Partitioning a Chordal Graph Into Transitive Subgraphs for Parallel Sparse Triangular Solution*

Barry W. Peyton†
Mathematical Sciences Section
Oak Ridge National Laboratory
P.O. Box 2008, Building 6012
Oak Ridge, Tennessee 37831-6367

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

and

Xiaoqing Yuan
IBM Canada Laboratory
1150 Eglinton Avenue East
North York, Ontario, Canada M3C 1H7

Submitted by Robert M. Guralnick

ABSTRACT

A recent approach for solving sparse triangular systems of equations on massively parallel computers employs a factorization of the triangular coefficient matrix to

*Written December 1992.

†E-mail: peyton@msr.epm.ornl.gov. The work of this author was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under Contract No. DE-AC05-85OR21400.

‡E-mail: apothen@narnia.uwaterloo.ca, na.pothen@na-net.ornl.gov. This author was supported by NSF grant CCR-9024954 and by U.S. Department of Energy grant DE-FG02-91ER25095 at the Pennsylvania State University and by the Canadian Natural Sciences and Engineering Research Council under grant OGP0008111 at the University of Waterloo.

 0024-3795/93/$6.00

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. 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.

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 algbebraically 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 $\alpha : V \to \{1, \ldots, |V|\}$ 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-Kees ordering for parallel sparse Cholesky factorization [6, 9].

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.

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

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 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 $u \in \text{adj}[v]$ such that $\{u\} \cup \text{adj}[u] \subset \{v\} \cup \text{adj}[v]$.[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 ith 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 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 $\lambda(\cdot) = 1$. We formalize the result suggested by this observation later in this section.

The following concepts will be used to define an interesting partition of $\text{adj}[v]$ in the case where $\lambda(v) \leqslant 2$. The *neighborhood* of a vertex v is denoted by $\text{nbd}[v] := \{v\} \cup \text{adj}[v]$. A vertex $u \in \text{adj}[v]$ is said to be *indistinguishable* from v if $\text{nbd}[u] = \text{nbd}[v]$; the set of neighbors indistinguishable from v will be denoted by $\text{adj}^0[v]$. A vertex $u \in \text{adj}[v]$ is said to *strictly outmatch* v if $\text{nbd}[u] \subset \text{nbd}[v]$. The set of vertices that strictly outmatch v will be written $\text{adj}^-[v]$; the set of vertices strictly outmatched by v will be written $\text{adj}^+[v]$. Finally, let $\text{adj}^*[v]$ consist of the vertices $u \in \text{adj}[v]$ for which $\text{nbd}[u]$ and $\text{nbd}[v]$ are incomparable. Some of these relationships in Figure 1 are: $a \in \text{adj}^-[e]$ and $e \in \text{adj}^+[a]$; $b \in \text{adj}^-[e]$ and $e \in \text{adj}^+[b]$; $e \in \text{adj}^-[f]$ and $f \in \text{adj}^+[e]$. There are no pairs of indistinguishable vertices in Figure 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

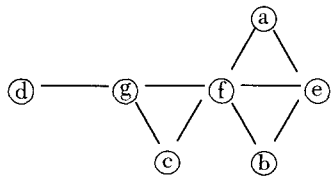

FIG. 1. Chordal graph with $\lambda(a) = \lambda(b) = \lambda(c) = \lambda(d) = 1$, $\lambda(e) = 2$, and $\lambda(f) = \lambda(g) = 3$.

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

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

introduced by Sherman [12] and in improving the time efficiency of the symbolic factorization step [3].

The reader may easily verify that the sets $\text{adj}^-[v]$, $\text{adj}^0[v]$, $\text{adj}^+[v]$, and $\text{adj}^*[v]$ form a partition of $\text{adj}[v]$. The following result shows that the vertices $v \in V$ for which $\lambda(v) \leqslant 2$ are precisely those vertices for which $\text{adj}^-[v]$, $\text{adj}^0[v]$, and $\text{adj}^+[v]$ form a partition of $\text{adj}[v]$ (i.e., $\text{adj}^*[v] = \varnothing$). Before reading the proof, the reader may find it helpful to verify the result for the graph in Figure 1.

