10/2-2-93 JS ①

ornl

ORNL/TM-12270

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Partitioning a Chordal Graph into Transitive Subgraphs for Parallel Sparse Triangular Solution

Barry W. Peyton
Alex Pothen
Xiaoqing Yuan

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM--12270

DE93 006614

Engineering Physics and Mathematics Division

PARTITIONING A CHORDAL GRAPH INTO TRANSITIVE SUBGRAPHS FOR PARALLEL SPARSE TRIANGULAR SOLUTION

Barry W. Peyton †
Alex Pothen ‡
Xiaoqing Yuan §

† Mathematical Sciences Section
Oak Ridge National Laboratory
P.O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367

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

§ IBM Canada Lab
1150 Eglinton Ave. East
North York, Ontario, M3C 1H7, Canada

Date Published: December 1992

Research was supported by the National Science Foundation under grant CCR-9024954, the U. S. Department of Energy under grant DE-FG02-91ER25095, the Canadian Natural Sciences and Engineering Research Council under grant OGP0008111, and the Applied Mathematical Sciences Research Program of the Office of Energy Research, U. S. Department of Energy.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
managed by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Contents

PARTITIONING A CHORDAL GRAPH INTO TRANSITIVE SUBGRAPHS FOR PARALLEL SPARSE TRIANGULAR SOLUTION

Barry W. Peyton
Alex Pothen
Xiaoqing Yuan

Abstract

A recent approach for solving sparse triangular systems of equations on massively parallel computers employs a factorization of the triangular coefficient matrix to obtain a representation of its inverse in product form. The number of general communication steps required by this approach is proportional to the number of factors in the factorization. The triangular matrix can be symmetrically permuted to minimize the number of factors over suitable classes of permutations, and thereby the complexity of the parallel algorithm can be minimized. Algorithms for minimizing the number of factors over several classes of permutations have been considered in earlier work.

Let $F = L+L^T$ denote the symmetric filled matrix corresponding to a Cholesky factor L, and let G_F denote the adjacency graph of F. In this paper we consider the problem of minimizing the number of factors over all permutations which preserve the structure of G_F. The graph model of this problem is to partition the vertices G_F into the fewest transitively closed subgraphs over all perfect elimination orderings while satisfying a certain precedence relationship. The solution to this chordal graph partitioning problem can be described by a greedy scheme which eliminates a largest permissible subgraph at each step. Further, the subgraph eliminated at each step can be characterized in terms of lengths of chordless paths in the current elimination graph. This solution relies on several results concerning *transitive perfect elimination orderings* introduced in this paper. We describe a partitioning algorithm with $\mathcal{O}(|V| + |E|)$ time and space complexity.

Keywords: chordal graph, directed acyclic graph, massively parallel computers, partitioned inverse, perfect elimination ordering, sparse triangular solution, transitive closure, transitive perfect elimination ordering

AMS(MOS) subject classifications: primary 65F50, 65F25, 68R10

1. Introduction

We consider a graph partitioning problem which arises in the development of a *partitioned inverse* approach to the solution of sparse triangular systems of equations on highly parallel computers. On such machines it is advantageous to compute the solution to a lower triangular system $L\underline{x} = \underline{b}$ by matrix-vector multiplication $\underline{x} := L^{-1}\underline{b}$ when there are several systems (not all available at the same time) involving the matrix L to be solved. This is due to the fact that there is much more parallelism to be exploited in the multiplication approach than in the conventional substitution algorithm. If we can find a factorization $L = \Pi_{i=1}^{t} P_i$, where each factor P_i has the property that P_i and P_i^{-1} have the same nonzero structure, then $L^{-1} = \Pi_{i=t}^{1} P_i^{-1}$ can be represented in a space-efficient manner, storing the t factors P_i^{-1} in the space required for L. Furthermore, the vector $\underline{x}$ can be computed as a sequence of t matrix-vector multiplication steps, exploiting parallelism fully within each step.

The number of factors t in the factorization of L is an important measure since it is proportional to the number of (expensive) router communication steps required by the parallel algorithm based on this approach; hence it is a good predictor of the running time of triangular solution on highly parallel machines like the Connection Machine CM-2. It has been recognized that the triangular matrix can be symmetrically permuted to minimize the number of factors, and hence several strategies for minimizing t over appropriate permutations of L have been considered in previous work [2,11].

Minimizing t over all symmetric permutations of L for which the permuted matrix remains lower triangular gives rise to a *directed acyclic graph (DAG) partitioning problem* [2]. After introducing some notation, we discuss this problem in some detail, after which we proceed with a description of the closely related partitioning problem addressed in this paper.

Let $G_d = (V, F)$ be the directed graph of the matrix L with vertices $V = \{1, \ldots, n\}$ corresponding to the columns of L and edges $E = \{(j, i) : i > j \text{ and } l_{i,j} \neq 0\}$. The edge (j, i) is directed from the lower numbered vertex j to the higher numbered vertex i. It follows that G_d is a directed acyclic graph (DAG). If there exists a directed path from a vertex j to another vertex i in G_d, then j is a *predecessor* of i, and i is a *successor* of j. An *ordering* of G_d is any bijection from V to the set $\{1, 2, \ldots, |V|\}$. A *topological ordering* is any ordering that, for every predecessor-successor pair, numbers the predecessor with a lower number than that received by the successor. Note that the initial ordering imposed on G_d by L is a topological ordering.

Given a set $X \subseteq V$, let $F_X \subseteq F$ be the set comprising every edge from a vertex in X to any vertex in the graph. The *edge subgraph* induced by F_X is the subgraph of G_d with edge set F_X and vertex set consisting of all vertices which are endpoints of these edges. (We will refer to this as the edge subgraph induced by X.) A directed graph is *transitively closed*, or more briefly *transitive*, if the existence of edges (u, v) and (v, w) implies the existence of edge (u, w).

We can now give a precise statement of the DAG partitioning problem:

Problem 1. *Given a DAG G_d, find an ordered partition $R_1 \prec R_2 \prec \cdots \prec R_t$ of its vertices such that*

1. *for every $v \in V$, if $v \in R_i$ then all predecessors of v belong to $R_1, \ldots, R_i$,*

2. *the edge subgraph induced by each R_i is transitively closed, and*

3. *t is minimum over all partitions that satisfy the first two properties.*

Problem 1 can be solved in $\mathcal{O}(|V|\,|F|)$ time and $\mathcal{O}(|F|)$ space when L is an arbitrary lower triangular matrix, or is obtained from the sparse LU factorization of an unsymmetric coefficient matrix [2]. However, if L is a Cholesky factor of a symmetric positive definite matrix, then there is a more efficient $\mathcal{O}(|V|)$ time and space partitioning algorithm [11]. We consider this latter case in more detail now since it will be helpful in describing the graph partitioning problem considered in this paper.

Let A be a symmetric positive definite matrix whose nonzeros are algebraically independent, and let $F = L + L^T$ denote the symmetric *filled matrix* corresponding to its Cholesky factor L. Then G_F, the adjacency graph of F, is a chordal graph.[1] The ordering $\alpha : V \rightarrow \{1, \ldots, |V|\}$ of the vertices of G that corresponds to the order in which the unknowns in the linear system are eliminated is a *perfect elimination ordering* (*PEO*) of G. In the case of sparse symmetric factorization, because G is a chordal graph, the transitive reduction of G_d (a data structure called the *elimination tree* [8]) can be used to obtain an extremely efficient $\mathcal{O}(|V|)$ time and space algorithm for solving the chordal DAG partitioning problem [11]. The only other data required are the outdegrees of the vertices in G_d, which are either already available or easily computed.

Further details on DAG partitioning problems connected with highly parallel algorithms for the solution of sparse triangular systems and computational results from a Connection Machine CM-2 implementation may be found in the papers [2,11]. The partitioned inverse approach has been shown to be normwise but not componentwise forward and backward stable when a certain scalar, which can be loosely described as a growth factor, is small; this scalar is guaranteed to be small when L is well-conditioned [5]. A comprehensive survey of the partitioned inverse approach to highly parallel sparse triangular solution is provided in [1].

The more general chordal graph partitioning problem addressed in this paper arises when we consider a larger class of elimination orderings for Cholesky factorization (thereby potentially reducing t further). Given the matrix A, we may compute an appropriate ordering in two steps: First, we compute the filled graph G_F for a Cholesky factor L by means of a primary fill-reducing ordering; then we compute a secondary reordering that minimizes the number of factors t in the triangular matrix over all reorderings of A that *preserve the structure of the filled graph G_F*. The computed ordering is then applied to the coefficient matrix A *before* the factorization is computed. When there are several systems to be solved involving the same triangular matrix, the use of an ordering for factorization that has been optimized for efficient parallel triangular solution is justified. This two-step approach is similar to that used to compute the Jess and Kees ordering for parallel sparse Cholesky factorization [6,9].

[1] Definitions of some technical terms will be deferred until later in the paper.

Given a chordal graph $G = (V, E)$ with vertices numbered in a *PEO*, we can associate a DAG G_d with G by directing each edge in E from the lower-numbered vertex to the higher-numbered vertex. The more general chordal graph partitioning problem may be stated as follows.

Problem 2. *Given a chordal graph $G = (V, E)$, compute a PEO, the associated DAG G_d, and an ordered partition $R_1 \prec R_2 \prec \cdots \prec R_t$ of its vertices such that*

1. *for every $v \in V$, if $v \in R_i$ then all predecessors of v belong to $R_1, \ldots, R_i$,*
2. *the edge subgraph induced by each R_i is transitively closed, and*
3. *t is minimum over all partitions that satisfy the first two properties for some DAG $\hat{G}_d$, where $\hat{G}_d$ ranges over all DAGs obtained from PEOs of G in the manner described above.*

In this paper we introduce an $\mathcal{O}(|V| + |E|)$ algorithm for solving Problem 2. Our solution, which we discuss briefly now, involves the lengths of certain *chordless*[2] paths in G. A vertex v is an *interior* vertex of a path if it lies on the path and is not an endpoint of the path. Observe that any vertex v is either an interior vertex on some chordless path in the graph, or else it is an endpoint of every chordless path on which it lies. In the former case, let $\lambda(v)$ denote the length of the longest chordless path in G which includes v in its interior. (Note that $\lambda(v) \geq 2$ for all such vertices.) In the latter case, let $\lambda(v) = 1$. The vertices $v \in V$ for which $\lambda(v) = 1$ or $\lambda(v) = 2$ have certain properties which will play a crucial role in our solution to Problem 2. Section 2 introduces a few of these properties.

