Printing Requisition / Graphic Services

Dept. No. 13875

Title or Description A Guide to Completion -- Staple

Date Aug. 31/82 **Date Required** ASAP **Account** 126-6602

Signature [signature] **Signing Authority** Alan Lenz

Department GS **Room** 3182 **Phone** 2191

Delivery ☐ Mail ☐ Via Stores ☑ Pick-up ☐ Other

1. Please complete unshaded areas on form as applicable. (4 part no carbon required).
2. Distribute copies as follows: White, Canary and Pink—Printing, Arts Library or applicable Copy Centre Goldenrod—**Retain.**
3. On completion of order, pink copy will be returned with printed material. Canary copy will be costed and returned to requisitioner, **Retain as a record of your charges.**
4. Please direct enquiries, quoting requisition number, to Printing/Graphic Services, Extension 3451.

Reproduction Requirements ☑ Offset ☐ Signs/Repro's ☐ Xerox **Number of Pages** 36 **Number of Copies** x 50

Type of Paper Stock ☑ Bond ☐ Book ☐ Cover ☐ Bristol ☐ Supplied

Paper Size ☑ 8½ x 11 ☐ 8½ x 14 ☐ 11 x 17

Paper Colour ☐ White ☐ Other **Ink** ☑ Black

Printing ☑ 1 Side ☐ 2 Sides **Numbering** to

Binding/Finishing Operations ☑ Collating ☐ Corner Stitching ☐ 3 Ring ☐ Tape ☐ Plastic Ring ☐ Perforating

Folding Finished Size 3 STAPLES **Cutting** Finished Size

Special Instructions
FRONTS & BACKS SUPPLIED
THREE STAPLES ALONG LEFT SIDE

Cost: Time/Materials	Fun.	Prod.Un. Cl.	Prod.Opr. No.	Mins.	Total
Signs/Repro's	1				
Camera	2				
Correcting & Masking Negatives	3				
Platemaking	4		1		
Printing	5		11		
Bindery	6		0	40	
			23	15	
Sub. Total Time					

Film Qty / Size	**Plates** Qty / Size & Type	Sub. Total Materials	
Paper Qty / Size	**Plastic Rings** Qty / Size	Prov. Tax	
Outside Services		Total	

Graphic Services
May 68 482-2

UNIVERSIDAD POLITECNICA DE VALENCIA

Jose Luis Oliver Herrero.
Universidad Politécnica de Valencia.
Dpto. Ing. Mecánica y Materiales.
P. O. Box 22012.
46022 VALENCIA.
SPAIN.
Phone (34)-6- 361 50 51, Ext. 128
Fax # (34)-6-360 31 78

19th September, 1988, Valencia,

UNIVERSITY OF WATERLOO
Department of Computer Science
Waterloo, Ontario N2l 3G1
CANADA

Dear Mrs. DeAngelis:

Thank you for your kindness. We have received the reports CS-78-12 and CS-78-30.

Enclosed, I send you check number 002182/2076 payable to COMPUTER SCIENCE DEPARTMENT, UNIVERSITY OF WATERLOO, for $ CAD. 8.

Sincerely yours

Prof. J. L. O. Herrero.

recd Oct. 3/88

University of Waterloo
Department of Computer Science
Waterloo, Ontario N2L 3G1

August 16, 1988

INVOICE

Jose Luis Clivet Herrero,
Universidad Politecnica de Valencia,
Dpto. Ing. Mecanica y Materiales,
P.O. Box 22012,
46022 Valencia,
SPAIN

REPORT(S) ORDERED

CS-78-12, CS-78-30

Please be advised that I was out of stock of these reports and had to have copies made for you.

TOTAL COST $8.00

Would you please make your cheque or international bank draft payable to the Computer Science Department, University of Waterloo and forward to my attention.

Thanking you in advance.

Yours truly,

Susan DeAngelis (Miss.)
Research Report Secretary
Computer Science Dept.

/sd

Encl.

UNIVERSIDAD POLITECNICA DE VALENCIA

Jose Luis Oliver Herrero.
Universidad Politécnica de Valencia.
Dpto. Ing. Mecánica y Materiales.
P. O. Box 22012.
46022 VALENCIA.
SPAIN.
Fax # (34)-6-360 31 78

12th July, 1988, Valencia,

UNIVERSITY OF WATERLOO
DEPARTMENT OF COMPUTER SCIENCE
Waterloo, Ontario N2L 3G1
CANADA

Dear Sir:

We should be very grateful if you could indicate us how to get copies of the REPORTS:

36 pgs.

- J. A. GEORGE, J. W. H. LIU., "A Fast Implementation of the Minimun Degree Algorithm Using Qoutient Graphs", REPORT CS-78-12, Department of Computer Science, University of Waterloo, 1978a.

- J. A. GEORGE., J. W. H. LIU., "User Guide for SPARSPAK: Waterloo Sparse Linear Equations Package", REPORT CS-78-30, Department of Computer Science, University of Waterloo, 1978b.

Yours sincerely

Jose Luis Oliver Herrero.

Printing Requisition / Graphic Services

15064

1. **Please complete unshaded areas** on form as applicable.
2. **Distribute copies** as follows: White and Yellow to Graphic Services. **Retain Pink Copies for your records.**
3. On completion of order the **Yellow copy will be returned** with the printed material.
4. **Please direct enquiries, quoting requisition number and account number,** to **extension 3451.**

TITLE OR DESCRIPTION: CS-78-12

DATE REQUISITIONED: Aug. 15

DATE REQUIRED: Aug. 16

ACCOUNT NO.: 1 1 2 6 1 4 4 3 1 1 0 2 1

REQUISITIONER – **PRINT**

PHONE: 2192

SIGNING AUTHORITY: Lisa DeAngelis

MAILING INFO – NAME | DEPT. | BLDG. & ROOM NO. | ☐ DELIVER ☑ PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES: 36 NUMBER OF COPIES: 1

TYPE OF PAPER STOCK: ☑ BOND ☐ NCR ____ PT. ☐ COVER ☐ BRISTOL ☐ SUPPLIED ☐ ____

PAPER SIZE: ☑ 8½ x 11 ☐ 8½ x 14 ☐ 11 x 17 ☐ ____

PAPER COLOUR: ☑ WHITE ☐ ____ INK: ☑ BLACK ☐ ____

PRINTING: ☐ 1 SIDE ____ PGS. ☑ 2 SIDES ____ PGS. NUMBERING: FROM ____ TO ____

BINDING/FINISHING: ☑ COLLATING ☑ STAPLING ☐ ____ HOLE PUNCHED ☐ PLASTIC RING

FOLDING/PADDING: CUTTING SIZE:

Special Instructions

COPY CENTRE — OPER. NO. | BLDG. | MACH. NO.

DESIGN & PASTE-UP — OPER. NO. | TIME | LABOUR CODE

D 0 1
D 0 1
D 0 1

TYPESETTING — QUANTITY

P A P 0 0 0 0 0 | T 0 1
P A P 0 0 0 0 0 | T 0 1
P A P 0 0 0 0 0 | T 0 1

PROOF

P R F
P R F
P R F

NEGATIVES — QUANTITY | OPER. NO. | TIME | LABOUR CODE

F L M | C 0 1
F L M | C 0 1
F L M | C 0 1
F L M | C 0 1
F L M | C 0 1

PMT

P M T | C 0 1
P M T | C 0 1
P M T | C 0 1

PLATES

P L T | P 0 1
P L T | P 0 1
P L T | P 0 1

STOCK

0 0 1
0 0 1
0 0 1
0 0 1

BINDERY

R N G | B 0 1
R N G | B 0 1
R N G | B 0 1
M I S 0 0 0 0 0 | B 0 1

OUTSIDE SERVICES

$ ____ COST

TAXES – PROVINCIAL ☐ FEDERAL ☐

GRAPHIC SERV. OCT. 85 482-2

78-13

A Purely Homomorphic Characterization of Recursively Enumerable Sets

K. CULIK II

University of Waterloo, Waterloo, Ontario, Canada

ABSTRACT. Characterizations of recursively enumerable sets as mappings of equality and minimal sets are given. An equality (minimal) set is the set of all (minimal) solutions of an instance of the Post correspondence problem where the solutions are viewed as strings. The main result is that every recursively enumerable set can be expressed (effectively) as a homomorphic image of a minimal set. From the algebraic point of view this seems to be the simplest characterization of recursively enumerable languages. A corollary of the main result is the solution of an open problem formulated by A. Salomaa. A purely homomorphic characterization of regular sets is derived. How such a characterization can be obtained for various time and space complexity classes for languages is outlined.

