SIAM REVIEW
Vol. 31, No. 1, pp. 1–19, March 1989

THE EVOLUTION OF THE MINIMUM DEGREE ORDERING ALGORITHM*

ALAN GEORGE† AND JOSEPH W.H. LIU‡

Abstract. Over the past fifteen years, the implementation of the minimum degree algorithm has received much study, and many important enhancements have been made to it. This paper describes these various enhancements, their historical development, and some experiments showing how very effective they are in improving the execution time of the algorithm. A shortcoming is also presented that exists in all of the widely used implementations of the algorithm, namely, that the quality of the ordering provided by the implementations is surprisingly sensitive to the initial ordering. For example, changing the input ordering can lead to an increase (or decrease) of as much as a factor of three in the cost of the subsequent numerical factorization. This sensitivity is caused by the lack of an effective tie-breaking strategy, and the authors' experiments illustrate the importance of developing such a strategy.

Key words. sparse matrices, minimum degree algorithm, computational complexity, ordering algorithms

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

1. Introduction.

Consider the n by n symmetric positive definite system of equations

$$Ax = b,$$

where n is large and A is sparse. When A is factored using Cholesky's method, it normally suffers some fill. Since PAP^T is also symmetric and positive definite for any permutation matrix P, we can instead solve the reordered system

$$(PAP^T)(Px) = Pb.$$

The choice of P can have a dramatic effect on the amount of fill that occurs during the factorization. Thus, it is standard practice to reorder the rows and columns of the matrix before performing the factorization.

The overall solution of a sparse positive definite system of equations is typically divided into four distinct independent phases:

(a) Find an appropriate ordering P for A.

(b) Set up a data structure for L, the Cholesky factor of PAP^T.

* Received by the editors March 22,1987; accepted for publication (in revised form) February 25, 1988.

† Department of Computer Science, University of Tennessee, Knoxville, Tennessee 37919. Present address: Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. This author is also affiliated with the Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, Tennessee, through the University of Tennessee/Oak Ridge National Laboratory Distinguished Scientist program. His research was supported in part by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems Inc., and by The Science Alliance, a state-supported program at the University of Tennessee.

‡ Department of Computer Science, York University, North York, Ontario, Canada M3J 1P3. This author's research was supported in part by Natural Sciences and Engineering Research Council of Canada grant A5509 and by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems Inc.

(c) Numerically factor PAP^T into LL^T.
(d) Solve $(LL^T)(Px) = Pb$.

Note that steps (a) and (b) depend only on the structure of A and are independent of its numerical values.

The problem of finding a *best* ordering for A in the sense of minimizing the fill is computationally intractable: an NP-complete problem [33]. We are therefore obliged to rely on heuristic algorithms. One of the most effective of these is the *minimum degree algorithm.* This algorithm is a symmetric analog of an algorithm proposed by Markowitz in 1957 for reordering equations arising in linear programming applications [28]. Loosely speaking, the Markowitz algorithm begins with the given matrix, and at each step of Gaussian elimination, row and column permutations are performed so as to minimize the product of the number of off-diagonal nonzeros in the pivot row and pivot column. Thus, one minimizes the amount of arithmetic that must be performed at each step of Gaussian elimination. This will also tend to minimize the amount of fill that occurs. Of course such a *local* minimization strategy will not in general provide a global minimum for either the arithmetic requirements or the fill. Nevertheless, the strategy has proved to be very effective in reducing arithmetic and fill. The cost of employing the strategy is almost always far outweighed by the savings in execution time that accrue. In practice, in order to preserve numerical stability, care must be taken to avoid using pivot elements that are too small in magnitude. Thus, there is a tradeoff between limiting fill and preserving numerical accuracy. A widely used modern implementation of Markowitz's basic strategy, along with many improvements, is Duff's MA28 code [2].

Tinney and Walker employed a similar strategy in solving the large sparse systems arising in the analysis of power systems [32]. These problems have symmetric structure and do not require interchanges for numerical stability provided that the pivots are taken from the diagonal of A. Thus, Tinney and Walker employed a symmetric version of the Markowitz strategy. That is, whenever rows were interchanged, the corresponding columns were interchanged as well, thus preserving the symmetric structure of the matrix. Subsequently, Rose [29] developed a graph theoretic model of the algorithm, and for reasons that will be clear in later parts of this paper, he renamed Tinney and Walker's symmetric analog of Markowitz's original strategy as the minimum degree algorithm.

In terms of the matrix A, the minimum degree algorithm can be described as follows. Generally, it works only with the structure of A and *simulates* in some manner the n steps of symmetric Gaussian elimination. At each step, a row and corresponding column interchange is applied to the part of the matrix remaining to be factored so that the number of nonzeros in the pivot row and column is minimized. (Note that since the structure of the matrix is symmetric, the number of nonzeros in the pivot row and pivot column is the same.) After n steps, the entire factorization has been simulated, and the order in which the pivot rows and columns were chosen is the ordering. An important observation, which we will focus on later, is that *ties* usually occur in the choice of the pivot row and column, which implies that the ordering obtained will depend on how these ties are resolved.

Rose's 1970 doctoral thesis work [30] did much to promote interest in the minimum degree algorithm, and its efficient implementation has received a great deal of attention over the past fifteen years. In particular, researchers at Harwell, Waterloo, and Yale have devoted much effort in this direction [8], [9], [11], [12], [17], [18], [21], [26]. A number of effective enhancements have been proposed and are now standard techniques in most state-of-the-art implementations. One of our objectives in this

article is to trace these developments and to provide some experimental results illustrating the extent to which the technology has advanced. A second objective of our work is to describe and highlight a significant shortcoming that exists in all of the widely used implementations of the algorithm. Specifically, the *quality* of the ordering provided by these implementations is quite sensitive to the initial ordering of A. This is a reflection of the fact that the quality of the ordering depends quite strongly on the way in which ties are resolved, and all of the implementations break ties more or less arbitrarily. Finding an effective *tie-breaking strategy* is one of the many interesting problems related to the algorithm that still remain unsolved.

The significance of a tie-breaking strategy for the minimum degree ordering algorithm has long been recognized by many researchers. The recent book by Duff, Erisman, and Reid [5] provides an interesting discussion of the effect of tie-breaking. In [6], the same authors demonstrate the effect by breaking ties based on the initial ordering, and provide results using a variety of initial orderings. The experimental results in this paper further illustrate the importance of developing an effective tie-breaking strategy.