From among all solutions to Problem 2, choose one for which $|R_1|$ is as large as possible. In Section 3 we show that R_1 is the unique set consisting of vertices v which satisfy $\lambda(v) \leq 2$, and also satisfy $\lambda(u) \leq 2$ for all $u \in adj[v]$ such that $\{u\} \cup adj[u] \subset \{v\} \cup adj[v]$.[3] This characterization moreover can be applied recursively to obtain the largest possible partition member R_i in the reduced graph $G \setminus (R_1 \cup \cdots \cup R_{i-1})$. As we shall see in Section 4, we can solve Problem 2 by using a simple greedy scheme that eliminates at the i-th step a maximum cardinality set R_i from the reduced graph. This greedy scheme is based on concepts associated with *transitive perfect elimination orderings* of subgraphs of G which are introduced in this paper.

The remainder of the paper is concerned with the expansion of this greedy scheme into an efficient algorithm for solving Problem 2. Section 5 develops two ideas needed for efficient implementation of the high-level scheme. Further details needed to realize our goal of an $\mathcal{O}(|V| + |E|)$ implementation are given in Section 6. A few concluding remarks are given in Section 7.

---

[2] A path is *chordless* if no edge in G joins two nonadjacent vertices on the path.

[3] The set $adj[v]$ contains all vertices joined to v by an edge in G.

2. Chordless paths and an adjacency set partition

Assume $G = (V, E)$ is a connected chordal graph,[4] and let the "length" parameters $\lambda(v)$, $v \in V$, be as defined in Section 1. Figure 2.1 displays a chordal graph for which $\lambda(a) = \lambda(b) = \lambda(c) = \lambda(d) = 1$, $\lambda(e) = 2$, and $\lambda(f) = \lambda(g) = 3$. It is interesting to note that the simplicial vertices[5] of the graph are a, b, c, and d: precisely the vertices for which $\lambda(.) = 1$. We formalize the result suggested by this observation later in this section.

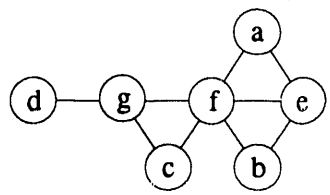

Figure 2.1: Chordal graph with $\lambda(a) = \lambda(b) = \lambda(c) = \lambda(d) = 1$, $\lambda(e) = 2$, and $\lambda(f) = \lambda(g) = 3$.

The following concepts will be used to define an interesting partition of $adj[v]$ in the case where $\lambda(v) \leq 2$. The *neighborhood* of a vertex v is denoted by $nbd[v] := \{v\} \cup adj[v]$. A vertex $u \in adj[v]$ is said to be *indistinguishable from* v if $nbd[u] = nbd[v]$; the set of neighbors indistinguishable from v will be denoted by $adj^0[v]$. A vertex $u \in adj[v]$ is said to *strictly outmatch* v if $nbd \setminus u] \subset nbd[v]$. The set of vertices that strictly outmatch v will be written $adj^-[v]$; the set of vertices strictly outmatched by v will be written $adj^+[v]$. Finally, let $adj^*[v]$ consist of the vertices $u \in adj[v]$ for which $nbd[u]$ and $nbd[v]$ are incomparable. Some of these relationships in Figure 2.1 are: $a \in adj^-[e]$ and $e \in adj^+[a]$; $b \in adj^-[e]$ and $e \in adj^+[b]$; $e \in adj^-[f]$ and $f \in adj^+[e]$. There are no pairs of indistinguishable vertices in Figure 2.1.

It is worth noting that some of these ideas have already played an important role in sparse matrix computations. In particular, vertex indistinguishability and outmatching play an interesting and vital role in efficient implementations of the minimum degree ordering heuristic [4]; vertex indistinguishability also plays a critical role in the subscript compression scheme introduced by Sherman [12] and in improving the time-efficiency of the symbolic factorization step [3].

The reader may easily verify that the sets $adj^-[v]$, $adj^0[v]$, $adj^+[v]$, and $adj^*[v]$ form a partition of $adj[v]$. The following result shows that the vertices $v \in V$ for which $\lambda(v) \leq 2$ are precisely those vertices for which $adj^-[v]$, $adj^0[v]$, and $adj^+[v]$ form a partition of $adj[v]$ (i.e., $adj^*[v] = \emptyset$). Before reading the proof, the reader may find it helpful to verify the result for the graph in Figure 2.1.

[4] A graph is *chordal* if every cycle containing more than three edges has a *chord* (i.e., an edge joining two non-adjacent vertices on the cycle).

[5] A vertex $v \in V$ is *simplicial* if the vertices of $adj[v]$ induce a complete subgraph of G (i.e., $adj[v]$ is a clique in G).

Lemma 1 (Adjacency-Partition Lemma). *The sets $adj^-[v]$, $adj^0[v]$, and $adj^+[v]$ form a partition of $adj[v]$ if and only if $\lambda(v) \leq 2$.*

Proof: We first prove the "only if" part by contraposition. Assume that $adj^-[v]$, $adj^0[v]$, and $adj^+[v]$ do *not* form a partition of $adj[v]$. It follows then that there exists a vertex $u \in adj^*[v]$, and thus we can choose $w_u \in nbd[u] - nbd[v] \neq \emptyset$ and $w_v \in nbd[v] - nbd[u] \neq \emptyset$. Note that w_u, u, v, and w_v are necessarily distinct, and moreover $[w_u, u, v, w_v]$ is a path in G. Since (w_u, v) and (u, w_v) clearly are not edges in G, the only other possible chord for the path is (w_u, w_v). If, however, w_u were joined to w_v by an edge in G, then $[w_u, u, v, w_v, w_u]$ would be a chordless cycle of length four, contrary to the chordality of G. It then follows that $[w_u, u, v, w_v]$ is a chordless path in G, and consequently we have $\lambda(v) \geq 3$.

We now prove the "if" part of the result, also by contraposition. Suppose $\lambda(v) \geq 3$, so that there exists a chordless path $[u, v, w, x]$ of length three in G with v in the interior. Clearly, $u \in nbd[v] - nbd[w]$ and $x \in nbd[w] - nbd[v]$, whence $w \in adj^*[v]$. It follows that $adj^-[v]$, $adj^0[v]$, and $adj^+[v]$ do not form a partition of $adj[v]$, thereby giving us the result. ∎

The vertices $v \in V$ for which $\lambda(v) \leq 2$ play a key role throughout the rest of the paper. The following properties of these vertices will be useful in later proofs. The reader may find it useful to confirm that the result holds for the vertices a, b, c, d, and e in Figure 2.1.

Lemma 2. *1. $\lambda(v) = 1$ if and only if v is simplicial; in which case $adj^-[v] = \emptyset$.*

2. If $\lambda(v) = 2$, then $|adj^-[v]| \geq 2$ and for every vertex $u \in adj^-[v]$ there exists a vertex $u' \in adj^-[v]$ for which $(u, u') \notin E$.

Proof: For the first statement we prove both directions by contraposition. If $\lambda(v) \geq 2$, then v is an interior vertex of some chordless path in G, say $[u, v, w]$. Whereas $u, w \in adj[v]$ and $(u, w) \notin E$, it follows that $adj[v]$ is not complete in G, whence v is not simplicial in G. Now assume v is not simplicial in G. Since $adj[v]$ is not complete in G, we can choose $u, w \in adj[v]$ for which $(u, w) \notin E$. The chordless path $[u, v, w]$ in G ensures that $\lambda(v) \geq 2$. To prove the last part of the first statement, assume that v is simplicial, so that $nbd[v]$ is complete in G. It follows that $nbd[v] \subseteq nbd[w]$ for every vertex $w \in adj[v]$, whence $adj^-[v] = \emptyset$.

To prove the second statement, assume that $\lambda(v) = 2$, and let $[u, v, u']$ be a chordless path in G of length two with v in the interior. It follows from the Adjacency-Partition Lemma that u belongs to one and only one of the sets $adj^-[v]$, $adj^0[v]$, or $adj^+[v]$. Since $u' \in nbd[v] - nbd[u]$, it follows that $u \in adj^-[v]$. By the same argument, $u' \in adj^-[v]$ too, whence $|adj^-[v]| \geq 2$, as required. To prove the last part of the second statement, again assume that $\lambda(v) = 2$; moreover, let $u \in adj^-[v] \neq \emptyset$, so that $nbd[u] \subset nbd[v]$. Choose a vertex $u' \in nbd[v] - nbd[u] \neq \emptyset$. Clearly $u' \notin adj[u]$, whence it follows that $u' \notin adj^0[v] \cup adj^+[v]$, and thus $u' \in adj^-[v]$. This concludes the proof. ∎

Here, also for later use, we verify that each of the sets $adj^0[v] \cup adj^+[v]$, $v \in V$, is complete (i.e., pairwise adjacent) in G.

Lemma 3. *The vertex set $adj^0[v] \cup adj^+[v]$ is complete in G for each $v \in V$.*

Proof: Let $v \in V$, and choose $w, w' \in adj^0[v] \cup adj^+[v]$. Since $nbd[v] \subseteq nbd[w]$, clearly $w' \in adj[w]$, whence $nbd[v]$ is complete in G. ∎

3. Transitive perfect elimination orderings

3.1. Definitions and notation

An *ordering of* G is a bijection

$$\alpha : V \to \{1, 2, \ldots, n\},$$

where $n := |V|$. For any vertex v of an ordered graph, let the *monotone adjacency set* of v be defined by

$$madj[v] := \{w \in adj[v] \mid \alpha(w) > \alpha(v)\}.$$

A *perfect elimination ordering* (PEO) of G is any ordering of G such that $madj[v]$ is complete in G for every vertex $v \in V$.