KEY WORDS AND PHRASES: formal languages, recursively enumerable sets, homomorphic characterization, equality sets

CR CATEGORIES: 5.22, 5.23, 5.25, 5.27

In several recent proofs of decidability results (e.g. [2, 3]) it turned out to be of crucial importance to effectively check whether two homomorphisms h_1, h_2 on a free monoid Σ^*, generated by an alphabet Σ, are equal on a certain subset of Σ^*, or alternatively, to find the language of all $w \in \Sigma^*$ for which $h_1(w) = h_2(w)$. Such languages were explicitly introduced as equality sets in [5] and further studied. We introduce here another group of languages called minimal sets which are defined as minima of equality sets using the terminology of [4], i.e. a minimal set is a subset of an equality set L containing all strings which have no proper prefix from L. These sets were also implicitly considered in [5], where it was noted that each equality set is a star language (star event in the sense of [1]) and where it was also shown that each minimal set is the minimal star root of an equality set.

Alternatively we can say that an equality set is the set of all solutions of an instance of the Post correspondence problem; a minimal set is the set of all its minimal solutions. Here we consider an instance with lists A, B of length n over alphabet Σ as homomorphisms A, B from $\{1, 2, \ldots, n\}^*$ to Σ^*, and its solution as a string over $\{1, \ldots, n\}$. Each equality set also contains ϵ, the empty string.

We are looking for characterizations of all recursively enumerable languages by mappings of minimal or equality sets. Our main result is that every recursively enumerable language can be expressed as a homomorphic image of a minimal set. As simple modifications of the proof of this main result we will obtain several other more complicated characterizations of recursively enumerable sets, one of them already shown in [5]. We will also solve an open problem from [5]. We show that the regular set is characterized by a

This research was supported by the National Research Council of Canada under Grant No. A7403.

Author's address: Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

A Fast Implementation of the Minimum Degree Algorithm Using Quotient Graphs*

by

Alan George†
and
Joseph W.H. Liu††

Research Report CS-78-12

July 1978

* Research supported in part by Canadian National Research Council Grant A-8111.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1.

†† Systems Dimensions Ltd., 111 Avenue Road, Toronto, Ontario, Canada.

Abstract

We describe a very fast implementation of the minimum degree algorithm, which is an effective heuristic scheme for finding low-fill orderings for sparse positive definite matrices. Our implementation has two important features; first, it is competitive in terms of speed with other implementations known to the authors, and second, its storage requirements are independent of the amount of fill suffered by the matrix during its symbolic factorization. Some numerical experiments are provided, comparing the performance of this new scheme to some existing minimum degree programs.

Computing Reviews Categories: 4.0, 5.14

Keywords and Phrases: sparse linear equations, quotient graphs, ordering algorithms, graph algorithms, mathematical software.

§1. Introduction

Consider the symmetric positive definite system of linear equations

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

where A is N by N and sparse. It is well known that if A is factored using Cholesky's method, it normally suffers some fill-in. Thus, if we intend to solve (1.1) by this method, it is usual to first find a permutation matrix P and solve the reordered system

$$(PAP^T)(Px) = Pb \ , \tag{1.2}$$

where P is chosen such that PAP^T suffers low fill-in when it is factored into LL^T.

A heuristic algorithm which experience suggests is very effective in finding such low-fill permutations is the so-called minimum degree algorithm [8]. This requires some form of simulation of the factorization (either explicit or implicit), since ordering decisions are made on the basis of the structure of the partially factored (reduced) matrices which normally appear in the usual implementation of Gaussian elimination. Specifically, at each step in the elimination, the algorithm permutes the part of the matrix remaining to be factored so that a column with the fewest nonzeros is in the pivot position.

Some previous implementations known to the authors store an explicit representation of the partially factored matrix. The best implementation we are aware of which uses this approach is contained in the Yale Sparse Matrix Package [2]. In the sequel, we will refer to this

code as MDE. Implementations of this type require quite flexible data structures, since the matrix structure changes as the (symbolic) factorization proceeds and the data structures must accommodate these changes. A more serious practical difficulty with using an explicit representation is that the maximum storage requirement is unpredictable, and may exceed the storage required for the resulting factor L.

In a recent paper, the authors described a "minimal storage" implementation of the minimum degree algorithm.[5]. The implementation requires only a few arrays of length N, in addition to that needed for the graph of A. Furthermore, the graph of A is preserved during execution of the algorithm. In the sequel we denote this implementation by MDI. Although MDI has obvious advantages in terms of storage when compared to MDE, the comparisons of execution times have been mixed. As the experiments in Section 5 show, for some problems the execution time of MDI is comparable or even better than that of MDE, but for others the penalty paid by MDI for low storage requirements can be a substantial increase in execution time over that of MDE.

The implementation we describe in this paper is endowed with most of the advantages of both MDE and MDI. Its storage requirement is comparable with MDI, and for some problems its execution time is substantially lower than that of either MDE or MDI. Its single disadvantage, shared by MDE and absent in MDI, is that the original graph of A is destroyed. Thus, if the graph of A must be preserved for future use, a copy of it must be made before the algorithm is executed.

Our algorithm makes heavy use of the notion of reachable sets, described in [5], and also relies upon an efficient implementation of the quotient graph model of symmetric factorization, introduced and analyzed in [6]. We briefly review these topics in Section 2 and Section 3.1 respectively, but for more details the reader is referred to the references cited. Since the algorithm is based on quotient graphs, we will refer to our implementation of the minimum degree algorithm as MDQ.

§2 Preliminaries on Models of Symmetric Factorization

As pointed out in the introduction, the minimum degree algorithm requires some form of simulation of the factorization process. In this section, we discuss the various ways in which this process can be simulated. The approach will be graph-theoretical and the reader is assumed to be familiar with standard graph theory notions and terminology, reference to which can be found in [1].

In subsequent sections, various graph-theory notions will be applied to different graphs. When the graph under consideration is not clear from context, we will add the appropriate subscript on the definition, Thus, the notation $Adj_G(y)$ will be used to denote the adjacent set of the node y in the graph G.

2.1 Elimination Graph Model

In this section we describe the graph theory approach to symmetric elimination as given by Parter [7] and Rose [8]. Let A be an N by N symmetric matrix. The labelled undirected graph of A, denoted

by $G^A = (X^A, E^A)$, is one for which X^A is labelled from 1 to N :

$$X^A = \{x_1, x_2, \ldots, x_N\} ,$$

and $\{x_i, x_j\} \in E^A$ if and only if $A_{ij} \neq 0$. For any N by N permutation matrix P , the unlabelled graphs of A and PAP^T are the same, but the associated labellings differ. Thus, the graph of A is a convenient vehicle for studying the structure of A , since no particular ordering is implied by the graph.

Now consider the symmetric factorization of A into LL^T . The sparsity changes (fill-in) can be modelled by a sequence of graph transformations on G^A . Let $G = (X, E)$ be a graph and $y \in X$. The elimination graph of G by y , denoted by G_y , is the graph

$$(X - \{y\}, E(X - \{y\}) \cup \{\{u, v\} \mid u,v \in Adj(y)\}) .$$

In words, G_y is obtained from G by deleting y and its incident edges, and then adding any edges to the remaining graph so that the set Adj(y) is a clique. This recipe is due to Parter [7].

With this definition, the process of symmetric elimination on A can be modelled as a sequence of elimination graphs $G_0, G_1, G_2, \ldots, G_{N-1}$, where

$$G_0 = G^A ,$$

and

$$G_i = (G_{i-1})_{x_i} = (X_i, E_i) , \quad i = 1,2,\ldots,N-1 .$$

Here $X_i = \{x_{i+1}, x_{i+2}, \ldots, x_N\}$, and it is straightforward to verify that

$$G_i \equiv G^{H_i} \quad ,$$

where H_i is the part of the matrix remaining to be factored after step i of the factorization has been completed. Thus, this model is quite explicit; the structure of G_i corresponds directly to the matrix H_i .