An outline of the paper follows. In §2 we describe the minimum degree algorithm in graph theory terms and provide a small example. In §3 we trace the development of the various enhancements that have been proposed to reduce the execution time of the algorithm, and §4 contains some numerical experiments showing the relative effectiveness of the enhancements. In §5 we provide some experiments which illustrate the importance of tie-breaking. Section 6 mentions briefly some implementation issues and describes how the implementation might be adapted to special structures arising in certain applications, such as the finite-element method and least squares problems. Section 7 contains some remarks on future research directions.

2. The basic minimum degree ordering.

2.1. Description of the minimum degree algorithm. We assume that the reader is familiar with basic graph theory notions, and the correspondence between undirected graphs and the structure of symmetric matrices. See, for example, Chapter 3 of [16]. Let G be an undirected graph and v a node of G. Throughout this paper, we shall use the notation $\mathrm{Adj}_G(v)$ to refer to the set of nodes adjacent to v in G. The degree of the node v in G will be denoted by $\mathrm{degree}_G(v)$, which is simply $|\mathrm{Adj}_G(v)|$.

The basic minimum degree ordering can be best described in terms of *elimination graphs*. Following Rose [29], we use the notation G_v to represent the elimination graph obtained after the elimination of the node v from the graph G. The graph G_v can be obtained by deleting the node v and its incident edges from G and then adding edges to make the nodes that were adjacent to v into a clique.

The basic algorithm is as shown in Fig.1, and captures the main idea of the ordering strategy. Implicitly assumed in this description is a scheme to represent the elimination graphs and to transform them. This allows the selection of a minimum degree node in the new elimination graph. Based on the transformation rule, we note that if a node v is not adjacent to y in G,

$$\mathrm{Adj}_{G_y}(v) = \mathrm{Adj}_G(v).$$

However, if $v \in \mathrm{Adj}_G(y)$, then we have

$$\mathrm{Adj}_{G_y}(v) = (\mathrm{Adj}_G(y) \cup \mathrm{Adj}_G(v)) - \{v, y\}.$$

Therefore, the degree of a node may change after the elimination graph transformation due to the deletion of edges incident to y and the possible addition of new

```
begin
    G ← given symmetric graph
    while G ≠ φ do
    begin
        select a node y of minimum degree in G and order y next
        G ← G_y
    end
end
```

FIG. 1. *Basic minimum degree algorithm.*

edges joining nodes adjacent to y. In other words, $\text{degree}_G(v)$ may be quite different from $\text{degree}_{G_y}(v)$. The following are simple observations on the relation between $\text{degree}_{G_y}(v)$ and $\text{degree}_G(v)$.

OBSERVATION 1. *If v is not adjacent to y, then* $\text{degree}_{G_y}(v) = \text{degree}_G(v)$.

OBSERVATION 2. *If v is adjacent to y, then*

$$\text{degree}_{G_y}(v) \geq \text{degree}_G(v) - 1 \geq \text{degree}_G(y) - 1.$$

The basic algorithm shown in Fig.1 is not completely specified, since at the node selection stage there may be several nodes of minimum degree. These "ties" must be resolved in some manner, usually arbitrarily. Fig.2 gives two minimum degree orderings by applying this algorithm to the 5×5 regular grid model problem (9-point difference), where the ties were resolved in different ways. Note that nodes associated with each small square are pairwise connected, and for simplicity, the grid lines in Fig.2 (and in later figures) are omitted. Different orderings can be obtained due to the freedom in the selection of minimum degree nodes. The example in Fig.2 is designed to illustrate this point.

4	8	11	7	3		4	7	21	12	3
20	21	22	16	15		11	16	22	15	8
10	14	23	13	9		20	19	23	18	17
6	18	24	25	5		6	14	24	13	9
2	17	12	19	1		2	10	25	5	1

FIG. 2. *Two minimum degree orderings on a 5×5 grid.*

Often, the quality of the resulting minimum degree ordering depends crucially on the "correct" node selection from the set of minimum degree nodes. We use the term *tie-breaking strategy* to refer to any scheme that provides a means of choosing a node from such a set. Some tie-breaking strategies will be discussed in later sections.

2.2. Quality of the minimum degree ordering. It is easy to see that a minimum degree ordering on a tree structure will be one that is a perfect ordering (with

no fill) and hence, a *minimum fill* ordering (that is, one with the least number of fills). But for general graphs, a minimum degree ordering may not be a minimum fill ordering. In [29], Rose provides an example to show this. Indeed, his example is a graph associated with a perfect elimination matrix; by definition, such matrices can always be ordered so that they suffer no fill. However, a minimum degree ordering of this example will create some fill from elimination. Recently, Hempel [24] gives a graph example where minimum degree can generate fill that is more than a constant factor greater than that given by a minimum fill ordering.

If we consider the $k \times k$ regular grid problem, a nested dissection ordering of it [14] is known to generate fills of no more than $O(k^2 \log_2 k)$ and is therefore optimal in the order of magnitude sense. Loosely speaking, it can also be regarded as a minimum degree ordering with a special way of selecting nodes of minimum degree. In other words, a special type of minimum degree ordering on the grid problem can be optimal (order of magnitude sense). Unfortunately, not all minimum degree orderings for the grid are optimal. More will be said about this later when we consider tie-breaking strategies.

3. Development of improvements to the minimum degree algorithm. It is apparent from the description of basic algorithm in Fig.1 that the transformation of the elimination graph

$$G \leftarrow G_y$$

is central to the implementation of this algorithm. In general, this elimination step creates fill among nodes adjacent to y. It is therefore necessary to have an efficient representation of the resulting elimination graph to accommodate additional fill (either implicitly or explicitly). Moreover, the degrees of nodes adjacent to y may change, so that a re-calculation of their degrees is required in preparation for the next node selection step. It has been recognized by researchers that this "degree update" is the most time consuming step of the entire algorithm.

In this section, we discuss various methods that have been developed to improve the performance of the basic minimum degree algorithm. They are presented in the sequence in which they appeared in the literature, and in each instance, to the best of our knowledge, we have attributed the techniques to those who first suggested them.

3.1. Mass elimination. In their study of the minimum degree algorithm on finite-element problems, George and McIntyre [21] observed that in the elimination of a node y of minimum degree, often there is a subset of nodes adjacent to y that can be eliminated immediately after y. This happens especially often in later stages of the elimination. The next theorem contains the theoretical basis of their observation, and we leave its proof to the readers.

THEOREM 3.1. *If y is selected as the minimum degree node in the graph G, then the subset*

$$Y = \{z \in \mathrm{Adj}_G(y) \mid \mathrm{degree}_{G_y}(z) = \mathrm{degree}_G(y) - 1\}$$

can be selected next (in any order) in the minimum degree algorithm.