LEMMA 2.1 (Adjacency-partition lemma). *For each vertex v of a chordal graph, the sets* $\text{adj}^-[v]$, $\text{adj}^0[v]$, *and* $\text{adj}^+[v]$ *form a partition of* $\text{adj}[v]$ *if and only if* $\lambda(v) \leqslant 2$.

Proof. We first prove the "only if" part by contraposition. Assume that $\text{adj}^-[v]$, $\text{adj}^0[v]$, and $\text{adj}^+[v]$ do *not* form a partition of $\text{adj}[v]$. It follows then that there exists a vertex $u \in \text{adj}^*[v]$, and thus we can choose $w_u \in \text{nbd}[u] - \text{nbd}[v] \neq \varnothing$, and $w_v \in \text{nbd}[v] - \text{nbd}[u] \neq \varnothing$. 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 $\lambda(v) \geqslant 3$.

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

The vertices $v \in V$ for which $\lambda(v) \leqslant 2$ 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 1.

LEMMA 2.2.

(1) *For each vertex v of a graph, $\lambda(v) = 1$ if and only if v is simplicial, in which case* $\text{adj}^-[v] = \varnothing$.

(2) *For each vertex v of a chordal graph, if $\lambda(v) = 2$, then* $|\text{adj}^-[v]| \geqslant 2$ *and for every vertex $u \in \text{adj}^-[v]$ there exists a vertex $u' \in \text{adj}^-[v]$ for which $(u, u') \notin E$.*

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

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. (Here, G is again assumed to be chordal.) It follows from the adjacency-partition lemma that u belongs to one and only one of the sets $\text{adj}^-[v]$, $\text{adj}^0[v]$, or $\text{adj}^+[v]$. Since $u' \in \text{nbd}[v] - \text{nbd}[u]$, it follows that $u \in \text{adj}^-[v]$. By the same argument, $u' \in \text{adj}^-[v]$ too, whence $|\text{adj}^-[v]| \geqslant 2$, as required. To prove the last part of the second statement, again assume that $\lambda(v) = 2$; moreover, let $u \in \text{adj}^-[v] \neq \emptyset$, so that $\text{nbd}[u] \subset \text{nbd}[v]$. Choose a vertex $u' \in \text{nbd}[v] - \text{nbd}[u] \neq \emptyset$. Clearly $u' \notin \text{adj}[u]$, whence it follows that $u' \notin \text{adj}^0[v] \cup \text{adj}^+[v]$, and thus $u' \in \text{adj}^-[v]$. This concludes the proof. ■

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

LEMMA 2.3. *The vertex set* $\text{adj}^0[v] \cup \text{adj}^+[v]$ *is complete in* G *for each* $v \in V$.

Proof. Let $v \in V$, and choose $w, w' \in \text{adj}^0[v] \cup \text{adj}^+[v]$. Since $\text{nbd}[v] \subseteq \text{nbd}[w]$, clearly $w' \in \text{adj}[w]$, whence $\text{nbd}[v]$ 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

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

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 3.1 (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 $G(X)$ will 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 $X = \mathrm{Sim}_G \neq \emptyset$, 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

$$R = \{v \in V \mid \lambda(v) \leqslant 2, \text{ and } \lambda(u) \leqslant 2 \text{ for every } u \in \mathrm{adj}^-[v]\}. \quad (1)$$

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 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 $u \in \mathrm{adj}^-[v]$, or equivalently, $u \prec v$ if $\mathrm{nbd}[u] \subset \mathrm{nbd}[v]$. 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 3.2 (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 $G(R)$

[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.

