ALGORITHMIC ASPECTS OF VERTEX ELIMINATION[†]

Donald J. Rose
Applied Mathematics, Aiken Computation Laboratory
Harvard University
Cambridge, Massachusetts

and

R. Endre Tarjan
Computer Science Division
Department of Electrical Engineering and Computer Sciences
Electronics Research Laboratory
University of California, Berkeley

[†]Research partially sponsored by the Office of Naval Research under contract N00014-67-A-0298-0034 at Harvard University, and by the National Science Foundation, Grant GJ-3560X and a Miller Research Fellowship at the University of California, Berkeley. Part of this work was completed while the first author was visiting the Computer Science Department at the University of Colorado and the second author was visiting the Weizmann Institute of Science, Rehovot, Israel.

Abstract

We consider a graph-theoretic elimination process which is related to performing Gaussian elimination on sparse symmetric and unsymmetric systems of linear equations. We discuss good algorithms for finding elimination orderings, showing that a generalization of breadth-first search, called lexicographic search, can be used to find perfect orderings in $O(n+e)$ time and minimal orderings in $O(ne)$ time, if the problem graph is undirected and has n vertices and e edges. We also give efficient (though slower) algorithms for generating such orderings on directed graphs. We claim that the minimum ordering problem for directed graphs is NP-complete, and conjecture that it is also NP-complete for undirected graphs. We include a brief discussion of the relation of elimination to transitive closure and discuss some unresolved, more general, issues.

Keywords: graph, directed graph, breadth-first search, lexicographic ordering, Gaussian elimination, sparse linear equations, perfect elimination, triangulated graph.

1. Introduction and Preliminaries

The process of vertex elimination on graphs is interesting, on one hand, since it models the process of performing Gaussian elimination (or LU factorization) on sparse systems of linear equations, say

(1) $$Mx = k$$

where $M = (m_{ij})$ is $n \times n$ (and nonsingular) and x and k are n-vectors. To do this we associate with M as in (1) an ordered (directed) graph $G_\alpha(M) = (V,E,\alpha)$ where α is a bijection which orders the vertices V. We construct the ordered graph $G_\alpha(M)$ as follows [12,20]: vertex $\alpha(i)$ corresponds to row i and variable i, and $(x_i,x_j) \in E$ (edge) if and only if $m_{ij} \neq 0$ and $i \neq j$. The unordered graph $G = (V,E)$ then corresponds to the equivalence class of matrices PMP^T, where P is any permutation matrix. Solving the system (1) using Gaussian elimination, in general, creates new nonzero elements; the edges corresponding to these new elements, say $F(G_\alpha)$, are called the "fill in". In order to make the elimination process efficient, we might, for example, like to minimize the fill in.

To make the connection more precise, we write M as

(2) $$M \equiv M^{(1)} = \begin{bmatrix} a & r \\ c & \bar{M} \end{bmatrix}$$

where a is 1×1 and $\bar{M}$ is $(n-1) \times (n-1)$. Gaussian elimination, viewed as LU factorization (L lower triangular, U upper triangular) proceeds, formally, from (2) to

(3) $$M^{(1)} = \begin{bmatrix} 1 & 0 \\ {}^{c}/a & I \end{bmatrix} \begin{bmatrix} a & r \\ 0 & M^{(2)} \end{bmatrix},$$
$$M^{(2)} = \bar{M} - {}^{cr}/a .$$

Then continuing recursively, we obtain the LU factorization of $M^{(2)} = L_2U_2$, say, giving, from (3) the LU factorization of M as

(4) $$M = \begin{bmatrix} 1 & 0 \\ {}^{c}/a & L_2 \end{bmatrix} \begin{bmatrix} a & r \\ 0 & U_2 \end{bmatrix} = LU .$$

The "fill in" at each stage in the process may occur because in (3) $M^{(2)}$ has, in general, fewer zeros than $\bar{M}_T$ (assuming no lucky cancellation). When M is symmetric $r = c^T$ in (2), in (3) $M^{(2)} = \bar{M} - {}^{cc}/a$, and we consider the symmetric factorization $M = LDL^T$, D a diagonal matrix. When L has ones on the diagonal, these factorizations are unique, and hence the zero-nonzero structure of L and U is independent of any specific numerical implementation.

