the four-noded quadrilateral

$I \times J$	10 × 10	10 × 20	10 × 30	10 × 40	10 × 50
N	121	231	341	451	561
E	420	830	1240	1650	2060
P_o	1441	2861	4281	5701	7121
F_o	12.18	12.54	12.66	12.72	12.76
P_{rcm}	1441†	2861†	4281†	5701†	7121†
P_{gk}	1441†	2861†	4281†	5701†	7121†
P_{slo}	1441†	2861†	4281†	5701†	7121†
F_{rcm}	12.18†	12.54†	12.66†	12.72†	12.76†
F_{gk}	12.18†	12.54†	12.66†	12.72†	12.76†
F_{slo}	12.18†	12.54†	12.66†	12.72†	12.76†
T_{rcm}	0.05	0.10	0.13	0.17	0.21
T_{gk}	0.17	0.23	0.33	0.46	0.54
T_{slo}	0.06	0.12	0.20	0.24	0.31
P_{rcm}/P_{slo}	1.00	1.00	1.00	1.00	1.00
P_{gk}/P_{slo}	1.00	1.00	1.00	1.00	1.00
T_{gk}/T_{rcm}	3.4	2.3	2.5	2.7	2.6
T_{slo}/T_{rcm}	1.2	1.1	1.5	1.4	1.5

See previous tables for explanation.

Table 6. Results for rectangular grid problems with the six-noded triangle

$I \times J$	10 × 10	10 × 20	10 × 30	10 × 40	10 × 50
N	441	861	1281	1701	2121
E	2160	4290	6420	8550	10,680
P_o	13,701	27,351	41,001	54,651	68,301
F_o	32.10	32.57	32.73	32.82	32.86
P_{rcm}	7905	17,809	27,649	37,489	47,329
P_{gk}	8770	18,656	28,406	38,156	47,906
P_{slo}	7951	18,743	29,135	39,931	50,565
F_{rcm}	18.75	21.34	22.06	22.41	22.62
F_{gk}	21.34	22.45	22.71	22.84	22.92
F_{slo}	18.96	22.51	23.31	23.93	24.22
T_{rcm}	0.22	0.43	0.67	0.86	1.14
T_{gk}	0.41	0.80	1.28	1.69	2.13
T_{slo}	0.31	0.68	1.00	1.38	1.70
P_{rcm}/P_{slo}	0.99	0.95	0.95	0.94	0.94
P_{gk}/P_{slo}	1.10	1.00	0.98	0.96	0.95
T_{gk}/T_{rcm}	1.9	1.9	1.9	2.0	1.9
T_{slo}/T_{rcm}	1.4	1.6	1.5	1.6	1.5

See previous tables for explanation.

Table 7. Results for rectangular grid problems with the eight-noded quadrilateral

$I \times J$	10 × 10	10 × 20	10 × 30	10 × 40	10 × 50
N	341	661	981	1301	1621
E	2260	4490	6720	8950	11,180
P_o	9351	18,651	27,951	37,251	46,551
F_o	28.10	28.67	28.86	28.96	29.02
P_{rcm}	9009	17,481	25,431	33,381	41,331
P_{gk}	8133	16,671	24,621	32,571	40,521
P_{slo}	7713	15,591	23,547	31,611	39,585
F_{rcm}	28.08	27.53	26.69	26.25	25.99
F_{gk}	24.59	25.64	25.39	25.26	25.18
F_{slo}	23.51	24.03	24.31	24.54	24.62
T_{rcm}	0.21	0.41	0.58	0.82	0.98
T_{gk}	1.62	1.59	1.90	2.54	2.91
T_{slo}	0.30	0.66	0.95	1.30	1.60
P_{rcm}/P_{slo}	1.17	1.12	1.08	1.06	1.04
P_{gk}/P_{slo}	1.05	1.07	1.05	1.03	1.02
T_{gk}/T_{rcm}	7.7	3.9	3.3	3.1	3.0
T_{slo}/T_{rcm}	1.4	1.6	1.6	1.6	1.6

See previous tables for explanation.

with larger aspect ratios. The performance of the Gibbs–King procedure is erratic; for square meshes it gives profiles and wavefronts which are 11% above those produced by the reverse Cuthill–McKee algorithm, whereas for meshes with larger aspect ratios the results for the two schemes are similar. As for all the examples analysed so far, the reverse Cuthill–McKee procedure is the fastest of the three schemes, with the Sloan scheme being appreciably faster than the Gibbs–King scheme.