Theorem 3.1 allows us to avoid some graph transformations and degree update steps, since it provides a set of nodes of minimum degree that may be selected next. Instead of having to perform the transformation $G \leftarrow G_y$ and the degree update, we can eliminate nodes in $Y \cup \{y\}$ simultaneously. This implies that the elimination graph transformation and degree update need only be performed once for the whole

set instead of $|Y \cup \{y\}|$ times. This will also save node selection time. We describe how to identify Y in §3.2.

To illustrate mass elimination, consider the actual application of the minimum degree algorithm to the 5×5 grid using the 9-point connectivity. Fig.3 shows an intermediate stage after 14 nodes have been eliminated during the ordering (compared with the left ordering in Fig. 2). Nodes marked with • have already been eliminated. It can be verified that the set of nodes adjacent to $y = 15$ in the current elimination graph is $\{16, 19, 20, 21, 22, 23, 24, 25\}$. (They are those uneliminated nodes that can be reached from the node y via eliminated nodes.) Therefore, the node $y = 15$ has minimum degree 8 and the node $z = 16$ satisfies the condition in Theorem 3.1. The nodes $\{y, z\}$ can be eliminated together.

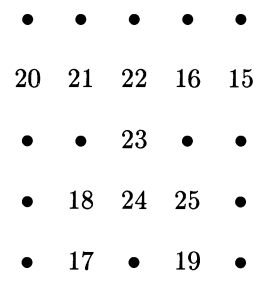

FIG. 3. *An intermediate stage of the minimum degree ordering.*

3.2. Indistinguishable nodes. The condition in Theorem 3.1 for mass elimination has a simple equivalent property. As before, let y be a node of minimum degree in G and $z \in \mathrm{Adj}_G(y)$. By the nature of the elimination graph transformation,

$$\mathrm{Adj}_{G_y}(z) = (\mathrm{Adj}_G(y) \cup \mathrm{Adj}_G(z)) - \{y, z\}.$$

THEOREM 3.2. *Let* $z \in \mathrm{Adj}_G(y)$. *Then* $\mathrm{degree}_{G_y}(z) = \mathrm{degree}_G(y) - 1$ *if and only if* $\mathrm{Adj}_G(z) \cup \{z\} = \mathrm{Adj}_G(y) \cup \{y\}$.

Proof. "*If*" *part*: If $\mathrm{Adj}_G(z) \cup \{z\} = \mathrm{Adj}_G(y) \cup \{y\}$ then

$$\mathrm{degree}_{G_y}(z) = |(\mathrm{Adj}_G(y) \cup \mathrm{Adj}_G(z)) - \{y, z\}| = |\mathrm{Adj}_G(y) - \{z\}| = \mathrm{degree}_G(y) - 1.$$

"*Only if*" *part*: Let $\mathrm{degree}_{G_y}(z) = \mathrm{degree}_G(y) - 1$. We first show that $\mathrm{Adj}_G(z) \cup \{z\} \subseteq \mathrm{Adj}_G(y) \cup \{y\}$. Assume for contradiction that there exists a node $x \in \mathrm{Adj}_G(z) - \mathrm{Adj}_G(y)$. Then we have

$$\mathrm{Adj}_{G_y}(z) = (\mathrm{Adj}_G(y) \cup \mathrm{Adj}_G(z)) - \{y, z\} \supseteq (\mathrm{Adj}_G(y) - \{z\}) \cup \{x\}.$$

This implies that $\mathrm{degree}_{G_y}(z) > \mathrm{degree}_G(y) - 1$, and hence proves the claim that

$$\mathrm{Adj}_G(z) \cup \{z\} \subseteq \mathrm{Adj}_G(y) \cup \{y\}.$$

However, y is selected to be a node of minimum degree so that the two sets must be the same. □

Together with Theorem 3.1, we note that the equivalent property in Theorem 3.2 can be used to identify nodes for mass elimination. We now show that it can also

help in reducing the number of degree updates. Formally, we say that two nodes u and v are *indistinguishable* in G [16] if

$$\mathrm{Adj}_G(u) \cup \{u\} = \mathrm{Adj}_G(v) \cup \{v\}.$$

Sometimes the set $\mathrm{Adj}_G(u) \cup \{u\}$ is referred to as the *neighborhood set* of u (see for example the book by Golumbic [23]). So, two nodes with identical neighborhood sets are indistinguishable. As shown in [16], this property is preserved under elimination graph transformation.

THEOREM 3.3. *If two nodes are indistinguishable in G, they remain indistinguishable in G_y.*

It is obvious that if two nodes are indistinguishable, their degrees must be the same. Theorem 3.3 says that if two nodes become indistinguishable at some stage of elimination, they will have identical neighborhood sets (and hence the same degree) thereafter. Furthermore, Theorems 3.1 and 3.2 imply that they can be eliminated together whenever one of them is picked for elimination.

Therefore, as far as the minimum degree algorithm is concerned, indistinguishable nodes can be merged together and treated as one. In this way, we need only to consider one *representative* from each group of indistinguishable nodes. In the literature, these representative nodes have been referred to as *supernodes*, *supervariables* [10], and *prototype nodes* [11].

The advantage of using indistinguishable nodes should be clear. We need to perform the degree update only on the representative nodes. This reduces the operating size of the current elimination graph in terms of both nodes and edges.

To illustrate this notion, consider the stage as given in Fig.3. The remaining 11 nodes can be divided into 5 groups of indistinguishable nodes: {15, 16}, {17, 18}, {19, 25}, {20, 21}, and {22, 23, 24}. In effect, we are now working on a graph with five (super) nodes rather than with eleven nodes.

3.3. Representation of elimination graphs. As noted earlier, a compact and efficient representation of the sequence of elimination graphs is crucial to the overall performance of the minimum degree ordering algorithm. The elimination graph transformation "$G \leftarrow G_y$" involves both deletion and addition of edges. In other words, the number of edges in the elimination graph may either increase or decrease depending on how the nodes adjacent to y are connected. The representational scheme must be able to accommodate such changes.

Speelpenning [31] used a *generalized element* approach to consider the elimination process on mesh problems arising from the finite-element method. Using this view, we can interpret each elimination graph as a collection of cliques or generalized elements. Indeed, the original graph with $|E|$ edges can be regarded as one with $|E|$ cliques, each consisting of two nodes (or equivalently an edge).