We assume that the matrix M has an LU factorization for any reordering of the system (1); i.e., that for any permutation matrix P,

$$M' = PMP^T = LU$$

(both L and U depending on P). Hence to solve (1) efficiently requires three phases:

(i) the choice of the ordering;

(ii) the calculation of the fill in, $F(G_\alpha)$; and

(iii) the numerical process.

We deal here with phases (i) and (ii) graph-theoretically. First some notation.

A <u>directed graph</u> is a pair $G = (V,E)$ where V is a finite set of $n = |V|$ elements called <u>vertices</u> and

$$E \subseteq \{(v,w) \mid v, w \in V, v \neq w\}$$

is a set of $e = |E|$ ordered vertex pairs called edges. Given $v \in V$, the set $A(v) = \{w \in V | (v,w) \in E\}$ is the set of vertices adjacent out from v. The notation $v \rightarrow w$ means $w \in A(v)$; $v \not\rightarrow w$ means $w \notin A(v)$. If $W \subseteq V$, the induced subgraph $G(W)$ of G is the subgraph $G(W) = (W, E(W))$ where $E(W) = \{(v,w) \in E | v,w \in W\}$. If $v \rightarrow w$ implies $w \rightarrow v$, G is called undirected. Two directed edges $v \rightarrow w$ and $w \rightarrow v$ are usually regarded as a single undirected edge, denoted by $v — w$.

For vertices $v, w \in V$, a v,w path of length k is a sequence of vertices $p = [v = v_1, v_2, \ldots, v_{k+1} = w]$ such that $v_i \rightarrow v_{i+1}$ for $1 \le i \le k$. Vertices v_i are assumed distinct except possibly that $v = w$ and then p is a cycle.

For a graph $G = (V,E)$ with $|V| = n$, an ordering of V is a bijection $\alpha: \{1,2,\ldots,n\} \leftrightarrow V$. $G_\alpha = (V,E,\alpha)$ is an ordered graph. For any vertex v, the deficiency $D(v)$ is the set of edges defined by

$$D(v) = \{(x,y) | x \rightarrow v, v \rightarrow y, x \not\rightarrow y, x \neq y\}.$$

The graph $G_v = (V - \{v\}, E(V - \{v\}) \cup D(v))$ is the v-elimination graph of G. For an ordered graph $G_\alpha = (V,E,\alpha)$, the elimination process $P(G_\alpha) = [G = G_0, G_1, \ldots, G_{n-1}]$ is the sequence of elimination graphs defined recursively by $G_0 = G$, $(G_i = G_{i-1})_{\alpha(i)}$ for $i = 1,2,\ldots,n-1$. If $G_i = (V_i, E_i)$ for $i = 0,1,\ldots,n-1$, the fill in $F(G_\alpha)$ is defined by

$$F(G_\alpha) = \bigcup_{i=1}^{n-1} \tau_i$$

where $\tau_i = D(\alpha(i))$ in G_{i-1}, and the elimination graph G_α^* is defined by $G_\alpha^* = (V, E \cup F(G_\alpha))$.

It is not hard to check that when G_α is the graph of M defined as above, then the elimination graph G_1 is the graph of $M^{(2)}$ in (3) and the deficiency $D(\alpha(1))$ corresponds to the numerical "fill in". Similarly the elimination graph G_i would correspond to the matrix $M^{(i+1)}$ as we proceeded recursively to factor $M^{(2)}$ as above. Finally the graph G_α^* is the graph of the matrix C whose strictly lower triangle is the strictly lower part of L and whose upper triangle is U.

Given a graph $G = (V,E)$, an ordering α of V is a perfect elimination ordering of G if $F(G_\alpha) = \phi$. A graph G which has a perfect elimination ordering is a perfect elimination graph. Note that while G may not be perfect G_α^* is always perfect. An ordering α is a minimal elimination ordering of G if no other ordering β satisfies $F(G_\beta) \subset F(G_\alpha)$ where the containment is proper. An ordering α is a minimum elimination ordering of G if no other ordering satisfies $|F(G_\beta)| < |F(G_\alpha)|$.

