

## COMPLEXITY OF FINDING EMBEDDINGS IN A $k$ -TREE\*

STEFAN ARNBORG†, DEREK G. CORNEIL‡ AND ANDRZEJ PROSKUROWSKI§

**Abstract.** A  $k$ -tree is a graph that can be reduced to the  $k$ -complete graph by a sequence of removals of a degree  $k$  vertex with completely connected neighbors. We address the problem of determining whether a graph is a partial graph of a  $k$ -tree. This problem is motivated by the existence of polynomial time algorithms for many combinatorial problems on graphs when the graph is constrained to be a partial  $k$ -tree for fixed  $k$ . These algorithms have practical applications in areas such as reliability, concurrent broadcasting and evaluation of queries in a relational database system. We determine the complexity status of two problems related to finding the smallest number  $k$  such that a given graph is a partial  $k$ -tree. First, the corresponding decision problem is NP-complete. Second, for a fixed (predetermined) value of  $k$ , we present an algorithm with polynomially bounded (but exponential in  $k$ ) worst case time complexity. Previously, this problem had only been solved for  $k = 1, 2, 3$ .

**Key words.** graph theory,  $k$ -trees, algorithm complexity, NP-complete

**AMS(MOS) subject classifications.** 05C10, 68C25, 90C39

**1. Motivation.** The class of  $k$ -trees is defined recursively as follows (see, for instance, Rose [15]). The complete graph with  $k$  vertices is a  $k$ -tree. A  $k$ -tree with  $n + 1$  vertices ( $n \geq k$ ) can be constructed from a  $k$ -tree with  $n$  vertices by adding a vertex adjacent to all vertices of one of its  $k$ -vertex complete subgraphs, and only to these vertices. In a given construction of a  $k$ -tree, the original  $k$ -complete subgraph is its *basis*. Any  $k$ -complete subgraph of a  $k$ -tree can be its basis (Proskurowski [13, Prop. 1.3]). A *partial  $k$ -tree* is a subgraph of a  $k$ -tree.

Our interest in the class of  $k$ -trees and their subgraphs is motivated by some practical questions about reliability of communication networks in the presence of constrained line- and site-failures (Farley [8], Farley and Proskurowski [9], Neufeldt and Colbourn [12], Wald and Colbourn [16]), concurrent broadcasting in a common medium network (Colbourn and Proskurowski [6]), reliability evaluation in complex systems (Arnborg [1]), and evaluation of queries in relational data base systems; for a survey see Arnborg [2]. For these problems restricted to partial  $k$ -trees, there exist efficient solution algorithms which exploit the following *separation property* of  $k$ -trees (Rose [15]): every minimal separator of a  $k$ -tree consists of  $k$  completely connected vertices. This property, together with the requirement that a graph is connected and does not contain a set of  $k + 2$  completely connected vertices, is a definitional property of  $k$ -trees (Rose [15, Thm. 1.1]).

Partial  $k$ -trees have a similar *bounded decomposability property* (cf. Arnborg and Proskurowski [3]): a sufficiently large partial  $k$ -tree can be disconnected by removal of at most  $k$  (separator) vertices so that each of the resulting connected components augmented by the completely connected separator vertices is a partial  $k$ -tree. Since the component partial  $k$ -trees of a partial  $k$ -tree interact only through the minimal separators of

\* Received by the editors May 31, 1984; accepted for publication (in revised form) October 7, 1986.

† Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, S-100 44 Stockholm, Sweden. The work of this author was supported in part by a grant from the Swedish Board of Technical Development, STU contract 83-3719.

‡ Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4. The work of this author was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Canada-France Science Exchange Program.

