USING DEFICIENCY MEASURE FOR TIEBREAKING THE MINIMUM DEGREE ALGORITHM

by

Ian A. Cavers

Technical Report 89-2

January 10, 1989

Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5 Canada

Using Deficiency Measure for Tiebreaking the Minimum Degree Algorithm

Ian A. Cavers

Technical Report 89-2
January 10, 1988

Department of Computer Science
The University of British Columbia
Vancouver, British Columbia
Canada V6T 1W5

Abstract

The minimum degree algorithm is known as an effective scheme for identifying a fill reduced ordering for symmetric, positive definite, sparse linear systems. Although the original algorithm has been enhanced to improve the efficiency of its implementation, ties between minimum degree elimination candidates are still arbitrarily broken. For many systems, the fill levels of orderings produced by the minimum degree algorithm are very sensitive to the precise manner in which these ties are resolved. This paper introduces several tiebreaking schemes for the minimum degree algorithm. Emphasis is placed upon a tiebreaking strategy based on the deficiency of minimum degree elimination candidates, which can consistently identify low fill orderings for a wide spectrum of test problems. The tiebreaking strategies are integrated into a quotient graph form of the minimum degree algorithm with uneliminated supernodes. Implementations of the enhanced forms of the algorithm are tested on a wide variety of sparse systems to investigate the potential of the tiebreaking strategies.

1 Introduction

This paper discusses the solution of large systems of linear algebraic equations of the form

$$Ax = b \tag{1}$$

where A is a sparse, symmetric, positive definite matrix of size NxN. Such problems arise from a broad spectrum of application areas found in many fields of science and engineering. A wide range of different methods have been proposed for their solution but this paper concentrates upon the ordering problem associated with direct methods based on the Cholesky factorization method.

For many problems, finding a "good" ordering or permutation can significantly reduce the computational effort and storage required when the equivalent system

$$PAP^T Px = Pb \tag{2}$$

is solved directly. Here P denotes a permutation matrix. Many criteria have been proposed as the basis of ordering algorithms including minimum arithmetic, minimum storage and minimum fill-in. As discussed by Rose [10], minimization of fill-in is an attractive goal because in practice it is a reasonable approximation of minimum arithmetic and if full advantage is taken of the Cholesky factor's sparsity, primary storage is minimized. Unfortunately, the minimum fill-in problem was shown by Yannakakis [12] to be an NP-complete problem and as a result only good heuristic algorithms are feasible for large sparse problems.

The minimum degree algorithm (or MDA), introduced by Tinney and Walker [11] and by Rose [10] in its graphical form, is the symmetric analog of a general ordering algorithm first suggested by Markowitz [9]. This heuristic algorithm has been widely accepted as a practical approach to the minimum fill-in ordering problem, which is effective in reducing fill and arithmetic costs for a wide range of problems.

The minimum degree ordering algorithm creates a fill reducing permutation by selecting a minimum degree node at each stage of the algorithm from a graph modelling

the unfactored portion of the matrix. When more than one minimum degree node exists in the current graph, an arbitrary selection is made from the group of acceptable elimination candidates. As will be demonstrated, the fill levels experienced by orderings produced by the MDA are sensitive to the precise manner in which these "ties" are resolved. This paper introduces tiebreaking strategies which remove some of the arbitrary nature of the MDA and result in orderings with improved characteristics.

A brief outline of the paper is as follows. Section 2 briefly reviews the MDA and many of the approaches which have been proposed to improve the efficiency of its implementation. After motivating the need for tiebreaking strategies in Section 3, a tiebreaking strategy based upon node deficiency is introduced in Section 4. In the next section several key issues central to a successful implementation of deficiency tiebreaking are discussed. Three additional secondary tiebreaking schemes are briefly outlined in Section 6. The performance of the tiebreaking schemes is demonstrated in Section 7 using a wide range of numerical experiments. Finally, some concluding remarks are provided in Section 8.

2 An Overview of the MDA

In this section a brief overview of the current status of the MDA and its role in the solution process is given, while identifying the particular form of the algorithm to which the tiebreaking strategies of Sections 4 and 6 are to be applied. It is assumed that the reader has a general knowledge of the graph-theoretic terminology associated with the study of ordering algorithms for sparse systems. In addition, familiarity with the special correspondence between the outer product form of the Cholesky method and elimination graphs is assumed. A thorough discussion of this material is provided by George and Liu in [6].

It is generally accepted that the complete solution process for sparse, symmetric, positive definite systems using the Cholesky method can be broken into four distinct and independent steps [4].

1. Ordering
   - Determination of a good ordering or permutation matrix P for A.

2. Symbolic Factorization
   - To exploit the general sparsity of the system, the nonzero structure of L is identified and an appropriate data structure initialized in preparation for the Cholesky factorization of PAP^T.

3. Numerical Factorization
   - Decompose PAP^T into LL^T using the Cholesky factorization method.

4. Solve
   - Solve the factored system $LL^T\,Px = Pb$, by solving two triangular systems.

The MDA has enjoyed increasing popularity as a practical, general purpose ordering algorithm which can be used to perform the first step of this solution process. The minimum fill-in heuristic approximates a local minimization of fill by minimizing the worst fill level that can be experienced at each step of the factorization process.

In some fashion all forms of the MDA rely upon a graphical simulation of the Cholesky factorization method. The simplest scheme is the *elimination graph model.* (See [4] and [10].) An elimination graph, G_i, is used to model the nonzero structure of the unfactored portion of the matrix at each step of the simulated process. The most elementary form of the MDA can be described very simply using this basic model. In the following algorithm $G_0 = (X, E)$ is the elimination graph representing the nonzero structure of the original matrix, while G_i represents the elimination graph after i nodes have been selected for the new ordering.

1. Form G_0 ; $\quad i := 0$

2. While $i < N$

   (a) Choose a minimum degree node, x, from G_i

   (b) Eliminate x from G_i to form G_{i+1}

   (c) i:= i + 1

In step 2b the selected node, x, is eliminated from the current elimination graph to form the new elimination graph. This transformation is accomplished by first removing x and all incident edges from the current elimination graph. The transformation is completed by adding edges to the graph so that the set of nodes adjacent to x in G_i form a clique in G_{i+1}.

Many different techniques have been employed to improve the modelling of the Cholesky factorization and increase the efficiency of resulting MDA implementations. One of the most successful models is the *quotient graph model* [4, 5, 6] in which quotient graphs replace elimination graphs. Similar representation schemes have been referred to as a generalized element model or superelements but this paper will augment with tiebreaking a form of the MDA based on the formalisms of quotient graphs.

In the quotient graph model the Cholesky factorization is modelled by a sequence of quotient graphs $G_0^q, G_1^q, \ldots, G_{N-1}^q$. When an uneliminated node, x, is chosen to be placed in the new ordering and eliminated from the current quotient graph, it is not explicitly removed from the graph as in the elimination graph model. Instead it is simply marked as a member of the set of eliminated nodes. To complete the transformation of the current graph into its successor, x is coalesced with all adjacent eliminated nodes to form a single *eliminated supernode*. The adjacency set of the new supernode is taken as the union of the adjacency sets of all nodes which were combined to from the eliminated supernode. For the remainder of the elimination process the newly formed supernode can be treated as a single eliminated node despite its origins.

The use of quotient graphs to model the Cholesky factorization is permissible because the MDA only requires that the degrees of all uneliminated quotient graph nodes in the corresponding elimination graph can be correctly determined. The *reachable set* of an uneliminated node in a quotient graph environment is equivalent to the node's adjacency set in the corresponding elimination graph and can be used for degree determination. The reachable set of node x in a quotient graph G_i^q, which has an associated set of eliminated nodes S_i, can be formally defined by the following equation. (The adjacency set of x in G_i^q is represented by $adj_{G_i^q}(x)$.)

$$reach_{G_i^q}(x, S_i) = \{\, y \in X - S_i - \{x\} \mid$$
$$(y \in adj_{G_i^q}(x)) \vee ((\exists z)(z \in S_i \wedge z \in adj_{G_i^q}(x) \wedge y \in adj_{G_i^q}(z)))\,\} \quad (3)$$

Less formally, the reachable set of x in G_i^q consists of all adjacent uneliminated nodes and those uneliminated nodes connected to x by a eliminated node path.

A more formal discussion of quotient graphs can be found in any of the references previously cited for the quotient graph model. Three aspects of this more formal treatment of the model will be used in subsequent presentations of the MDA and are briefly outlined at this point. It is assumed that $G = (X, E)$ represents the graph of the nonzero structure of the system's original matrix.

The *component partitioning* $C(S_i)$ can be defined by

$$C(S_i) = \{\, Y \subseteq S_i \mid G(Y) \text{ is a connected subgraph} \,\} \quad (4)$$

where $G(Y)$ is a subgraph of G consisting of nodes in the set Y. Each member of the component partitioning is a connected subset of eliminated nodes. As a result, each member of $C(S_i)$ corresponds to a single eliminated supernode in quotient graph G_i^q. (The coalesced set represented by an eliminated supernode may be a singleton set.)

In turn the *partitioning of* X *induced by* S_i can be defined as

$$Q(S_i) = \{\, \{y\} \mid y \in X - S_i\} \cup C(S_i). \quad (5)$$

Each member of $Q(S_i)$ is simply a node in the quotient graph G_i^q. The *quotient graph* of G with respect to $Q(S_i)$ by definition produces G_i^q [4].

$$G_i^q = G/Q(S_i) \quad (6)$$

There are several advantages to using quotient graphs as the underlying model of an MDA implementation. The most significant benefit is that the storage requirements of the algorithm can be determined a priori. Unlike the elimination graph model the original data structure used to represent the graph of the original matrix provides an

upper bound on the storage requirements of the entire modelling process[4]. In addition, the performance of the algorithm does not suffer as a consequence of obtaining good storage characteristics and efficient degree determination is also possible.

The efficiency of the quotient graph form of the MDA can be enhanced by the identification of *indistinguishable sets* and the formation of *uneliminated supernodes*[5]. Uneliminated supernodes are also used by the symmetric code of the Yale Sparse Matrix Package [2] but the term *prototype node* is used in place of uneliminated supernode. A similar scheme employing *supervariables* is also used by Duff and Reid [1].

Two nodes $x, y \in X - S_i$ are said to be indistinguishable in quotient graph G_i^q if the following expression is true.

$$reach_{G_i^q}(x, S_i) \cup \{x\} = reach_{G_i^q}(y, S_i) \cup \{y\} \tag{7}$$

Two important theorems have been proven by George and Liu [5] which permit the successful integration of the concept of indistinguishable nodes, in the form of uneliminated supernodes, into the MDA. Firstly, it was shown that once a group of nodes are identified as indistinguishable they will remain indistinguishable throughout the elimination process until eliminated themselves. Secondly, it was shown that if one node of an indistinguishable set was chosen for elimination, that all remaining members would be of minimum degree after the elimination of the first node. As a result when one node of an indistinguishable set is targeted for elimination, the entire set can be eliminated in a single step of *mass elimination.*

These two results, together with the fact that indistinguishable nodes must have the same degree, permit an indistinguishable set to be treated as a single unit throughout the elimination process. Once an indistinguishable set of uneliminated nodes has been identified they can be coalesced to form an uneliminated supernode by choosing a representative from the set and removing all other members, and incident edges, from the current quotient graph. When the representative is selected for elimination all members of the indistinguishable set are placed in the new ordering. If an uneliminated node encounters an uneliminated supernode in its reachable set during a degree calculation, the degree value must, of course, be incremented by the number of nodes represented by the supernode.