2.2 Reachable Set Characterization

A useful notion in the study of symmetric factorization in sparse linear systems is the reachable set [5]. It can be used to model the factorization process in terms of the original graph structure.

Let $G = (X, E)$ be a given undirected graph. Consider a subset $S \subset X$ and a node $y \notin S$. The node y is said to be reachable from a node x through x if there exists a path $(x, s_1, \ldots, s_t, y)$ such that $s_i \in S$ for $1 \leq i \leq t$. Since t can be zero, any node in $Adj(y)-S$ is reachable from y through S . The reachable set of y through S is then defined to be

$$Reach(y, S) = \{x \in X-S \mid x \text{ is reachable from } y \text{ through } S\} .$$

Consider the graph example in Figure 2.1. Let $S = \{s_1, s_2, s_3, s_4\}$. We have

$$Reach(y, S) = \{a, b, c\}$$

because of the paths

$$(y, s_2, s_4, a)$$
$$(y, b)$$
$$(y, s_1, c) \quad .$$

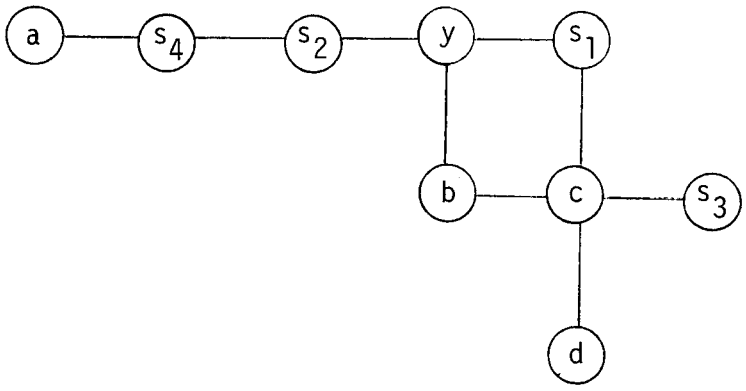


Figure 2.1 Example to illustrate reachable sets

The reachable set notion can be used to characterize the adjacency structure of elimination graphs. Let $G^A = (X^A, E^A)$ be the graph of A, with $X = \{x_1, x_2, \ldots, x_N\}$.

Let $G_0, G_1, \ldots, G_{N-1}$ be the sequence of elimination graphs as defined by the nodes $x_1, x_2, \ldots, x_N$. Define $S_i = \{x_1, \ldots, x_i\}$ for $i = 1,\ldots,N$ with $S_0 = \phi$. The following result relates the structures of the elimination graph G_i and the original graph G^A through reachable sets.

Theorem 2.1 [5] Let y be a node in the elimination graph $G_i = (X_i, E_i)$. The set of nodes adjacent to y in the graph G_i is given by

$$\text{Reach}_{G^A}(y, S_i) \quad .$$

□

The result means that the elimination process can be modelled by the subsets S_i and the Reach operator on the original graph G^A, since they represent implicitly the sequence of elimination graphs.

For convenience in later discussions, we introduce two more definitions that are related to the reachable set notion. Consider a graph $G = (X, E)$. Let $S \subset X$ and $y \notin S$. The neighborhood set of y in S is the subset

$$\text{Nbrhd}(y, S) = \{s \in S \mid s \text{ is reachable from } y \text{ through } S\} \quad .$$

In other words, this set $\text{Nbrhd}(y, S)$ contains those nodes in S through which the reach set $\text{Reach}(y, S)$ can be determined. The closure of y in S is defined to be

$$\text{Closure}(y, S) = \text{Nbrhd}(y, S) \cup \{y\} \cup \text{Reach}(y, S) \quad .$$

In Figure 2.1, we have

$$\text{Nbrhd}(y, S) = \{s_1, s_2, s_4\}$$

and $$\text{Closure}(y, S) = \{s_1, s_2, s_4, y, a, b, c\} \quad .$$

2.3 Quotient Graph Model

In [6], the authors have introduced the quotient graph model for the study of the Gaussian elimination process. Its primary advantage is that it lends itself to very efficient computer implementation in simulating the factorization. We now introduce notations and definitions for the model.

Let $G = (X, E)$ be a graph with X the set of nodes and E the set of edges. For a subset $S \subset X$, $G(S)$ will be used to refer to the subgraph $(S, E(S))$ of G, where

$$E(S) = \{\{u, v\} \in E \mid u,v \in S\} \quad .$$

The central notion in the new model is that of a quotient graph [3]. Let P be a partitioning of the node set X:

$$P = \{Y_1, Y_2, \ldots, Y_p\} \quad .$$

That is, $\bigcup_{k=1}^{p} Y_k = X$ and $Y_i \cap Y_j = \phi$ for $i \neq j$. The quotient graph of G with respect to P is defined to be the graph $(P, \mathcal{E})$, where $\{Y_i, Y_j\} \in \mathcal{E}$ if and only if $Y_i \cap \mathrm{Adj}(Y_j) \neq \phi$. This graph will be denoted by G/P.

Consider the graph in Figure 2.2. If $P = \{\{a, b, c\}, \{d, e\}, \{f\}, \{g, h, i\}\}$ is the partitioning, the quotient graph G/P is given as shown.

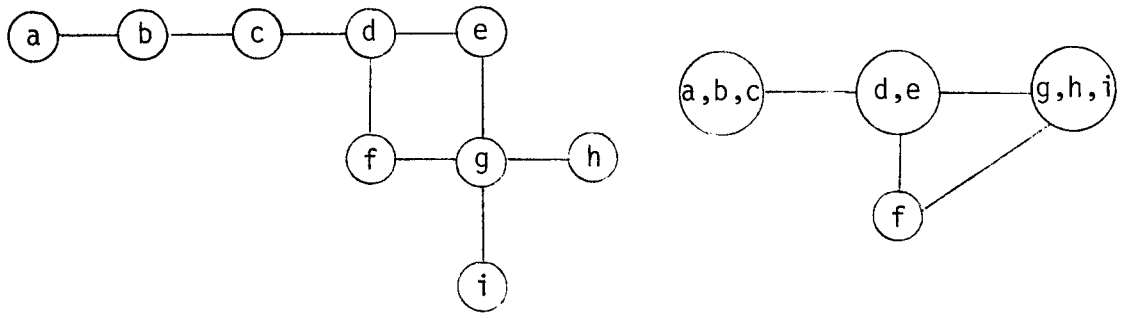


Figure 2.2 A quotient graph

An important type of partitioning is that defined by connected components. Let S be a subset of the node set X. The component partitioning $C(S)$ of S is defined as

$$C(S) = \{Y \subset S \mid G(Y) \text{ is a connected component in the subgraph } G(S)\} .$$

A closely related type of partitioning turns out to be relevant in the modelling of Gaussian elimination. The partitioning on X induced by S is defined to be

$$\overline{C}(S) = C(S) \cup \{\{x\} \mid x \notin S\} .$$

That is, the partitioning $\overline{C}(S)$ consists of the component partitioning of S and the remaining nodes of the graph G.

Consider the graph in Figure 2.1. Let S be the subset $\{a, b, d, f, h\}$. The component partitioning $C(S)$ is given by

$$C(S) = \{\{a, b\}, \{d, f\}, \{h\}\}$$

so that $\overline{C}(S)$ has seven members and the quotient graph $G/\overline{C}(S)$ is given in Figure 2.3. Here, double circles are used to indicate partition members in $C(S)$.

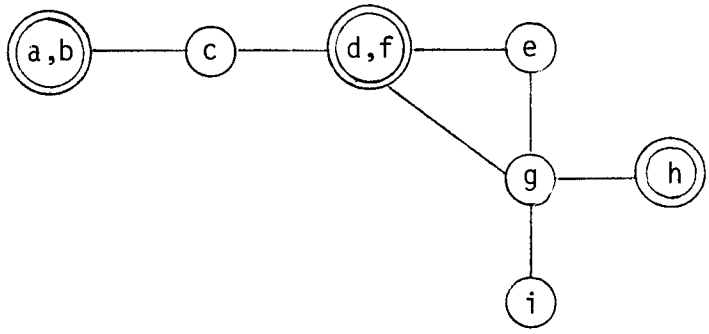


Figure 2.3 The induced quotient graph $G/\overline{C}(S)$.