When the matrices M are symmetric, giving undirected graphs, the corresponding perfect elimination graphs have a special and interesting structure. For example, calling a graph triangulated if every cycle of length $\ell > 3$ has a chord (an edge joining two non-consecutive vertices of the cycle) we have

Theorem A [4,20,22]. An undirected graph G is a perfect elimination graph if and only if it is triangulated.

Further characterizations of triangulated graphs have been given in [12,20]. These graphs arise in other contexts and have also been called chordal [8] and rigid circuit graphs [4]. In [9] Gavril has constructed an $O(n^{2.81})$ algorithm for testing when an undirected graph is triangulated and in [8] he has presented efficient algorithms for finding all maximal cliques, maximum cliques, minimum colorings, maximum independent sets, and minimum clique coverings in tirangulated graphs (for arbitrary graphs, these problems are NP-complete). These algorithms depend upon exploiting the necessary perfect elimination ordering. Assuming that such an ordering is given, it is easy to implement Gavril's algorithms to run in $O(n+e)$ time. ($O(n+e)$ is optimum to within a constant factor; the time bounds he gives can be improved.) Several important classes of graphs, such as trees, k-trees [21], and interval graphs [7,11], are triangulated, and a recognition algorithm for triangulated graphs can be used to recognize interval graphs efficiently [2,7,11]. For a general undirected graph $G = (V,E)$, Ohtsuski [18,19] has given an $O(ne)$ algorithm for finding a minimal ordering, which can equivalently be regarded as a minimal set of edges F such that $\hat{G} = (V,E \cup F)$ is triangulated.

Here we present in § 2 an algorithmic discussion of minimal and perfect orderings for general directed graphs giving reasonably efficient algorithms for finding such orderings. We also discuss the relation between perfect elimination and transitive closure and claim that the problem of finding a minimum ordering on a directed graph is NP-complete. We specialize in § 3 to undirected graphs and present a version of breadth-first search, called a lexicographic search, which gives an $O(n+e)$ algorithm for generating perfect orderings (and testing for triangulation) and an $O(ne)$ algorithm for generating minimal orderings. The latter algorithm also calculates the fill in produced by the ordering (which is something that Ohtsuki's algorithm does not do). We conclude in § 4 with some remarks summarizing the situation.

2. General Properties of Minimal and Perfect Orderings.

In this section we give some properties of minimal and perfect orderings for general directed graphs which lead to efficient ways to calculate such orderings. The results and algorithms also apply to undirected graphs where directed edges (u,v) and (v,u) can then be regarded as an undirected edge $\{u,v\}$. The details of our discussion are available in [24].

Given any ordering α we first characterize the fill in $F(G_\alpha)$ as follows.

Proposition 1. Let $G_\alpha = (V,E,\alpha)$ be an ordered graph. Then (v,w) is an edge of $G^*_\alpha = (V, E \cup F(G_\alpha))$ if and only if there exists a path $\mu = [v = v_1, v_2, \ldots, v_{k+1} = w]$ in G_α such that

(P) $\alpha^{-1}(v_i) < \min(\alpha^{-1}(v), \alpha^{-1}(w))$ for $2 \leq i \leq k$.

Proposition 1 leads to

```
Algorithm FILL:
    for i:=1 until n-1 do begin
        v:=α(i);
search: for each w such that (P)
            do add (v,w) to F(G_α);
    end;
```

If search is implemented efficiently each such search has a time bound of $O(e)$, hence FILL has a time bound of $O(ne)$.

Proposition 2. Let $G = (V,E)$ have a perfect elimination ordering α. Let $x \in V$ and let $(w,y), (y,z) \in E \cup D(x)$ with $\alpha^{-1}(y) < \min(\alpha^{-1}(w), \alpha^{-1}(z))$. Then $(w,z) \in E \cup D(x)$.