The identification of all indistinguishable sets in a quotient graph is a very difficult task. Fortunately, George and Liu [5] have proposed an efficient mechanism which can be used to identify sets of indistinguishable uneliminated nodes after each elimination step and associated quotient graph transformation. Although the scheme can not guarantee to identify all indistinguishable sets, in practice the scheme has been shown to find an acceptable proportion.

Once a group of indistinguishable nodes has been identified and coalesced into a single supernode, several aspects of the MDA benefit from their formation. The presence of uneliminated supernodes in a quotient graph generally reduces the number of costly degree updates that are required. Only the degree of a supernode's representative ever needs recalculation. In addition, when a supernode is eliminated in a mass elimination step only one set of degree updates and one quotient graph transformation is required for the entire group of indistinguishable nodes. Finally, the formation of uneliminated supernodes helps to reduce the number and complexity of search paths required by reachable set determination.

If the node selection criteria of the MDA is to be strictly observed the degree of an uneliminated supernode should take into account the number of indistinguishable nodes associated with the representative. The choice of a minimum degree node by the MDA, however, can be thought of in terms of choosing the node whose elimination will result in the formation of the smallest clique. When a supernode is deleted by a single mass elimination step the size of the resulting clique is independent of the number of indistinguishable nodes actually represented by the supernode. This motivates the use of *external degrees* as introduced by Liu [8], for uneliminated supernodes. Liu has shown that the use of external degrees improves the quality of orderings produced by the modified form of the MDA.

The MDA based upon a quotient graph model with indistinguishable nodes and external degrees is the form of the algorithm to which tiebreaking strategies are to be applied. The following pseudo-code algorithm summarizes this existing form of the MDA. It should be noted that step 3d should only be considered as a formalism used to simplify the presentation of the algorithm. Each quotient graph is not formed from the original graph of the matrix but from its quotient graph predecessor. A formal

algorithm for successive quotient graph formation can be found in [5], as well as an algorithm for the identification of indistinguishable sets.

MDA: Quotient Graph Model With Uneliminated Supernodes

Let

D_i = the degree of node i

S = the set of eliminated nodes

$G(=(X,E))$ = the graph of the system's original coefficient matrix

$G^q(=(X^q,E^q))$ = the current quotient graph

$save$ = variable for saving a reachable set

$expand(x) = \{x\} \cup \{$all other members of the supernode represented by $x\}$

$size(W)$ = the number of members in set W,
with each supernode $x \in W$ counting $|expand(x)|$

1. $S := \emptyset$, $G^q := G$

2. For all nodes $k \in X$
   $D_k := |adj_G(k)|$

3. While $S \neq X$ do

   (a) Choose a node $t \in X^q$, for which
   $D_t = \min_{i \in X^q - C(S)}(D_i)$ (ties arbitrarily broken)

   (b) $save := reach_{G^q}(t, C(S))$

   (c) $S := S \cup expand(t)$
   (The nodes of a mass elimination are recorded in an arbitrary order.)

   (d) $G^q := G/Q(S)$

   (e) Identify all sets of indistinguishable nodes amongst the members of *save* and form a supernode for each set. Update G^q and X^q appropriately.

   (f) For each $y \in save$
   $D_y := size(reach_{G^q}(y, C(S)))$

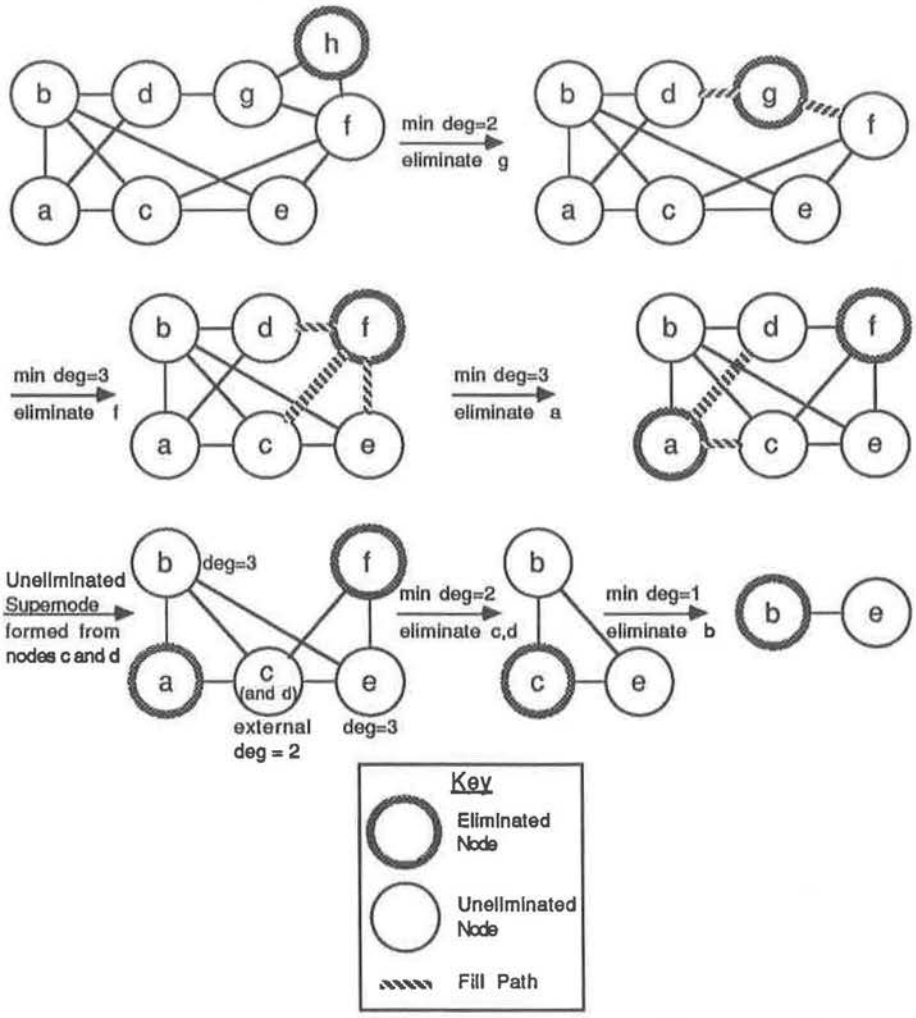

Figure 1: Application of the MDA to a small example.

Figure 1 illustrates the application of the quotient graph form of the MDA with indistinguishable nodes to a small example in which the first node has already been eliminated. When a supernode is formed it will be represented for the process by one member chosen from the group of coalesced nodes. Each *fill path* corresponds to a fill entry in L.

Although not considered by the tiebreaking discussion of this paper, two additional enhancements of the MDA have been proposed. The Yale Sparse Matrix Package [2] uses a technique referred to as *incomplete degree updating* to delay degree updating of *outmatched* nodes until absolutely necessary. Using the *multiple elimination scheme*

introduced by Liu [8], degree updating is delayed until an entire independent set of minimum degree nodes has been eliminated. It should be emphasized that forms of the MDA enhanced by the multiple elimination scheme will not necessarily produce a minimum degree ordering.

3 Tiebreaking Motivation

For a tiebreaking strategy to have a significant impact upon the quality of orderings produced, there must be sufficient opportunity to apply a tiebreaking criterion. In the majority of practical test problems considered in Section 7 a significant fraction of node selections require an arbitrary choice between two or more minimum degree elimination candidates. As an example, consider the nine point problem of Section 7, consisting of 4225 nodes. When the MDA with indistinguishable nodes is applied to the graph of this problem, with its original labelling, 2181 node selections were made. (Each selection of a supernode contributes one unit towards this total.) From this total 2167 tiebreaking opportunities were identified. Because the choice between minimum degree candidates is arbitrary, this large number of tiebreaking opportunities means that there is a profusion of valid minimum degree orderings for this problem.

In an actual MDA implementation the choice between minimum degree candidates is not random. One particular node from a group of minimum degree candidates is always selected; perhaps the first candidate of the original ordering is chosen. As a result, the fill levels of several different minimum degree orderings for the same problem can be observed by applying the MDA implementation to graphs of the same problem with different initial labellings.

Table 1 summarizes the fill values observed for six different minimum degree orderings of the 4225 node nine point problem. Each fill value records the number of new nonzero entries introduced into the factor L. The initial labelling used for the first ordering was the standard lexicographic ordering of the finite differencing problem, while the five remaining fill values correspond to orderings produced when this initial labelling was randomly permuted. From this relatively small sample of the problem's of minimum degree orderings, the potential impact of an effective tiebreaking strategy is

Ordering	MDA Fill
1	99644
2	108222
3	113026
4	111345
5	110660
6	115643

Table 1: Fill Fluctuations of Minimum Degree Orderings

immediately apparent. The fill value of ordering 1 is approximately 14% smaller than the fill value observed for ordering 6. These results demonstrate the sensitivity of fill levels to the particular manner in which ties between minimum degree candidates are resolved.

Due to the complexity of the factor and solve steps of the solution process, even a small reduction in fill can lead to a substantial reduction in the effort required by the remainder of the solution process. To illustrate this effect, the numerical solution of the 4225 node nine point problem was computed using orderings 1 and 6. Timings of the explicit factorization and solve steps of the solution process were recorded. (See Section 7 for a description of the testing environment.) Ordering 6 required 816 CPU seconds for the factorization and 50 CPU seconds for the solution of the resulting triangular systems. The fill reduced ordering reduced the factorization CPU requirements by 24% to 621 seconds, while the number of CPU seconds required for the triangular solutions was reduced to 44.

4 Primary Tiebreaking of the MDA

This section discusses the development of a tiebreaking strategy for the MDA and the integration of this scheme into the version of the MDA outlined in Section 2. The node selection criterion forming the basis of this strategy relies upon the comparison of *deficiency* values for the minimum degree candidates. The primary goal of the strategy is to produce an enhanced MDA which can consistently select fill-reduced orderings. A

secondary goal of the strategy is to stabilize the fill levels of orderings with respect to a particular problem's initial labelling.

4.1 Other Tiebreaking Research

Other than the ideas proposed by this paper, the only other known active research into the tiebreaking problem is that of George and Liu [7]. Their approach to tiebreaking the MDA is very different from the strategy to be subsequently discussed. They suggest the use of a profile reduction algorithm on the matrix before applying the MDA. Their preliminary results have shown that this method is effective on a 180-by-180 square grid problem using a nine point differencing molecule.

4.2 The Deficiency Tiebreaking Strategy

When Rose [10] first described a graph theoretic form of the MDA, a second heuristic algorithm for the ordering problem was also proposed. In a similar fashion to the MDA, the *minimum deficiency algorithm* approximates a global minimization of fill using a local minimization. Instead of eliminating the nodes of minimum degree at each stage, however, the node selected by this alternate algorithm must have minimum deficiency.

Assuming that the minimum deficiency algorithm uses a quotient graph model of the factorization process, the deficiency of node $x \in X^q$, *def(x)*, can be defined in the following manner. To simplify this initial discussion it is assumed that uneliminated supernodes were not implemented by the modelling process.

$$def_{G^q}(x) = \{\, \{z,y\} \subseteq X^q - C(S) \mid (z \neq y) \wedge (\, \{z,y\} \subseteq reach_{G^q}(x, C(S))\,) \wedge (z \notin reach_{G^q}(y, C(S)))\,\} \quad (8)$$

Alternatively, the deficiency of node x in the corresponding elimination graph consists of the set of all pairs of nodes in *adj(x)* which are not adjacent. In other words, the deficiency of x is the set of new edges amongst members of its adjacency set which are required to make *adj(x)* into a clique.