§ Computer and Information Science Department, University of Oregon, Eugene, Oregon 97403. The work of this author was supported in part by the U.S.-Sweden Scientific Cooperation Program of the National Science Foundation through grant INT-8318441.

the  $k$ -tree, solutions to subproblems on the component partial  $k$ -trees may be combined to form a solution for the given partial  $k$ -tree. Using a succinct representation of a bounded number of optimal solutions to such subproblems (cf. Corneil and Keil [7]), Arnborg and Proskurowski [4] develop a general algorithm paradigm to solve efficiently many difficult problems on graphs with this bounded decomposability property. The algorithms presented there solve NP-hard optimization problems (like vertex cover, chromatic number, graph reliability) for partial  $k$ -trees given with a suitable embedding  $k$ -tree. However, the presentation skirts two major problems: one is finding an embedding, a  $k$ -tree of which the given graph is a subgraph; the other is finding the minimal value of such  $k$ , which is important since the algorithms, though linear in the size of the input, are exponential or even superexponential in  $k$ . For small values of  $k$ , recognition and embedding problems for partial  $k$ -trees have been solved by reducing a given graph according to a complete set of safe reduction rules, i.e., application of the reduction rules (in any order) reduces a graph to the empty graph iff it is a partial  $k$ -tree (see Wald and Colbourn [16] for  $k = 2$ , and Arnborg and Proskurowski [3] for  $k = 3$ ). In this paper, we first address the question of determining the minimum value of  $k$  for which a given graph is a partial  $k$ -tree. We show that the decision version of this PARTIAL  $K$ -TREE problem is NP-complete. We then follow a "brute-force" approach to finding a  $k$ -tree embedding of the given graph through examination of all its  $k$ -vertex separators. This yields a polynomial time algorithm, assuming a fixed value of  $k$ . Not surprisingly, this new algorithm is inferior to the reduction rules algorithms for  $k = 2$  and  $k = 3$ .

**2. Definitions.** A graph  $G$  with vertex set  $V$  and edge set  $E$  will be denoted  $G(V, E)$ . The cardinality of the vertex set will be called the *size* of  $G$ . A *partial graph* of  $G$  contains all its vertices and a subset of its edges, whereas a *subgraph* of  $G$  has a subset of both edges and vertices of  $G$ . A *supergraph* of  $G$  is any graph of which  $G$  is a partial graph. For general graph theoretical concepts, the reader is advised to consult a standard text, e.g., Bondy and Murty [5].

A *clique* in a graph  $G(V, E)$  is a maximal complete subgraph of  $G$ . The *clique number*  $\omega(G)$  is the size of a largest clique in  $G$ . For any vertex  $v \in V$ , the (*open*) *neighborhood* of  $v$  is defined as the set of all vertices adjacent to  $v$ ,  $\Gamma(v) = \{u : (u, v) \in E\}$ . The *closed neighborhood* of  $v$  contains also the vertex  $v$ . The *degree* of  $v$  is the size of its neighborhood,  $\deg(v) = |\Gamma(v)|$ , and  $\Delta(G) = \max_{v \in V} \deg(v)$ . A vertex  $v$  is *simplicial* if the subgraph of  $G$  induced by  $\Gamma(v)$  is complete. A graph  $G$  is  *$k$ -decomposable* iff either  $G$  has  $k + 1$  or fewer vertices or there is a subgraph  $S$  of  $G$  with at most  $k$  vertices such that  $G - S$  is disconnected, and each of the connected components of  $G - S$  augmented by  $S$  with completely connected vertices is  $k$ -decomposable. A graph is *chordal* (or *triangulated*) if every cycle of length greater than three has a chord. Clearly,  $k$ -trees are examples of chordal graphs. An *elimination scheme* of a graph is an ordering  $\pi$  of its vertices. The *filled graph* of  $G(V, E)$  w.r.t.  $\pi$  is the graph  $G(V, E \cup F^\pi)$ , where  $F^\pi$  are the fill edges. An edge  $(u, w)$  is a *fill edge* if there is a vertex  $v$  preceding  $u$  and  $w$  in  $\pi$  such that both  $u$  and  $w$  are adjacent to  $v$  via original or fill edges but not to each other. The complete set of fill edges is easily obtained by examining vertices in order  $\pi$ . A graph  $G$  has a *perfect elimination scheme*, i.e., an elimination scheme with no fill edges (cf. Rose [14]), if there exists an order of eliminating the vertices of  $G$  such that each vertex is simplicial at the time of elimination. It is well known (Rose [14]) that a graph is chordal iff it has a perfect elimination scheme, and that every edge-minimal chordal supergraph of a graph  $G$  is the filled graph of  $G$  w.r.t. some elimination scheme. Given  $A$ , a complete subgraph of graph  $G$ , we say that  $G$  is an  *$A$ -chordal path* if there exists a perfect elimination scheme  $\pi$  such that if  $u$  immediately follows  $v$  in  $\pi$  then  $u$  and  $v$  are adjacent, and the vertices

in  $A$  are last in  $\pi$ . A chordal graph  $G$  is a *chordal path* iff there is an  $A$  for which  $G$  is an  $A$ -chordal path.