We now study the relevance of quotient graphs in symmetric factorization. Consider the factorization of a matrix A into LL^T. The process can be interpreted as a sequence of elimination graphs

$$G_0, G_1, \ldots, G_{N-1} ,$$

where G_i precisely reflects the structure of the matrix remaining to be factored after the i-th step of the Gaussian elimination.

Alternatively, the process can also be represented as a sequence of quotient graphs, which may be regarded as <u>implicit representations</u> of the elimination graphs $\{G_i\}$. Let $\{x_1, x_2, \ldots, x_N\}$ be the sequence of node elimination in the graph $G = G^A$. As in Section 2.1, let $S_i = \{x_1, \ldots, x_i\}$ for $1 \le i \le N$ and $S_0 = \phi$.

Consider the partitioning $\overline{C}(S_i)$ induced by S_i and the corresponding quotient graph

$$G/\overline{C}(S_i) = (\overline{C}(S_i), \mathcal{E}_i) \ .$$

We shall denote this quotient graph by G_i . In this way, we obtain a sequence of quotient graphs

$$G_1, G_2, \ldots, G_N \ .$$

The following result shows that, indeed, the quotient graph $G_i = G/\overline{C}(S_i)$ can be regarded as an implicit representation of the elimination graph G_i .

Theorem 2.2 For $y \notin S_i$,

$$\text{Reach}_{G_i}(\{y\}, C(S_i)) = \{\{x\} \mid x \in \text{Reach}_G(y, S_i)\} \ .$$

Proof: Consider $x \in \text{Reach}_G(y, S_i)$. If x and y are adjacent in G , so are $\{x\}$ and $\{y\}$ in G_i . Otherwise, there exists a path $y, s_1, \ldots, s_t, x$ in G where $\{s_1, \ldots, s_t\} \subset S_i$. Let C be the component in $G(S_i)$ containing $\{s_1, \ldots, s_t\}$. Then we have a path $\{y\}, C, \{x\}$ in G_i so that

$$\{x\} \in \text{Reach}_{G_i}(\{y\}, C(S_i)) \ .$$

Conversely, consider any $\{x\} \in \text{Reach}_{G_i}(\{y\}, C(S_i))$. There exists a path

$$\{y\}, C_1, \ldots, C_t, \{x\}$$

in G_i where each $C_j \in C(S_i)$. If $t = 0$, then x and y are adjacent in the original graph G . If $t > 0$, by the definition of $C(S_i)$, t cannot be greater than one; that is, the path must be

$$\{y\}, C, \{x\} \quad .$$

Since $G(C)$ is a connected subgraph, we can obtain a path from y to x through $C \subset S_i$ in the graph G . Hence

$$x \in \text{Reach}_G(y, S_i) \quad .$$

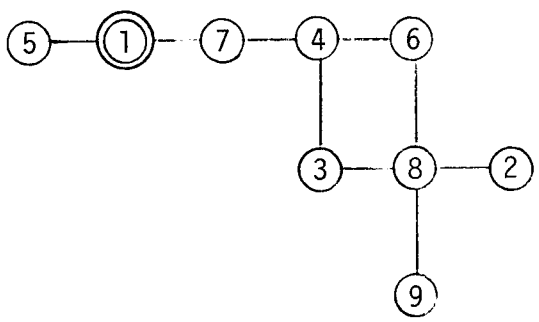
□

Figure 2.4 contains an example of a quotient graph sequence. Partition members in $C(S_i)$ are marked in double circles for clarity.

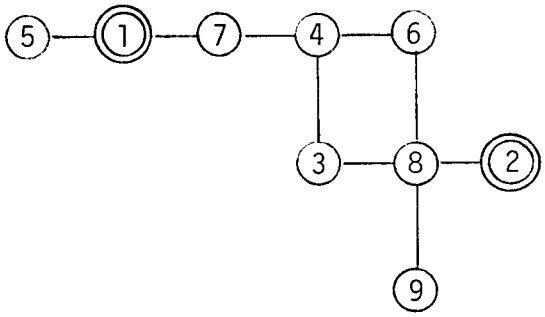
As a theoretical tool for studying and understanding the symmetric factorization, the use of quotient graphs does not appear to be better than the elimination graphs or the reachable set notion. However, the new model can be implemented very efficiently both in terms of time and space. In what follows, we shall show that it can be implemented in-place.

Lemma 2.3 Let $G = (X, E)$ be a given graph, and let $S \subset X$, where $G(S)$ is a connected subgraph. Then

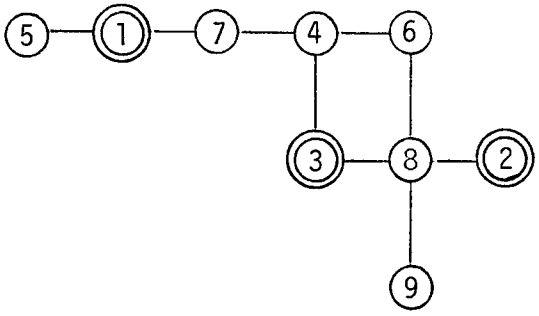
$$\sum_{x \in S} |\text{Adj}(x)| \geq |\text{Adj}(S)| + 2(|S| - 1) \quad .$$



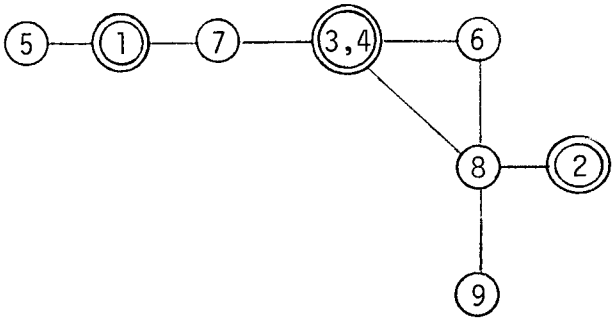
G_1



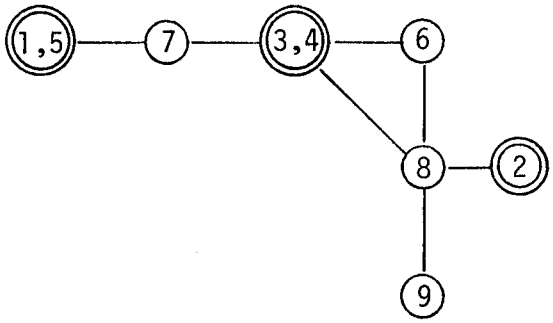
G_2



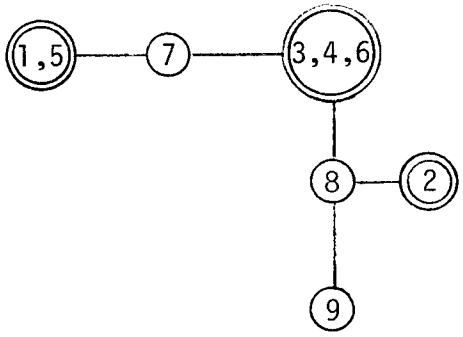
G_3



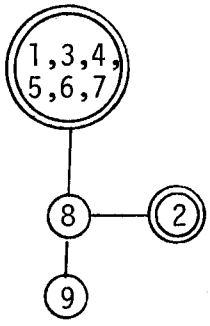
G_4



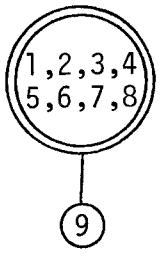
G_5



G_6



G_7



G_8

Figure 2.4 A sequence of quotient graphs

Proof: Since $G(S)$ is connected, there are at least $|S|-1$ edges in the subgraph. These edges are counted twice in $\sum_{x \in S} |Adj(x)|$ and hence the result.

□

We now show that the edge set sizes of the quotient graphs G_i cannot increase with increasing i. Let $x_1, x_2, \ldots, x_N$ be the node sequence, let

$$G_i = (X_i, E_i) \quad , \quad 0 \leq i \leq N \quad ,$$

be the corresponding elimination graph sequence, and let

$$\mathcal{G}_i = (C(S_i), \mathcal{E}_i), \quad 1 \leq i \leq N \quad ,$$

be the corresponding quotient graph sequence.