In this paper we will be interested in perfect elimination orderings that are "partially specified" in the following sense. An *incomplete ordering of* G relative to a vertex set $X \subseteq V$ is a mapping

$$\alpha : V \to \{1, 2, \ldots, |X| - 1, |X|, n + 1\}$$

such that α restricted to X is a bijection from X to $\{1, 2, \ldots, |X|\}$ and $\alpha(v) = n + 1$ for each vertex $v \in V - X$. For convenience we shall refer to such an incomplete ordering of G as an *ordering of* $G(X)$. Whenever $X = V$, clearly the "incomplete" ordering is an ordering of G. A *perfect elimination ordering of* $G(X)$ is an ordering of $G(X)$ such that $madj[v]$ is complete in G for every vertex $v \in X$. (We emphasize that $G(X)$ *does not* refer to the subgraph induced by the vertex set X, and that in the previous sentence $madj[v]$ is complete in the graph G and not in the subgraph induced by X.) Note that any incomplete PEO can be "completed" into a PEO of G.

Unless G is a complete graph, there are some sets $X \subset V$ for which there exists no PEO of $G(X)$. The following result identifies every vertex set $X \subseteq V$ for which there exists a PEO of $G(X)$.

Proposition 1 (Shier [13]). *Let $X \subseteq V$. There exists a PEO of $G(X)$ if and only if the vertices of every chordless path in G joining two vertices in $V - X$ are included in $V - X$.*

A *transitive ordering* of $G(X)$ is any ordering of $G(X)$ for which the following property holds: If $\alpha(u) < \alpha(v) < \alpha(w)$ and $(u, v), (v, w) \in E$, then $(u, w) \in E$. Note that the vertices u and v are necessarily taken from X (because $\alpha(u) < \alpha(v) < n + 1$), while the vertex w may be taken from either X or $V - X$. A *transitive perfect elimination ordering* (TEO) of $G(X)$ is any ordering of $G(X)$ that is both a PEO of $G(X)$ and a transitive ordering of $G(X)$. Any vertex set $X \subseteq V$ for which there exists a TEO of $G(X)$ shall henceforth be called a *T-set of* G.

Due to the additional transitivity condition, the collection of T-sets of G is generally much smaller than the collection of vertex sets $X \subseteq V$ for which merely a PEO of $G(X)$ exists. For example, while there exists a PEO of $G(V)$ for every chordal graph G, it is not the case that there exists a TEO of $G(V)$ for every chordal graph G. On the contrary, V is *not* a T-set for most chordal graphs $G = (V, E)$. Indeed, any chordal graph G for which V is a T-set is also a member of another major class of perfect graphs known as *comparability graphs.*[6] In other words, if a chordal graph G is not also a comparability graph, then V is not a T-set of G. Note, however, that a graph G can be both a chordal graph and a comparability graph without possessing a TEO of $G(V)$. That is, there exist graphs which are both chordal and comparability graphs, but for which the set of transitive orderings is disjoint from the set of perfect elimination orderings. An example is P_4, the path on four vertices.

Though V is not a T-set for most chordal graphs $G = (V, E)$, T-sets nevertheless exist for *any* chordal graph G. For example, consider the vertex set $X = Sim_G \neq \emptyset$, where Sim_G is the set of simplicial vertices of G. It is easy to verify that any ordering of $G(X)$ is a TEO of $G(X)$, and hence X is a T-set of G.

3.2. The T-set of maximum cardinality

In this subsection we show that G has a unique maximum-cardinality T-set R, and that this set is given by

$$R = \{v \in V \mid \lambda(v) \leq 2, \text{ and } \lambda(u) \leq 2 \text{ for every } u \in adj^-[v]\}. \tag{3.1}$$

More specifically, we will show that (a) the vertex set R is a T-set of G, and (b) for any T-set $\hat{R}$ of G we have $\hat{R} \subseteq R$. (The reader can, with some care, verify that these two statements hold for the graph in Figure 2.1 ($R = \{a, b, c, d, e\}$).

Toward that goal, we first characterize the TEOs of $G(R)$. The outmatching relation on V is the key concept needed to obtain the result. Henceforth, for any pair of vertices $u, v \in V$, we shall write $u \prec v$ if $u \in adj^-[v]$, or equivalently, $u \prec v$ if $nbd[u] \subset nbd[v]$. The relation $\prec$ clearly imposes a strict partial order on the vertex set. An ordering α of $G(X)$ is *consistent with the partial order* $\prec$ if $u \prec v$ implies that $\alpha(u) < \alpha(v)$. The following result says that the TEOs of $G(R)$ are precisely the orderings of $G(R)$ that are consistent with the partial order $\prec$.

Theorem 3.1 (*TEO* **Theorem**). *An ordering α of $G(R)$ is a TEO of $G(R)$ if and only if α is consistent with the partial order $\prec$.*

Proof: First we show that any ordering α of $G(R)$ that is consistent with the partial order $\prec$ is a PEO of $G(R)$. Let α be any ordering of $G(R)$ for which $\alpha(u) < \alpha(v)$ whenever $u \prec v$. From (3.1) and the Adjacency-Partition Lemma, it follows that for each vertex $v \in R$ the sets $adj^-[v]$, $adj^0[v]$, and $adj^+[v]$ form a partition of $adj[v]$. Furthermore, our assumption that α is consistent with the partial order $\prec$ implies that for each vertex $v \in R$, the set $madj[v]$ includes no vertices from $adj^-[v]$, and hence

[6] An arbitrary graph $G = (V, E)$ is a *comparability* graph if there exists a transitive ordering of $G(V)$; each comparability graph is associated in a natural way with a finite partially ordered set.

contains only vertices from $adj^0[v] \cup adj^+[v]$. From Lemma 3 it follows that $madj[v]$ is complete in G for every every vertex $v \in R$, and α is therefore a PEO of $G(R)$.

Next we show that any ordering α of $G(R)$ that is consistent with the partial order $\prec$ is also transitive, and hence a TEO of $G(R)$. Assume the ordering α of $G(R)$ is *not* transitive. There exist then vertices $u, v \in R$ and $w \in V$ such that $\alpha(u) < \alpha(v) < \alpha(w)$, $(u,v),(v,w) \in E$, and $(u,w) \notin E$. From (3.1) and the Adjacency-Partition Lemma, it follows that $adj^-[v]$, $adj^0[v]$, and $adj^+[v]$ form a partition of $adj[v]$. Consequently, since $u, w \in adj[v]$ and $(u,w) \notin E$, we have $u, w \in adj^-[v]$. Since $\alpha(v) < \alpha(w)$, the ordering α clearly is not consistent with the partial order $\prec$, and thus we have proven the "if" part of the result.

To complete the proof, we show that any TEO of $G(R)$ is consistent with the partial order $\prec$. Let α be any ordering of $G(R)$ that is not consistent with $\prec$. Then for some vertex $v \in R$ there exists a vertex $u \in adj^-[v]$ such that $\alpha(v) < \alpha(u)$. Now by (3.1) and Lemma 2, $\lambda(v) = 2$ and moreover there exists a vertex $w \in adj^-[v]$, $w \neq u$, that is not adjacent to u. If $\alpha(w) < \alpha(v)$, then we have $\alpha(w) < \alpha(v) < \alpha(u)$, $(w,v),(v,u) \in E$, and $(w,u) \notin E$, whence α is not a transitive ordering of $G(R)$. If on the other hand $\alpha(w) > \alpha(v)$, then $u, w \in madj[v]$ and $(w,v) \notin E$, whence α is not a PEO of $G(R)$. In either case, α is not a TEO of $G(R)$, and this concludes the proof. ∎

That the vertex set R is a T-set of G follows immediately from the TEO Theorem. We now show that any T-set of G is contained in R.

Theorem 3.2. *For any T-set $\hat{R}$ of G, we have $\hat{R} \subseteq R$.*

Proof: To prove the result it suffices to show that for every vertex $v \in V - R$ there exists no T-set that contains v. We therefore choose a vertex $v \in V - R$ and consider in turn the following two mutually exclusive cases, at least one of which must hold true:

1. $\lambda(v) \geq 3$.
2. $\lambda(v) = 2$, but $\lambda(u) \geq 3$ for some vertex $u \in adj^-[v]$.

Assume first that $\lambda(v) \geq 3$, and let $[u,v,w,x]$ be a chordless path of length three in G with v in the interior. Let α moreover be any PEO of $G(\hat{R})$ where $v \in \hat{R}$. It suffices to show that α is not a transitive ordering of $G(\hat{R})$. Since $v \in \hat{R}$, we have $\alpha(v) \neq \alpha(w)$; there are, therefore, two cases to consider. Consider first the case where $\alpha(v) < \alpha(w)$. Since α is a PEO of $G(\hat{R})$, it follows that $\alpha(u) < \alpha(v) < \alpha(w)$. Such an ordering cannot be a transitive ordering of $G(\hat{R})$ because $(u,v),(v,w) \in E$, but $(u,w) \notin E$. Now consider the case where $\alpha(w) < \alpha(v)$. Since α is a PEO of $G(\hat{R})$, it follows that $\alpha(x) < \alpha(w) < \alpha(v)$. Such an ordering cannot be a transitive ordering of $G(\hat{R})$ because $(x,w),(w,v) \in E$, but $(x,v) \notin E$.