This not only gives a conceptually different view of the elimination process, but also provides a compact scheme to represent the elimination graphs. The advantage of this representation in terms of storage is based on the following observations.

Let $\{K_1, K_2, \cdots, K_q\}$ be the set of cliques for the current graph G. Assume that y is a node of minimum degree selected for elimination. Let $\{K_{s_1}, \cdots, K_{s_t}\}$ be the subset of cliques to which y belongs. Transforming the elimination graph requires two steps.

(1) Remove the cliques $K_{s_1}, \cdots, K_{s_t}$ from $\{K_1, K_2, \cdots, K_q\}$.

(2) Add the new clique $K = (K_{s_1} \cup \cdots \cup K_{s_t}) - \{y\}$ into the clique set.

The next theorem is obvious and is the basis of the advantage of this generalized element representation scheme.

THEOREM 3.4. $|K| < \sum_{i=1}^{t} |K_{s_i}|$.

This theorem says that in using the generalized element approach for the representation of the sequence of elimination graphs, the amount of storage required will *never* exceed the amount needed to represent the original graph. The storage requirement to carry out the minimum degree algorithm is therefore known beforehand in spite of the rather dynamic nature of the elimination process. An economical representation of the structures has an added advantage of reducing processor time in their manipulations.

The use of the generalized element storage scheme appears in the Harwell code MA27 [9], the Waterloo Sparse Matrix Package (SPARSPAK) [20], and the Yale Sparse Matrix Package (YSMP) [11]. This representation technique has been referred to as a generalized element model, superelements [10], [13], and the quotient graph model [17].

Apparently, the developers of YSMP were the first to adopt this approach. Both the Harwell code and YSMP use linked lists to implement the set of generalized elements. In the SPARSPAK implementation, linked lists of subarrays are used to take advantage of the initial data representation of the structure of the given matrix.

We again use the example shown in Fig.3 to illustrate this generalized element approach. At this point, the elimination graph has four cliques: {15, 16, 19, 22, 23, 24, 25}, {15, 16, 20, 21, 22}, {17, 18, 20, 21, 22, 23, 24}, and {17, 18, 19, 24, 25}. Note that each clique is determined by the adjacent set of a maximal connected set of eliminated nodes. Upon the elimination of the nodes {15, 16} (by mass elimination), there will be three cliques left: {19, 20, 21, 22, 23, 24, 25}, {17, 18, 20, 21, 22, 23, 24}, and {17, 18, 19, 24, 25}. Notice that the amount of storage required is less than before.

3.4. Incomplete degree update. The notion of indistinguishable nodes in §3.2 can be regarded as an extension of the condition used for mass elimination. The use of indistinguishable nodes helps not only at elimination but also at the time of degree update. By merging indistinguishable nodes together, we need only to recompute the degrees of the representatives. Here, we consider yet a further generalization which was first used in the YSMP implementation [11]. This technique speeds up the minimum degree algorithm by avoiding the degree computation for nodes that are known *not* to be minimum degree.

Following [11], given two nodes u and v in the graph G, the node v is said to be *outmatched* by u if

$$\mathrm{Adj}_G(u) \cup \{u\} \subseteq \mathrm{Adj}_G(v) \cup \{v\}.$$

A direct consequence of this condition is that $\mathrm{degree}_G(u) \leq \mathrm{degree}_G(v)$. More importantly, this property is preserved under elimination graph transformation [26].

THEOREM 3.5. *If the node v is outmatched by u in G, it is also outmatched by u in the graph G_y.*

COROLLARY 3.6. *If a node v becomes outmatched by u in the elimination process, the node u can be eliminated before v in the minimum degree ordering algorithm.*

An important consequence of Corollary 3.6 is that if v becomes outmatched by u at some stage during the elimination, it is not necessary to update the degree of v until the node u has been eliminated. In the case of the indistinguishable property, degree updates are avoided since some degrees can be deduced from the representatives.

Here, using the outmatched property, degree updates are skipped for those nodes which will not participate in the next round of minimum degree selection.

Again using the example in Fig.3, we note that the nodes in {22,23,24} are all outmatched by each of the remaining nodes, namely, {15,16,17,18,19,20,21,25}. In other words, it is unnecessary to recompute the degrees of nodes 22, 23, and 24 until all others have been eliminated. This is indeed a very powerful technique in improving the performance of the ordering algorithm.

3.5. Element absorption. The technique of *element absorption* was first proposed by Duff and Reid in [10]. This idea can be most easily explained in terms of the generalized element approach described in §3.3. Recall that this approach provides a compact representation of elimination graphs by storing cliques rather than edges. Let $\{K_1, K_2, \cdots, K_q\}$ be the set of cliques for the current graph G.

THEOREM 3.7. *If $K_s \subseteq K_t$ for some s and t, then the graph G can be represented by the clique set $\{K_1, K_2, \cdots, K_q\} - K_s$.*

In Theorem 3.7, the clique K_s can be viewed as being *absorbed* by the clique K_t. The technique of element absorption is to remove any detected clique redundancy in the representation. Reducing the number of cliques will speed up the ordering process since less overhead in manipulating the set of cliques is involved.

A good example to illustrate element absorption is the star graph (see [16]) and its variants. Consider a graph with eight nodes; its structure is given by the set of four cliques: {1, 5}, {2, 5, 6}, {3, 5, 6, 7}, and {4, 5, 6, 7, 8}. After the elimination of the nodes 1, 2 and 3 (in that order by the minimum degree algorithm), the technique of element absorption reduces the number of cliques to only one: namely, {4, 5, 6, 7, 8}. The new cliques {5}, {5, 6}, and {5, 6, 7} formed after the elimination of nodes 1, 2 and 3 can all be absorbed into the clique {4, 5, 6, 7, 8}. For larger versions of this example a great reduction in degree update time can be achieved.

3.6. Multiple elimination. The incomplete degree update described in §3.4 can also be regarded as a technique using *delayed* degree update. The enhancement of *multiple elimination* proposed by Liu [26] extends this idea of delaying the degree update. Instead of performing a degree update step after each minimum degree node selection and elimination graph transformation, the technique of multiple elimination postpones the degree update step to a later stage. The readers should be cautioned that multiple elimination is almost an opposite notion to mass elimination discussed in §3.1.

The basis for multiple elimination is quite simple. It makes use of the observation that in the elimination of the node y from the graph G, the structure associated with nodes *not* in $\mathrm{Adj}_G(y)$ remains unchanged. The idea is to suspend the degree update for nodes in $\mathrm{Adj}_G(y)$ and select a node with the same degree as y in the remaining subgraph $G-(\mathrm{Adj}_G(y)\cup\{y\})$. This process is repeated until there are no nodes in the remaining subgraph with degree $\mathrm{degree}_G(y)$. A degree update step is then performed.