A  $k$ -chordal graph is a chordal graph  $G$  for which  $\omega(G) = k + 1$ . Thus, in every perfect elimination scheme of a  $k$ -chordal graph, the neighborhood of every vertex, when eliminated, induces  $K_i$ , a complete graph with  $i$  vertices,  $i \leq k$ . We notice that a  $k$ -tree with more than  $k$  vertices is a  $k$ -chordal graph. Since the neighborhood of a simplicial vertex  $u$  in a  $k$ -chordal graph is a completely connected set of at most  $k$  vertices that separate  $u$  from the rest of the graph, any  $k$ -chordal graph is  $k$ -decomposable. Given a graph  $G$ , we define  $k_c(G)$  to be the minimum  $k$  such that  $G$  is a partial  $k$ -tree. Similarly,  $k_{c'}(G)$  is defined to be the minimum  $k$  such that  $G$  is a partial  $k$ -chordal graph. Not surprisingly, we have the following lemma relating  $k_c(G)$  and  $k_{c'}(G)$  for any graph  $G$ .

LEMMA 2.1. For any graph  $G$  that is not a complete graph,  $k_c(G) = k_{c'}(G)$ .

Proof. From the definitions, we see that  $k_{c'}(G) \leq k_c(G)$ . To show that  $k_c(G) \leq k_{c'}(G)$ , we let  $G'$  be a  $k_{c'}(G)$ -chordal supergraph of  $G$ . Since  $G'$  is  $k_{c'}(G)$ -decomposable, it follows from Arnborg and Proskurowski [3, Thm. 2.7] that  $G'$  is also a partial  $k_{c'}(G)$ -tree, and so is  $G$ .  $\square$

A *block* of a graph  $G$  is a maximal set of vertices with the same closed neighborhood. Clearly, the blocks of  $G$  partition  $V$ . A *block-contiguous* elimination scheme is one in which the vertices of each block are eliminated contiguously.

Yannakakis [17] introduced the notion of chain graph: a bipartite graph  $G(A \cup B, E)$  is a *chain graph* if the neighbors of the nodes in  $A$  form a chain, i.e., there exists a bijection  $\tau : A \leftrightarrow \{1, 2, \dots, |A|\}$  such that  $\tau(u) < \tau(v)$  iff  $\Gamma(u) \supseteq \Gamma(v)$ . Such a permutation  $\tau$  is called a *chain order*. The neighbors of the nodes of  $B$  also form a chain, and thus the definition is unambiguous. Given a bipartite graph  $G(A \cup B, E)$  and an ordering  $\tau$  of  $A$ , a  $(G, \tau)$ -*chain graph* is any bipartite graph  $G'(A \cup B, E \cup E')$  for which  $\tau$  is a chain graph order. For a given bipartite graph  $G(A \cup B, E)$ , the graph  $C(G)$  is formed from  $G$  by adding edges to form complete subgraphs on  $A$  and  $B$ . The following lemma relates chain graphs and chordal graphs.

LEMMA 2.2 (Yannakakis [17, Lemma 2.1]). A bipartite graph  $G(A \cup B, E)$  is a chain graph iff  $C(G)$  is chordal.  $\square$

In fact, we can strengthen this statement by a more detailed characterization of the chordal graph  $C(G)$ .

COROLLARY 2.3. A bipartite graph  $G(A \cup B, E)$  is a chain graph iff  $C(G)$  is an  $A$ -chordal path.  $\square$

If  $V$  is the vertex set of a graph  $G$  and  $\tau$  is a permutation of  $V$ , then we define the *linear cut value* of  $G$  w.r.t.  $\tau$  as

$$c_\tau(G) = \max_{1 \leq i < |V|} |\{(u, v) \in E : \tau(u) \leq i < \tau(v)\}|.$$

The MINIMUM CUT LINEAR ARRANGEMENT problem (MCLA) is defined as follows: Given a graph  $G(V, E)$  and a positive integer  $k$ , does there exist a permutation  $\tau$  of  $V$  such that  $c_\tau(G) \leq k$ ? MCLA is NP-complete (see, for instance, Garey and Johnson [10]). In the next section, we use this fact to show the NP-completeness of the following PARTIAL K-TREE recognition problem: Given a graph  $G$  and an integer  $k$ , is  $k_c(G) \leq k$ ?