for which $\alpha(u) < \alpha(v)$ whenever $u \prec v$. From (1) and the adjacency-partition lemma, it follows that for each vertex $v \in R$ the sets $\text{adj}^-[v]$, $\text{adj}^0[v]$, and $\text{adj}^+[v]$ form a partition of $\text{adj}[v]$. Furthermore, our assumption that α is consistent with the partial order $\prec$ implies that for each vertex $v \in R$, the set $\text{madj}[v]$ includes no vertices from $\text{adj}^-[v]$, and hence contains only vertices from $\text{adj}^0[v] \cup \text{adj}^+[v]$. From Lemma 2.3 it follows that $\text{madj}[v]$ is complete in G for 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 (1) and the adjacency-partition lemma, it follows that $\text{adj}^-[v]$, $\text{adj}^0[v]$, and $\text{adj}^+[v]$ form a partition of $\text{adj}[v]$. Consequently, since $u, w \in \text{adj}[v]$ and $(u, w) \notin E$, we have $u, w \in \text{adj}^-[v]$. 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 $u \in \text{adj}^-[v]$ such that $\alpha(v) < \alpha(u)$. Now by (1) and Lemma 2.2, $\lambda(v) = 2$ and moreover there exists a vertex $w \in \text{adj}^-[v]$, $w \neq u$, 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 $u, w \in \text{madj}[v]$ 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 3.3. *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) $\lambda(v) \geqslant 3$.
(2) $\lambda(v) = 2$, but $\lambda(u) \geqslant 3$ for some vertex $u \in \text{adj}^-[v]$.

Assume first that $\lambda(v) \geqslant 3$, 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 $u \in \text{adj}^-[v]$. 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 $G(\hat{R})$ 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 2.2, there exists another vertex $w \in \text{adj}^-[v]$ 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 mem-

[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}, $\lambda_{G_i}(v)$, $\text{adj}_{G_i}[v]$, etc. to distinguish these items from the corresponding items for a different graph.

bers 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 2. 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 1, the T-partition obtained by this process has members $R_1 = \{a, b, c, d, e\}$ and $R_2 = \{f, g\}$.

It is not difficult to show that this process obtains a minimum-cardinality T-partition of V_G, and hence a solution to Problem 2. First we show that $\tau(H) \leqslant \tau(G)$ for any induced subgraph H of G, after which the main result of this section can be obtained by a simple induction argument.

LEMMA 4.1. *For any induced subgraph H of G, we have $\tau(H) \leqslant \tau(G)$.*

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 $\text{madj}_H[v] \subseteq \text{madj}_G[v]$, with $\text{madj}_G[v]$ complete in G. It follows therefore that $\text{madj}_H[v]$ 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 $\tilde{R}_i = \hat{R}_i \cap X$, $1 \leqslant i \leqslant t$. 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

$i \leftarrow 1$;
$G_1 \leftarrow G$;
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

FIG. 2. Greedy partitioning scheme for which each R_i is the maximum-cardinality T-set of G_i.

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 4.2. *The greedy partitioning scheme in Figure 2 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 $n \leqslant 2$. Let G be a graph with $n \geqslant 3$ 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 2. Clearly $\tau(G) = s \leqslant t$; thus to prove the result it suffices to show that $t \leqslant s$.

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 3.3 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 4.1 that

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

In consequence we have $t \leqslant s$ 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 2.

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 simplical vertex examined as a candidate for elimination. We will show that, within the context of our algorithm, $v \in R$ if and only if

$$\mathrm{adj}_G[v] - \hat{R} \subseteq \mathrm{adj}_G^0[v] \cup \mathrm{adj}_G^+[v]. \tag{2}$$

To enable the test in (2) to accurately distinguish members from nonmembers 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., $|\mathrm{adj}_G[v]|$). 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 $u \in \mathrm{adj}_G^-[v]$, we have $\mathrm{nbd}_G[u] \subset \mathrm{nbd}_G[v]$, 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 $u \in \mathrm{adj}_G^-[v]$ *before* examination of v, so that whenever the algorithm finally tests whether or not a vertex v satisfies (2), it will have already examined and, if called for, eliminated, every member of $\mathrm{adj}_G^-[v]$.

We have incorporated these two ideas into the algorithm shown in Figure 3. 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 $C = \mathrm{Sim}_G$. 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

```
H ← G; R̂ ← ∅; C ← Sim_G;
while C ≠ ∅ do
    Choose v ∈ C for which deg_G(v) is minimum;
    C ← C − {v};
    if adj_G[v] − R̂ ⊆ adj⁰_G[v] ∪ adj⁺_G[v] then
        H′ ← H \ {v}; R̂ ← R̂ ∪ {v};
        for w ∈ Sim_H′ − Sim_H do
            C ← C ∪ {w};
        end for
        H ← H′;
    end if
end while
```

FIG. 3. 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 $G(R)$.