Corollary 1. If $G = (V,E)$ is a perfect elimination graph and x is anv vertex, the elimination graph G_x is also a perfect elimination graph.

Corollary 2. If $G = (V,E)$ is a perfect elimination graph with $x \in V$, $D(x) = \phi$, there is a perfect elimination ordering α with $\alpha(1) = x$.

To test for perfection we use

```
Algorithm PERFECT:
    i:=1;
    while a vertex x in graph has D(x) = ϕ do begin
        α(i):=x;
        delete x and edges containing x from graph;
        i:i+1;
    end;
    comment if i = n+1 then graph is perfect and a perfect ordering has been calculated,
        otherwise the graph is not perfect;
```

Proposition 2 guarantees the correctness of algorithm PERFECT. If a few list-processing tricks are used to keep track of the sets $D(x)$, PERFECT can be implemented to run in $O(ne)$ time. While these implementations of algorithms FILL and PERFECT could be regarded as straightforward, we can show that our algorithms cannot be improved too much without finding a new and better transitivity-testing algorithm. In particular, we can show that any algorithm which computes an ordering's fill in can be used to compute the transitive closure of a graph, and any algorithm which tests whether a graph has a perfect elimination order can be used to test a graph for transitivity. Specifically we have the following results.

Proposition 3. Given an acyclic graph G , we can construct in $O(n+e)$ time a graph G_2 with $2n$ vertices and $n+e$ edges and an ordering α_2 such that the edges in $F((G_2)_{\alpha_2})$ correspond one-to-one with the edges in the transitive closure of G .

Thus any algorithm for computing fill in can be converted into an algorithm (with the same time and space requirements, to within a constant factor) for computing the transitive closure of an acyclic graph. (The requirement that the graph be acyclic is not a significant restriction; see [6,17].) Thus the fill in problem is at least as hard as the transitive closure problem.

Proposition 4. Given an acyclic digraph G , we can construct in $O(n+e)$ time a graph G_3 with $3n+1$ vertices and $3e+3n$ edges such that G is transitive if and only if G_3 is perfect elimination.

Thus any algorithm for testing whether a graph is perfect elimination can be used to test a graph for transitivity, at a cost of only a constant factor in the running time. Munro [17] has shown that the transitive closure of a graph can be computed in $O(n^{2.81})$ time using Strassen's fast matrix multiplication method [27]. Various problems, including transitive reduction [1] and Boolean matrix multiplication [6] are known to be computationally equivalent to transitive closure. There may be a way to solve the fill in and perfect ordering problems in $O(n^{2.81})$ time, by reducing them to transitive closure problems, but any improvement beyond $O(n^{2.81})$ would improve the best bound known for Boolean matrix multiplication.

To justify our minimal elimination ordering algorithm we use

Proposition 5. Let $G = (V,E)$ be a perfect elimination graph. Suppose $G' = (V, E \cup F)$ with $F \neq \emptyset$, $E \cap F = \phi$ is also a perfect elimination graph. Then there exists $f \in F$ such that $G' - f = (V, E \cup F - \{f\})$ is a perfect elimination graph.

Proposition 5 easily implies

Theorem 1. Let $G_\alpha = (V,E,\alpha)$ be an ordered graph. Then α is a minimal ordering if and only if for each $f \in F(G_\alpha)$, $G^*_\alpha - f = (V, E \cup F(G_\alpha) - \{f\})$ is not a perfect elimination graph.

In the undirected case this implies

Corollary 3. Let $G_\alpha = (V,E,\alpha)$ be an undirected, ordered graph. Then α is a minimal ordering if and only if each $f \in F(G_\alpha)$ is a unique chord of a cycle of length four in G^*_α.