3. NP-completeness of PARTIAL K-TREE. In this section we will use the concepts of chain graph and chordal path to prove that the PARTIAL K-TREE problem is at least as difficult as the MCLA problem, in the standard sense of polynomial reducibility.

Given an arbitrary graph  $G(V, E)$ , we will construct a bipartite graph  $G'(A \cup B, E')$  in the following way: Each vertex  $x \in V$  is represented by  $\Delta(G) + 1$  vertices in  $A$  and

$\Delta(G) - \deg(x) + 1$  vertices in  $B$ . We let  $A_x$  (resp.  $B_x$ ) denote the set of vertices in  $A$  (resp.  $B$ ) which represents  $x$ . Each edge  $e \in E$  is represented by two vertices in  $B$ ; this set of vertices is denoted  $B_e$ . Edges in  $E'$  are of the following two types: (i) all vertices in  $A_x$  are adjacent to all vertices in  $B_x$ ; (ii) all vertices in  $A_x$  are adjacent to both vertices in  $B_e$  if  $x$  is an endpoint of  $e$ . As an example of this construction, see the graph in Fig. 1. We note that the vertex sets  $A_a, B_a$  and  $B_e$  form the blocks of  $C(G)$ .

Before proving the main result of this section, we relate block-contiguous elimination schemes of a given graph  $G$  to chordal supergraphs of  $G$ .

LEMMA 3.1. *Let  $H$  be a minimal chordal supergraph of  $G$ . Then there exists a block-contiguous elimination order  $\pi$  such that  $H$  is the filled graph of  $G$  w.r.t.  $\pi$ .*

*Proof.* As stated in § 2,  $H$  is the filled graph of  $G$  w.r.t. some elimination scheme. A vertex is simplicial iff all other vertices in its block are also simplicial. Since the elimination of any vertex preserves this property, any chordal graph has a block-contiguous perfect elimination scheme. Given such a scheme any ordering of the vertices in a block determines a block-contiguous perfect elimination scheme. If  $u$  and  $v$  belong to the same block of  $G$ , then the addition of fill edge  $(v, w)$  implies the addition of fill edge  $(u, w)$ . Thus the blocks of  $G$  form a refinement (possibly trivial) of the blocks of  $H$ . We now set  $\pi$  to be a block-contiguous perfect elimination scheme for  $H$  which is also block-contiguous for  $G$ .  $\square$

We now establish the relationship between the linear cut value of a graph  $G$  and values of  $k'$  such that  $C(G)$  is a partial  $k'$ -tree.

LEMMA 3.2. *Given a graph  $G$  and a positive integer  $k$ ,  $G$  has a minimum linear cut value  $k$  w.r.t. some permutation  $\pi$  iff the corresponding graph  $C(G)$  is a partial  $k'$ -tree for  $k' = (\Delta(G) + 1)(|V| + 1) + k - 1$ .*

*Proof.* Since a  $k'$ -tree is a  $k'$ -chordal graph, it follows from Lemma 3.1 that if  $C(G)$  is a partial  $k'$ -tree then there exists a block-contiguous elimination scheme  $\pi'$  such that no vertex has degree greater than  $k'$  when it is eliminated. Let  $F$  be the filled graph of  $C(G)$  w.r.t. this permutation  $\pi'$ .  $F$  is also a supergraph of the filled graph of  $C(G)$  w.r.t.