The number of edges needed to make a node's adjacency set a clique is precisely the amount of fill that will be experienced if that node is eliminated. Thus by selecting

an elimination candidate of minimal deficiency, the minimum deficiency algorithm is actually choosing the node whose elimination produces the least fill. Although the cost of maintaining an updated record of the size of each active node's deficiency throughout the factorization process is prohibitive, the notion of deficiency can be used in a more restricted fashion as the basis of a tiebreaking strategy for the MDA.

Each cycle of the basic MDA arbitrarily selects the next node for elimination from the group of minimum degree nodes in the current quotient graph. In terms of an elimination graph modelling, a minimum degree node x is chosen because its adjacency set will form the smallest clique upon x's elimination. This approximates the local minimum fill criteria by minimizing the worst case of possible *fill damage* resulting from an elimination. The worst possible fill damage is realized when no edges already exist amongst the members of $adj(x)$ before x's elimination. Thus with respect to the local minimization of fill, the best minimum degree candidate to choose for elimination is the node which has the most edges already existing amongst members of its adjacency set. This corresponds to the minimum degree node which has the smallest deficiency set. This observation forms the basis of the primary tiebreaking strategy.

At each stage of the modelled factorization, the proposed tiebreaking strategy directs that the node with the smallest deficiency be selected for elimination from the group of minimum degree candidates. As a small example, consider the two subgraphs of an elimination graph illustrated by Figure 2. Assuming that the minimum degree of all nodes in the current elimination graph is 4, both x and y would be candidates for elimination. However, in this case the new tiebreaking strategy would choose node x because it has the smaller deficiency set.

The tiebreaking scheme can be viewed in an alternate fashion. As mentioned previously, a rigorous local minimization of fill at each stage of the elimination process proves very costly. An application of the MDA's minimum degree criterion can be thought of as a step in which the set of all uneliminated nodes is reduced to a more manageable subset within which a minimum fill node is likely to be found. The tiebreaking strategy then applies the minimum fill criterion to this smaller set, hopefully finding a minimum or near minimum fill candidate.

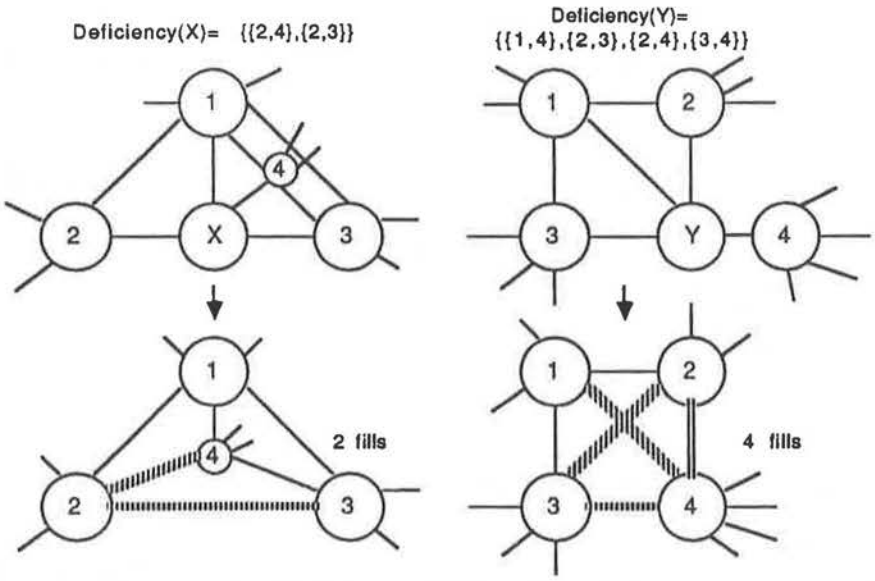

Figure 2: Deficiency Tiebreaking Example

It should be noted that even with the new tiebreaking strategy, "ties" may still exist between elimination candidates. In such cases a minimum degree, minimal deficiency node is arbitrarily selected. (Secondary tiebreaking schemes are briefly discussed in Section 6.)

4.3 Deficiency Calculations

As discussed in Section 4.2, the deficiency of a node is a set of edges. The MDA requires only the size of deficiency sets rather than their actual membership. As a result, in subsequent discussion the term *deficiency* will also be used to refer to the number of edges in the set. A preliminary algorithm for the calculation of deficiencies in quotient graphs, without indistinguishable sets, is considered before extending the algorithm to include uneliminated supernodes.

The calculation of a node's deficiency is fairly straightforward. A count of the number of edges needed to make the reachable set of a node, x, into a clique in the corresponding elimination graph is required. From an implementation standpoint, it is easier to count edges which already exist amongst the nodes of x's elimination graph adjacency set. In quotient graph terms, this corresponds to the number of pairs of

nodes in x's reachable set which are adjacent or connected by an eliminated node path. This value is essentially the complement of deficiency and will be referred to as a node's *connectivity.*

Once its connectivity is calculated, a node's deficiency can be determined by subtracting the actual connectivity observered from the maximum connectivity value possible for a node of its degree. (The degree and connectivity of node x are referred to as *deg(x)* and *conn(x)* respectively.)

$$\begin{aligned} def(x) &= maxconn(x) - conn(x) \\ &= (deg(x) * (deg(x) - 1))/2 - conn(x) \end{aligned} \tag{9}$$

To calculate the connectivity of a node x, the reachable set of each node in x's reachable set can be searched for other nodes which are also members of x's reachable set. This observation is formalized in the following pseudocode algorithm for calculating the connectivity of node x in a quotient graph $G^q = (X^q, E^q)$ without uneliminated supernodes.

1. $conn(x) := 0$
2. For each $y \in reach_{G^q}(x, S)$
   $conn(x) := conn(x) + |reach_{G^q}(x, S) \cap reach_{G^q}(y, S)|$
3. $conn(x) := conn(x)/2$

This form of the connectivity calculation algorithm neglects the added complications introduced when an uneliminated supernode is encountered in x's reachable set. When a supernode is found in a node's reachable set during a degree determination, all members of the indistinguishable set contribute towards the degree value. This allows the MDA to successfully predict the size of the clique that would be formed upon the node's elimination from the corresponding elimination graph. While the MDA attempts to minimize the size of clique formed by each elimination, the tiebreaking scheme attempts to choose the node for which the fewest new edges are required to form the minimal clique. Thus when calculating a normal node's connectivity in a quotient graph, all

supernodes encountered should be considered in their expanded form so that all paths explicitly and implicitly represented in the quotient graph are accounted for.

The following pseudocode algorithm presents a revised form of the connectivity calculation which allows for uneliminated supernodes. This form of the algorithm is intended for the calculation of a normal uneliminated node's connectivity. The connectivity of supernodes themselves will be considered during the following subsection. (The function *size* was declared in Section 2.)

Let

$superlength(x)$ = the number of indistinguishable nodes represented by x

1. $conn(x) := 0$
2. For each $y \in reach_{G^q}(x, S)$
   (a) $conn(x) := conn(x)$
   $+ superlength(y) * size(\ reach_{G^q}(x, S) \cap reach_{G^q}(y, S)\ )$
   (b) If $superlength(y) > 1$ then
   $conn(x) := conn(x) + superlength(y) * (superlength(y) - 1)$
3. $conn(x) := conn(x)/2$

Step 2a of the algorithm performs the basic step of connectivity accumulation and is an obvious extension of the previous algorithm. Step 2b of the algorithm, however, was introduced so that the internal connectivity of all supernodes in x's reachable set is included. As a trivial example, in Figure 3 nodes 4 and 6 both have a maximal connectivity of 6 while the connectivity of node 5 is 9.

As can be seen from equation 9, the minimization of deficiency corresponds to a maximization of connectivity. This permits the formulation of the tiebreaking strategy in terms of connectivity values. Because slightly fewer calculations are required to determine connectivity values, the implementation developed for testing used connectivity maximization. As a result, in subsequent discussion connectivity may replace deficiency when discussion of issues close to the implementation are considered.

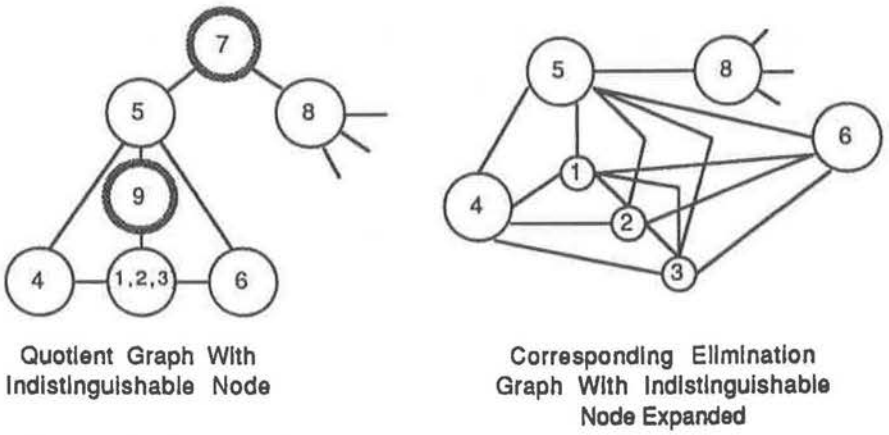

Figure 3: Supernode Expansion for Connectivity Calculation

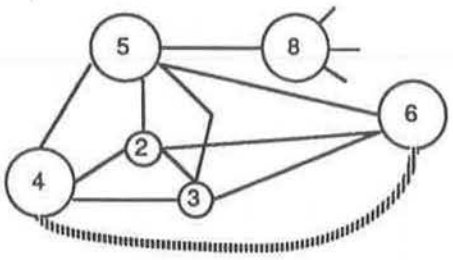

Figure 4: Elimination of a Supernode Representative

4.4 The Deficiency, Degree and Elimination of Supernodes

With the introduction of the new tiebreaking strategy, it must be determined if supernodes may still be considered as a single unit when chosen for elimination. The resolution of this situation will dictate whether the external or true degree of a supernode should be considered by the selection process of the tiebreaking MDA. In addition the calculation of a supernode's deficiency must be considered to decide how the other indistinguishable nodes represented by the supernode should influence its own deficiency calculation.

Assume for the moment that the nature of a supernode's degree and deficiency have been settled upon. The introduction of the tiebreaking scheme may hamper the mass elimination of uneliminated supernodes. Although all remaining nodes of a coalesced set will be indistinguishable and of minimum degree after their representative's elimination, they may not be of minimal deficiency as well. Momentarily ignoring the constraints of minimum degree, consider the elimination of the representative of the supernode illustrated in Figure 3. (Node 1 is assumed to be the representative.) The resulting elimination graph is shown in Figure 4. After the elimination of node 1, the adjacency set of each member of the remaining indistinguishable set, $\{2,3\}$, now forms a clique. As a result, the elimination of the entire indistinguishable set, originally represented by the supernode, cannot result in more fill than if the representative were eliminated alone. Although this is a trivial example, it motivates the following theorem which allows the generalization of this observation to all uneliminated supernodes. (The size of the deficiency set of node x in quotient graph G_i^q is referred to as $def_{G_i^q}(x)$.)

Theorem 1 *Let Y represent a set of ℓ indistinguishable, uneliminated nodes $\{y_1, y_2, \ldots, y_\ell\}$ in the quotient graph $G_i^q = (X_i^q, E_i^q)$, which have not been coalesced to form a supernode. ($Y \subseteq X^q - S_i$.) If $y_k \in Y$ is eliminated from G_i^q to form G_{i+1}^q, for all remaining nodes of Y, $y_i \in Y - \{y_k\}$, $def_{G_{i+1}^q}(y_i) = 0$.*

Proof: During the course of this proof elimination graphs are considered instead of quotient graphs because of their simpler but equivalent representation of the unfactored portion of the matrix under consideration. It can easily be shown that the degree,

deficiency and indistinguishable properties of uneliminated nodes are invariant under the transformation from a quotient graph into its equivalent elimination graph.