Hence to find a minimal elimination ordering we use

```
Algorithm MINIMAL:
    pick any ordering $\alpha$ of graph;
    calculate $G' = G^*_\alpha$ and $F = F(G_\alpha)$ ;
    while $f \in F$ such that $G'-f$ is a perfect elimination graph do delete $f$ from $G'$ ;
    find a perfect ordering $\beta$ of $G'$ ;
    comment $\beta$ is a minimal ordering of $G$ ;
```

Theorem 1 guarantees the correctness of this algorithm. MINIMAL has an obvious time bound of $O(n^7)$ if PERFECT is used to test the condition in the while loop, since $|F| \leq n^2$. Some subtleties based on the proof of Proposition 5 can be used to improve the time bound to $O(n^4)$.

We conclude this section with a result which implies that a problem important in practice, that of finding minimal orderings, may be very hard in general.

Theorem 2. The problem of determining whether a given directed graph $G = (V,E)$ has an ordering α which produces a fill in $F(G_\alpha)$ with $F(G_\alpha) \leq k$ is NP-complete.

3. Undirected Elimination and Lexicographic Search.

In this section $G = (V,E,\alpha)$ will be an undirected ordered graph; our discussion here summarizes [23]. For each $v \in V$ let $m_\alpha(v)$ be the vertex w such that $v — w$, $\alpha^{-1}(v) < \alpha^{-1}(w)$, and $\alpha^{-1}(w)$ is minimum. Then we have the following characterization of perfection.

Theorem 3. The ordering α is a perfect elimination ordering of G if and only if for $v — w$ with $\alpha^{-1}(v) < \alpha^{-1}(w)$ we have $m_\alpha(v) - w$ or $m_\alpha(v) = w$.

This theorem leads to algorithms that may be used to test an ordering of an undirected graph for perfection in $O(n+e)$ time rather than the obvious $O(ne)$ and to compute $F(G_\alpha)$ in $O(n+e')$ time where $e' = |E \cup F(G_\alpha)|$. This suggests the possibility that elimination orderings are easier to find on undirected graphs than on directed graphs, a possibility we now confirm.

To find minimal orderings and perfect orderings on undirected graphs, we use a lexicographic ordering scheme which is a special type of breadth-first search. The vertices of the graph are numbered in the order from n to 1. During the search, each vertex v has an associated label $L(v)$ consisting of a set of numbers selected from $\{1,2,\ldots,n\}$, ordered in decreasing order. Given two labels $L_1 = [p_1,\ldots,p_k]$ and $L_2 = [q_1,\ldots,q_\ell]$, we define $L_1 < L_2$ if, for some j, $p_i = q_i$ for $i = 1,2,\ldots,j-1$ and $p_j < q_j$, or if $p_i = q_i$ for $i = 1,2,\ldots,k$ and $k < \ell$. $L_1 = L_2$ if $p_i = q_i$ for $i = 1,\ldots,k$ and $k = \ell$.

```
Algorithm LEX M:
        assign the label $\phi$ to all vertices;
        for i:=n step -1 until 1 do begin
select: pick an unnumbered vertex $v$ with largest label;
            $\alpha(i)$:=v;
update: for each unnumbered vertex $w$ such that there is a path $[v = v_1,v_2,\ldots,v_{k+1} = w]$ with
            $v_j$ unnumbered and label($v_j$) < label(w) for $j = 2,3,\ldots,k$ do add i to label(w);
        end;
```

We call any ordering α which can be generated by LEX M a lexicographic ordering. The labels $L(v)$ which give the final values of label(v) are related to the fill in produced by α.

Proposition 6. If α is a lexicographic ordering of an undirected graph $G = (V,E)$, then in G^*_α $L(w) = \{\alpha^{-1}(v) \mid w — v \text{ and } \alpha^{-1}(w) < \alpha^{-1}(v)\}$ for all $w \in V$.

Proposition 6 is used to prove

Theorem 4. Let α be a lexicographic ordering of $G = (V,E)$. Then any edge $v — w \in F(G_\alpha)$ is the unique chord of some cycle of length four in G^*_α.

Then by Corollary 3 we have

Corollary 4. Every lexicographic ordering of an undirected graph is minimal.

Algorithm LEX M can be implemented to run in $O(ne)$ time since each iteration of the for loop update requires at most $O(e)$ time.