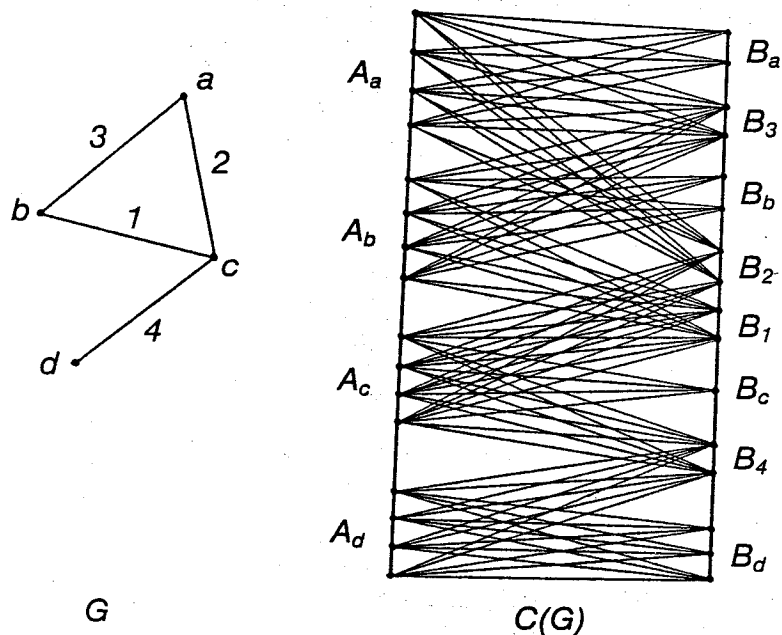
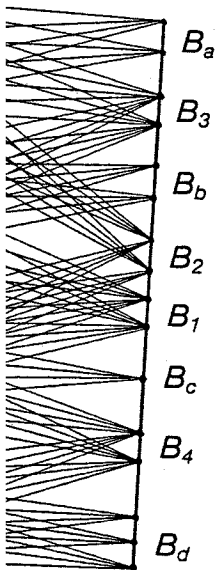


FIG. 1. Example of the bipartite graph construction.

the set of vertices in  $A$  (resp. two vertices in  $B$ ; this set of types: (i) all vertices in  $A_x$  are adjacent to both vertices in  $B_e$  if  $x$  see the graph in Fig. 1. We note  $G$ ).  
 ate block-contiguous elimination  
 h of  $G$ . Then there exists a block-graph of  $G$  w.r.t.  $\pi$ .  
 v.r.t. some elimination scheme also simplicial. Since the elimination graph has a block-contiguous ordering of the vertices in a block. If  $u$  and  $v$  belong to the same the addition of fill edge  $(u, w)$  of the blocks of  $H$ . We now set  $F$  which is also block-contiguous  
 ar cut value of a graph  $G$  and  
 $k$ ,  $G$  has a minimum linear cut  
 ing graph  $C(G)$  is a partial  
 from Lemma 3.1 that if  $C(G)$   
 mination scheme  $\pi'$  such that  
 l. Let  $F$  be the filled graph of  
 the filled graph of  $C(G)$  w.r.t.



any perfect elimination order of  $F$ . But since  $F$  is chordal, its edges between  $A$  and  $B$  form a chain graph by Lemma 2.2, and it is easy to see that  $F$  has a perfect elimination ordering starting with all vertices in  $A$  in reverse chain ordering, which is also contiguous in the blocks  $A_v$ . Without loss of generality we can assume that  $\pi'$  is such an ordering. Let  $\pi$  be the ordering of blocks in  $A$  induced by  $\pi'$ . Assume without loss of generality that the vertices of  $G$  are numbered in order  $\pi$  and are identified by their numbers. Consider the graph resulting from elimination of vertices in the first  $i - 1$  blocks of  $C(G)$ . In this graph, each vertex of  $A_i$  is adjacent to: the other  $\Delta(G)$  vertices in  $A_i$ ; the  $\Delta(G) + 1$  vertices in each of  $A_{i+1}, \dots, A_{|V|}$ ; the  $\Delta(G) + 1 - \deg_G(j)$  vertices in  $B_j$  for  $j = 1, \dots, i$  (these are fill edges except for  $j = i$ ); and the two vertices in  $B_e$  for each edge  $e$  incident to at least one vertex in  $\{1, \dots, i\}$  (these are fill edges for  $e$  not incident to  $i$ ). These adjacencies sum up to

$$\Delta(G) + (\Delta(G) + 1)(|V| - i) + (\Delta(G) + 1) \times i - \sum_{j=1}^i \deg_G(j) + 2|E_1^i| + 2|E_2^i|,$$