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 1. Initially, $C = \mathrm{Sim}_G = \{a, b, c, d\}$. 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 2.2 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 nonsimplicial, 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 $f \in \mathrm{adj}_G^*[g]$ remains uneliminated, and e will pass the test because the vertex $f \in \mathrm{adj}_G^+[e]$ is the only neighbor of e in the reduced graph. The vertex f (degree five in 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 $g \in \mathrm{adj}_G^*[f]$ 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 *nondecreasing* 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 3 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 nondecreasing 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 $u \in \mathrm{adj}_G[v] \cap \tilde{R}$. Since $u \in \tilde{R} \subseteq R$, by (1) we have $\lambda_G(u) \leqslant 2$, and thus by the adjacency-partition lemma the sets $\mathrm{adj}_G^-[u]$, $\mathrm{adj}_G^0[u]$, and $\mathrm{adj}_G^+[u]$ form a partition of $\mathrm{adj}_G[u]$. It follows that u belongs to one of the three sets $\mathrm{adj}_G^-[v]$, $\mathrm{adj}_G^0[v]$, and $\mathrm{adj}_G^+[v]$. Now consider a vertex $w \in \mathrm{adj}_G[v] - \tilde{R}$. Since v passes the test for inclusion in $\hat{R}$, it follows that $w \in \mathrm{adj}_G^0[v] \cup \mathrm{adj}_G^+[v]$. We have therefore shown that $\mathrm{adj}_G^-[v]$, $\mathrm{adj}_G^0[v]$, and $\mathrm{adj}_G^+[v]$ form a partition of

$\text{adj}_G[v]$, whence $\lambda_G(v) \leqslant 2$ by the adjacency-partition lemma. Since $\text{adj}_G[v] - \tilde{R} \subseteq \text{adj}_G^0[v] \cup \text{adj}_G^+[v]$, we have $\text{adj}_G^-[v] \subseteq \tilde{R} \subseteq R$; hence, by (1), $\lambda_G(u) \leqslant 2$ for each vertex $u \in \text{adj}_G^-[v]$. It follows by (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 $\text{adj}_G^-[v] \subseteq \hat{R}$. Consider a vertex $u \in \text{adj}_G^-[v]$. By (1), $\lambda_G(u) \leqslant 2$; moreover, since $v \in R$ and $\text{adj}_G^-[u] \subset \text{adj}_G^-[v]$, it follows by (1) that $u \in R$. From $\text{nbd}_G[u] \subset \text{nbd}_G[v]$ 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 $\text{adj}_G^-[v] \subseteq \hat{R}$.

Let $\tilde{R}$ be the set of vertices already selected for elimination by the algorithm *immediately after the last vertex of* $\text{adj}_G^-[v]$ *has been selected for inclusion in* $\hat{R}$, so that we have $\text{adj}_G^-[v] \subseteq \tilde{R}$. It follows by applying the adjacency-partition lemma to $v \in R$ that $\text{adj}_G^-[v]$, $\text{adj}_G^0[v]$, and $\text{adj}_G^+[v]$ form a partition of $\text{adj}_G[v]$, and thus we have $\text{adj}_G[v] - \tilde{R} \subseteq \text{adj}_G^0[v] \cup \text{adj}_G^+[v]$. In consequence, v is simplicial in the reduced graph $G \backslash \tilde{R}$ (by Lemma 2.3) 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 $u \in \text{adj}_G^-[v]$, 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 $\text{adj}_G^-[v]$ 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 $\text{adj}_G^-[v]$ 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 nondecreasing, which concludes the proof. ■

6. IMPLEMENTING THE GREEDY SCHEME

Repeated application of the algorithm in Figure 3 to a chordal graph gives us an algorithm that implements the greedy partitioning scheme in Figure 2. 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 $w \in \mathrm{Sim}_{H'} - \mathrm{Sim}_H$ in Figure 3). 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 (2) would require examination of the set $\mathrm{adj}_G[w]$ for each vertex $w \in \mathrm{adj}_G[v] - \hat{R}$, 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 (2).

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-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 $1 \leqslant j \leqslant n$. 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

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

and let $\mathrm{mdeg}_G(v_j)$, the *monotone degree* of v_j, be given by

$$\mathrm{mdeg}_G(v_j) := |\mathrm{madj}_G[v_j]|.$$

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

PROPOSITION 6.1 (Liu and Mirzaian [9]). *We have* $v_j \in \mathrm{Sim}_G$ *if and only if* $\deg_G(v_j) = \mathrm{mdeg}_G(v_{f_j})$.