If we are only interested in whether G has a perfect ordering, we can modify LEX M by substituting the following statement for update:

```
update 2: for each unnumbered vertex $w \in adj(v)$ do add i to label(v);
```

The resulting algorithm, called LEX P, can be implemented to run in $O(n+e)$ time and will generate a perfect ordering if G has any perfect ordering. An $O(n+e)$ algorithm based on Theorem 3 can be used to check whether the generated ordering is perfect. Thus there is an $O(n+e)$ algorithm to check whether an undirected graph is triangulated, and to find a perfect ordering if it is.

The lexicographic ordering scheme may be interesting in its own right, in addition to its use for minimal and perfect orderings. Surprisingly, a similar ordering scheme can be used to solve multi-processor scheduling problems [3,15,26].

4. Remarks

(i) The results of the previous two sections seem to indicate that, algorithmically, directed graph elimination is more complicated than undirected graph elimination. For example, if there are algorithms for directed fill and perfection with time bounds as good as the undirected case, then there must be new algorithms for transitivity problems. We have also shown that the minimum ordering problem is NP-complete for directed graphs but can only conjecture this for undirected graphs.

Further evidence of the difficulty of understanding directed elimination appears in the work of Haskins and Rose [12] and Kleitman [16]. In [12], an attempt is made to characterize perfect elimination digraphs by conditions on paths and vertex separation. These conditions are equivalent to perfect elimination in the undirected case, were shown to be necessary in directed graphs and were conjectured also to be sufficient. However in [16], it was shown that such conditions involved behavior on a finite set of paths and no such conditions can be sufficient. It is tempting to speculate that some relationship exists between this observation and NP-completeness of directed minimum fill.

(ii) It is not known how different the quantities $|F(G_\alpha)|/|E|$ and $|F(G_\beta)|/|E|$ can be for α a <u>minimal</u> elimination order and β a <u>minimum</u> elimination order, nor how well certain commonly used ordering heuristics work [20]. For one important special case, the grid graph, G_n, with n^2 vertices (and $O(n^2)$ edges), there exist such α and β such that $|F(G_\alpha)|/|E| = O(n)$ and $|F(G_\beta)|/|E| = O(\log n)$; furthermore the standard "minimum degree" heuristic seems to give an ordering γ with $|F(G_\gamma)|/|E| = O(\log n)$. See [10,13,25] for further discussions.

(iii) To solve the linear system (1), we have allowed reordering among the class PMP^T . It is of some combinatorial interest (although less numerical interest due to the use of pivoting to reduce round-off error) to examine reorderings among the class PMQ , P and Q both permutation matrices. One would then study "bipartite elimination" defining the bipartite graph of M as $B(M) = (R,C,E,\alpha,\beta)$ where

α $\{1,2,\dots,n\} \leftrightarrow R$ (rows)

β $\{1,2,\dots,n\} \leftrightarrow C$ (columns)

and $\{\alpha(i), \beta(j)\} \in E$ iff $m_{ij} \neq 0$. If M is nonsingular, there always exist P and Q (α and β) such that $\{\alpha(i), \beta(i)\} \in E$, $1 \leq i \leq n$. Once such a "diagonal" or matching has been found and the natural identification $R \leftrightarrow C$ is made, one can again consider digraph elimination in an appropriate digraph $G_\gamma = (V,E,\gamma)$. For any "diagonal" the vertex partition of V induced by the strongly connected components of G is the same, but the corresponding strong components themselves can be different. No substantial study of bipartite elimination has been made although some of the above results are reported in [5]. For example, it is not known whether one should find a diagonal first or let fill in serve as part of the diagonal or whether optimizing over PMQ can substantially improve optimizing over PMP^T when M has $M_{ii} \neq 0$, a priori, as when M is symmetric positive definitive. The PMQ elimination problem may be algorithmically interesting since it bears this relationship to the bipartite matching problem (see [14] for an efficient algorithmic solution to the latter problem).

References

[1] A. Aho, M.Garoy, and J. Ullman, "The transitive reduction of a directed graph", SIAM Journal on Computing, Vol. 1 (1972), pp. 131-137.