Two nodes, x and y, are indistinguishable in an elimination graph G_i if the following expression is true.

$$adj_{G_i}(x) \cup \{x\} = adj_{G_i}(y) \cup \{y\} \tag{10}$$

Let $G_i = (X_i, E_i)$ and $G_{i+1} = (X_{i+1}, E_{i+1})$ be the elimination graphs corresponding to the quotient graphs G_i^q and G_{i+1}^q respectively. As well, let $T = \mathrm{adj}_{G_i}(y_i) - Y$, $1 \leq i \leq \ell$. By the definition of indistinguishable nodes given in equation 10, the membership of set T is independent of the particular value of i selected.

A node in an elimination graph will have a nonzero deficiency if at least two members of its adjacency set are not adjacent to each other. From the alternate definition of indistinguishable nodes given in equation 10, any pair of nodes $y_n, y_m \in Y$ in G_i must be adjacent. Thus no new fill edges are possible between the members of Y. As well, independent of the particular node chosen for elimination, no new fill edges are possible between a pair of nodes $\{t_i, y_i\}$, where $t_i \in T$ and $y_i \in Y$. By the definition of set T, each member of Y is already adjacent to all members of T. Thus when a member of Y, y_k, is chosen for elimination, the only possible source of new edges is amongst pairs of nodes in T.

When y_k is eliminated, fill edges are added amongst pairs of nodes in T which are not already adjacent and $adj_{G_i}(y_k)$ becomes a clique in G_{i+1}. The number of fill edges introduced depends upon the value of $def_{G_i}(y_k)$. In any event, after the elimination all possible edges must exist amongst the members of T, $Y - \{y_k\}$ and between all node pairs with one node from each of these sets. As a result, for all possible $y_i \in Y - \{y_k\}$, $adj_{G_{i+1}}(y_i)$ is a clique and $def_{G_{i+1}}(y_i) = 0$. Ξ

Without the results of Theorem 1 it might have been necessary to eliminate each member of a supernode separately. After the removal of each indistinguishable node, the deficiency of the smaller supernode would have had to have been reevaluated in preparation for the next minimum degree node selection. Having to follow such a scheme would negate many of the advantages observed for uneliminated supernodes in

Section 2.

Fortunately, Theorem 1 allows the complete integration of the MDA, using quotient graphs and uneliminated supernodes, with the proposed deficiency tiebreaking strategy. When a supernode is selected for elimination it can be treated as a single unit and eliminated with one step of mass elimination. In addition, since uneliminated supernodes are manipulated in their compressed form, the notion of external degrees will be employed by the deficiency tiebreaking version of the MDA.

The only unresolved issue is the calculation of a supernode's deficiency. As far as the tiebreaking strategy is concerned, all that matters is the number of new fill entries a supernode's elimination will create. As shown in the proof of Theorem 1, any new *fill paths* created must be amongst members of the representative's reachable set in the current quotient graph. (A new fill path in a quotient graph is equivalent to a new fill edge in the corresponding elimination graph.) Thus the deficiency of the representative is the same regardless of whether or not the other members of the coalesced indistinguishable set are considered. As a result the deficiency of a supernode can be calculated from its compressed structure using the algorithm presented in Section 4.3, as though the representative was a normal uneliminated node in the quotient graph.

4.5 Degree and Connectivity Updating

As previously mentioned, the cost of maintaining the current deficiency (or connectivity) value for all uneliminated nodes throughout the elimination process is unacceptable. As will be discussed in Section 5.1, only a very restricted subset of uneliminated nodes will actually have their connectivity value recorded at any particular time. The discussion in the remainder of this subsection ignores this possibility and assumes that all affected nodes need connectivity updating. The resulting updating algorithm is easily modified to reflect the constraints of a particular connectivity maintenance scheme.

The introduction of the deficiency tiebreaking extension has increased the complexity of updating required after each step of the MDA. Although the subset of uneliminated nodes requiring degree updating remains the eliminated node's reachable set, connectivity updating must be applied to a broader class of nodes.

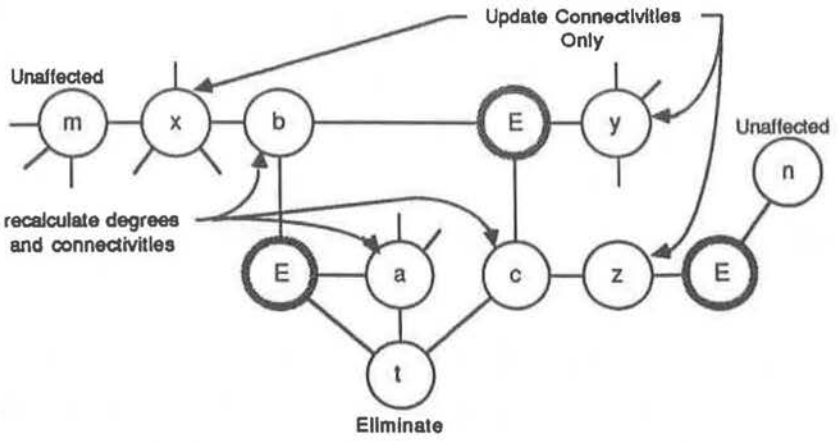

Figure 5: Connectivity and Degree Updating

Consider the elimination of node t from the portion of the quotient graph, G_i^q, illustrated in Figure 5. With respect to degree and connectivity updating, the remaining nodes of the quotient graph can be grouped into three categories. Letting S_i represent the set of eliminated nodes in G_i^q, labelled as "E" nodes, the first category consists of all nodes in $reach_{G_i^q}(t, S_i)$. The nodes of this set have their own reachable sets altered by the elimination of t and therefore need their degree recalculated. A node's reachable set also plays a central role in the calculation of its connectivity. Consequently the nodes of this first category also require a reevaluation of their connectivity.

The second category of uneliminated nodes are those which require a recalculation of their connectivity value but not being members of $reach_{G_i^q}(t, S_i)$, have unaffected reachable sets. Formally, the members of this category consist of all nodes in the set

$$\Big(\bigcup_{x \in reach_{G_i^q}(t,S_i)} (reach_{G_i^q}(x, S_i)) \Big) - (reach_{G_i^q}(t, S_i) \cup \{t\}). \tag{11}$$

Because of their distance from the elimination node t, members of this category will often be referred to as *satellite* nodes.

From the formal definition given above, the reachable set of each satellite node must contain one or more members of $reach_{G_i^q}(t, S_i)$. When the reachable sets of category one nodes change upon t's elimination, their contribution to a satellite node's connectivity may change. However, it is impossible for t's elimination to decrease a satellite

node's connectivity because by definition a satellite node may not be a member of $reach_{G_i^q}(t, S_i)$.

Finally, the third category of uneliminated nodes are those which do not have their degree or connectivity affected by the elimination. Each of these nodes is insulated from the elimination by at least two uneliminated nodes in all paths connecting them to node t.

For all nodes of category one, it is necessary to completely recalculate their connectivity. Instead of recalculating the connectivity of each satellite node, however, it is possible to modify their current connectivity value to reflect the connectivity changes resulting from those nodes with altered reachable sets. As an example, when t is eliminated from G_i^q to form the new quotient graph G_{i+1}^q, the reachable set of b is altered, possibly changing the connectivity of node y. If the reachable set of b in G_i^q is still available, the connectivity of y can be modified in the following manner to reflect the change in b's reachable set. (The functions *size* and *superlength* were introduced in Sections 2 and 4.3 respectively.)

$$\begin{aligned} conn(y) \;:=\; & conn(y) - 1/2 * superlength_{G_i^q}(b) * size(\, reach_{G_i^q}(y, S_i) \\ & \cap\, reach_{G_i^q}(b, S_i)\,) + 1/2 * superlength_{G_{i+1}^q}(b) \\ & * size(\, reach_{G_{i+1}^q}(y, S_{i+1}) \cap\, reach_{G_{i+1}^q}(b, S_{i+1})\,) \end{aligned} \tag{12}$$

The origin of this modification is clear upon consideration of step 2a of the final algorithm presented in Section 4.3. The factors of $1/2$ are required in the modification formula to avoid double counting. If such a correction is performed for each node of y's reachable set whose own reachable set has been altered by t's elimination, the connectivity updating of node y can be accomplished without using the calculation algorithm outlined in Section 4.3.

The correction of y's connectivity by node b can only increase the connectivity total or leave it unchanged. The potential difference in y's connectivity is dependent upon the change in b's reachable set upon t's elimination. Momentarily ignoring the possibility that new indistinguishable sets might be identified upon t's elimination, the only node which can be removed from b's reachable set is node t itself. Because a satellite node's reachable set can not contain t by definition, this change cannot

affect y's connectivity value. The elimination of t can increase the number of nodes in b's reachable set, however, by providing a new eliminated node bridge to additional uneliminated nodes. It is this potential increase in b's reachable set that may cause an increase in y's connectivity.

Based on these observations, the update step of equation 12 can be simplified by introducing the difference set

$$\begin{aligned} diff(b) &:= reach_{G_{i+1}^q}(b, S_{i+1}) - reach_{G_i^q}(b, S_i) \\ &:= reach_{G_i^q}(t, S_i) - reach_{G_i^q}(b, S_i) \end{aligned} \tag{13}$$

and by assuming that the *superlengths* of nodes have not changed.

$$\begin{aligned} conn(y) &:= conn(y) \\ &\quad +1/2 * superlength_{G_i^q}(b) * size(diff(b) \cap reach_{G_{i+1}^q}(y, S_{i+1})) \end{aligned} \tag{14}$$

The assumption that superlengths have not changed requires that the updating of satellite connectivity values is performed before the search for new indistinguishable sets. This is a realistic implementation assumption. In practice, it has been confirmed that the modification of the connectivity values for satellite nodes is more efficient than a total recalculation of their values.

To formalize the discussion presented in this subsection, the following algorithm is proposed to coordinate all necessary degree and connectivity updating required after each elimination. It is assumed that the selected node, t, has already been eliminated from the quotient graph but that the reachable sets of all category one nodes before the elimination are available. (The functions *size* and *superlength* were introduced in Sections 2 and 4.3 respectively.)

Let

D_x = the degree of node x

1. For each $b \in reach_{G_i^q}(t, S_i)$

   (a) $diff(b) := reach_{G_i^q}(t, S_i) - reach_{G_i^q}(b, S_i)$

(b) For each $y \in (reach_{G_i^q}(b, S_i) - (reach_{G_i^q}(t, S_i) \cup \{t\}))$

$$conn(y) := conn(y) + 1/2 * superlength_{G_i^q}(b) * size(diff(b) \cap reach_{G_{i+1}^q}(y, S_i \cup \{t\}))$$

2. For each $b \in reach_{G_i^q}(t, S_i)$

   (a) $D_b := size(reach_{G_{i+1}^q}(b, S_i \cup \{t\}))$

   (b) Recalculate the connectivity of b using the algorithm of Section 4.3.

The updating of category one and two nodes is performed separately so that in the enhanced MDA, the search for indistinguishable sets can be performed between the two stages of the updating process. The modification of satellite connectivities is carried out before the formation of new supernodes so that the *superlengths* are unchanged from the previous graph. The recalculation of degree and connectivity of nodes in category one is performed after the search so that advantage can be taken of newly formed uneliminated supernodes.