Now suppose that $\lambda(v) = 2$, but $\lambda(u) \geq 3$ for some vertex $u \in adj^-[v]$. Again let α be any PEO of $G(\hat{R})$ where $v \in \hat{R}$; it again suffices to show that α is not a transitive ordering of $G(\hat{R})$. First, by the argument in the preceding paragraph it is impossible for α to be a transitive ordering of of $G(\hat{R})$ if $u \in \hat{R}$, and thus we assume that $u \notin \hat{R}$; that is, we assume that $\alpha(u) = n + 1$. By Lemma 2, there exists another vertex $w \in adj^-[v]$ such that $(w,u) \notin E$. Note that $[u,v,w]$ is a chordless path in G. Since $\alpha(v) < \alpha(u) = n + 1$, we must have $\alpha(w) < \alpha(v) < \alpha(u)$ in order for α to be

a *PEO* of $G(\hat{R})$. Such an ordering however cannot be a transitive ordering of $G(\hat{R})$ because $(w,v),(v,u) \in E$, but $(w,u) \notin E$. This concludes the proof. ∎

4. A greedy scheme for the chordal partitioning problem

We can partially reduce the graph G by choosing a T-set $\hat{R}$ of G and removing the vertices in $\hat{R}$ from G in the order specified by a TEO of $G(\hat{R})$; we then complete the reduction of G to the null graph by applying this process recursively to the reduced graph $G \setminus \hat{R}$.

Suppose the graph G is reduced to the null graph after the removal of t distinct T-sets, each ordered by a TEO. Define $G_1 := G$, and let $G_2, G_3, \ldots, G_{t+1}$ be the sequence of reduced graphs obtained at the end of each "block" elimination step. (Note that G_{t+1} is the empty graph.) Let $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_t$ be the sequence of T-sets, so that $\hat{R}_i$ is removed from G_i by a TEO of $G_i(\hat{R}_i)$ to obtain the reduced graph $G_{i+1} = G_i \setminus \hat{R}_i$. We shall refer to any vertex set partition $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_t$ obtained by this process as a *T-partition* of V_G;[7] we shall refer to any PEO of G generated by this process as a *compound* TEO of G with respect to the T-partition $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_t$.

Note that the solution to Problem 2 consists of a compound TEO, along with its associated T-partition and DAG, for which t, the number of members in the partition, is as small as possible. Let $\tau(G)$ be the minimum value t for which there exists a T-partition $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_t$ of V_G. Consider a greedy approach for generating a T-partition of V by eliminating the T-set of maximum cardinality at each major step, as shown in Figure 4.1. We let $R_1, R_2, \ldots, R_t$ be the T-partition of V_G obtained by this process. For the graph in Figure 2.1, the T-partition obtained by this process has members $R_1 = \{a,b,c,d,e\}$ and $R_2 = \{f,g\}$.

$i \leftarrow 1$;
$G_1 \leftarrow G$;
while $G_i \neq \emptyset$ **do**
    Let R_i be the maximum-cardinality T-set of G_i;
    Compute $G_{i+1} \leftarrow G_i \setminus R_i$,
        with R_i removed in a TEO of $G_i(R_i)$;
    $i \leftarrow i + 1$;
end while

Figure 4.1: Greedy partitioning scheme for which each R_i is the maximum-cardinality T-set of G_i.

It is not difficult to show that this process obtains a minimum-cardinality T-partition

[7] Henceforth we will incorporate the graph into our notation as a subscript when needed. For example, if G has been reduced to G_i, we might write V_{G_i}, $\lambda_{G_i}(v)$, $adj_{G_i}[v]$, etc. to distinguish these items from the corresponding items for a different graph.

of V_G, and hence a solution to Problem 2. First we show that $\tau(H) \leq \tau(G)$ for any induced subgraph H of G, after which the main result of this section can be obtained by a simple induction argument.

Lemma 4. *For any induced subgraph H of G, we have $\tau(H) \leq \tau(G)$.*

Proof: Let $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_t$ be a T-partition of V_G, and let α be a compound TEO of G with respect to $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_t$. Consider the subgraph H of G induced by $X \subseteq V$ and the unique ordering β of H that is *consistent* with α in the sense that $\beta(u) < \beta(v)$ whenever $u, v \in X$ and $\alpha(u) < \alpha(v)$. Now, for every vertex $v \in X$ we have $madj_H[v] \subseteq madj_G[v]$, with $madj_G[v]$ complete in G. It follows therefore that $madj_H[v]$ is complete in H for every vertex $v \in X$, whence β is a PEO of H.

Let $\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_t$ be the partition of X defined by $\tilde{R}_i = \hat{R}_i \cap X$, $1 \leq i \leq t$. To prove the result it suffices to show that β is a compound TEO of H with respect to $\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_t$. Clearly, β is a "block" ordering of V_H, consecutively numbering the vertices in $\tilde{R}_i$ before numbering next those in $\tilde{R}_{i+1}$. In the previous paragraph we showed that β is a PEO of H. To complete the proof, it suffices to show that β restricted to $\tilde{R}_i$ is a transitive ordering of $H_i(\tilde{R}_i)$. Toward that end, assume that $u, v \in \tilde{R}_i$, $w \in X$, $\beta(u) < \beta(v) < \beta(w)$, and $(u, v), (v, w) \in E_H$. It follows that $u, v \in \hat{R}_i$, $\alpha(u) < \alpha(v) < \alpha(w)$, and $(u, v), (v, w) \in E_G$. Since α is a compound TEO of G with respect to the T-partition $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_t$, we have $(u, w) \in E_G$, which in turn implies that $(u, w) \in E_H$, thereby giving us the result. ∎

Theorem 4.1. *The greedy partitioning scheme in Figure 4.1 generates a minimum-cardinality T-partition of V_G.*

Proof: We prove the result by induction on $n = |V_G|$. Clearly, the result is true for $n \leq 2$. Let G be a graph with $n \geq 3$ vertices, and assume the greedy scheme produces a minimum-cardinality T-partition for any graph with fewer vertices. Let $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_s$ be a T-partition of V_G for which $s = \tau(G)$, and let $R_1, R_2, \ldots, R_t$ be the T-partition of V_G generated by the greedy scheme in Figure 4.1. Clearly $\tau(G) = s \leq t$; thus to prove the result it suffices to show that $t \leq s$.

Since the greedy scheme applied to G processes the reduced graph $G \setminus R_1$ precisely as it does when applied directly to $G \setminus R_1$, it follows by the induction hypothesis that $R_2, R_3, \ldots, R_t$ is a minimum-cardinality T-partition of $V_G - R_1$, and thus we have $t - 1 = \tau(G \setminus R_1)$. Now, Theorem 3.2 implies that $\hat{R}_1 \subseteq R_1$, whence $G \setminus R_1$ is an induced subgraph of $G \setminus \hat{R}_1$. Whereas $\hat{R}_2, \hat{R}_3, \ldots, \hat{R}_s$ is a T-partition of $V_G - \hat{R}_1$, it follows by Lemma 4 that

$$t - 1 = \tau(G \setminus R_1) \leq \tau(G \setminus \hat{R}_1) \leq s - 1.$$

In consequence we have $t \leq s$ as required. ∎

5. Computing a maximum-cardinality T-set

This section introduces an algorithm for computing the maximum-cardinality T-set R and a TEO of $G(R)$. The algorithm removes one simplicial vertex after another from

the graph so that upon termination the vertices of R have been eliminated and the order in which they were eliminated is a TEO of $G(R)$. Using this algorithm, Section 6 presents the implementation details needed for a linear-time implementation of the greedy scheme in Figure 4.1.

The algorithm introduced in this section is based on two simple ideas. As the algorithm eliminates simplicial vertices from the graph, new simplicial vertices appear in the reduced graph. The first, and most important, idea incorporated into the algorithm is a technique for determining whether or not a "candidate" simplicial vertex in the reduced graph is a member of R and hence should be eliminated. Let $\hat{R}$ denote the set of vertices that have been eliminated thus far by the algorithm, and let v be the next simplicial vertex examined as a candidate for elimination. We will show that, within the context of our algorithm, $v \in R$ if and only if

$$adj_G[v] - \hat{R} \subseteq adj_G^0[v] \cup adj_G^+[v]. \tag{5.1}$$

To enable the test in (5.1) to accurately distinguish members from non-members of R, the order in which the candidate simplicial vertices are examined must be carefully prescribed. The second idea incorporated into the algorithm deals with this issue. Let $deg_G(v)$ be the degree of a vertex v in G (i.e., $|adj_G[v]|$). At each step, the algorithm chooses as the next vertex to examine for elimination a candidate simplicial vertex u for which $deg_G(u)$ is *minimum*. Whenever $u \in adj_G^-[v]$, we have $nbd_G[u] \subset nbd_G[v]$, whence $deg_G(u) < deg_G(v)$. We therefore incorporated this particular ordering of the candidates into the algorithm to enforce examination of the vertex $u \in adj_G^-[v]$ *before* examination of v, so that whenever the algorithm finally tests whether or not a vertex v satisfies (5.1), it will have already examined, and if called for, eliminated, every member of $adj_G^-[v]$.

We have incorporated these two ideas into the algorithm shown in Figure 5.1. The algorithm collects the eliminated vertices in the set $\hat{R}$. The set C contains the *candidate* simplicial vertices belonging to the current elimination graph. Initially $C = Sim_G$. As the computation proceeds each "successful" candidate is eliminated from both the graph and the set C. When elimination of a successful candidate v results in a new simplicial vertex w in the reduced graph, the algorithm places w in C where it will be examined later for possible elimination.