In essence, before each degree update step, an *independent* set of nodes with minimum degree is selected. The fact that the set is independent allows the delay of degree update until the entire set is determined and eliminated. As noted in [26], this modification does not always result in a genuine minimum degree ordering, but it very rarely produces an ordering that is inferior to that provided by the "true" minimum degree algorithm. Indeed, the quality of the resulting ordering is maintained while the execution time of the ordering is reduced.

To understand multiple elimination, we consider the 5×5 grid in Fig.4 (with nine-point connectivity), where eight nodes have been eliminated. As before, nodes

marked with • have already been eliminated. At this stage, the minimum degree is 5. The set {9, 10, 11, 12} forms an independent set with degree 5. (Note that {15, 17, 19, 20} and {12, 15, 20} are two other different independent sets with degree 5.) Eliminating them altogether before a degree update helps to reduce the update time. We need to update the degree of nodes in {13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25} only once after the elimination of {9, 10, 11, 12}. On the other hand, using the conventional eliminate-update approach, each node in {15, 16, 17, 18, 19, 20, 21, 25} would be updated twice during the course of eliminating the nodes 9, 10, 11, 12 in four separate steps.

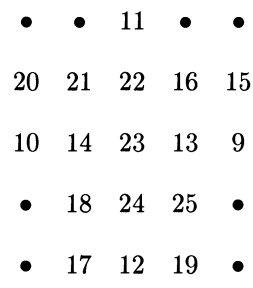

FIG. 4. *An intermediate stage of the minimum degree ordering.*

3.7. External degree. In [26], Liu suggests the use of *external* degree instead of true degree in the minimum degree algorithm. In the conventional scheme, the degree used is the number of adjacent nodes in the current elimination graph. By the external degree of a node, we mean the number of nodes adjacent to it that are not indistinguishable from itself.

The motivation comes from the underlying reason for the success of the minimum degree ordering in terms of fill reduction. Eliminating a node of minimum degree implies the formation of the smallest possible clique due to elimination. Since we are using the technique of mass elimination (§3.1), the size of the resulting clique after mass elimination is the same as the external degree of nodes eliminated by the mass elimination step. For example, in Fig.3, since nodes 15 and 16 are indistinguishable, the external degree of node 15 is 7 (and its true degree is 8). Experimental results in [26] show that using external degree rather than true degree yields a reduction in the number of nonzeros in the factor matrix of 3 to 7 percent for the grid model problem.

4. Timing results of various improvement techniques. So far in the literature, there has been no formal analysis of the time complexity of the minimum degree ordering algorithm. Such an analysis appears to be very difficult and will be further complicated by the various enhancements to the basic scheme that have been described in the previous section. As partial compensation for the lack of a theoretical analysis, in this section we provide some timing statistics for the various improvement techniques. Our objective is to illustrate to the reader the very substantial gains in efficiency that these enhancements provide. We use the minimum degree ordering algorithm developed by Liu [26], which is now incorporated in SPARSPAK [20]. In order to illustrate the effectiveness of the various techniques, we mask off each improvement strategy in the basic code and obtain timing results.

In all the experiments, the internal data structure to represent elimination graphs is the quotient graph scheme described in [17]. Fixing the data structure allows us to gauge the importance of each improvement technique on the algorithm. Using a different technique for representing the elimination graphs, such as generalized elements or linked lists, would undoubtedly change the absolute times reported. However, we do not think the relative ranking of the various techniques, in terms of their effectiveness in reducing execution time, would be changed significantly.

We choose as our test problem the 180×180 regular grid model problem corresponding to the use of a nine-point difference operator. For this matrix example, the number of unknowns is 32,400 and the number of off-diagonal nonzeros in the original matrix is 128,522. We first note that for this particular example, the quality of the resulting minimum degree ordering using external degree is better than that with true degree in terms of fill reduction. The numbers of off-diagonal nonzeros in the Cholesky factors are listed below.

External Degree	1,180,771
True Degree	1,273,517

To make a uniform comparison, we used external degree for all the experiments reported hereafter. In Table 1 we have tabulated the number of off-diagonal nonzeros in the Cholesky factors and the amount of time in CPU seconds on a SUN 3/50 for each version of the minimum degree algorithm. In [3] and [4] Duff provides some timing statistics of various minimum degree ordering implementations during the period 1970 to 1981. Our timing results here are given in correspondence with each enhancement technique and can be regarded as supplements to Duff's statistics.

TABLE 1
Statistics on various versions of minimum degree algorithm.

Version	Minimum Degree Algorithm	Off-diagonal Factor Nonz	Ordering Time
Md#1	Final minimum degree	1,180,771	43.90
Md#2	Md#1 without multiple elimination	1,374,837	57.38
Md#3	Md#2 without element absorption	1,374,837	56.00
Md#4	Md#3 without incomplete deg update	1,374,837	83.26
Md#5	Md#4 without indistinguishable nodes	1,307,969	183.26
Md#6	Md#5 without mass elimination	1,307,969	2289.44

It should be emphasized that the effectiveness of each technique in reducing the ordering time is problem dependent. Here, we have simply used the regular grid as an example for illustration. However, the relative effectiveness of each technique in Table 1 is quite representative of other problems tested. The version Md#1 is the current version of the minimum degree ordering routine in SPARSPAK. All techniques are removed from Md#1 one by one to eventually give Md#6. Therefore, Md#6 represents the basic scheme with no enhancement other than the use of clique representation (§3.3). That is, it is almost the same as the original algorithm as described by Tinney and Walker [32] or by Rose [29], except that external degree is used. We did not optimize the codes for versions Md#2-Md#6, but rather simply commented out the part associated with the enhancement in version Md#1. We do not believe that such optimization would change the numbers in any significant way.

From Table 1, there is an overall reduction of ordering time by a factor of more than 50. Indeed, the savings have not even included the possible reduction due to the use of some generalized element representation.

Another aspect worth mentioning is the number of off-diagonal nonzeros in the Cholesky factors. For this example, the final version MD#1 produces an ordering with 10 percent fewer nonzeros than that provided by those with some of the enhancements removed. This suggests that Liu's multiple elimination technique can have an important effect on the *quality* of the ordering produced by the algorithm, in addition to reducing its the execution time. We shall return to this point in the next section.