Table 7 shows the results for the various meshes of eight-noded quadrilaterals. As for the preceding cases, the profiles for the initial labellings are further from their optimal values than might be expected. For the square mesh, the Sloan scheme reduces the profile and root-mean-square wavefront by roughly 18%, with a reduction of 15% for a mesh with an aspect ratio of five. In all cases, this algorithm yields the best reductions in profile and wavefront, with the reverse Cuthill–McKee scheme giving the worst reductions. The results for the reverse Cuthill–McKee procedure are 17% above those of the Sloan procedure for square meshes, but its relative performance improves as the aspect ratio of the mesh increases. Similar results are observed for the Gibbs–King scheme except that its profiles and root-mean-square wavefronts exceed those of the Sloan scheme by smaller margins. With regard to speed, the reverse Cuthill–McKee and Sloan algorithms are substantially faster than the Gibbs–King algorithm.

CONCLUSION

The performances of the reverse Cuthill–McKee, Gibbs–King and Sloan algorithms have been compared directly using two sets of test problems. The first set of problems, collected by Everstine, suggests that the reverse Cuthill–McKee scheme, as implemented by George and Liu [3], is the least successful of the three schemes for reducing profile and root-mean-square wavefront. Even though it is the fastest algorithm tested, its average and worst-case results are substantially poorer than those of the Gibbs–King or Sloan algorithms. Overall, for Everstine's test cases, the Sloan procedure gives the most consistent reductions in profile and wavefront. The Gibbs–King procedure yields results which are worse than those of the Sloan scheme but better than those of the reverse Cuthill–McKee method. For the second set of test problems, comprised of rectangular meshes of linear and quadratic finite elements, the relative performances of the three algorithms are similar to those for Everstine's cases. The major difference is that the spread of the results is reduced significantly.

REFERENCES

1. A. Jennings, *Matrix Computation for Engineers and Scientists*. John Wiley, New York (1977).
2. R. L. Taylor, *The Finite Element Method* (Ed. by O. C. Zienkiewicz), Chapter 24. McGraw-Hill, London (1977).
3. A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ (1981).
4. E. Cuthill and J. McKee, Reducing the bandwidth of sparse symmetric matrices. *Proc. ACM Natl Conf.* Association for Computing Machinery, pp. 157–172. Brandon Press, NJ (1969).
5. I. P. King, An automatic reordering scheme for simultaneous equations derived from network systems. *Int. J. Numer. Meth. Engng* **2**, 523–533 (1970).
6. N. E. Gibbs, A hybrid profile reduction algorithm. *ACM Trans. Math. Software* **2**, 378–387 (1976).
7. J. G. Lewis, Implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms. *ACM Trans. Math. Software* **8**, 180–189 (1982).
8. S. W. Sloan, An algorithm for profile and wavefront reduction of sparse matrices. *Int. J. Numer. Meth. Engng.* **23**, 239–251 (1986).
9. S. W. Sloan, A FORTRAN program for profile and wavefront reduction. *Int. J. Numer. Meth. Engng* (submitted).
10. R. J. Melosh and R. M. Bamford, Efficient solution of load deflection equations. *J. Struct. Div., ASCE* **95**, 661–676 (1969).

11. B. M. Irons, A frontal solution program for finite element analysis. *Int. J. Numer. Meth. Engng* **2,** 5–32 (1970).
12. G. C. Everstine, A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront. *Int. J. Numer. Meth. Engng* **14,** 837–853 (1979).
13. B. A. Armstrong, Near-minimal matrix profiles and wavefronts for testing nodal resequencing algorithms. *Int. J. Numer. Meth. Engng* **21,** 1785–1790.