where  $E_1^i$  is the set of edges with exactly one vertex in  $\{1, \dots, i\}$ , and  $E_2^i$  the set with both vertices in  $\{1, \dots, i\}$ . Obviously,  $\sum_{j=1}^i \deg_G(j) = 2|E_2^i| + |E_1^i|$ , so the degree of a vertex in  $A_i$  simplifies to  $(\Delta(G) + 1)(|V| + 1) - 1 + |E_1^i|$ . Since  $E_1^i$  is the set of edges between vertices in  $\{1, \dots, i\}$  and vertices in  $\{i + 1, \dots, |V|\}$ , in this particular ordering  $\pi$ , the maximum size of  $E_1^i$  over all  $i$  is the linear cut value of  $G$  w.r.t.  $\pi$ . This value also determines the maximum size of a clique in  $C(G)$ . We have thus shown that the  $k'$ -chordality implies the existence of a linear arrangement with the cut value  $k$ . Conversely, the existence of an ordering  $\pi$  w.r.t. which  $G$  has a linear cut value  $k$  implies that the largest clique in  $F$ , the filled graph of  $C(G)$  w.r.t.  $\pi'$ , has size  $k' + 1$  (by examination of an induced ordering  $\pi'$  of its vertices.) This completes the proof.  $\square$

**THEOREM 3.3.** *The PARTIAL K-TREE problem is NP-complete.*

*Proof.* (Hardness for NP): This follows from Lemma 3.2 and the fact that  $C(G)$  can be constructed from  $G$  in polynomial time. (Membership in NP): For a suitable (nondeterministic) choice of vertex order, the elimination process is easily turned into a polynomial time verification that a graph is a partial  $k$ -tree.  $\square$

Let us define a  $k$ -chordal path to be a  $k$ -chordal graph which is also a chordal path. A  $k$ -interval graph is an interval graph derived from a set of intervals, no  $k + 2$  of which have a nonempty intersection (i.e., it has clique number  $k + 1$  or less). We now have:

**COROLLARY 3.4.** *The following problems are NP-complete:*

- (i) *Given graph  $G$  and integer  $k$ , is  $G$  a partial  $k$ -chordal path?*
- (ii) *Given graph  $G$  and integer  $k$ , is  $G$  a partial  $k$ -interval graph?*

*Proof.* Statement (i) follows from Theorem 3.3, Lemma 2.1 and Corollary 2.3. A result of Gilmore and Hoffman [11] says that graph  $G$  is an interval graph iff the cliques of  $G$  can be numbered  $C_1, C_2, \dots, C_m$  such that for each node  $x, x \in C_i \cap C_j, (i < j)$  implies that  $x \in C_l$  for all  $l$  such that  $i < l < j$ . So the class of  $k$ -interval graphs is contained in the class of  $k$ -chordal graphs but contains the class of  $k$ -chordal paths, from which (ii) follows.  $\square$

**4. Recognition of partial  $k$ -trees for a fixed value of  $k$ .** In the preceding section, we have shown that the partial  $k$ -tree recognition problem is NP-complete if  $k$  is part of the problem's instance. Since the proof of our NP-completeness result (Theorem 3.3) builds on the value of  $k$  growing polynomially with the size of the graph, one could expect the complexity of partial  $k$ -tree recognition for fixed  $k$  to grow quickly with  $k$ . However, when the value of  $k$  is fixed (i.e., all instances of the problem refer to the same  $k$  value), the complexity status of the problem changes, as any dependence on  $k$  is considered

constant. The recognition problems for partial 2- and 3-trees have been solved previously (cf. Wald and Colbourn [16] and Arnborg and Proskurowski [3]) by exhibiting complete sets of safe reduction rules, reducing to the empty graph precisely those graphs in the relevant class.

Another approach proves successful for general, fixed values of  $k$ . This new approach uses a dynamic programming technique in evaluating feasibility of proposed partial embeddings of subgraphs of the given graph in a  $k$ -tree. Although there might be many such embeddings, the set of all possible minimal separators in all embeddings has cardinality bounded by a polynomial in the size of the graph (the number of vertices), due to the fact that all such separators have size  $k$ . Our algorithm considers the connected components into which  $k$ -element vertex sets separate the graph and decides their embeddability in the order of their increasing sizes. Thus, successful embedding attempts (which assume completely connected minimal separators) can be subsequently used to embed a union of such connected components.