4.6 The MDA With Deficiency Tiebreaking

The tiebreaking version of the MDA presented in this section is a modified form of the quotient graph MDA with indistinguishable nodes discussed in Section 2. Following the tiebreaking extension, a node from the minimum degree set with maximal connectivity, in comparison to other minimum degree nodes, is selected for elimination at each stage. Both external degree and external connectivity values are used for all uneliminated supernodes. The degree updating section of the original algorithm has been expanded to handle connectivity updating as described in Section 4.5. Finally, it is assumed that all connectivity values are originally calculated or recalculated using the algorithm of Section 4.3.

As previously mentioned, to maintain the connectivity of every uneliminated node throughout the simulated factorization would be unacceptable. Section 5.1 will describe the maintenance of a data structure referred to as the *connectivity list* which will store a strictly limited subset of node–connectivity pairs. Connectivity updates will actually be performed for only those nodes which are included in the connectivity list. (Functions

size and *expand* were introduced in Sections 2, while *superlength* and *diff* were defined in Sections 4.3 and 4.5 respectively.)

MDA Enhanced With Deficiency Tiebreaking

Let

D_x = the degree of node x
$minD$ = set of minimum degree nodes
$conn(x)$ = the current connectivity value of node x
S = the set of eliminated nodes
$G\,(=(X,E))$ = the graph of the system's original coefficient matrix
$G^q\,(=(X^q,E^q))$ = the current quotient graph
$saverch(x)$ = storage for the reachable set of node x

1. $S := \emptyset$, $G^q := G$
2. Initialize the connectivity list
3. For each node $k \in X$
   $D_k := |adj_G(k)|$
4. While $S \neq X$ do
   (a) $minD := \{\, x \mid (x \in X^q - C(S)) \wedge (D_x = \min_{i \in X^q - C(S)}(D_i)) \,\}$
   (b) Choose a node $t \in minD$, for which
   $conn(t) = \max_{i \in minD}(conn(i))$
   (c) $saverch(t) := reach_{G^q}(t, C(S))$
   (d) For each $x \in reach_{G^q}(t, C(S))$
   $saverch(x) := reach_{G^q}(x, C(S))$
   (e) $S := S \cup expand(t)$
   (The nodes of a mass elimination are recorded in an arbitrary order.)
   (f) $G^q := G/Q(S)$
   (g) For each $x \in saverch(t)$

i. $diff(x) := saverch(t) - saverch(x)$

ii. For each $y \in (saverch(x) - (saverch(t) \cup \{t\}))$

$conn(y) := conn(y) + 1/2 * superlength(x)$

$* size(diff(x) \cap reach_{G^q}(y, C(S)))$

(h) Identify sets of indistinguishable nodes amongst the members of $saverch(t)$ and form a supernode for each set. Update G^q and X^q appropriately.

(i) For each $x \in saverch(t)$

i. $D_x := size(reach_{G^q}(x, C(S))$

ii. recalculate the connectivity of x

Figure 6 illustrates the application of this enhanced algorithm to a small example. In each successive quotient graph, the minimum degree nodes are identified by a small asterisk beside each potential candidate, along with its current connectivity. The application of the tiebreaking MDA to this graph results in three fills. Each is illustrated by a dashed path between two uneliminated nodes. The application of the normal MDA with arbitrary tiebreaking to this example could produce up to a maximum of five fills.

5 Important Implementation Considerations

The introduction of deficiency tiebreaking results in a substantial increase in the complexity of the enhanced MDA's implementation. This section briefly discusses two of the most sensitive implementation issues. In addition, the particular approach taken for the implementation of reachable sets is shown to permit a modification of the tiebreaking MDA which dramatically increases the efficiency of an implementation.

5.1 Connectivity Lists

As first discussed in previous sections, it would be impractical to maintain an updated connectivity value for each uneliminated node throughout the elimination process. An alternative is to keep up to date the connectivity of a smaller subset of nodes, which is guaranteed to include all minimum degree elimination candidates. To assist in the node

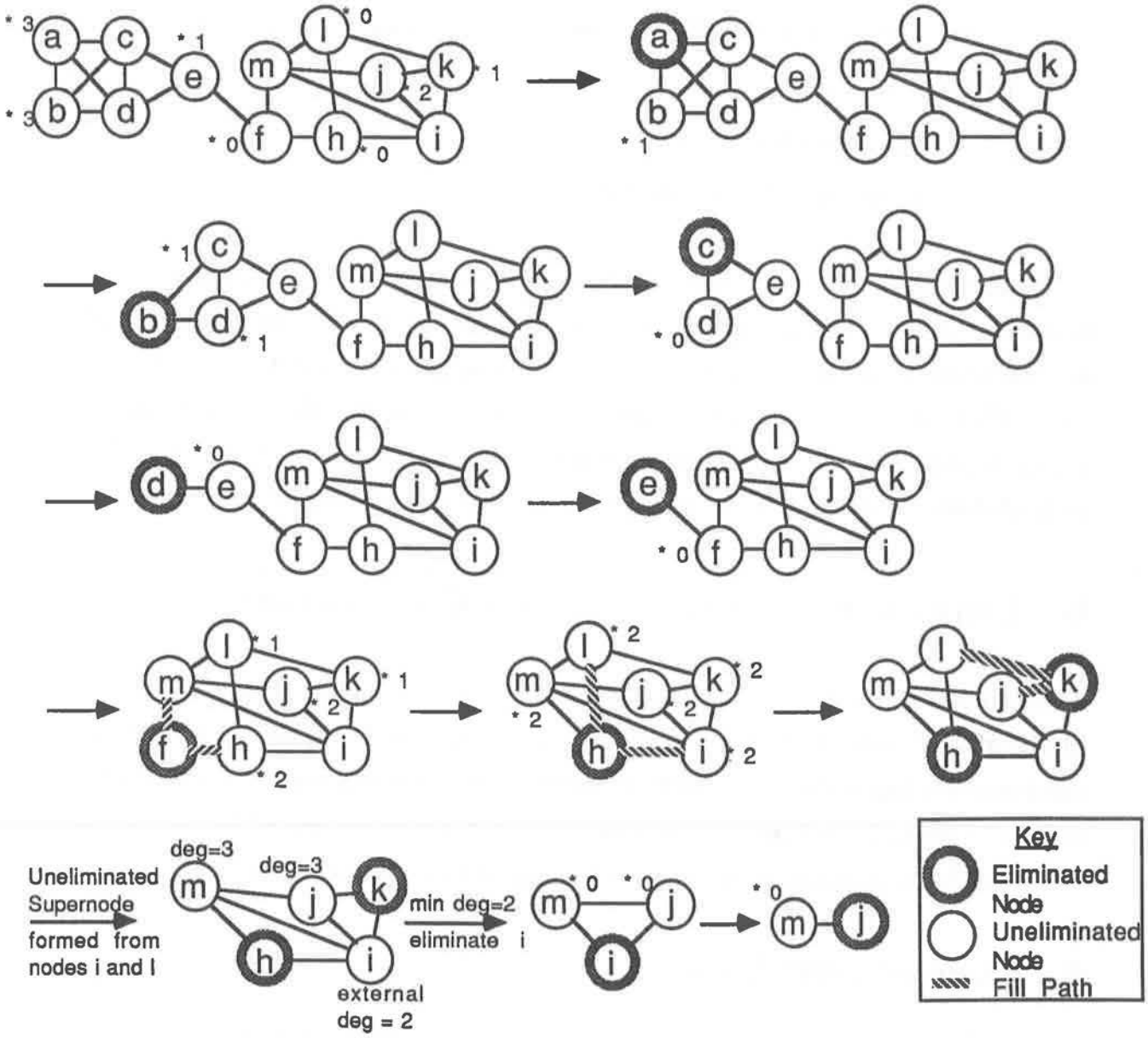

Figure 6: MDA Example Using Deficiency Tiebreaking

selection process, this subset of nodes is grouped together into a central *connectivity list* sorted according to decreasing connectivity values. At each step of the algorithm a node is selected for elimination using a sequential search for the first minimum degree candidate in the connectivity list. It is important to reemphasize that after a node is eliminated, connectivity updating is only conducted for nodes which will remain members of the connectivity list.

The connectivity list was actually implemented as a doubly linked list. As the elimination process proceeds, degree and connectivity values change, requiring nodes to be inserted into or deleted from the connectivity list. If the connectivity list is not implemented as a doubly linked list, the deletion of nodes becomes very inefficient for large problems. As an example, if a tiebreaking MDA implementation with a singly linked connectivity list is applied to the 10,000 node five point problem of Section 7, an increase in execution time of approximately 25% is observed.

To further increase the efficiency of the deletion and insertion operations, an index for the connectivity list was introduced. The index consists of a linked list of index nodes, each of which records the length and position of blocks of the connectivity list with a common connectivity value. Experience has shown that for a typical problem, the connectivity list consists of relatively small number of different blocks of nodes with common connectivity values. The index nodes are sorted within the linked list of the index by decreasing connectivity block values.

The performance of the tiebreaking MDA implementation is very sensitive to the precise manner in which membership of the connectivity list is regulated. An `integer` variable, referred to as `maxcondeg`, was introduced to manage the restricted subset of nodes included in the connectivity list. The connectivity list is guaranteed to consist of all active uneliminated nodes, whose degree is less than or equal to `maxcondeg`.

When the ordering program begins, `maxcondeg` is set equal to the minimum degree of all nodes in the quotient graph of the original system. The initial connectivity list is constructed from all nodes, and their corresponding connectivities, which possess this minimum degree. As nodes are eliminated from the sequence of quotient graphs or absorbed into uneliminated supernodes they are removed from the connectivity list.

If the connectivity list becomes empty during the elimination process, **maxcondeg** is increased to the new minimum degree value, and the connectivity list is augmented with all active, uneliminated nodes of this degree.

As the elimination process proceeds, the degrees of active uneliminated nodes can vary quite substantially from their initial values. If the degree of a node becomes less than or equal to the current value of **maxcondeg**, its connectivity is calculated and the node inserted into the connectivity list. However, if the degree of a list member rises above the current **maxcondeg** value, the node is removed from the connectivity list. If such a node were to remain in the connectivity list, its connectivity value would have to be totally recalculated. In addition, the node would have to be deleted from the connectivity list and then reinserted with its new value. Essentially the same amount of work is required if the node were to be removed from the connectivity list and reinserted when it is of minimum degree once again later in the elimination process. However, by removing the connectivity list entry all additional connectivity updating, which would have been required before the node once again became a minimum degree candidate, is also avoided.

Ignoring this aspect of the connectivity list maintenance scheme severely reduces the overall efficiency of the tiebreaking MDA implementation. Consider, for example, the nine point problem of Section 7 with 1089 nodes. During its simulated factorization, if all nodes introduced into the connectivity list remain members until eliminated or absorbed into a supernode as a nonrepresentative member, 72% of the connectivity updates and 96% of the connectivity recalculations are performed on nodes with degrees larger than **maxcondeg**. This translates into an increase in the overall execution time by a factor of approximately two. If the graph's minimum degree falls below **maxcondeg**, however, it does not pay to lower the **maxcondeg** value to match this change.

In all of the test problems considered in Section 7, the general trend during the elimination process is towards the selection of nodes of increasing degree, with blocks of nodes possessing a common degree eliminated without interruption. The number of nodes eliminated with a degree less than the current **maxcondeg** value varies between problems, but in each case this class of node selections was a definite minority. In addition, it was found that when the minimum degree dropped below the **maxcondeg**

level, in general relatively few nodes were eliminated at this reduced value before the minimum degree once again equalled **maxcondeg**. From these observations it is clear that the maxcondeg value should not be lowered when the minimum degree of the current graph is temporarily reduced. The connectivity list is kept intact and all necessary updating is performed in anticipation of the minimum degree's return to the **maxcondeg** level.