Before proving the algorithm correct we examine how it processes the graph shown in Figure 2.1. Initially, $C = Sim_G = \{a, b, c, d\}$. It is trivial to verify that each of these vertices will pass the test for inclusion in $\hat{R}$ when it is finally examined by the algorithm. (It can be proven formally using Lemma 2 and the Adjacency-Partition Lemma). The vertex d (degree one in G) will be removed first, whereupon the newly simplicial vertex g will be added to the candidate set C. The vertices a, b, and c, each of degree two in G, will be removed next in succession. Observe that after the removal of these vertices, e has become simplicial and f remains non-simplicial, whence $C = \{e, g\}$. The algorithm will next examine either e or g for inclusion in $\hat{R}$. (Both are of degree three in G.) No matter which is examined first, g will fail the test because the vertex $f \in adj_G^*[g]$ remains uneliminated, and e will pass the test because the vertex $f \in adj_G^+[e]$ is the only neighbor of e in the reduced graph. The vertex f (degree five in

$H \leftarrow G$; $\hat{R} \leftarrow \emptyset$; $C \leftarrow Sim_G$;
while $C \neq \emptyset$ **do**
  Choose $v \in C$ for which $deg_G(v)$ is minimum;
  $C \leftarrow C - \{v\}$;
  if $adj_G[v] - \hat{R} \subseteq adj^0_G[v] \cup adj^+_G[v]$ **then**
    $H' \leftarrow H \setminus \{v\}$; $\hat{R} \leftarrow \hat{R} \cup \{v\}$;
    for $w \in Sim_{H'} - Sim_H$ **do**
      $C \leftarrow C \cup \{w\}$;
    end for
    $H \leftarrow H'$;
  end if
end while

Figure 5.1: High-level algorithm for computing the maximum-cardinality T-set R and a TEO of $G(R)$. Upon termination, $\hat{R} = R$ and the elimination sequence is a TEO of $G(R)$.

G) becomes simplicial upon the removal of e, but upon examining it the algorithm will reject it for membership in $\hat{R}$ because the vertex $g \in adj^+_G[f]$ remains uneliminated. The algorithm thus terminates with $\hat{R} = R = \{a, b, c, d, e\}$ as required.

While the primary purpose of the following result is to prove the algorithm correct, it also shows that the minimum degree among the candidates is *non-decreasing* as the algorithm proceeds. This property of the algorithm provides the implementation presented in Section 6 with efficient access to the minimum-degree members of C.

Theorem 5.1. *The set of vertices $\hat{R}$ removed by the algorithm in Figure 5.1 is precisely the maximum cardinality T-set R. Furthermore, the order in which the vertices are removed is a TEO of $G(R)$ and the minimum degree among the vertices of C is non-decreasing as the algorithm proceeds.*

Proof: Let $\hat{R}$ be the set of vertices removed by the algorithm. We first show that $\hat{R} \subseteq R$. Toward that end, let $\tilde{R}$ denote the set of vertices already selected for elimination at some point during the computation, and let v be the next vertex selected for elimination. To prove that $\hat{R} \subseteq R$, it suffices to prove the following: if $\tilde{R} \subseteq R$, then $v \in R$.

Let $\tilde{R}$ and v be as stated above, and consider a vertex $u \in adj_G[v] \cap \tilde{R}$. Since $u \in \tilde{R} \subseteq R$, by (3.1) we have $\lambda_G(u) \leq 2$, and thus by the Adjacency-Partition Lemma the sets $adj^-_G[u]$, $adj^0_G[u]$, and $adj^+_G[u]$ form a partition of $adj_G[u]$. It follows that u belongs to one of the three sets $adj^-_G[v]$, $adj^0_G[v]$, and $adj^+_G[v]$. Now consider a vertex $w \in adj_G[v] - \tilde{R}$. Since v passes the test for inclusion in $\hat{R}$, it follows that $w \in adj^0_G[v] \cup adj^+_G[v]$. We have therefore shown that $adj^-_G[v]$, $adj^0_G[v]$, and $adj^+_G[v]$ form a partition of $adj_G[v]$, whence $\lambda_G(v) \leq 2$ by the Adjacency-Partition Lemma. Since $adj_G[v] - \tilde{R} \subseteq adj^0_G[v] \cup adj^+_G[v]$, we have $adj^-_G[v] \subseteq \tilde{R} \subseteq R$; hence, by (3.1), $\lambda_G(u) \leq 2$ for each vertex $u \in adj^-_G[v]$. It follows by (3.1) then that $v \in R$, giving us $\hat{R} \subseteq R$ as

required. This concludes the first part of the proof.

We now complete the proof that $\hat{R} = R$ by showing that $\hat{R}$ is not properly contained in R. By way of contradiction assume that $\hat{R} \subset R$. Choose $v \in R - \hat{R}$ for which $deg_G(v)$ is *minimum*. We first show that $adj_G^-[v] \subseteq \hat{R}$. Consider a vertex $u \in adj_G^-[v]$. By (3.1), $\lambda_G(u) \leq 2$; moreover, since $v \in R$ and $adj_G^-[u] \subset adj_G^-[v]$, it follows by (3.1) that $u \in R$. From $nbd_G[u] \subset nbd_G[v]$ we have $deg_G(u) < deg_G(v)$, and thus by the minimality of $deg_G(v)$ among the vertices of R excluded from $\hat{R}$, it follows that $u \in \hat{R}$, thereby giving us $adj_G^-[v] \subseteq \hat{R}$.

Let $\tilde{R}$ be the set of vertices already selected for elimination by the algorithm *immediately after the last vertex of* $adj_G^-[v]$ *has been selected for inclusion in* $\hat{R}$, so that we have $adj_G^-[v] \subseteq \tilde{R}$. It follows by applying the Adjacency-Partition Lemma to $v \in R$ that $adj_G^-[v]$, $adj_G^0[v]$, and $adj_G^+[v]$ form a partition of $adj_G[v]$, and thus we have $adj_G[v] - \tilde{R} \subseteq adj_G^0[v] \cup adj_G^+[v]$. In consequence, v is simplicial in the reduced graph $G \setminus \hat{R}$ (by Lemma 3) and also henceforth satisfies the test for inclusion in $\hat{R}$. Observe that the algorithm has not yet examined v for inclusion in $\hat{R}$, because $deg_G(u) < deg_G(v)$ for any vertex $u \in adj_G^-[v]$, and moreover u becomes simplicial in the reduced graph no later than v does. The algorithm therefore eventually examines v sometime *after* eliminating the last member of $adj_G^-[v]$ and includes it in $\hat{R}$, despite our assumption to the contrary. From this contradiction we conclude that $\hat{R} = R$.

To conclude the argument, note that the test for inclusion in $\hat{R}$ ensures that for every vertex $v \in \hat{R} = R$ the vertices of $adj_G^-[v]$ precede v in the elimination sequence. The elimination sequence is therefore, by the TEO Theorem, a TEO of $G(\hat{R})$. Finally, note that the test for inclusion in $\hat{R}$ also ensures that $deg_G(w) > deg_G(v)$ for each new simplicial vertex w resulting from the elimination of v. In consequence, the minimum degree among the vertices in C is non-decreasing, which concludes the proof. ∎

6. Implementing the greedy scheme

Repeated application of the algorithm in Figure 5.1 to a chordal graph gives us an algorithm that implements the greedy partitioning scheme in Figure 4.1. With careful attention to certain implementation details, we can obtain an algorithm whose runtime is linear in the number of vertices and edges in the chordal graph.

Two implementation issues in particular must be successfully dealt with in order to achieve a linear-time algorithm. First, we need an efficient technique for detecting *new* simplicial vertices (i.e., the vertices $w \in Sim_{H'} - Sim_H$ in Figure 5.1). Liu and Mirzaian [9] showed how to use a previously computed PEO and certain vertex degree information in the graph to devise a simple and efficient test for simpliciality. We briefly discuss this test in Section 6.1.

Second, we need an efficient way to implement the test for membership of a candidate simplicial vertex in R. Note that straightforward determination of whether or not a vertex v satisfies (5.1) would require examination of the set $adj_G[w]$ for each vertex $w \in adj_G[v] - \hat{R}$, which is far too costly. We show in Section 6.2 that judicious use of vertex degree information leads to a simple and efficient test that is equivalent to (5.1).

Other implementation issues are fairly straightforward and will be dealt with when

we look at the detailed algorithm in Section 6.3. In Section 6.4 we show that the time-complexity of the algorithm is $\mathcal{O}(|V| + |E|)$.

6.1. An efficient test for simpliciality

In their efficient implementation of the Jess and Kees reordering algorithm, Liu and Mirzaian [9] address the issue of how to determine when a vertex has become simplicial in the reduced graph. Their approach requires a perfect elimination ordering β of the chordal graph. Throughout the rest of Section 6 we will often subscript the vertices with their position in this PEO; that is, we will let $V_G = \{v_1, v_2, \ldots, v_n\}$, where $\beta(v_j) = j$ for $1 \leq j \leq n$. Note that a PEO can be computed in $\mathcal{O}(|V| + |E|)$ time using the maximum cardinality search algorithm [14].

For each vertex v_j, let f_j be the index given by

$$f_j := \min\{k \mid v_k \in nbd_G[v_j]\},$$

and let $mdeg_G(v_j)$, the *monotone degree* of v_j, be given by

$$mdeg_G(v_j) := |madj_G[v_j]|.$$

The following result is Theorem 3.5 in Liu and Mirzaian [9].

Proposition 2 (Liu and Mirzaian [9]). *We have $v_j \in Sim_G$ if and only if $deg_G(v_j) = mdeg_G(v_{f_j})$.*

In order to use the simpliciality test of Proposition 2, the algorithm will maintain the degree values $deg_H(v_j)$ and $mdeg_H(v_j)$ in the variables $deg(v_j)$ and $mdeg(v_j)$ respectively, where H is the current reduced graph.

6.2. An efficient test for membership in R

As noted earlier, a naive implementation of the test in (5.1) is far too expensive to lead to a linear-time implementation. The following result provides us with an efficient alternative to (5.1).