A final point to note is that the reduction in the number of nonzeros in the factor matrix of 10 percent will lead to a much larger relative reduction in the numerical factorization time. We provide some experiments in the next section to illustrate this fact.

5. On tie-breaking strategies.

5.1. An example: regular grid. Let A be a matrix associated with the $k \times k$ regular grid. It is known that the best ordering on A will yield a Cholesky factor with at least $O(k^2 \log_2 k)$ nonzeros [25]. Furthermore, it is shown by George [14] that a nested dissection ordering on the grid is one such optimal ordering (optimal in the order of magnitude sense). Nested dissection can be viewed as a minimum degree ordering with a perfect tie-breaking strategy for the grid (strictly speaking, only for the torus graph).

On the other hand, it has been a long-outstanding question as to how bad a minimum degree ordering can be for the grid problem. Or, to phrase the question differently, can the minimum degree algorithm produce an ordering for A for which the resulting Cholesky factor has more than $O(k^2 \log_2 k)$ nonzeros? This problem can be interpreted as a search for an imperfect tie-breaking strategy for the minimum degree algorithm on the grid. Recently, Berman and Schnitger have devised one such tie-breaking scheme [1], yielding an ordering for which the Cholesky factor is shown to have $O(k^{2 \log_3 4})$ nonzeros.

For the regular grid, there is therefore a wide class of minimum degree orderings, ranging from the optimal nested dissection ordering to the ordering by Berman and Schnitger. This shows the significance of choosing an effective tie-breaking strategy for the minimum degree algorithm. In the next section, we further illustrate its significance by considering a number of existing implementations of the minimum degree ordering.

5.2. Current implementations: random tie-breaking. Perhaps the most well-known implementations of the minimum degree algorithm are those found in the Harwell MA27 code [9], the Waterloo SPARSPAK package [20], and the Yale sparse matrix package (YSMP) [11]. They incorporate most of the enhancements described in §3. It is not our objective here to compare the relative execution speed of these three implementations. Instead, we want to illustrate the practical importance of having an effective tie-breaking strategy.

In some sense, Liu's multiple elimination technique [26] provides a limited form of tie-breaking for the minimum degree algorithm. It forces the algorithm to search for a maximal set of independent minimum degree nodes before a degree update is performed. Other than this, none of the three implementations (Harwell, SPARSPAK, YSMP) has incorporated any tie-breaking strategy. The selection of the next mini-

mum degree node from the candidate set is, in effect, *random*, since the initial ordering essentially determines the way ties are resolved.

To illustrate this, we again use the 180 × 180 grid problem. A random permutation is applied to rearrange the rows and columns before passing the matrix to the minimum degree ordering routine. In other words, the algorithm is ordering a different form of the same matrix problem. This is repeated ten times and we have tabulated in Table 2 the best and the worst orderings for all three implementations. As a basis for comparison, we have also included the case when the matrix problem is presented in a row-by-row initial ordering. In the table, "Factor Nonz" is the number of off-diagonal nonzeros in the resulting Cholesky factor (in thousands), and "Factor Opns." is the number of multiplicative operations to perform the factorization (in millions of operations).

TABLE 2
Best and worst orderings from ten runs on the 180 × 180 *grid with different initial orderings.*

Minimum Degree Code	Row-by-row Initial Ordering			Ten Random Initial Orderings					
				Best Min Deg Ordering			Worst Min Deg Ordering		
	Order Time	Factor Nonz	Factor Opns.	Order Time	Factor Nonz	Factor Opns.	Order Time	Factor Nonz	Factor Opns.
Harwell	115.2	1,605K	132.0M	116.2	1,731K	144.6M	116.2	1,821K	174.9M
Sparspak	43.9	1,180K	62.2M	58.5	1,567K	118.0M	58.8	1,651K	139.7M
YSMP	49.0	1,241K	71.9M	52.9	1,710K	139.5M	53.3	1,868K	191.3M

Results in Table 2 demonstrate that a random form of tie-breaking (as basically used by the three implementations) is clearly not enough to ensure a good quality ordering from a practical standpoint. Indeed, among the three YSMP minimum degree orderings tabulated, there is a difference of about 50 percent in terms of factor matrix nonzeros and over 160 percent in terms of factorization operation counts. The difference is even more dramatic among all the orderings in Table 2. The worst one requires more than *three* times as many arithmetic operations as are required by the best ordering. Note that they are all variants of the same basic minimum degree algorithm on the same matrix problem. This illustrates the practical importance of an effective tie-breaking strategy for the minimum degree algorithm.

"Order Time" in Table 2 refers to the time required to determine a minimum degree ordering from each of the three software packages. Care was taken to exclude any pre-processing time to transform input matrix structure and post-processing time to manipulate the resulting ordering. The results in the table illustrate that the quality of the ordering can have a significant impact on the ordering time.

It should be emphasized that for large problems the ordering time used by the minimum degree algorithm is insignificant compared to the actual numerical factorization time. Even a small percentage reduction in factorization operations would easily offset a relatively large increase in ordering time that might result from adding sophisticated tie-breaking strategies to the algorithm. To illustrate this, we actually performed the numerical factorization and triangular solution of the 180 × 180 grid problem using the best ordering from Table 2 (SPARSPAK minimum degree with row-by-row initial ordering). On a SUN 3/50 the numerical factorization required 1847.08 CPU seconds (about 62.2 million arithmetic operations) and the forward and backward solution required 78.52 seconds. The ordering time of 43.9 seconds is a small fraction of the overall cost.

For the sake of comparison, we also solved the same linear system using the worst minimum degree ordering from SPARSPAK. The numerical factorization required

4529.24 CPU seconds for the 139.7 million operations. This illustrates the importance of devising an effective tie-breaking strategy for reducing fill (and operations). The potential pay-off in the subsequent numerical phase can be tremendous.

5.3. Tie-breaking based on preordering. The experimental results in §5.2 suggest that significant gain in terms of both storage and arithmetic operations can be achieved if more care is exercised in selecting nodes of minimum degree. One idea is to fix the initial ordering by rearranging the adjacency structure before passing it to the ordering routine. Of course, the preordering strategy itself should not be sensitive to the initial ordering.

Much more research is required to understand the impact of preordering. However, we shall provide some preliminary experimental results to indicate that this approach can be promising. Our results will be based on the SPARSPAK minimum degree ordering routine. From Table 2, the row-by-row initial ordering appears to be better than a random initial ordering on the 180×180 grid for SPARSPAK.