It is not difficult to create an artificial example in which the connectivity list consists of a majority of the uneliminated nodes of the current quotient graph throughout the elimination process, or a problem in which the overlying index becomes comparable in length to the connectivity list itself. Fortunately, during the application of the tiebreaking MDA to the more practical examples of Section 7, the connectivity list and indexes did not exhibit these undesirable characteristics. In general the connectivity list consisted of only a small fraction of the uneliminated nodes and the number of different connectivity values was usually very small.

5.2 Reachable Set Storage

As a direct consequence of the introduction of connectivity values, the tiebreaking MDA implementation exhibits a large increase in the number of reachable set requests when applied to most problems. To calculate the connectivity of node x, its reachable set and the reachable set of each member of x's reachable set is required. To avoid the redundant calculation of reachable sets, long term storage of their membership was introduced. Once a node's reachable set has been requested, it is stored in an updated form until the node is eliminated or absorbed into an uneliminated supernode.

The introduction of long term storage of reachable sets substantially increases the storage requirements of the tiebreaking MDA from those of the normal algorithm. For each test problem considered by Section 7, however, the maximum level of primary storage required for reachable sets was substantially less than the storage needed to represent the lower adjacency structure of L using the uncompressed scheme discussed in [6]. Between problems the maximum level of primary reachable set storage ranged from 30% to 70% of L's storage requirements. As a result, the storage requirements of

the tiebreaking MDA are comparable to that of the symbolic factorization step, which requires the simultaneous storage of A's adjacency structure and L's lower adjacency structure.

Unfortunately, without prior experience the reachable set storage requirements for a particular problem cannot be accurately predicted before the application of the tiebreaking MDA. However, a general allocation rule is discussed in Section 7 to assist in the allocation of reachable set storage for practical problems.

The elimination graph model experienced similar problems when allocating storage for adjacency sets. In general it could not be guaranteed that sufficient storage was allocated without attempting to order the particular problem. These difficulties resulted in the rejection of the elimination graph model as a basis of MDA implementations. The storage of reachable sets, however, does not present such a formidable problem. If a shortage of storage is encountered, all saved reachable sets do not have to be maintained. The necessary information to rebuild a reachable set is always stored implicitly within the current quotient graph. In the current implementation of the tiebreaking MDA reachable sets were only stored to help reduce execution times of the resulting implementation.

Although they will not be discussed in detail in this paper, a number of possible schemes could be introduced to alleviate the storage difficulties of the existing implementation. The most obvious scheme is to introduce a more sophisticated routine to manage the central storage of reachable sets. If the free space list for set storage was empty and more space required, reachable sets not currently in use could be selected for destruction. Various criteria for this selection process could be developed. The only restriction is that before each elimination step the reachable sets of the selected node and all nodes whose connectivity may change after the elimination must be stored. Using such a scheme it would be possible to reduce the storage allocation for reachable sets when memory was a problem, or increase the allocation if a faster execution time was desired. In addition, such an enhancement would essentially alleviate the problem of whether a particular storage allocation was sufficient to allow the ordering program to run to completion on a given problem.

5.3 Degree and Reachable Set Updating

When a node, t, is eliminated from the current quotient graph the reachable sets, and hence degrees, of all nodes $x \in reach_{G^q}(t, S)$ are affected. To update the reachable set and degree of each x, all affected reachable sets could be completely recalculated using the new quotient graph. However, the reachable sets of each node x prior to the elimination of node t must be recorded in reachable set storage. As a result, an alternative updating approach is possible in which each stored reachable set $reach_{G^q}(x, S)$ is simply modified by removing t and augmenting the set with any new members.

In step 4g of the enhanced MDA algorithm presented in Section 4.6, a difference set, *diff(x)* := *saverch(t)* − *saverch(x)*, is calculated for each member of t's old reachable set during the updating of satellite connectivities. As described in Section 4.5, *diff(x)* represents the new members of x's reachable set which are now reachable through the eliminated node t. Instead of totally recalculating each reachable set, the difference sets can be used to augment the stored reachable set of each node x.

The difference set degree updating approach was selected for the implementation of the tiebreaking MDA. The degree updating shown in step 4(i)i of the algorithm in Section 4.6 is expanded to include reachable set updating and moved to step 4g, which was originally composed entirely of connectivity updating. The modified step is illustrated in the following revised fragment of the original algorithm.

(g) For each $x \in saverch(t)$

i. $diff(x) := saverch(t) - saverch(x)$

ii. $reach_{G^q}(x, C(S)) := (saverch(x) \cup diff(x)) - \{t\}$

iii. $D_x := size(reach_{G^q}(x, C(S))$

iv. For each $y \in (saverch(x) - (saverch(t) \cup \{t\}))$

$conn(y) := conn(y) + 1/2 * superlength(x)$

$* size(diff(x) \cap reach_{G^q}(y, C(S)))$

If explicit recalculation is used in place of the updating schemes outlined above, the execution times for the deficiency tiebreaking MDA recorded in Section 7 can be expected to increase 14 to 18% for most problems.

6 Secondary Tiebreaking Strategies

Even with the deficiency tiebreaking strategy proposed in Section 4, the enhanced algorithm is still not completely deterministic. Opportunities for the application of secondary tiebreaking strategies arise when more than one node is of minimum degree of and minimal deficiency. In general such opportunities arose in approximately 45 to 70% of all node selections when the tiebreaking MDA was applied to the test problems of Section 7. In the present form of the tiebreaking MDA a node is arbitrarily selected from the group of qualified candidates at each elimination step. Although the deficiency tiebreaking strategy will be shown to have improved the stability and fill level of orderings, the precise manner in which these ties are broken still has a significant impact on the quality of orderings produced for a subset of the problems.

Although deficiency tiebreaking is the primary focus of this paper, the remainder of this section briefly outlines three different criteria which were pursued as secondary tiebreaking strategies for the deficiency tiebreaking MDA. The central goals of the strategies are the same as those discussed in Section 4 for deficiency tiebreaking. No attempt will be made to detail the implementation of these strategies.

The strategies outlined in this section are based upon criteria which emphasize the importance of increasing the global nature of the complete algorithm. The tiebreaking MDA is based upon the approximate local minimization of fill at each elimination step. No consideration is made of how the elimination of a particular node affects the uneliminated nodes of the remaining graph. Each secondary tiebreaking scheme chooses the minimum degree, minimal deficiency node at each step whose elimination most benefits the uneliminated nodes of the remaining graph. Different interpretations of what is viewed as being the most beneficial results in the three different approaches taken.

Although a secondary tiebreaking scheme cannot affect the level of fill suffered at a particular elimination step, the introduction of the different sets of fill edges can change the number of additional fills produced during the elimination of the graph's remaining nodes. The first secondary tiebreaking scheme enhances the existing strategy by selecting the minimum degree, minimal deficiency candidate whose set of new fill

edges results in the most positive change in overall connectivity of the remaining graph. Using this additional criterion in the selection process, it is hoped that the general level of connectivity in each successive elimination graph will be kept higher than if random selections were made. By controlling the connectivity level in this fashion, it may be possible to subsequently select minimum degree nodes with larger connectivities and reduce the amount of fill experienced.

The second secondary tiebreaking scheme is essentially a refinement of the previous strategy. If the tiebreaking scheme is to have any noticeable effect on the quality of orderings produced, the elimination of each block of minimum degree nodes must be disturbed in some fashion. In the strategy outlined in the previous subsection, among all potential candidates the node chosen for elimination had to have the most positive effect on the connectivity of the remaining graph. As a result, a node could be selected based on the fact that its elimination would increase the connectivity level of a group of higher degree nodes, whose elimination was not imminent. This would not directly affect the nodes in the minimum degree block.

Instead of calculating the effect of a candidate's elimination on the connectivity of all remaining uneliminated nodes, only minimum degree satellite nodes and members of $reach_{G^q}(candidate, S)$ whose degree after the elimination will not be larger than the candidate's minimum degree are considered. The minimum degree, minimal deficiency candidate whose elimination has the most positive effect on the connectivity of this subset of uneliminated nodes will be selected.

The previously outlined secondary tiebreaking strategies have both concentrated upon how a candidate's elimination would affect the connectivity of the remaining graph. Connectivities, however, only form the basis of the tiebreaking scheme for a node selection process which is dominated by nodal degrees. The criterion of the final secondary tiebreaking strategy attempts to focus attention on the degree of the graph's remaining nodes.

The elimination of different minimum degree, minimal deficiency candidates can have a substantially different affect upon the distribution of degree values within the remaining graph. The third secondary tiebreaking strategy selects the candidate whose

elimination would result in the most positive change in the size of the block of minimum degree nodes. This helps to prolong the elimination of lower degree nodes. In general it is more desirable to eliminate a minimum degree node with its smaller fill producing potential. In addition, by keeping the minimum degree block as large as possible, the primary tiebreaking scheme can be applied more effectively.

7 Numerical Experiments

This section summarizes the numerical experiments conducted to evaluate the MDA enhanced by the proposed tiebreaking strategies. Although an emphasis is placed upon the deficiency tiebreaking approach, implementations of the normal MDA and versions of the algorithm enhanced by both primary and secondary tiebreaking strategies were applied to a wide variety of test problems. In addition, the effect of an ordering's quality upon the factor and solve steps of the solution process was investigated for a subset of the test problems.

Three programs, referred to as *norm*, *tie* and *sec*, were developed to permit the testing of the proposed tiebreaking strategies. The program *norm* is an implementation of the normal minimum degree algorithm presented in Section 2 and follows many aspects of the implementation approach suggested by George and Liu [6]. The *tie* program is an implementation of the normal algorithm enhanced by deficiency tiebreaking. Three important aspects of this implementation were discussed in Section 5. Finally the program *sec* actually refers to a collection of three different programs in which *tie* is augmented with one of the three secondary tiebreaking strategies of Section 6. An additional program, referred to as *solve*, was implemented and encompasses the remaining three steps of the solution process. A brief description outlining the particular algorithms selected for this program is given in Section 7.4.

7.1 Testing Environment

All testing discussed in this section was carried out on a Sun 3/50, running SunOS 3.2. All code for programs under analysis was written in Sun Pascal. Timings of the

programs were carried out using the built in Sun Pascal function `clock`.

7.2 Test Problems

A wide variety of symmetric, sparse matrices were selected for the testing of the ordering algorithms. In some cases only the nonzero structure of the system was provided, while for other problems the complete system was available. This subsection provides a brief description of each group of test problem and its origins.

For the first test problem only the nonzero structure of the matrix was available. The graph was created by randomly selecting an initial size, r, for each node's initial adjacency set and by choosing r nodes of the graph at random to be members of its adjacency set. The average degree of the 400 node graph created in this fashion was 2.49 and is referred to as the *random graph* in subsequent discussion.

The next two groups of problems arise from the solution of partial differential equations on an nxn grid. Each problem results in a graph of $N = (n+1)^2$ nodes. The first set of problems arises from the solution of the Poisson equation on the unit square, using Dirichlet boundary conditions and a five point finite differencing molecule. The four values $n = 19, 29, 39$ and 99, were selected to produce graphs of 400, 900, 1600 and 10000 nodes respectively.

The second set of problems arise from the application of finite elements to a partial differential equation. The resulting system has a banded matrix, which is equivalent in structure to a matrix arising from the application of finite differences to a differential equation using a nine point molecule. Only grids with $n = 2^t$ were considered. The four values $t = 4, 5, 6$ and 7 were selected to produce graphs consisting of 289, 1089, 4225 and 16641 nodes respectively.