Proposition 3. *Suppose the algorithm in Figure 5.1 is currently testing the simplicial vertex $v \in C$ for elimination, and let $\hat{R}$ be the set of vertices removed from G thus far by the algorithm. We then have (5.1) if and only if*

$$|nbd_G[u] - \hat{R}| = |nbd_G[v] - \hat{R}| \qquad \text{for every } u \in nbd_G[v] \cap \hat{R}. \tag{6.1}$$

Proof: Let v and $\hat{R}$ be as stated, and choose a vertex $u \in nbd_G[v] \cap \hat{R}$. Because u was simplicial in the reduced graph from which it was removed, it follows that $nbd_G[u] - \hat{R}$ is complete in G. Since v belongs to the clique $nbd_G[u] - \hat{R}$, the following statement holds true:

$$nbd_G[u] - \hat{R} \subseteq nbd_G[v] - \hat{R} \qquad \text{for every } u \in nbd_G[v] \cap \hat{R}. \tag{6.2}$$

Assume that (6.1) holds. It follows then from (6.2) that

$$nbd_G[u] - \hat{R} = nbd_G[v] - \hat{R} \qquad \text{for every } u \in nbd_G[v] \cap \hat{R}. \tag{6.3}$$

Choose a vertex $w \in adj_G[v] - \hat{R}$. To show that (5.1) holds, it suffices to show that $nbd_G[v] \subseteq nbd_G[w]$. Let $x \in nbd_G[v]$. If x belongs to the clique $nbd_G[v] - \hat{R}$ from which w was taken, clearly $x \in nbd_G[w]$ as required. If on the other hand $x \in nbd_G[v] \cap \hat{R}$, then from (6.3) we have $w \in nbd_G[v] - \hat{R} = nbd_G[x] - \hat{R}$, whence $x \in nbd_G[w]$, completing the first half of the argument.

Now assume that (5.1) holds, and choose a vertex $u \in nbd_G[v] \cap \hat{R}$. To show that (6.1) holds, it suffices (by (6.2)) to show that

$$nbd_G[v] - \hat{R} \subseteq nbd_G[u] - \hat{R}.$$

Clearly, v belongs to both sets. Let $w \neq v$ belong to $nbd_G[v] - \hat{R}$. It follows by (5.1) that $w \in adj^0_G[v] \cup adj^+_G[v]$. In consequence, $nbd_G[v] \subseteq nbd_G[w]$; hence $u \in nbd_G[w]$, and thus $w \in nbd_G[u] - \hat{R}$, which completes the proof. ∎

To test for (6.1), our algorithm must accurately maintain the variable $deg(u) = |adj_G[u] - \hat{R}|$ for *eliminated* vertices $u \in \hat{R}$ as well as uneliminated vertices $u \in V_G - \hat{R}$.

6.3. Implementation details

The algorithm introduced in Figure 6.1 (along with Figures 6.2, 6.3, and 6.4) implements the greedy scheme introduced in Figure 4.1. That is, it generates the minimum-cardinality T-partition $R_1, R_2, \ldots, R_t$, where each partition member R_i is the unique maximum-cardinality T-set of the reduced graph $G_i = G \setminus \{R_1 \cup \cdots \cup R_{i-1}\}$, and it also generates a compound TEO of G with respect to the T-partition $R_1, R_2, \ldots, R_t$. For efficient access to the candidate simplicial vertex of smallest degree in G_i, the algorithm maintains a collection of sets $C[d]$ $(1 \leq d \leq n)$, where $C[d]$ contains the current candidate simplicial vertices w for which $deg_{G_i}(w) = d$. We now discuss other details of the implementation.

Initialization for the algorithm is performed by the procedure INITIALIZE shown in Figure 6.2. This procedure initializes S_1 to Sim_G (see Proposition 2), each candidate set $C[d]$ to the empty set, and each marker variable $mark(v_j)$ to an appropriate integer value. The various values taken on by the marker variables $mark(v_j)$ during the course of the algorithm have the meanings given below:

$$mark(v_j) = \begin{cases} 0 & \text{if } v_j \text{ has been eliminated during an earlier major step} \\ 1 & \text{if } v_j \text{ has been eliminated during the current major step} \\ 2 & \text{if } v_j \text{ is simplicial, but not yet chosen for elimination} \\ 3 & \text{if } v_j \text{ is not yet simplicial,} \end{cases}$$

where each major step is a single iteration of the main **while** loop.

An iteration of the main **while** loop in Figure 6.1 removes the vertices of the maximum-cardinality T-set R_i from the reduced graph G_i, generating a TEO of $G_i(R_i)$ as the elimination sequence for the set. Note that the set $S_i = Sim_{G_i}$ is available at the

Input: A chordal graph $G = (V, E)$; for each vertex $v_j \in V$, $deg(v_j)$ $(= deg_G(v_j))$, $mdeg(v_j)$ $(= mdeg_G(v_j))$, and $adj_G[v_j]$, sorted in ascending order by the numbers assigned by the initial PEO.
Output: Upon termination, $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_t$ is precisely the minimum-cardinality T-partition $R_1, R_2, \ldots, R_t$, where each partition member R_i is the maximum-cardinality T-set of the reduced graph $G_i = G \setminus \{R_1 \cup \cdots \cup R_{i-1}\}$. The PEO α (computed in Figure 6.4) is a compound TEO of G with respect to the T-partition $R_1, R_2, \ldots, R_t$.

```
INITIALIZE(mark(*), C[*], S_1);    /*Figure 6.2*/
r ← 0; i ← 1; G_1 ← G; U ← V;
while G_i ≠ ∅ do
    d_max ← 0; d_min ← |V|;
    for v_j ∈ S_i do
        d_max ← max{d_max, deg(v_j)};
        d_min ← min{d_min, deg(v_j)};
        C[deg(v_j)] ← C[deg(v_j)] ∪ {v_j};
    end for
    for v_j ∈ U do olddeg(v_j) ← deg(v_j) end for
    R̂_i ← ∅; S_{i+1} ← ∅; U ← ∅;
    while d_min ≤ d_max do
        for each vertex v_j ∈ C[d_min] do
            C[d_min] ← C[d_min] − {v_j};
            if IN_TSET(v_j) = 1 then    /*Figure 6.3*/
                ELIMINATE(v_j);    /*Figure 6.4*/
            else
                S_{i+1} ← S_{i+1} ∪ {v_j};
            end if
        end for
        while C[d_min] = ∅ and d_min ≤ d_max do
            d_min ← d_min + 1;
        end while
    end while
    G_{i+1} ← G_i \ R̂_i; i ← i + 1;
    for v_j ∈ R̂_i do mark(v_j) ← 0 end for
end while
```

Figure 6.1: Detailed implementation of scheme in Figure 4.1.

```
procedure INITIALIZE(mark(*), C[*], S_1)
S_1 ← ∅;
for d ∈ {1, 2, ..., n} do C[d] ← ∅ end for
for j ∈ {1, 2, ..., n} do
    if deg(v_j) = mdeg(v_{f_j}) then
        mark(v_j) ← 2; S_1 ← S_1 ∪ {v_j};
    else mark(v_j) ← 3; end if
end for
```

Figure 6.2: Initialization procedure: initializes data structures for main **while** loop.

```
boolean function IN_TSET(v_j)
IN_TSET ← 1;
for each vertex v_k ∈ adj_G[v_j] do
    if mark(v_k) = 1 and deg(v_k) ≠ deg(v_j) + 1 then
        IN_TSET ← 0;
    end if
end for
```

Figure 6.3: Boolean function that tests for membership in the maximum-cardinality T-set R_i.

```
procedure ELIMINATE(v_j)
mark(v_j) ← 1; R̂_i ← R̂_i ∪ {v_j}; U ← U − {v_j};
r ← r + 1; α(v_j) ← r;
for each vertex v_k ∈ adj_G[v_j] in ascending order do
    deg(v_k) ← deg(v_k) − 1;
    if mark(v_k) ≥ 2 then
        Update f_k if necessary; U ← U ∪ {v_k};
        if k < j then mdeg(v_k) ← mdeg(v_k) − 1;
        if deg(v_k) = mdeg(v_{f_k}) and mark(v_k) = 3 then
            mark(v_k) ← 2;
            C[olddeg(v_k)] ← C[olddeg(v_k)] ∪ {v_k};
            d_max ← max{d_max, olddeg(v_k)};
        end if
    end if
end for
```

Figure 6.4: Elimination procedure: updates data structures to reflect the selection of v_j for elimination.

beginning of the i-th iteration. The first **for** loop computes the minimum and maximum degrees encountered among the vertices of Sim_{G_i} (d_{max} and d_{min}, respectively), and also places each simplicial vertex v_j in the appropriate candidate set $C[deg_{G_i}(v_j)]$. The algorithm maintains the degree value $deg_{G_i}(v_j)$ in the variable $olddeg(v_j)$.

The second **for** loop updates $olddeg(v_j)$ for each vertex v_j whose degree was reduced during the preceding major step. To do this efficiently, the algorithm maintains a set U, which contains every uneliminated vertex whose degree has been reduced during the current major step.

As long as there remain candidate simplicial vertices to be processed, the algorithm examines those of minimum degree in G_i (i.e., those in $C[d_{min}]$). For each vertex $v_j \in C[d_{min}]$, the boolean function IN_TSET (see Figure 6.3) uses the current degree information to determine if v_j satisfies the test for elimination given in (6.1). In Figure 6.3, note that

$$deg(v_k) = |adj_G[v_k] - \hat{R}_i| = |nbd_G[v_k] - \hat{R}_i|$$

and

$$deg(v_j) = |adj_G[v_j] - \hat{R}_i| = |nbd_G[v_j] - \hat{R}_i| - 1.$$

If v_j is not to be eliminated at this step, the algorithm then places v_j in the set of simplicial vertices S_{i+1}, where it will be processed (and eliminated) during the next iteration of the main **while** loop. Otherwise, the procedure shown in Figure 6.4 selects v_j for elimination and updates the current T-set $\hat{R}_i$ and the relevant marker and degree

variables. More specifically, while the degree variables of the neighbors of v_j are updated, new simplicial vertices detected in $adj_G[v_j] - \hat{R}_i$ (see Proposition 2) are placed in the appropriate candidate set. The set U of uneliminated vertices whose degrees have been reduced is also updated.