Theorem 2.4 For $1 \leq i \leq N$,

$$|\overline{C}(S_{i+1})| \leq |\overline{C}(S_i)|$$

and

$$|\mathcal{E}_{i+1}| \leq |\mathcal{E}_i| \quad .$$

Proof: Since $C(S_i)$ is the set of components in the subgraph $G(S_i)$, we have

$$|C(S_{i+1})| \leq |C(S_i)| + 1 \quad .$$

However, $|\overline{C}(S_i)| = |C(S_i)| + N - i$, so that

$$\begin{aligned} |\overline{C}(S_{i+1})| &= |C(S_{i+1})| + N - i - 1 \\ &\leq |C(S_i)| + N - i = |\overline{C}(S_i)| \quad . \end{aligned}$$

For the inequality on the edge size, consider $C(S_{i+1})$. If $\{x_{i+1}\} \in C(S_{i+1})$, clearly $|\mathcal{E}_{i+1}| = |\mathcal{E}_i|$. Otherwise, the node x_{i+1} is merged with some components in $C(S_i)$ to form a new component in $C(S_{i+1})$. But then Lemma 2.3 applies, so that $|\mathcal{E}_{i+1}| < |\mathcal{E}_i|$. Hence in all cases,

$$|\mathcal{E}_{i+1}| \leq |\mathcal{E}_i| \quad .$$

□

The next theorem illustrates the advantage of the quotient graph model over the elimination graph model, and is one of the primary motivations for our introduction of the model.

Theorem 2.5

$$\max_{1 \leq i \leq N} |\mathcal{E}_i| \leq |E| \leq \max_{0 \leq i \leq N} |E_i| \quad .$$

Proof: The first inequality follows from Theorem 2.4, and the fact that $|E_0| = |E|$ implies the second one.

□

The advantage of the quotient graph model over the reachable set approach is, in fact, implicit in the proof of Theorem 2.2. To determine the set $\text{Reach}_G(y, S_i)$, paths emanating from the node y through the subset S_i have to be traced. These paths can be of arbitrary lengths (less than N), and many may be traversed unnecessarily. For example, in the graph of Figure 2.4, to find $\text{Reach}_G(x_7, S_7)$, the three paths

$$(x_7, x_1, x_5)$$
$$(x_7, x_4, x_6, x_8)$$
$$(x_7, x_4, x_3, x_8)$$

have to examined, of which the last one is unnecessary.

However, the determination of $\text{Reach}_{G_i}(\{y\}, C(S_i))$ in the quotient graph involves paths of length at most two (see proof of Theorem 2.2). Indeed, the time required can be shown to be proportional to the size $|\text{Reach}_{G_i}(\{y\}, C(S_i))|$.

§3. Description of the Minimum Degree Algorithm

3.1 Using Elimination Graphs

The minimum degree algorithm is probably the most widely used heuristic algorithm for finding low fill orderings. In terms of elimination graphs, it can be best described as follows. Let $G_0 = (X, E)$ be an unlabelled graph.

Step 1 (Initialization) Set $i \leftarrow 1$.

Step 2 (Minimum degree selection) In the elimination graph G_{i-1} , choose x_i to be a node such that

$$|\text{Adj}_{G_{i-1}}(x_i)| = \min_{y \in X_{i-1}} |\text{Adj}_{G_{i-1}}(y)| \quad ,$$

where $G_{i-1} = (X_{i-1}, E_{i-1})$.

Step 3 (Graph transformation) Form the new elimination graph

$$G_i = (G_{i-1})_{x_i} \quad .$$

Step 4 (Loop or stop) $i \leftarrow i+1$. If $i > |X|$, stop. Otherwise, go to Step 2.

In effect, the algorithm orders a given graph by selecting nodes of minimum degree from elimination graphs (hence the name minimum degree algorithm). The formation of the elimination graph in Step 3 is necessary to facilitate the selection of the next node to be ordered. Different versions of the algorithm can be formulated for different ways of computing degrees of nodes in the elimination graph.

An extremely good implementation of this formation can be found in the Yale Sparse Matrix Package [2]. The sequence of elimination graphs is represented explicitly in a linked list structure. Interested readers are referred to that reference for details.

3.2 Using Reachable Sets

Theorem 2.2 relates the adjacency structure of the elimination graph to that of the original graph. With this simple connection, the minimum degree algorithm can be restated in terms of reachable sets.

Step 1 (Initialization) $S \leftarrow \phi$

Step 2 (Selection) Pick a node $y \in X-S$ such that

$$|\text{Reach}(y, S)| = \min_{x \in X-S} |\text{Reach}(x, S)| .$$

Number the node y next.

Step 3 (Implicit transformation) Set $S \leftarrow S \cup \{y\}$.

Step 4 (Loop or stop) If $S = X$, stop. Otherwise, go to Step 2.

It should be noted that in the actual implementation, the number $|Reach(x, S)|$ for $x \in X-S$ should be stored to avoid re-computation. This means in the implicit transformation step, some of these numbers have to be updated to reflect the change in the subset S . The following result is quoted from [5].

Theorem 3.1 For $u \in X-S$,

$$Reach(u, S\cup\{y\}) = Reach(u, S)$$

if $u \notin Reach(y, S)$.

□

This result justifies keeping the numbers $|Reach(x, S)|$ since many of them remain unchanged after the node y is numbered. Indeed, the numbering of node y only affects the degrees of the nodes in $Reach(y, S)$.

In [5], the authors have implemented this approach with some refinements to speed up the execution of the algorithm. The main advantage of this implementation is its small and predictable storage requirements. Moreover, it preserves the adjacency structure of the given graph.

3.3 Using Quotient Graphs

The selection step in the minimum degree algorithm depends on the degrees of the nodes in the elimination graph. The reformulation of the algorithm in Section 3.2 makes use of the connection between the degree of a node in the elimination graph and the reachable set. In view of Theorem 2.2, we can describe the algorithm in terms of quotient graphs.

Step 1 (Initialization) Set $S \leftarrow \phi$, $G = G/\overline{C}(\phi)$.

Step 2 (Selection) Pick a node in $y \in X\text{-}S$ such that

$$|\text{Reach}_G(\{y\}, C(S))| = \min_{x \in X\text{-}S} |\text{Reach}_G(\{y\}, C(S))| \quad .$$

Number the node y next.

Step 3 (Quotient graph transformation) $S \leftarrow S \cup \{y\}$ and put $G \leftarrow G/\overline{C}(S)$.

Step 4 (Loop or stop) If $S = X$, stop. Otherwise, go to Step 2.

Basically, there is not much difference in the formulation of the algorithm in terms of reachable sets and quotient graphs. They both employ the notion of reachable sets; one on the original graph and the other on quotient graphs. Their major difference lies in their implementation. The important features of the quotient graph implementation are described in Section 5.

In [5], the authors discussed various enhancements in the implementation of the algorithm by the reachable set approach. Some of these novel features are applicable in the use of quotient graphs. In what follows, we shall quote results from [5] and discuss how we can apply them to quotient graphs.

Minimum Degree Selection

For each execution of Step 2 of the minimum degree algorithm as described, a node of minimum degree is selected and numbered. The following theorem shows that a set of nodes can be numbered at one time at the expense of some minimal extra work.

Theorem 3.2 [5] Let y be a node in $X-S$ satisfying

$$|\text{Reach}(y, S)| = \min_{x \notin S} |\text{Reach}(x, S)| \quad .$$

Let Y be the set

$$Y = \{x \in \text{Reach}(y, S) \cup \{y\} \mid \text{Adj}(x) \subset \text{Closure}(y, S)\} \quad .$$

Then the nodes in Y can be numbered next consecutively in the minimum degree algorithm. □

This result can be translated in a straightforward way to quotient graphs. Consider the quotient graph

$$G = G/\overline{C}(S) \quad .$$

If y is a node in $X-S$ such that

$$|\text{Reach}_G(\{y\}, C(S))| = \min_{x \notin S} |\text{Reach}_G(\{x\}, C(S))| \quad ,$$

then the nodes in the set

$$Y = \Big\{x \mid \{x\} \in \text{Reach}_G(\{y\}, C(S)) \cup \{\{y\}\} \quad \text{and} \quad \text{Adj}_G(\{x\}) \subset \text{Closure}_G(\{y\}, C(S))\Big\}$$

can be ordered next in the algorithm.