In order to use the simpliciality test of Proposition 6.1, the algorithm will maintain the degree values $\deg_H(v_j)$ and $\mathrm{mdeg}_H(v_j)$ in the variables $\deg(v_j)$ and $\mathrm{mdeg}(v_j)$ 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 (2) is far too expensive to lead to a linear-time implementation. The following result provides us with an efficient alternative to (2).

PROPOSITION 6.2. *Suppose the algorithm in Figure 3 is currently testing the simplicial vertex $v \in C$ for elimination, and let $\hat{R}$ now denote the subset of R containing those vertices that have been eliminated thus far. We then have (2) if and only if*

$$\left|\mathrm{nbd}_G[u] - \hat{R}\right| = \left|\mathrm{nbd}_G[v] - \hat{R}\right| \quad \text{for every} \quad u \in \mathrm{nbd}_G[v] \cap \hat{R}. \quad (3)$$

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

$$\mathrm{nbd}_G[u] - \hat{R} \subseteq \mathrm{nbd}_G[v] - \hat{R} \quad \text{for every} \quad u \in \mathrm{nbd}_G[v] \cap \hat{R}. \quad (4)$$

Assume that (3) holds. It follows then from (4) that

$$\mathrm{nbd}_G[u] - \hat{R} = \mathrm{nbd}_G[v] - \hat{R} \quad \text{for every} \quad u \in \mathrm{nbd}_G[v] \cap \hat{R}. \quad (5)$$

Choose a vertex $w \in \mathrm{adj}_G[v] - \hat{R}$. To show that (2) holds, it suffices to show that $\mathrm{nbd}_G[v] \subseteq \mathrm{nbd}_G[w]$. Let $x \in \mathrm{nbd}_G[v]$. If x belongs to the clique $\mathrm{nbd}_G[v] - \hat{R}$ from which w was taken, clearly $x \in \mathrm{nbd}_G[w]$ as required. If on the other hand $x \in \mathrm{nbd}_G[v] \cap \hat{R}$, then from (5) we have $w \in \mathrm{nbd}_G[v] - \hat{R} = \mathrm{nbd}_G[x] - \hat{R}$, whence $x \in \mathrm{nbd}_G[w]$, completing the first half of the argument.

Now assume that (2) holds, and choose a vertex $u \in \mathrm{nbd}_G[v] \cap \hat{R}$. To show that (3) holds, it suffices [by (4)] to show that

$$\mathrm{nbd}_G[v] - \hat{R} \subseteq \mathrm{nbd}_G[u] - \hat{R}.$$

Clearly, v belongs to both sets. Let $w \neq v$ belong to $\mathrm{nbd}_G[v] - \hat{R}$. It follows by (2) that $w \in \mathrm{adj}_G^0[v] \cup \mathrm{adj}_G^+[v]$. In consequence, $\mathrm{nbd}_G[v] \subseteq \mathrm{nbd}_G[w]$; hence $u \in \mathrm{nbd}_G[w]$, and thus $w \in \mathrm{nbd}_G[u] - \hat{R}$, which completes the proof. ∎

To test for (3), our algorithm must accurately maintain the variable $\deg(u) = |\mathrm{adj}_G[u] - \hat{R}|$ 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 4 (along with Figures 5, 6, and 7) implements the greedy scheme introduced in Figure 2. 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 a

Input: A chordal graph $G = (V, E)$; for each vertex $v_j \in V$, $\deg(v_j)$ [$= \deg_G(v_j)$], $\mathrm{mdeg}(v_j)$ [$= \mathrm{mdeg}_G(v_j)$], and $\mathrm{adj}_G[v_j]$, 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 7) 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 5*/
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*/
                ELIMINATE(v_j);   /*Figure 7*/
            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
    for v_j ∈ R̂_i do mark(v_j) ← 0 end for
    G_{i+1} ← G_i \ R̂_i; i ← i + 1;
end while
```

Fig. 4. Detailed implementation of scheme in Figure 2.