This suggests the use of some form of profile ordering as a preordering method. We used the *reverse Cuthill–McKee* ordering as implemented in [16] for our purpose. The overall strategy can then be viewed as

$$A \xrightarrow{RCM} \tilde{A} = P_r A P_r^T \xrightarrow{MD} P \tilde{A} P^T$$

where P_r is the reverse Cuthill–McKee (RCM) ordering on the given matrix A, and P is the minimum degree (MD) ordering on the permuted matrix $\tilde{A} = P_r A P_r^T$.

This initial reordering has a remarkable effect on the 180×180 grid problem. In this case, the resulting minimum degree ordering is the same irrespective of how we randomly permute the original matrix A. The intermediate reverse Cuthill–McKee ordering removes the randomness in the row/column arrangement before presenting the matrix to the minimum degree algorithm. For this problem, the number of off-diagonal nonzeros then becomes 1,205,768 and the number of factorization operation count is about 67.6 million arithmetic operations (irrespective of how the matrix is initially ordered). This compares quite favorably with the best minimum degree ordering in Table 2.

6. Odds and ends.

6.1. Implementational issues. The actual performance of a minimum degree code depends quite heavily on its implementation. Here we simply provide a list of relatively important issues that one encounters during the implementation of a minimum degree algorithm with the enhancements described in §3:

(a) how to represent degrees,
(b) how to implement elimination graphs using the generalized element model,
(c) how to detect and store indistinguishable nodes,
(d) how to detect and store outmatched nodes,
(e) how to recompute degrees.

These design issues are crucial to the overall performance of the ordering algorithm. For more detailed information, readers can consult the Harwell MA27 code, the Waterloo SPARSPAK and the Yale YSMP programs to see how these issues are dealt with in each of these implementations.

It may be worthwhile to point out one major difference among these three implementations. Both MA27 and YSMP use linked lists in the representation of the sequence of elimination graphs for the generalized element model. By Theorem 3.4,

the amount of storage required will never exceed the storage requirement for the linked list representation of the original given graph. On the other hand, SPARSPAK uses a quotient graph storage scheme as described in [17], whereby the ordering is performed "in-place" within the given adjacency structure representation. Higher overhead in manipulating the structure is required in exchange for a reduction of approximately 50 percent in storage requirements.

6.2. Adaptation to some applications. The minimum degree algorithm (with the enhancements) is a general purpose ordering scheme that can be applied to any symmetric matrix structure (or undirected graph). It is interesting to note that in some situations, it may be advantageous to tailor the general minimum degree algorithm to specific applications. In this section, we shall mention two such applications.

Consider computing the least squares solution of the large $m \times n$ sparse system $Ax \approx b$, where $m \geq n$ and A has full rank. It is well-known that such a problem can be solved via the orthogonal decomposition of the matrix A into $Q \begin{pmatrix} R \\ 0 \end{pmatrix}$, where Q is an orthogonal matrix and R is upper triangular. The solution to the least squares problem above is known to be mathematically equivalent to solving the normal equations $A^T Ax = A^T b$. Based on this connection, George and Heath [15] provide an overall scheme to solve the sparse system $Ax \approx b$. In particular, they suggest solving the equivalent system $(AP)(P^T x) \approx b$, where P is a minimum degree ordering of the symmetric matrix $A^T A$. This is motivated by the observation that the factor R is mathematically the same as the Cholesky factor of the matrix $A^T A$ so that using a minimum degree ordering of $A^T A$ will result in a sparse factor R.

To use the minimum degree algorithm in this setting, one can form the structure of the matrix $M = A^T A$ and then apply the ordering algorithm to M. A natural question is whether one can adapt the algorithm to determine a minimum degree ordering for M directly from the structure of A. We first note that the structure of each row of A corresponds to a clique in the graph associated with M. Therefore we can use the generalized element approach to represent M as a sequence of cliques consisting of the rows of A. Since this is an intermediate data structure used by the minimum degree algorithm, one can easily adapt the algorithm by allowing a set of cliques as the initial graph representation. This will save the extra step of forming the graph structure of the matrix $M = A^T A$ explicitly. Adapting the algorithm so that it efficiently finds an ordering of M, given A, is also important in the context of solving sparse indefinite systems using Gaussian elimination with partial pivoting [19], [22].

A different application is in the numerical solution of sparse linear systems arising from the *finite-element* method. The structure of the matrix from such systems is governed by the underlying finite-element mesh. The nodes associated with each element form a clique.

In practice, the matrix structure is often provided in terms of the elements; that is, it is given in the form of node membership in each element. Of course, the adjacency structure of the matrix can be generated based on this element-node membership information and then passed to the minimum degree algorithm. An alternative is to perform the ordering directly on the element-node information without forming the adjacency structure.

It is interesting to note that this is exactly the same situation as in the case of the sparse least squares problem. Consider the structure of the matrix B with each row corresponding to an element of the finite-element mesh and each column to a node. The structure of each row of B is given by the node membership of the associated

element. Then, it can be verified that the structure of the finite-element matrix is the same as the structure of B^TB. Therefore, a minimum degree ordering of the finite-element matrix can be obtained by performing the ordering on B in the same manner as the sparse least squares problem.

This approach has another important advantage. In the use of higher order elements, the representation of the finite-element matrix structure implicitly by B usually requires significantly less storage than the explicit adjacency structure representation. For example, consider the $k \times k$ grid where each square/element is associated with nine variables/nodes (with one interior node, one at each corner, and one along each side). There are a total of $(2k-1)^2$ nodes, $(k-1)^2$ of which are interior nodes, k^2 corner nodes, and the rest edge nodes. Using the adjacency structure representation, we need storage amounting to about $60k^2$ items ($8k^2$ for interior nodes, $24k^2$ for corner nodes, and $28k^2$ for edge nodes). On the other hand, the storage requirement for the structure of B is only $18k^2$ ($9k^2$ for element-node information, and $9k^2$ for node-element relation).

7. Future research directions. The various enhancements to the basic minimum degree ordering algorithm over the past fifteen years have made the algorithm a truly practical approximate solution to the NP-complete minimum fill ordering problem [33]. The ordering time has been drastically reduced, as our example with an over 50-fold reduction in ordering time in Table 1 illustrates. This makes the ordering time a small fraction of the overall solution time for a given large sparse problem.

In the authors' opinion, the next significant advance of this algorithm will probably be the development of effective and practical tie-breaking strategies in the selection of minimum degree nodes. Given the possible variation in the quality of the orderings as exemplified by experimental results in Table 2, significant reduction in factorization storage requirement can be attained by having a good tie-breaking strategy. More importantly, reducing the number of nonzeros in the Cholesky factor implies an even more substantial reduction in operation counts (and hence factorization time). Indeed, the potential gain makes it apparently very worthwhile to invest more time in the ordering phase to determine a better ordering.