Consider the graph example in Figure 3.1. The subset S contains 27 nodes marked in shade. The corresponding quotient graph is also given in Figure 3.1. To simplify the notation, we use y to stand for the node $\{y\}$ in the quotient graph where $y = a, b, c, d, \ldots$.

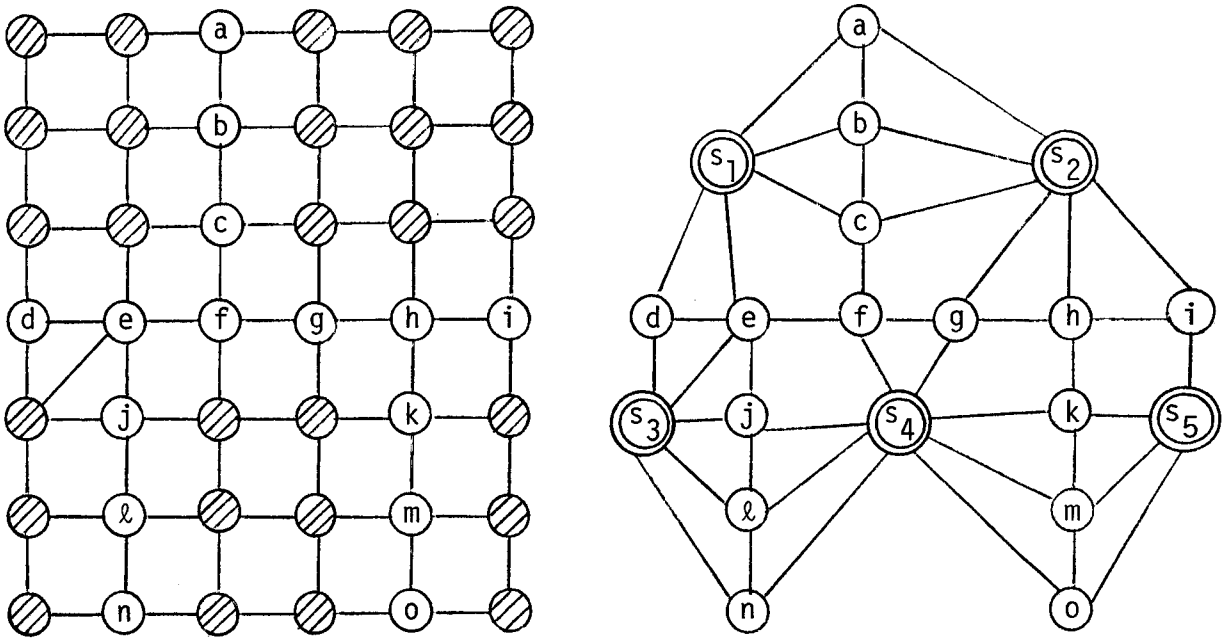
In the example, the node a is one of minimum degree, where

$$\text{Reach}_G(a, C(S)) = \{b, c, d, e, g, h, i\}$$

$$\text{Nbrhd}_G(a, C(S)) = \{s_1, s_2\} .$$

But $$\text{Adj}_G(b) = \{a, c, s_1, s_2\}$$

which is contained in $\text{Closure}_G(a, C(S))$. In other words, the nodes a and b can be numbered consecutively.



Original graph G Quotient graph $G/\overline{C}(S)$

Figure 3.1

Degree Update

The remark in Section 3.2 about degree update also applies to the quotient graph formulation. If y is the node selected in Step 2, only the degrees of the nodes in

$$\text{Reach}_G(\{y\}, C(S))$$

need be updated; the others remain unchanged.

In [5], the authors have also introduced an enhancement in the degree update procedure, whereby a set of nodes can have their degrees updated together. We quote the result and adapt it to quotient graphs.

Theorem 3.3 Let U be the set of nodes whose degree $|\text{Reach}(x, S)|$ for $x \in U$ need to be computed. Let $u \in U$ satisfy the condition

$$\text{Reach}(u, S) = \text{Adj}(\text{Nbrhd}(u, S)) \quad .$$

Consider $v \in U$ with $\text{Nbrhd}(v, S) = \text{Nbrhd}(u, S)$. If $\text{Adj}(v) \subset \text{Closure}(u, S)$, then

$$|\text{Reach}(v, S)| = |\text{Reach}(u, S)| \quad .$$

□

To illustrate the applicability of the result to quotient graphs, we consider the example in Figure 3.2. It is in fact a continuation of the elimination process from the example in Figure 3.1.

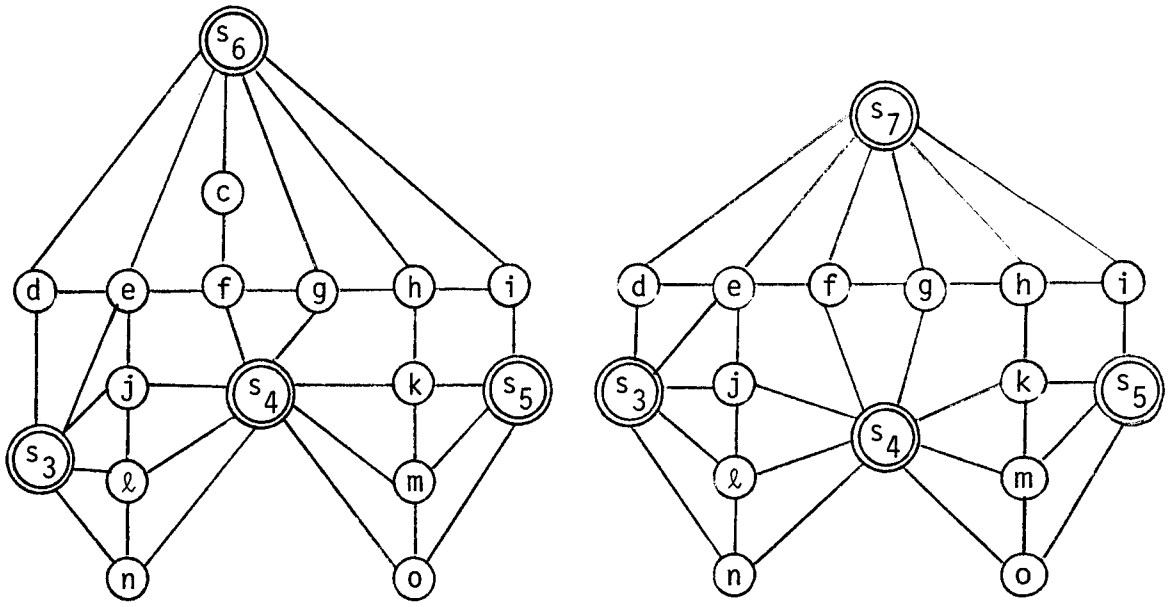


Figure 3.2

Consider the elimination of the node c . In this case,

$$U = \{d, e, f, g, h, i\}$$

contains the nodes whose degrees have to be updated. Let S be the set of eliminated nodes (including c), and $G = G/\overline{C}(S)$. Consider the node $f \in U$. Clearly, we have

$$\text{Nbrhd}_G(f, C(S)) = \{s_4, s_7\} \quad ,$$

so that

$$\text{Reach}_G(f, C(S)) = \text{Adj}_G(\text{Nbrhd } (f, C(S))) \quad .$$

Furthermore, the node $g \in U$ satisfies

$$\text{Nbrhd}_G(g, C(S)) = \{s_4, s_7\}$$

and

$$Adj_G(g) = \{f, h, s_4, s_7\} \quad .$$

Hence, the nodes f and g have the same degree through S . (Their degrees are 11.)

§4. Some Details About the Implementation

4.1 Computer Representation of Quotient Graphs

A graph is completely determined by its adjacency structure, that is, the set of adjacency lists for every node in the graph. We represent such a structure by storing the adjacency lists sequentially in a one-dimensional array along with an index array of length N containing pointers to the beginning of each adjacency list in the storage array. An example is shown in Figure 4.1, where the lists are separated for illustrative purposes.