```
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
```

Fig. 5. Initialization procedure: initializes data structures for main **while** loop.

candidate simplicial vertex of smallest degree in G_i, the algorithm maintains a collection of sets $C[d]$ $(1 \leqslant d \leqslant n)$, where $C[d]$ contains the current candidate simplicial vertices w for which $\deg_{G_i}(w) = d$. Since vertices are both added to and removed from these sets, they should be implemented as a collection of doubly linked lists. Because no vertex appears in more than one set at a time, only three n-vectors are required: one for the first pointers into the lists, two more for the backward and forward links. We now discuss other details of the implementation.

Initialization for the algorithm is performed by the procedure INITIALIZE shown in Figure 5. This procedure initializes S_1 to Sim_G (see Proposition 6.1), each candidate set $C[d]$ to the empty set, and each marker variable $\mathrm{mark}(v_j)$ to an appropriate integer value. The various values taken on by the marker variables $\mathrm{mark}(v_j)$ during the course of the algorithm have the meanings given below:

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

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

```
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
```

Fig. 6. 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
```

Fig. 7. Elimination procedure: updates data structures to reflect the selection of v_j for elimination.

An iteration of the main **while** loop in Figure 4 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 $S_i = \mathrm{Sim}_{G_i}$ is available at the beginning of the ith 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 $\mathrm{olddeg}(v_j)$.

The second **for** loop updates $\mathrm{olddeg}(v_j)$ 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) uses the current degree information to determine if v_j satisfies the test for elimination given in (3). In Figure 6, note that

$$\deg(v_k) = \left|\mathrm{adj}_G[v_k] - \hat{R}_i\right| = \left|\mathrm{nbd}_G[v_k] - \hat{R}_i\right|$$

and

$$\deg(v_j) = \left|\mathrm{adj}_G[v_j] - \hat{R}_i\right| = \left|\mathrm{nbd}_G[v_j] - \hat{R}_i\right| - 1.$$

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 7 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 $\text{adj}_G[v_j] - \hat{R}_i$ (see Proposition 6.1) 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 $\text{adj}_G[v_j]$ 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 $\text{mdeg}(v_k)$ have been correctly updated before they are used in simpliciality tests. In Figure 7, 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 $\text{adj}_G[v_k]$ that has not yet been chosen for elimination. If $f_k = j$, where v_j is the vertex just chosen for elimination, then $\text{adj}_G[v_k]$ 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, that the algorithm in Figure 4 correctly implements the greedy scheme in Figure 2 follows immediately from the fact that each iteration of the main **while** loop implements the algorithm in Figure 3.

6.4. Complexity Analysis

In this section we verify that the algorithm in Figure 4 runs in time proportional to $|V| + |E|$. Recall that the algorithm in Figure 4 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 $v_j \in \text{adj}_{G_i}[v_k]$ 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 4 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 2 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.

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.

REFERENCES

1 F. L. Alvarado, A. Pothen, and R. S. Schreiber, Highly parallel sparse triangular solution, in *Graph Theory and Sparse Matrix Computation* (J. A. George, J. R. Gilbert, and J. W. H. Liu, Eds.), Springer-Verlag, vol. 56, pp. 141–158 (1993).

2 F. L. Alvarado and R. S. Schreiber, Optimal parallel solution of sparse triangular systems, *SIAM J. Sci. Comput.*, 14:446–460 (1993).

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

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

5 N. J. Higham and A. Pothen, The stability of the partitioned inverse approach to parallel sparse triangular solution, Tech. Report CS-92-52, Computer Science, Univ. of Waterloo, Oct. 1992: *SIAM J. Sci. Comput.*, to appear.

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

7 J. W. H. Liu, Reordering sparse matrices for parallel elimination, *Parallel Comput.* 11:73–91 (1989).

8 J. W. H. Liu, The role of elimination trees in sparse factorization, *SIAM J. Matrix 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. Discrete 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, Tech. Report CS-93-27, Computer Science, Univ. Waterloo, July 1993; submitted.

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

12 A. H. Sherman, On the Efficient Solution of Sparse Systems of Linear and Nonlinear Equations, Ph.D. Thesis, Yale Univ., 1975.

13 D. R. Shier, Some aspects of perfect elimination orderings in chordal graphs, *Discrete 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).

Received 18 December 1992; final manuscript accepted 11 May 1993