For both the five and nine point problems the initial labelling of their corresponding graph was based upon a standard lexicographic ordering of the grid's unknowns.

The next pair of problems consist of the nonzero structures of two systems used by George and Liu [3, 6] when testing their implementations. The graphs are typical of those that may be found in the study of heat conduction or structural analysis. The

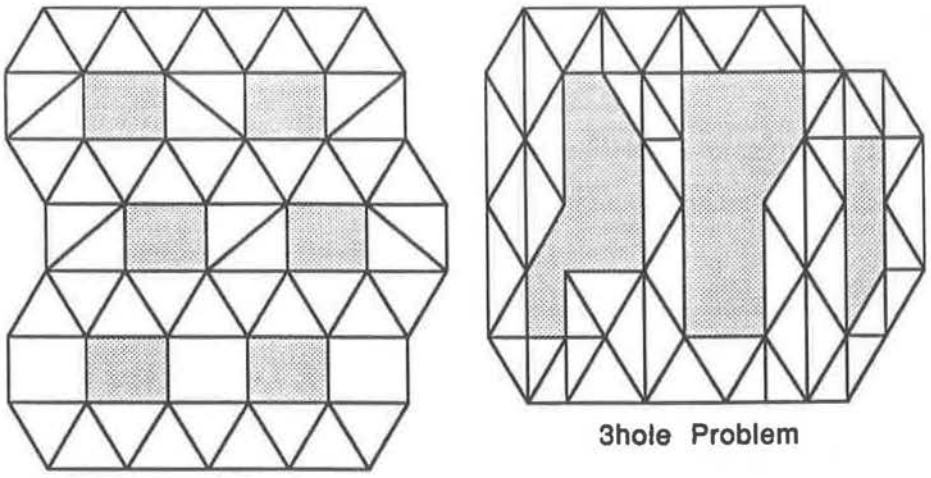

Figure 7: The Initial Mesh of *6hole* and *3hole*

triangular mesh structures illustrated in Figure 7 were used to derive the *6hole* and *3hole* test problems. Each triangle of the basic structures is subdivided by a factor of 3 resulting in 9 smaller triangles. Both graphs produced by this method consisted of 316 nodes and were given an arbitrary initial labelling.

The final group of 14 test problems was selected from the Harwell-Boeing collection of sparse matrix examples. Table 2 contains a list of the selected problems, their size and a short description briefly outlining the origin of each system. The key listed for each problem is the original key assigned to the system in the Harwell-Boeing collection and will be used to identify each problem during the remainder of this section. All matrices selected were symmetric but the actual nonzero values of the systems were only available for approximately half of the problems.

7.3 Deficiency Tiebreaking Test Results

So that comparison could be made between the tiebreaking MDA and the original quotient graph MDA, both the *norm* and *tie* programs were applied to all test problems discussed in the previous section. The goals of the new tiebreaking strategy were to reduce the fill observed for the MDA orderings and to increase the fill stability of orderings for problems sensitive to different initial labellings. To study the effectiveness

Key	N	Description
ERIS1176	1176	Symmetric Pattern of Erisman, Summer 1973
BCSPWR09	1723	Symmetric Structure Representation of Western US Power Network
BCSSTK09	1083	Symmetric Stiffness Matrix, Square Plate Clamped
BCSSTK10	1086	Symmetric Stiffness Matrix, Buckling of Hot Washer
CAN1072	1072	Symmetric Pattern from Cannes, Lucien Marro, June 1981
DWT2680	2680	Symmetric Connection Table from DTNSRDC, Washington
LSHP3025	3025	Symmetric Matrix form A. George's L-shaped Problems 1978
NOS3	960	Symmetric Matrix, FE Approximation to Biharmonic Operator on Plate
PLAT1919	1919	Splatzman Symmetric Finite Difference, Three Ocean Model
BLCKHOLE	2132	Connecitivity Structure of a Geodesic Dome on a Coarse Base
BCSSTK19	817	Symmetric Stiffness Matrix, Part of a Suspension Bridge
685BUS	685	Admittance Matrix, 685 Bus Power System, D.F.Tylavsky, July 1985
1138BUS	1138	Admittance Matrix, 1138 Bus Power System, D.F.Tylavsky, July 1985
BCSSTK26	1922	Symmetric Stiffness Matrix, Reactor Containment Floor-Section

Table 2: Selected Harwell-Boeing Test Problems

of the tiebreaking MDA with respect to either goal it is necessary to consider more than one initial labelling for each problem. As a result, the testing of each problem was expanded to include five random relabellings of each graph, in addition to the original labelling.

Table 3 provides a summary of the effect of the tiebreaking enhancement of the MDA upon fill levels for all 25 problems. The mean fill value and standard deviation for both the *norm* and *tie* programs was calculated using the fill values of all six initial orderings for each problem. The final column of the table shows the percentage by which the mean fill value of the normal algorithm is reduced by the tiebreaking enhancement. All fill values record the number of new fill entries in L and not the total number of nonzero entries in the factor.

The table also records the mean number of CPU seconds required by the *norm* and *tie* programs when applied to the six initial orderings of each problem. It should be noted that for testing purposes the 25 problems were divided into three broad classes based upon their storage requirements. (Dynamic memory allocation is not possible in

	Normal MDA			Tiebreaking MDA			
Key	Mean Fill	Std. Dev.	Mean Time	Mean Fill	Std. Dev.	Mean Time	% Improvement in Mean Fill
random graph	977	20.5	3.08	966	8.9	3.59	1.1
5Pt 400	2633	127	3.48	2447	50.5	4.69	7.1
5Pt 900	7898	412	8.59	7235	174	11.7	8.4
5Pt 1600	17092	562	18.7	15135	319	22.1	11.4
5Pt 10000	182168	3049	260.7	142119	1104	176.5	22.0
9Pt 289	2576	70.7	2.52	2417	96.9	3.93	6.2
9Pt 1089	18396	674	13.1	15629	630	18.7	15.0
9Pt 4225	109757	5537	67.9	90224	3186	92.6	17.8
9Pt 16641	629647	45368	519.3	490111	23742	415.2	22.2
3hole	1589	15.2	2.08	1524	22.4	2.85	4.1
6hole	1751	59.6	2.16	1684	18.4	2.82	3.8
ERIS1176	648	36.3	25.8	553	14.3	24.9	14.7
BCSPWR09	2149	16.7	14.7	2088	14.6	10.1	2.8
BCSSTK09	48154	3623	41.7	42244	4097	71.3	12.3
BCSSTK10	11811	260	23.4	11486	383	36.5	2.8
CAN1072	14032	391	22.1	13787	173	28.2	1.8
DWT2680	40604	707	61.9	39025	597	82.0	3.9
LSHP3025	63859	2138	38.0	61626	1709	52.5	3.5
NOS3	22804	670	20.8	20737	385	30.8	9.1
PLAT1919	49751	524	42.9	50996	1248	69.3	-2.5
BLCKHOLE	47220	729	29.0	45690	1467	40.0	3.2
BCSSTK19	4068	167	8.82	4098	254	10.1	-0.7
685BUS	1703	46.9	5.25	1614	28.5	5.69	5.2
1138BUS	689	9.7	6.35	644	3.6	4.79	6.5
BCSSTK26	26791	583	78.6	26977	676	89.7	-0.7

Table 3: Ordering Summary

Pascal and a recompilation of the programs for each different problem was avoided.) It was observed that the amount of excess memory declared for reachable set and adjacency set storage could effect the timings observed as much as 15 to 20%. As a result, rigorous comparison of ordering times should only be made within the confines of a particular test problem.

Of all test problems, the MDA enhanced with deficiency tiebreaking was the most successful when applied to the five and nine point problems sets. The mean fill value for the five point problems was improved by 7.1 to 22%, while the mean fill was improved by 6.2 to 22.2% for the nine point problems. The fill improvement is the most pronounced for the largest problem in each class, with the *tie* fill levels for all initial orderings well below their *norm* counterparts.

These values represent significant reductions in fill levels. In each class of problems the fill improvements observed increased as larger and larger graphs were considered. This trend was paralleled by an increasing proportion of tiebreaking opportunities in which the deficiency tiebreaking strategy actually contributed towards a node selection. For both the five point problem of 10000 nodes and the nine point problem of 16641 nodes, in approximately 95% of all tiebreaking opportunities there were nodes of minimum degree with differing connectivity values.

Although six different orderings is a relatively small sample of initial labellings, the standard deviation values recorded in Table 3 for the five point problems show that for each system the tiebreaking MDA has also increased the fill stability of the orderings. The standard deviation values have been reduced by 43 to 63%. The fill values for the tiebreaking ordering of the nine point problems also showed an increased stability except for the 289 node problem.

For both the five and nine point problem sets, the CPU time required by the *tie* program was comparable to the requirements of the *norm* program. One very encouraging trend observed in these timings is the general reduction in the CPU requirements of the *tie* program with respect to the *norm* program as the size of the problems increased. In fact for the largest problem in each set, the tiebreaking algorithm required approximately 20 to 30% less CPU time. It is possible for the tiebreaking program

to require less time because of the reachable set updating scheme introduced to the enhanced algorithm, which allows reachable set recalculations to be avoided.

The success of deficiency tiebreaking on the five and nine point problems is in stark contrast to the results for the *random graph*. The ineffectiveness of the tiebreaking scheme on this problem is a direct consequence of the small number of tiebreaking opportunities for which the scheme was able to affect the node selection process. In only 12% of all tiebreaking opportunities was there differing connectivity values amongst the minimum degree candidates.

These results are typical of random graphs. In general, graphs produced in this fashion lack the local nature of their adjacency structure, with very few short node cycles being present. They typically have very low initial connectivity values or no connectivity at all, as was the case for the random graph considered. The connectivity of nodes in several randomly produced graphs was traced through the factorization process. In each case the connectivity of essentially all nodes being selected was 0 until 80 to 90% of the nodes had been eliminated. At this point, the tiebreaking scheme applied to the remaining node selections has little effect on the overall fill level observed. Fortunately, very few practical problems exhibit such a total lack of local nature to their graphs.

The deficiency tiebreaking MDA was moderately successful on the *3hole* and *6hole* grid problems, reducing the mean fill levels by 4.1 and 3.8% respectively. In addition, the fill reduction for the *6hole* problem was accompanied by a large increase in fill stability. Once again the CPU time requirements of the *tie* program were comparable to those of *norm*.

The effect of the tiebreaking strategy on the problems selected from the Harwell-Boeing sparse matrix collection were mixed. For approximately one half of the problems, the *tie* program produced orderings with substantially reduced levels of fill, in comparison to the *norm* orderings. The reduction in the mean fill values ranged from 3.5% for the *LSHP3025* problem to 14.7% for the *ERIS1176* problem. The majority of this subset of problems also showed a substantial reduction in the standard deviation of fill values for the *tie* orderings in comparison to the *norm* orderings.

Of the remaining seven Harwell-Boeing test problems four showed a very slight improvement in fill values. Their ordering stability increased or decreased slightly, except for *CAN1072*, which showed a substantial reduction in the standard deviation of fill values for the *tie* program. The tiebreaking strategy essentially had no effect upon the mean fill levels for the *BCSSTK19* and *BCSSTK26* problems and both showed a slight reduction in ordering stability. Finally, for *PLAT1919* an increase in the mean fill value of 2.5% for the *tie* program ordering was accompanied by a large decrease in ordering stability as shown by the standard deviation values.

Once again the CPU seconds required by the *tie* ordering program were comparable to the time requirements of *norm*. In addition, within each of the Harwell-Boeing test problems, timings are essentially independent of the initial ordering. This stability is also observed in all other test problems.