Note that the procedure ELIMINATE must process the members of $adj_G[v_j]$ in ascending order by their numbering in the initial PEO. This is needed to enable efficient updating of the parameters f_k and to ensure that the values $mdeg(v_k)$ have been correctly updated before they are used in simpliciality tests. In Figure 6.4, we have not shown the details of how f_k is updated. Efficient access to f_k can be obtained by maintaining a pointer to the first vertex in the ordered list $adj_G[v_k]$ that has not yet been chosen for elimination. If $f_k = j$, where v_j is the vertex just chosen for elimination, then $adj_G[v_k]$ must be searched to the right of v_j for the new *first* uneliminated vertex, and the pointer must be adjusted accordingly.

After the algorithm examines v_j for possible elimination, it then increases d_{min} if necessary. That d_{min} cannot possibly decrease during the course of a major step was shown in Theorem 5.1. After computing $\hat{R}_i$ $(= R_i)$, the algorithm then eliminates $\hat{R}_i$ from the graph and marks each vertex of $\hat{R}_i$ as eliminated from the graph.

Finally, observe that the algorithm in Figure 6.1 correctly implements the greedy scheme in Figure 4.1 follows immediately from the fact that each iteration of the main **while** loop implements the algorithm in Figure 5.1.

6.4. Complexity analysis

In this section we verify that the algorithm in Figure 6.1 runs in time proportional to $|V| + |E|$. Recall that the algorithm in Figure 6.1 requires

1. a PEO of G, and
2. sorted adjacency lists so that neighbors can be processed in ascending order by their labels in the PEO.

The first can be obtained in $\mathcal{O}(|V| + |E|)$ time using the maximum cardinality search algorithm [14]; the second can be obtained in $\mathcal{O}(|V| + |E|)$ time by careful application of a bin sort. It is worth pointing out that in our application, the PEO and sorting can be obtained as a by-product of the symbolic factorization step, and thus are available at no extra cost in computation time. (For further details consult Liu [7].)

The total work associated with the procedure INITIALIZE is clearly proportional to $|V|$. Because $S_i \subset \hat{R}_i$ at each major step i, the total work performed by the **for** loop that distributes the members of S_i among the candidate sets is also proportional to $|V|$. Each vertex is eliminated from the graph once, and thus the work associated with the procedure ELIMINATE is $\mathcal{O}(|V| + |E|)$. Note that each vertex is eliminated either by the major step during which it first becomes simplicial or by the next major step. As a result, each vertex is examined for possible elimination no more than twice, and consequently the work associated with the boolean function IN_TSET is also $\mathcal{O}(|V| + |E|)$. For each vertex $v_j \in U$ whose "old" degree is updated by the algorithm at major step $i + 1$, we have $v_j \in adj_{G_i}[v_k]$ for some vertex $v_k \in \hat{R}_i$; that is, to each vertex $v_j \in U$ there corresponds one or more edges which were removed from the graph during the

previous major step i. In consequence, the total work spent updating the variables $olddeg(v_j)$ $(1 \leq j \leq n)$ is $\mathcal{O}(|V| + |E|)$.

Finally, we consider the work expended by the **while** loop that updates d_{min}. During any given iteration of the main **while** loop, the work performed updating d_{min} is bounded above by the maximum of $deg_G(v)$ over all vertices v examined for possible elimination during the step. Since each vertex is examined for possible elimination no more than twice during the course of the algorithm, it follows that the total work spent updating d_{min} is $\mathcal{O}(|V| + |E|)$. From this and the foregoing observations, it follows that the time complexity of the algorithm in Figure 6.1 is $\mathcal{O}(|V| + |E|)$. Note that the space complexity is also $\mathcal{O}(|V| + |E|)$.

7. Concluding remarks

In this paper we have developed an $\mathcal{O}(|V| + |E|)$ algorithm for solving the graph partitioning problem stated as Problem 2 in Section 1. Two new ideas—TEOs and T-sets—enabled us to devise a simple greedy scheme that solves Problem 2. We then provided a high-level description of an algorithm for computing a maximum-cardinality T-set R, along with the required TEO of $G(R)$. Careful implementation provides us with a detailed $\mathcal{O}(|V| + |E|)$ algorithm that implements the greedy scheme, and thus solves Problem 2.

The approach taken in this paper has the virtue of simplicity and provides insight into the essential features of this fairly involved graph partitioning problem. A forthcoming paper [10] will present an implementation of a variant of the greedy scheme in Figure 4.1 that processes a clique tree representation of G, rather than the conventional representation by adjacency lists. The new clique tree algorithm makes use of some interesting new concepts about separators in the clique intersection graph of the chordal graph.

Acknowledgement

The third author would like to thank Professor Joseph Liu for the guidance and encouragement he received when he was a student at York University.

8. References

[1] F. L. Alvarado, A. Pothen, and R. S. Schreiber. Highly parallel sparse triangular solution. Technical Report CS-92-51, Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1, Oct. 1992. To appear in Sparse Matrix Computations: Graph Theory Issues and Algorithms, J. A. George and J. R. Gilbert and J. W. H. Liu (eds.), Springer Verlag. (IMA volumes in Mathematics and its Applications).

[2] F. L. Alvarado and R. S. Schreiber. Optimal parallel solution of sparse triangular systems. SIAM J. Sci. Stat. Comput., to appear, 1992.

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

[4] J. A. George and J. W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989.

[5] N. J. Higham and A. Pothen. The stability of the partitioned inverse approach to parallel sparse triangular solution. Technical Report CS-92-52, Computer Science, University of Waterloo, Oct. 1992. Submitted to *SIAM J. Sci. Stat. Comput.*

[6] J. G. Lewis, B. W. Peyton, and A. Pothen. A fast algorithm for reordering sparse matrices for parallel factorization. *SIAM J. Sci. Stat. Comput.*, 6:1146–1173, Nov. 1989.

[7] J. W. H. Liu. Reordering sparse matrices for parallel elimination. *Parallel Computing*, 11:73–91, 1989.

[8] J. W. H. Liu. The role of elimination trees in sparse factorization. *SIAM J. Mat. Anal. Appl.*, 11:134–172, 1990.

[9] J. W. H. Liu and A. Mirzaian. A linear reordering algorithm for parallel pivoting of chordal graphs. *SIAM J. Disc. Math.*, 2:100–107, 1989.

[10] B. W. Peyton, A. Pothen, and X. Yuan. A clique tree algorithm for partitioning a chordal graph into transitive subgraphs. Work in preparation, 1992.

[11] A. Pothen and F. L. Alvarado. A fast reordering algorithm for parallel sparse triangular solution. *SIAM J. Sci. Stat. Comput.*, 13:645–653, 1992.

[12] A. H. Sherman. *On the efficient solution of sparse systems of linear and nonlinear equations.* PhD thesis, Yale University, 1975.

[13] D. R. Shier. Some aspects of perfect elimination orderings in chordal graphs. *Discr. Appl. Math.*, 7:325–331, 1984.

[14] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.

INTERNAL DISTRIBUTION

1. B.R. Appleton
2-3. T.S. Darland
4. E.F. D'Azevedo
5. J.M. Donato
6. J.J. Dongarra
7. G.A. Geist
8. M.R. Leuze
9. E.G. Ng
10. C.E. Oliver
11-15. B.W. Peyton
16-20. S.A. Raby
21. C.H. Romine
22. T.H. Rowan
23-27. R.F. Sincovec
28-32. R.C. Ward
33. P.H. Worley
34. Central Research Library
35. ORNL Patent Office
36. K-25 Appl Tech Library
37. Y-12 Technical Library
38. Lab Records Dept - RC
39-40. Laboratory Records Dept

EXTERNAL DISTRIBUTION

41. Cleve Ashcraft, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346
42. Donald M. Austin, 6196 EECS Bldg., University of Minnesota, 200 Union St., S.E., Minneapolis, MN 55455
43. Robert G. Babb, Oregon Graduate Institute, CSE Department, 19600 N.W. von Neumann Drive, Beaverton, OR 97006-1999
44. Clive Baillie, Physics Department, Campus Box 390, University of Colorado, Boulder, CO 80309
45. Lawrence J. Baker, Exxon Production Research Company, P.O. Box 2189, Houston, TX 77252-2189
46. Jesse L. Barlow, Department of Computer Science, Pennsylvania State University, University Park, PA 16802
47. Edward H. Barsis, Computer Science and Mathematics, P.O. Box 5800, Sandia National Laboratories, Albuquerque, NM 87185
48. Chris Bischof, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
49. Ake Bjorck, Department of Mathematics, Linkoping University, S-581 83 Linkoping, Sweden
50. Jean R. S. Blair, Department of Computer Science, Ayres Hall, University of Tennessee, Knoxville, TN 37996-1301
51. Roger W. Brockett, Wang Professor of Electrical Engineering and Computer Science, Division of Applied Sciences, Harvard University, Cambridge, MA 02138
52. James C. Browne, Department of Computer Science, University of Texas, Austin, TX 78712

53. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307

54. Donald A. Calahan, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109

55. John Cavallini, Deputy Director, Scientific Computing Staff, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20585

56. Ian Cavers, Department of Computer Science, University of British Columbia, Vancouver, British Columbia V6T 1W5, Canada

57. Tony Chan, Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90024

58. Jagdish Chandra, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709

59. Eleanor Chu, Department of Mathematics and Statistics, University of Guelph, Guelph, Ontario, Canada N1G 2W1

60. Melvyn Ciment, National Science Foundation, 1800 G Street N.W., Washington, DC 20550

61. Tom Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853

62. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720

63. Andy Conn, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598

64. John M. Conroy, Supercomputer Research Center, 17100 Science Drive, Bowie, MD 20715-4300

65. Jane K. Cullum, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598

66. George Cybenko, Center for Supercomputing Research and Development, University of Illinois, 104 S. Wright Street, Urbana, IL 61801-2932

67. George J. Davis, Department of Mathematics, Georgia State University, Atlanta, GA 30303

68. Tim A. Davis, Computer and Information Sciences Department, 301 CSE, University of Florida, Gainesville, FL 32611-2024

69. John J. Dorning, Department of Nuclear Engineering Physics, Thornton Hall, McCormick Road, University of Virginia, Charlottesville, VA 22901