[2] K. Booth, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, private communication (1974).

[3] E.G. Coffman, Jr. and R.L. Graham, "Optimal scheduling for two processor systems", Acta Informatica, Vol. 1 (1972), pp. 220-213.

[4] G.A. Dirac, "On rigid circuit graphs", Abh. Math. Sem. Univ. Hamburg, Vol. 25 (1961), pp. 71-76.

[5] A. Dulmage and N. Mendelsohn, "Graphs and matrices", Graph Theory and Theoretical Physics, F. Harary, ed., Academic Press, N.Y. (1967), pp. 167-227.

[6] M. Fischer and A. Meyer, "Boolean matrix multiplication and transitive closure", Twelfth Annual Symposium on Switching and Automata Theory (1971), pp. 129-131.

[7] D.R. Fulkerson and O.A. Gross, "Incidence matrices and interval graphs", Pacific Journal of Mathematics, Vol. 15 (1965), pp. 835-855.

[8] F. Gavril, "Algorithms for minimum coloring, maximum clique, minimum covering by cliques and maximum independent set of a chordal graph", SIAM Journal on Computing, Vol. 1 (1972), pp. 180-187.

[9] F. Gavril, "An $O(n^{2.81})$ algorithm for testing chordality of graphs", unpublished manuscript, Syracuse University (1974).

[10] J.A. George, "Nested dissection of a regular finite element mesh", SIAM Journal on Numerical Analysis, Vol. 10 (1973), pp. 345-363.

[11] P.C. Gilmore and A.J. Hoffman, "A characterization of comparability graphs and of interval graphs", Canadian Journal of Mathematics 6, Vol. 16 (1964), pp. 539-548.

[12] L. Haskins and D. Rose, "Toward characterization of perfect elimination digraphs", SIAM Journal on Computing, Vol. 2 (1973), pp. 217-224.

[13] A. Hoffman, M. Martin, and D. Rose, "Complexity bounds for regular finite difference and finite element grids", SIAM Journal on Numerical Analysis, Vol. 10 (1973), pp. 364-369.

[14] J. Hopcroft and R. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing, Vol. 2 (1972).

[15] T.C. Hu, "Parallel sequencing and assembly line problems", Operations Research, Vol. 9 (1961), pp. 841-848.

[16] D. Kleitman, "A note on perfect elimination digraphs", SIAM Journal on Computing, Vol. 3 (1974), pp. 280-282.

[17] I. Munro, "Efficient determination of the strongly connected components and transitive cosure of a directed graph", manuscript.

[18] T. Ohtsuki, "A fast algorithm for finding an optimal ordering in the vertex elimination of a graph", SIAM Journal on Computing, submitted.

[19] T. Ohtsuki, "A graph-theoretic algorithm for optimal pivoting order of sparse matrices", Proceedings Sixth Annual Hawaii International Conference on System Sciences (1973).

[20] D. Rose, "A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations", Graph Theory and Computing, R. Read, ed., Academic Press, N.Y. (1973), pp. 183-217.

[21] D. Rose, "On simple characterizations of k-trees", Discrete Mathematics, Vol. 7 (1974), pp. 317-322.

[22] D. Rose, "Triangulated graphs and the elimination process", Journal of Mathematical Analysis and Applications, Vol. 32 (1970), pp. 597-609.

[23] D. Rose and R. Tarjan, "Algorithmic aspects of vertex elimination on graphs", Memorandum No. ERL-M483, Electronics Research Laboratory, University of California, Berkeley (1974)

[24] D. Rose and R. Tarjan, "Algorithmic aspects of vertex elimination on directed graphs", manuscript.

[25] D. Rose and G. Whitten, "Automatic nested dissection", Proceedings ACM Annual Conference (1974), pp. 82-88.

[26] R. Sethi, "Scheduling graphs on two processors", SIAM Journal on Computing, to appear.

[27] V. Strassen, "Gaussian elimination is not optimal", Numerical Mathematics, Vol. 13 (1964), pp. 354-356.