The tiebreaking strategy has clearly been shown to be very effective on the five and nine problems, and it is suspected that the scheme would be equally effective on problems arising from other discretization molecules or banded matrices. However, it is difficult to predict to what extent the tiebreaking MDA will reduce fill for other more general problems. It is clear that the initial graph of the system must have a certain local nature, missing from most random graphs, for the tiebreaking algorithm to have sufficient opportunity to influence the selection process and effect fill levels. The initial connectivity of a graph is often a good indication of this property but many exceptions do exist. Connectivity only considers the number of three member cycles and totally ignores the four member cycles of these graphs, which quickly result in increased connectivity as the first elimination take place. Except for this local structure, no other common characteristic of the initial graphs was identified, for which *tie* was most successful. There are many variable graph characteristics which together contribute towards the overall fill level of a given ordering.

As outlined in Section 5.2, in the current implementation of the *tie* program, once a node's reachable set is requested it is stored until the node is eliminated or absorbed into an uneliminated supernode as a nonrepresentative member. It is not difficult to create an example for which the storage of reachable sets would quickly expand to unacceptable levels. However, for all 25 test problems, the reachable set storage

requirements were well bounded and in each case the amount of primary reachable set storage required was comparable to the size of the communal adjacency list storage. It is important to note that in the current implementation the structure used for reachable sets is very simplistic and requires one integer of overhead storage for each integer of primary storage.

For the five and nine point problems, the primary storage required for reachable sets ranged from 0.94 to 1.3 times the size of the graph's initial adjacency lists. For the *3hole* and *6hole* problems, the size of the primary reachable set storage required was approximately the same or slightly smaller than the size of the appropriate adjacency lists. Finally, the ratio of the maximum primary reachable set storage required to the size of the adjacency lists for the Harwell-Boeing test problems ranged from 0.38 for *ERIS1176* to 2.0 for *BCSSTK09*. The latter problem was an exception, with the next highest reachable set storage requirement being for the *BLCKHOLE* problem with a ratio of 1.4. For the majority of Harwell-Boeing problems, however, the required size of primary reachable set storage was less than the size of the problem's adjacency lists. As a conservative rule of thumb, selecting the size of primary reachable set storage to be 1.4 times the length of the adjacency list structure seems the most reasonable when very little is known about a new problem.

As previously mentioned, the reachable set storage scheme selected for implementation is very simplistic. It represents the most storage intensive extreme of a whole spectrum of possible storage schemes. The favorable timing results observed for the *tie* program will allow for considerable experimentation with storage schemes which are more restrictive. For many problems, such as the large five and nine point problems, the size of the reachable set storage would have to be substantially restricted, forcing many reachable set recalculations, before the *tie* program would require more CPU time than the *norm* module.

7.4 Factor and Solve Timings

For a select group of problems, the effect of a fill reduced ordering upon the remainder of the solution process was investigated by comparing the factorization and solution

step timings for different orderings. For each of the nine problems considered, the system was solved using the *norm* ordering exhibiting the highest level of fill and the *tie* ordering with the lowest level of fill.

The timings were taken using the *solve* program previously mentioned. The program consists of separate independent modules for the symbolic factorization, explicit factorization and solve steps of the solution process. The explicit factorization and solve steps were implemented using data structures and algorithms selected from the discussion of these two steps presented in [6]. The system's original matrix before factorization and the factor L after the Cholesky decomposition are both stored in the same data structure, which is based upon the uncompressed scheme of George and Liu. The same data structure can be used to store the original matrix and the factor because the inner product form of Cholesky factorization was used to create L. Finally, in the *solve* module the solution of the two lower triangular systems, $Ly = Pb$ and $L^T Px = y$, is calculated using the outer product and inner product forms of triangular solution respectively.

Table 4 summarizes the factorization and solve timings observed for each of the nine problems considered. Two lines of data are recorded for each problem. The *norm* ordering was used to produce the data of the first line, while the data recorded in the second was observed for the *tie* ordering. The values in the third column of the table record the number of fills entries for each ordering. The final two columns record the CPU seconds required to factor the original matrix and solve the two triangular systems.

In each case, the CPU seconds saved in the factor and solve steps was substantial when fill reduced orderings were used. As expected, due to the higher complexity of the factorization process, the reduction in the factor time began to outweigh the reduction in solve time as the size of problems increased. In all cases the savings in factorization time alone more than recovered any additional time required by the *tie* program over that needed by *norm*. For some larger problems the savings in factor time were larger than the *tie* ordering time itself. In addition, for a given reduction in fill, the factorization time in most cases, is reduced by a larger factor. For example, a 24% reduction in fill for the 10000 node five point problem translates into a 42% reduction

Name		Fill	Factor Time	Solve Time
5Pt 400	Normal	2792	5.93	1.42
	Tiebreaking	2393	4.68	1.28
5Pt 900	Normal	8334	22.4	3.97
	Tiebreaking	7007	17.0	3.4
5Pt 1600	Normal	17653	57.5	8.10
	Tiebreaking	14691	42.1	6.97
5Pt 10000	Normal	184646	1287	79.6
	Tiebreaking	140470	749.9	63.1
9Pt 289	Normal	2676	7.28	1.45
	Tiebreaking	2278	5.63	1.32
9Pt 1089	Normal	19223	81.3	8.97
	Tiebreaking	14586	48.7	7.23
9Pt 4225	Normal	115643	816.4	50.4
	Tiebreaking	83742	416.6	38.3
BCSSTK09	Normal	54062	529.9	24.0
	Tiebreaking	37501	269.0	17.7
NOS3	Normal	23570	137.3	11.7
	Tiebreaking	20118	103.8	10.3

Table 4: Solution Time Summary

in the factorization time, while for BCSSTK09 a 30% reduction in fill decreases the corresponding factorization time by 50%.

7.5 Secondary Tiebreaking Test Results

As described in Section 6, three different secondary tiebreaking schemes were proposed for the deficiency tiebreaking MDA. The three strategies were implemented by augmenting the *tie* program to produce the *sec* group of programs. Each *sec* program was applied to all test problems described in Section 7.2.

The first two secondary tiebreaking strategies failed to meet the goals of Section 6. There was no observed improvement in fill levels for the orderings of the test problems, using either secondary tiebreaking algorithm, in comparison to the orderings produced by the MDA enhanced by deficiency tiebreaking alone. Nor was there a significant improvement in ordering stability. In addition ordering times were dramatically increased.

When the deficiency tiebreaking MDA was augmented by the third secondary tiebreaking scheme, the resulting program was very successful for the nine point problems and the Harwell-Boeing example referred to as *BCSSTK19*. For the remainder of the test problems no improvement in ordering quality was observed.

For each of the four nine point problems, fill levels were completely stabilized at a very low fill level. For each of the group's three smaller problems, the fill level for each initial labelling differed by only one fill entry from the lowest fill level observed when *tie* was applied to the same group of initial orderings. The 16641 problem also exhibited a similar stabilization and each fill value was within 0.3% of the lowest fill value observed for a *tie* ordering of the problem. The timings for the *sec* program, using the third secondary tiebreaking criterion, are not as high as for the other two strategies. However, CPU time requirements did increase from those of *tie* on the same graphs by a factor ranging from 1.4 for the 289 node problem to 3.0 for the 16641 node version. This increase in the relative time requirements is matched by an increase in the average number of nodes considered by each secondary tiebreaking. In addition, the timings were found to be relatively independent of the initial ordering.

When the *tie* ordering program was applied to the problem *BCSSTK19*, there was

no apparent improvement in the quality of orderings from those of the *norm* program. However, in comparison to the *norm* orderings, a 9% reduction in the mean fill level for orderings of this problem was achieved by the *sec* program. In addition, the standard deviation was reduced from 167 for *norm* to 33. The ordering time required by the *sec* program is 3.0 and 2.6 times that required by *norm* and *tie* for this problem.

8 Concluding Remarks

This paper has introduced the use of nodal deficiency as the basis of a tiebreaking strategy for the minimum degree algorithm. Deficiency tiebreaking has been successfully integrated into the quotient graph form of the MDA and has been shown to be compatible with the mass elimination of indistinguishable sets represented by uneliminated supernodes. To avoid the practical limitations of Rose's minimum deficiency algorithm [10], the *connectivity list* data structure was introduced to maintain the deficiency values of a strictly regulated subset of uneliminated nodes.

The MDA enhanced by deficiency tiebreaking was found to produce orderings with improved quality for a variety of different sparse problems. The goals of reduced fill and increased fill stability were met, at least in part, for more than two thirds of the test problems. For each sparse example, the CPU requirements of the deficiency tiebreaking MDA implementation were comparable to the requirements of the normal MDA implementation. In addition, the timings of the tiebreaking MDA were found to be essentially independent of the initial labelling assigned to the graph of the system's matrix.

The deficiency tiebreaking strategy was found to be the most effective on the five and nine point finite differencing problems, with reductions of up to 22% in the mean fill level observed for the two largest sparse systems. For the majority of problems in these group a substantial increase in the stability of fill levels with respect to the initial labelling was also observed. The enhanced MDA was generally ineffective, however, on randomly produced graphs. This emphasized that if the tiebreaking strategy is to have a positive impact on ordering quality, the initial graph of a system's matrix must exhibit a certain *local nature*. The tiebreaking strategy is less successful on systems

whose initial graphs have few short node cycles. In such cases insufficient opportunities to apply the tiebreaking criterion arise before the majority of node eliminations have occurred.

Implementations of the deficiency tiebreaking MDA discussed introduced the long term storage of reachable sets, substantially increasing the storage requirements of the enhanced MDA implementation. However, the increased levels of storage, introduced to reduce timings, are not essential to the execution of the algorithm because reachable sets are always implicitly represented by the current quotient graph. The favorable timings observed for the tiebreaking MDA will permit the consideration of less storage intensive implementations as practical alternatives.

Finally, this paper considered three secondary tiebreaking strategies for the deficiency tiebreaking minimum degree algorithm. One secondary tiebreaking strategy, which attempts to maximize the number of minimum degree nodes after each elimination, dramatically stabilized the fill levels of orderings for the nine point problems and reduced the fill levels observed for the orderings of one symmetric stiffness matrix. Otherwise, the secondary tiebreaking schemes failed to meet the goals of reduced fill levels and increased fill stability.

References

[1] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software*, 9(3):302–325, September 1983.

[2] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman. Yale sparse matrix package I. the symmetric codes. *International Journal For Numerical Methods In Engineering*, 18(8):1145–1151, 1982.

[3] J. A. George and J. W. H. Liu. An automatic nested dissection algorithm for irregular finite element problems. *SIAM Journal of Numerical Analysis*, 15(5):1053–1069, October 1978.

[4] J. A. George and J. W. H. Liu. A quotient graph model for symmetric factorization. In I. S. Duff and G. W. Stewart, editors, *Sparse Matrix Proceedings*, pages 154–175. SIAM, 1978.

[5] J. A. George and J. W. H. Liu. A fast implementation of the minimum degree algorithm using quotient graphs. *ACM Transactions on Mathematical Software*, 6(3):337–358, September 1980.

[6] J. A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Inc., New Jersey, 1981.

[7] J. A. George and J. W. H. Liu. The evolution of the minimum degree ordering algorithm. (to appear in SIAM Review), 1988.

[8] J. W. H. Liu. Modification of the minimum-degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software*, 11(2):141–153, June 1985.

[9] H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, 3:255–269, 1957.

[10] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Read, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, New York, 1972.

[11] W. F. Tinney and J. W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proceedings of the IEEE*, 55(11):1801–1809, November 1967.

[12] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):77–79, March 1981.