In §5, we have offered a "partial" solution to the tie-breaking problem based on preordering. The reverse Cuthill–McKee profile ordering is used to pre-order a matrix before the minimum degree algorithm is applied. Some success in "stabilizing" the overall ordering phase is reported in §5.

Another promising avenue for tie-breaking strategies is based on the notion of independent sets. As pointed out in §5, Liu's multiple elimination technique [26] provides a form of tie-breaking by selecting a *maximal independent set* of nodes of minimum degree. An apparently better strategy would be to select at each stage a *maximum* independent set of minimum degree nodes. In view of the NP-completeness of this maximum independent set problem, we must instead rely on practical approximate solutions to find a "nearly" maximum independent set from the set of minimum degree nodes. Some of these ideas are currently under investigation by the authors.

Another related problem of practical interest is to generate a fill-reducing ordering appropriate for *parallel elimination.* Some recent progress in this area can be found in [7], [27]. In terms of the minimum degree algorithm, we are interested in finding a suitable tie-breaking strategy so that the resulting minimum degree ordering is appropriate for parallel elimination.

We have tried the following strategy. At a given stage of the ordering, let S be the set of nodes already eliminated. If there is more than one node in the uneliminated

set with the current minimum degree, we select a node y with the smallest connected component containing y in the subgraph of $S \cup \{y\}$. Intuitively, we try to enlarge the smaller components of the subgraph defined by S within the guideline of the minimum degree algorithm. Experiments on this (and other slight variants of this) tie-breaking strategy were tried. However, in terms of parallelism, the resulting orderings are not significantly different from those reported in [27].

REFERENCES

[1] P. BERMAN AND G. SCHNITGER, *On the performance of the minimal degree heuristic for Gaussian elimination*, Dept. of Computer Science, Pennsylvania State University, 1987.

[2] I. S. DUFF, *MA28 - A set of* FORTRAN *subroutines for sparse unsymmetric linear equations*, Tech. Rept. AERE R-8730, Harwell, U.K., 1977.

[3] ———, *Research directions in sparse matrix computations*, in Studies in Numerical Analysis, Volume 24, G. Golub, ed., Mathematical Association of America, 1984, pp. 83–139.

[4] ———, *A sparse future*, in Sparse Matrices and Their Uses, I. S. Duff, ed., Academic Press, 1982, pp. 1–29.

[5] I. S. DUFF, A. M. ERISMAN, AND J. K. REID, *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, U.K., 1987.

[6] I. S. DUFF, A. M. ERISMAN, AND J. K. REID, *On George's nested dissection method*, SIAM J. Numer. Anal., 13 (1976), pp. 686–695.

[7] I. S. DUFF, N. I. M. GOULD, M. LESCRENIER, AND J. K. REID, *The multifrontal method in a parallel environment*, Tech. Rept. CSS 211, Computer Science and Systems Division, Harwell, 1987.

[8] I. S. DUFF AND J. K. REID, *A comparison of sparsity orderings for obtaining a pivotal sequence in Gaussian elimination*, J. Inst. Maths. Appl., 14 (1974), pp. 281–291.

[9] ———, *MA27 - A set of* FORTRAN *subroutines for solving sparse symmetric sets of linear equations*, Tech. Rep. AERE R 10533, Harwell, U.K., 1982.

[10] ———, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[11] S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ, AND A. H. SHERMAN, *The Yale sparse matrix package I. the symmetric codes*, Internat. J. Numer. Methods Engrg., 18 (1982), pp. 1145–1151.

[12] S. C. EISENSTAT, M. H. SCHULTZ, AND A. H. SHERMAN, *Algorithms and data structures for sparse symmetric Gaussian elimination*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 225–237.

[13] ———, *Applications of an element model for Gaussian elimination*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 85–96.

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

[15] J. A. GEORGE AND M. T. HEATH, *Solution of sparse linear least squares problems using Givens rotations*, Linear Algebra Appl., 34 (1980), pp. 69–83.

[16] J. A. GEORGE AND J. W. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[17] ———, *A fast implementation of the minimum degree algorithm using quotient graphs*, ACM Trans. Math. Software, 6 (1980), pp. 337–358.

[18] ———, *A minimal storage implementation of the minimum degree algorithm*, SIAM J. Numer. Anal., 17 (1980), pp. 282–299.

[19] J. A. GEORGE, J. W. LIU, AND E. G. NG, *A data structure for sparse* QR *and* LU *factors*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 100–121.

[20] ———, *User's guide for SPARSPAK: Waterloo sparse linear equations package*, Tech. Rep. CS-78-30 (revised), Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1980.

[21] J. A. GEORGE AND D. R. MCINTYRE, *On the application of the minimum degree algorithm to finite element systems*, SIAM J. Numer. Anal., 15 (1978), pp. 90–111.

[22] J. A. GEORGE AND E. G. NG, *Symbolic factorization for sparse Gaussian elimination with partial pivoting*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 877–898.

[23] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[24] C. HEMPEL, *Minimum degree can miss by more than a constant factor*, Tech. Rep.; unpublished manuscript, Dept. of Computer Science, Cornell University, 1985.

[25] A. J. HOFFMAN, M. S. MARTIN, AND D. J. ROSE, *Complexity bounds for regular finite difference and finite element grids*, SIAM J. Numer. Anal., 10 (1973), pp. 364–369.

[26] J. W. LIU, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.

[27] ———, *Reordering sparse matrices for parallel elimination*, Tech. Rep. CS-87-01, Dept. of Computer Science, York University, Downsview, Ontario, Canada, 1987.

[28] H. M. MARKOWITZ, *The elimination form of the inverse and its application to linear programming*, Management Sci., 3 (1957), pp. 255–269.

[29] D. J. ROSE, *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, 1972, pp. 183–217.

[30] ———, *Symmetric elimination on sparse positive definite systems and potential flow network problem*, Ph.D. thesis, Harvard University, 1970.

[31] B. SPEELPENNING, *The generalized element method*, Tech. Rep. UIUCDCS-R-78-946, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, IL, 1978.

[32] W. F. TINNEY AND J. W. WALKER, *Direct solution of sparse network equations by optimally ordered triangular factorization*, Proc. IEEE, 55 (1967), pp. 1801–1809.

[33] M. YANNAKAKIS, *Computing the minimum fill-in is* NP*-complete*, SIAM J. Algebraic Discrete Methods, 2 (1981), pp. 77–79.