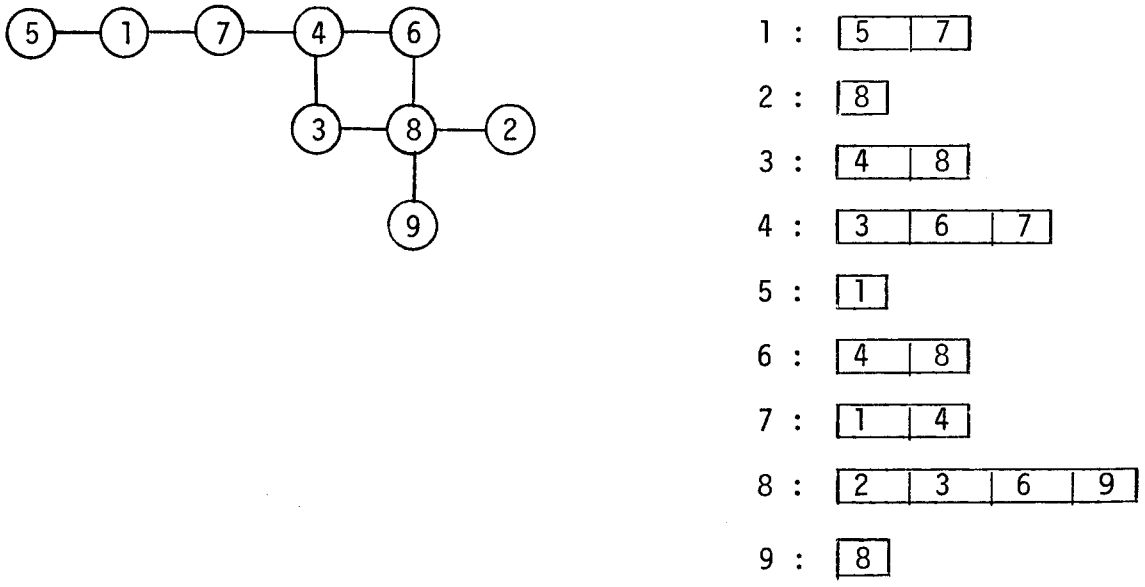


Figure 4.1 A graph and its representation

Given a graph $G = (X, E)$ and a subset S. How can we represent the quotient graph

$$G = G/\overline{C}(S)$$

using the space provided by the adjacency structure of the original graph?

Consider a connected component Y in the subgraph $G(S)$. By the result of Lemma 2.3, we have

$$\sum_{y \in Y} |Adj(y)| \geq |Adj(Y)| + 2(|Y| - 1) \quad ,$$

or

$$|Adj(Y)| \leq \sum_{y \in Y} |Adj(y)| - 2(|Y| - 1) \quad .$$

Therefore, there are always enough locations for $Adj(Y)$ from those for $Adj(y)$, $y \in Y$. Moreover, for $Y \neq \phi$, there is always a surplus of $2(|Y|-1)$ locations which can be utilized for links or pointers.

In the graph example in Figure 4.2, let the subset $S = \{1, 2, 3, 4, 5\}$. Figure 4.2 shows the way the quotient graph $G = G/\overline{C}(S)$ is stored.

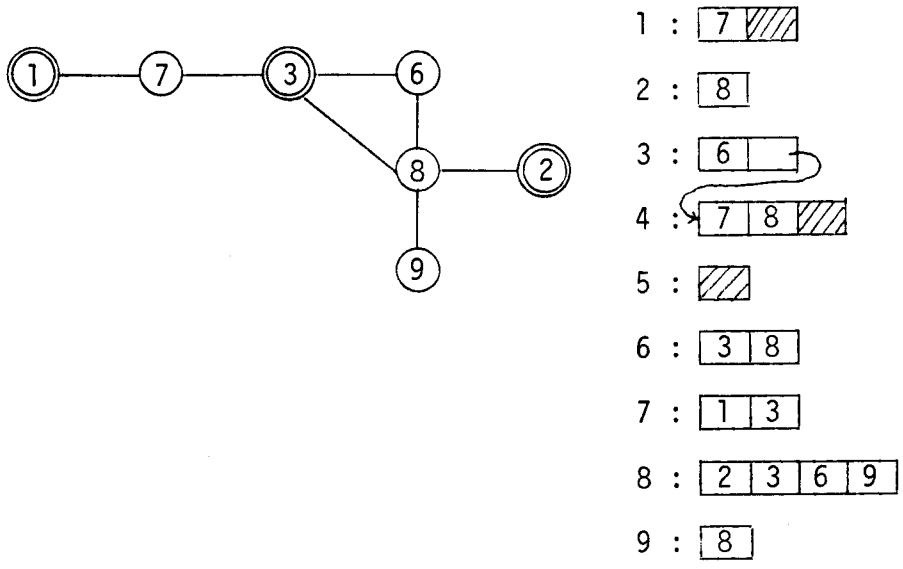


Figure 4.2 A quotient graph representation

The components in $G(S)$ are given by

$$\mathcal{C}(S) = \{\{1, 5\}, \{2\}, \{3, 4\}\} \quad .$$

Consider finding the adjacent set of the component $\{3, 4\}$ in the quotient graph representation. It can be retrieved by first exhausting the current list for node 3; the link to node 4 then gives the remaining neighbors of the component.

Another point that is worth noting in the example is the use of representatives for component members. The components $\{1, 5\}$ and $\{3, 4\}$ are now represented by the nodes 1 and 3 respectively. In other words, the nodes 5 and 4 will be ignored in future processing, although the storage for their original adjacent lists may be used for the quotient graph representation.

In our implementation, the set S and its component representatives are maintained by a three-state array SMASK where

1) SMASK(i) = 0 if node $i \notin S$
2) SMASK(i) > 0 if node i is a representative of a component of S
3) SMASK(i) < 0 if node i is in S and can be ignored.

4.2 Quotient Graph Transformation

In the minimum degree algorithm, the selection of node y puts it into the subset S, so it is necessary to do the transformation

$$G/\overline{\mathcal{C}}(S) \rightarrow G/\overline{\mathcal{C}}(S \cup \{y\}) \quad .$$

It is easy to verify that

$$\{y\} \cup \text{Nbrhd}(y, S)$$

is a connected component in the subgraph $G(S \cup \{y\})$. A natural choice of representative for this new component subset of S is the node y . The transformation can be described as follows:

Step 1 (New adjacent set) Determine the

$$\text{sets } R = \{x \mid \{x\} \in \text{Reach}_G(\{y\}, C(S))\} .$$

Step 2 (In-place transformation) Use the node y as the representative of the new quotient member $\{y\} \cup \text{Nbrhd}(y, S)$. Reset

$$\text{Adj}(y) \leftarrow R$$

using the adjacency list storage from nodes in $\text{Nbrhd}(y, S)$ if necessary.

Step 3 (Neighbor update) For $u \in R$, put

$$\text{Adj}(u) \leftarrow (\text{Adj}(u) - \text{Nbrhd}(y, S)) \cup \{y\} .$$

4.3 Mass Elimination and Degree Update

The enhancements to the minimum degree algorithm are essentially direct applications of Theorems 3.1 and 3.2, both of which involve a test of whether a relation of the following kind is true.

$$\text{Adj}_G(x) \subset \text{Closure}_G(y, C(S)) .$$

The condition is usually tested for numerous x after $\text{Reach}_G(y, C(S))$ has been generated. In order to facilitate this test, in our implementation we have a marker array MARKER which is initially zero. In generating

$$\text{Reach}_G(y, C(S)) \quad ,$$

we set the MARKER values of all the nodes in $\text{Closure}_G(y, C(S))$ to a nonzero quantity. In this way, the above condition can be tested simply by examining the MARKER values of the neighbors of the node x. Of course, the MARKER values have to be reset to zero before the next reach operation is performed.