70. Iain Duff, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England

71. Patricia Eberlein, Department of Computer Science, SUNY at Buffalo, Buffalo, NY 14260

72. Stanley Eisenstat, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

73. Lars Elden, Department of Mathematics, Linkoping University, 581 83 Linkoping, Sweden

74. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742

75. Albert M. Erisman, Boeing Computer Services, Engineering Technology Applications, ETA Division, P.O. Box 24346, MS-7L-20 Seattle, WA 98124-0346

76. Geoffrey C. Fox, Northeast Parallel Architectures Center, 111 College Place, Syracuse University, Syracuse, NY 13244-4100

77. Paul Frederickson, Los Alamos National Laboratory, Center for Research on Parallel Computing, MS B287, Los Alamos, NM 87545

78. Fred N. Fritsch, L-316, Computing and Mathematics Research Division, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550

79. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650

80. K. Gallivan, Computer Science Department, University of Illinois, Urbana, IL 61801

81. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47405

82. Feng Gao, Department of Computer Science, University of British Columbia, Vancouver, British Columbia V6T 1W5, Canada

83. David M. Gay, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

84. C. William Gear, NEC Research Institute, 4 Independence Way, Princeton, NJ 08540

85. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Road, Ottawa, Ontario, Canada K1A 0R8

86. J. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

87. John R. Gilbert, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto CA 94304

88. Gene H. Golub, Department of Computer Science, Stanford University, Stanford, CA 94305

89. Joseph F. Grcar, Division 8245, Sandia National Laboratories, Livermore, CA 94551-0969

90. John Gustafson, Ames Laboratory, Iowa State University, Ames, IA 50011

91. Per Christian Hansen, UCI*C Lyngby, Building 305, Technical University of Denmark, DK-2800 Lyngby, Denmark

92. Richard Hanson, IMSL Inc., 2500 Park West Tower One, 2500 City West Blvd., Houston, TX 77042-3020

93. Michael T. Heath, National Center for Supercomputing Applications, 4157 Beckman Institute, University of Illinois, 405 North Mathews Avenue, Urbana, IL 61801-2300

94. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P.O. Box 481, Houston, TX 77001

95. Nicholas J. Higham, Department of Mathematics, University of Manchester, Grt Manchester, M13 9PL, England

96. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332

97. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550

98. Ilse Ipsen, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

99. Barry Joe, Department of Computer Science, University of Alberta, Edmonton, Alberta T6G 2H1, Canada

100. Lennart Johnsson, Thinking Machines Inc., 245 First Street, Cambridge, MA 02142-1214

101. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309

102. Bo Kagstrom, Institute of Information Processing, University of Umea, 5-901 87 Umea, Sweden

103. Malvyn H. Kalos, Cornell Theory Center, Engineering and Theory Center Bldg., Cornell University, Ithaca, NY 14853-3901

104. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Bldg. 221, Argonne, IL 60439

105. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

106. Robert J. Kee, Division 8245, Sandia National Laboratories, Livermore, CA 94551-0969

107. Kenneth Kennedy, Department of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77001

108. Thomas Kitchens, Department of Energy, Scientific Computing Staff, Office of Energy Research, ER-7, Office G-236 Germantown, Washington, DC 20585

109. Richard Lau, Office of Naval Research, Code 111MA, 800 Quincy Street, Boston Tower 1, Arlington, VA 22217-5000

110. Alan J. Laub, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106

111. Robert L. Launer, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709

112. Charles Lawson, MS 301-490, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109

113. Peter D. Lax, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012

114. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815

115. John G. Lewis, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346

116. Jing Li, IMSL Inc., 2500 Park West Tower One, 2500 City West Blvd., Houston, TX 77042-3020

117. Heather M. Liddell, Center for Parallel Computing, Department of Computer Science and Statistics, Queen Mary College, University of London, Mile End Road, London E1 4NS, England

118. Arno Liegmann, c/o ETH Rechenzentrum, Clausiusstr. 55, CH-8092 Zurich, Switzerland

119. Joseph Liu, Department of Computer Science, York University, 4700 Keele Street, North York, Ontario, Canada M3J 1P3

120. Robert F. Lucas, Supercomputer Research Center, 17100 Science Drive, Bowie, MD 20715-4300

121. Franklin Luk, Department of Computer Science, Amos Eaton Building - #131, Rensselaer Polytechnic Institute, Troy, NY 12180-3590

122. Thomas A. Manteuffel, Department of Mathematics, University of Colorado - Denver, Campus Box 170, P.O. Box 173364, Denver, CO 80217-3364

123. Consuelo Maulino, Universidad Central de Venezuela, Escuela de Computacion, Facultad de Ciencias, Apartado 47002, Caracas 1041-A, Venezuela

124. James McGraw, Lawrence Livermore National Laboratory, L-306, P.O. Box 808, Livermore, CA 94550

125. Paul C. Messina, Mail Code 158-79, California Institute of Technology, 1201 E. California Blvd., Pasadena, CA 91125

126. Cleve Moler, The Mathworks, 325 Linfield Place, Menlo Park, CA 94025

127. Neville Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green Street, Urbana, IL 61801

128. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742

129. James M. Ortega, Department of Applied Mathematics, Thornton Hall, University of Virginia, Charlottesville, VA 22901

130. Charles F. Osgood, National Security Agency, Ft. George G. Meade, MD 20755

131. Chris Paige, McGill University, School of Computer Science, McConnell Engineering Building, 3480 University Street, Montreal, Quebec, Canada H3A 2A7

132. Roy P. Pargas, Department of Computer Science, Clemson University, Clemson, SC 29634-1906

133. Beresford N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720

134. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706

135. Dan Pierce, Boeing Computer Services, P.O. Box 24346, M/S 7L-21 Seattle, WA 98124-0346

136. Robert J. Plemmons, Departments of Mathematics and Computer Science, Box 7311, Wake Forest University, Winston-Salem, NC 27109

137. Jesse Poore, Department of Computer Science, Ayres Hall, University of Tennessee, Knoxville, TN 37996-1301

138–142. Alex Pothen, Computer Science Department, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

143. Yuanchang Qi, IBM European Petroleum Application Center, P.O. Box 585, N-4040 Hafrsfjord, Norway

144. Giuseppe Radicati, IBM European Center for Scientific and Engineering Computing, via del Giorgione 159, I-00147 Roma, Italy

145. John K. Reid, Numerical Analysis Group, Central Computing Department, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England

146. Werner C. Rheinboldt, Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA 15260

147. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907

148. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore Laboratory, Livermore, CA 94550

149. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706

150. Edward Rothberg, Department of Computer Science, Stanford University, Stanford, CA 94305

151. Axel Ruhe, Dept. of Computer Science, Chalmers University of Technology, S-41296 Goteborg, Sweden

152. Joel Saltz, ICASE, MS 132C, NASA Langley Research Center, Hampton, VA 23665

153. Ahmed H. Sameh, Center for Supercomputing R&D, 1384 W. Springfield Avenue, University of Illinois, Urbana, IL 61801

154. Michael Saunders, Systems Optimization Laboratory, Operations Research Department, Stanford University, Stanford, CA 94305

155. Robert Schreiber, RIACS, Mail Stop 230-5, NASA Ames Research Center, Moffet Field, CA 94035

156. Martin H. Schultz, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

157. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006

158. Lawrence F. Shampine, Mathematics Department, Southern Methodist University, Dallas, TX 75275

159. Andy Sherman, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

160. Kermit Sigmon, Department of Mathematics, University of Florida, Gainesville, FL 32611

161. Horst Simon, Mail Stop T045-1, NASA Ames Research Center, Moffett Field, CA 94035

162. Anthony Skjellum, Lawrence Livermore National Laboratory, 7000 East Ave., L-316, P.O. Box 808 Livermore, CA 94551

163. Danny C. Sorensen, Department of Mathematical Sciences, Rice University, P.O. Box 1892, Houston, TX 77251

164. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742

165. Paul N. Swartztrauber, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307

166. Philippe Toint, Dept. of Mathematics, University of Namur, FUNOP, 61 rue de Bruxelles, B-Namur, Belgium

167. Bernard Tourancheau, LIP, ENS-Lyon, 69364 Lyon cedex 07, France

168. Henk van der Vorst, Dept. of Techn. Mathematics and Computer Science, Delft University of Technology, P.O. Box 356, NL-2600 AJ Delft, The Netherlands

169. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853

170. Jim M. Varah, Centre for Integrated Computer Systems Research, University of British Columbia, Office 2053-2324 Main Mall, Vancouver, British Columbia V6T 1W5, Canada

171. Udaya B. Vemulapati, Dept. of Computer Science, University of Central Florida, Orlando, FL 32816-0362

172. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665

173. Phuong Vu, Cray Research, Inc., 19607 Franz Rd., Houston, TX 77084

174. Daniel D. Warner, Department of Mathematical Sciences, O-104 Martin Hall, Clemson University, Clemson, SC 29631

175. Robert P. Weaver, 1555 Rockmont Circle, Boulder, CO 80303

176. Mary F. Wheeler, Rice University, Department of Mathematical Sciences, P.O. Box 1892, Houston, TX 77251

177. Andrew B. White, Computing Division, Los Alamos National Laboratory, P.O. Box 1663, MS-265, Los Alamos, NM 87545

178. Margaret Wright, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

179. David Young, University of Texas, Center for Numerical Analysis, RLM 13.150, Austin, TX 78731

180–184. Xiaoqing Yuan, IBM Canada Lab, 1150 Eglinton Avenue, North York, Ontario, Canada M3C 1H7

185. Guodong Zhang, CONVEX Computer Corporation, 3000 Waterview Pkwy, P.O. Box 833851, Richardson, TX 75083-3851

186. Earl Zmijewski, Department of Computer Science, University of California, Santa Barbara, CA 93106

187. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P.O. Box 2001 Oak Ridge, TN 37831-8600

188–197. Office of Scientific & Technical Information, P.O. Box 62, Oak Ridge, TN 37831

END

DATE
FILMED
3/9/93