§5 Numerical Experiments and Concluding Remarks

In this section we report on some numerical experiments showing the performance of our implementation of the minimum degree algorithm (MDQ), based on the use of quotient graphs. As a basis for comparison, we have used two other implementations, one from the Yale Sparse Matrix package which we denote by MDE [2], and one from [5], which we denote by MDI. The MDE subroutine is a conventional implementation, in the sense that it represents the elimination graphs $G_0, G_1, \ldots, G_{N-1}$ explicitly in a linked list structure. The MDI implementation, on the other hand, uses only the original graph, and does not change it during the execution of the algorithm. Information that is required about the G_i is obtained through generation of the appropriate reachable sets, as described in detail in [5]. The names MDQ, MDE, and MDI are intended to suggest quotient graph, explicit, and implicit methods of representing the elimination graphs.

Our first set of test problems were chosen to obtain evidence bearing on the asymptotic behaviour of our program. We used the set of "graded-L" mesh problems from [3, page 318], which consists of a sequence of similar problems typical of those arising in finite element applications. The results of these runs are summarized in Table 5.1. Execution times are in seconds on an IBM 360/75 computer. The programs are all written in Fortran, and were compiled using the optimizing version of the IBM H-level compiler.

N	EXECUTION TIME						STORAGE REQUIREMENTS					
	MDE	÷|E|	MDI	÷|E|	MDQ	÷|E|	MDE	÷|E|	MDI	÷|E|	MDQ	÷|E|
265	.89	1.19	1.20	1.61	.66	.887	7147	9.61	4139	5.56	4832	6.49
406	1.44	1.25	1.97	1.71	1.14	.987	11151	9.65	6371	5.52	7463	6.46
577	2.18	1.32	2.95	1.77	1.67	1.01	16735	10.11	9083	5.48	10664	6.44
778	3.30	1.47	3.89	1.73	2.37	1.05	24697	10.99	12275	5.46	14435	6.42
1009	4.56	1.56	5.14	1.75	2.92	1.00	32513	11.10	15974	5.45	18776	6.41
1270	5.76	1.56	6.30	1.72	4.16	1.12	42151	11.40	20099	5.43	23687	6.40
1561	7.40	1.62	7.98	1.75	4.88	1.07	53215	11.67	24731	5.42	29168	6.40
1882	9.67	1.75	9.87	1.79	6.26	1.14	65723	11.93	29843	5.41	35219	6.39
2233	11.55	1.76	11.98	1.83	6.88	1.05	82585	12.60	35435	5.41	41840	6.39
2614	14.63	1.90	13.63	1.77	8.45	1.10	97121	12.64	41507	5.40	49031	6.38
		($x10^{-3}$)		($x10^{-3}$)		($x10^{-3}$)						

Table 5.1 Execution Times and Storage Requirements for the Three Implementations on the Graded L Problems

In terms of execution time, the MDQ implementation enjoys a significant advantage over the other two implementations. Both MDI and MDQ appear to execute in time very close to proportional to $|E|$ for these problems, but the MDE execution time appears to be superlinear in $|E|$. However, because of differences in constants of proportionality, MDE is still quite competitive with MDI until N is several thousand.

Turning now to storage requirements, it is evident that the MDI and MDQ subroutines require storage proportional to $|E|$. On the other hand, MDE explicitly generates a sequence of graphs $G_0, G_1, \ldots, G_{N-1}$, and the maximum storage required may greatly exceed $|E|$. The entries in the storage column for MDE were obtained by monitoring the maximum storage used by the working storage arrays. In general, for implementations which store the elimination graphs explicitly, the user must estimate the maximum storage requirement, and provide at least that much to allow the program to execute. This is a major disadvantage not shared by MDI and MDQ, since they use a fixed, predictable amount of storage.

It is interesting to note that the storage required by the subroutine MDQ is actually <u>less</u> than that required by MDI. However, MDI does not change the input graph structure, since it generates reachable sets via the original graph G_0. On the other hand, MDQ generates a sequence of graphs starting with the original, and does not preserve its input graph. Thus, to be fair in the comparison, we have added space required to retain the input graph to the MDQ storage requirements. Alternatively, one could preserve the input by writing it on auxiliary storage, and then reading it after the ordering had been found. The apparent storage advantage of MDI over MDQ would then be removed.

Although MDI and MDQ will always enjoy a storage advantage over MDE, the magnitude of this advantage is problem dependent. Moreover, it turns out that the execution time advantage enjoyed by MDQ over MDE in the above problems is not indicative of the results for other problems. Sometimes the lower storage requirement of MDQ and MDI over MDE is paid for through substantially increased execution time. To illustrate this, we applied the three subroutines to a collection of problems from various applications, including finite element problems and normal equations of sparse least square problems arising in surveying and linear programming. The results of these experiments are summarized in Table 5.2.

The results in Table 5.2 show that none of the methods is uniformly faster than the others. However, the "AVERAGE" row in the table suggests to us that the MDQ implementation is the method of choice. Specifically, if we make the reasonable assumption that the cost of running a program is proportional to the product of execution time and storage used, the average costs of MDE, MDI, and MDQ are 11.20, 14.07, and 7.16 respectively. (In many installations, storage requirements are weighted more heavily, thus making MDI and MDQ more favorable compared to MDE.) We feel that these costs, combined with the predictability of storage requirements, puts MDQ in a very strong position.

N	EXECUTION TIME			STORAGE REQUIREMENTS		
	MDE	MDI	MDQ	MDE	MDI	MDQ
1176	4.86	4.43	3.05	5.91	2.91	4.30
663	.33	.18	.22	.56	.71	.55
822	1.38	9.27	2.78	1.20	1.06	1.04
822	5.46	30.22	8.09	3.41	1.22	1.37
1250	1.40	12.32	3.59	1.90	1.63	1.64
796	.94	7.52	2.27	1.21	1.04	1.04
700	7.44	28.75	8.84	5.15	.98	1.05
936	3.65	3.98	2.62	2.77	1.47	1.72
1009	4.39	4.72	3.06	3.25	1.59	1.88
1089	4.26	4.66	3.19	3.32	1.72	2.02
1440	4.80	6.41	4.31	3.85	2.25	2.62
1180	3.24	4.11	3.50	2.99	1.84	2.14
1377	3.36	4.79	3.85	3.52	2.14	2.49
1138	3.54	4.94	3.54	2.97	1.78	2.06
1141	4.09	5.16	3.57	2.99	1.77	2.06
1349	5.30	5.87	3.89	4.09	2.12	2.49
AVERAGE	3.65	8.58	3.77	3.07	1.64	1.90
				($\times 10^4$)	($\times 10^4$)	($\times 10^4$)

Table 5.2 Comparison of the performance of the MDE, MDI and MDQ implementations of the minimum degree algorithms applied to 16 test problems.

References

[1] C. Berge, The Theory of Graphs and its Applications, John Wiley and Sons Inc., New York, (1962).

[2] S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A.H. Sherman, "Yale Sparse Matrix Package I. The Symmetric Codes", Research Report 112, Computer Science Department, Yale University.

[3] Alan George and Joseph W.H. Liu, "Algorithms for Matrix Partitioning and the Numerical Solution of Finite Element Systems", SIAM J. Numer. Anal., 15 (1978), pp. 297-327.

[4] Alan George and Joseph W.H. Liu, "An Automatic Nested Dissection Algorithm for Irregular Finite Element Problems", SIAM J. Numer. Anal., (to appear).

[5] Alan George and Joseph W.H. Liu, "A Minimal Storage Implementation of the Minimum Degree Algorithm", SIAM J. Numer. Anal., (to appear).

[6] Alan George and Joseph W.H. Liu, "A Quotient Graph Model for Symmetric Factorization", to appear in the Proc. 1978 Symposium on Sparse Matrix Computations and their Applications, Nov. 2-3, 1978, Knoxville, TN.

[7] S.V. Parter, "The Use of Linear Graphs in Gauss Elimination", SIAM Rev., 3 (1961), pp. 364-369.

[8] D.J. Rose, "A Graph Theoretic Study of the Numerical Solution of Sparse Positive Definite Systems", in Graph Theory and Computing, R.C. Read, editor, Academic Press (1972).

[9] D.J. Rose, R.E. Tarjan, and G.S. Lueker, "Algorithmic Aspects of Vertex Elimination on Graphs", SIAM J. on Computing, 5 (1975), pp. 266-283.