ALGORITHM 4.1 for the recognition of partial  $k$ -trees.

INPUT: A graph  $G$ , with  $n$  vertices.

OUTPUT: YES or NO.

DATA STRUCTURE: Family of  $k$ -element vertex sets which are separators of  $G$ . For each such set  $S$ , there is a set of  $l$  connected components of  $G$  into which  $G$  is separated by removal of  $S$ . Denoting  $S$  by  $C_i$ , we denote by  $C_i^j$ ,  $1 \leq j \leq l$  the subgraphs of  $G$ , each induced by  $S$  and the vertices of the corresponding connected component, with the addition of edges required to make the subgraph induced by  $S$  complete. Each such  $C_i^j$  has an answer YES or NO (whether it is embeddable in a  $k$ -tree or not) determined during the computation.

METHOD: {find the graphs  $C_i$  and  $C_i^j$ }  
 for each set  $S$  of  $k$  vertices in  $G$  do  
   if  $S$  is a separator of  $G$   
     then insert  $C_i = S$  and the corresponding graphs  $C_i^j$  into the data structure  
 end-do  
 sort all graphs  $C_i^j$  by increasing size  
 {examine graphs  $C_i^j$  from smallest to largest and determine whether the graph is a partial  $k$ -tree}  
 A graph  $C_i^j$  of size  $k + 1$  is a partial  $k$ -tree: set its answer to YES.  
 for each graph  $C_i^j$  an increasing order of size  $h$  do  
   for each  $v \in C_i^j$  do  
     examine all  $k$ -vertex separators  $C_m$  contained in  $C_i \cup \{v\}$ ;  
     consider all  $C_m^l$  in  $(C_i^j - C_i) \cup C_m$  which are partial  $k$ -trees.  
     if their union, over all  $l$ 's and all  $m$ 's, contains  $C_i^j - C_i$   
       then set the answer for  $C_i^j$  to YES and exit-do.  
   end-do  
   if no answer was set for  $C_i^j$   
     then set the answer for  $C_i^j$  to NO.  
   if  $G$  has a separator  $C_m$  such that all  $C_m^l$  graphs have answer YES  
     then  $G$  is a partial  $k$ -tree: return (YES).  
   if each separator  $C_m$  of  $G$  has a  $C_m^l$  with answer NO  
     then  $G$  is not a partial  $k$ -tree: return (NO).  
 end-do  
 {end of the algorithm}

have been solved previously (see [3]) by exhibiting completely those graphs in the

values of  $k$ . This new approach; feasibility of proposed partial embeddings has characteristics (the number of vertices), due to the algorithm considers the connected components of the graph and decides their embedding. Successful embedding attempts can be subsequently used to

which are separators of  $G$ . Connected components of  $G$  induced by  $S$  and  $\bar{S}$ , each induced by  $S$  and  $\bar{S}$ , component, with the addition induced by  $S$  complete. Whether it is embeddable computation.

graphs  $C_i^j$  into the data

and determine whether

et its answer to YES.  
: h do

tained in  $C_i \cup \{v\}$ ;  
h are partial  $k$ -trees.  
ontains  $C_i^j - C_i$   
d exit-do.

aphs have answer YES

answer NO  
)).

In the algorithm above we use  $C_i^j$  to denote both a graph and its vertex set, depending on context. This slightly inaccurate usage will continue in the verification below. The worst case time complexity of the algorithm is fairly straightforwardly bounded by a polynomial in  $n$ , the size of  $G$ , since all the operations (searches and checks) can be performed efficiently and there is a limited (polynomially bounded) number of them.

**THEOREM 4.2.** *The execution time of the partial  $k$ -tree recognition algorithm using suitably chosen data structures is of order  $\mathcal{O}(n^{k+2})$ .*

*Proof.* The algorithm examines at most all  $k$ -element vertex sets; there are  $\mathcal{O}(n^k)$  of those, and it takes  $\mathcal{O}(n^2)$  time to check if one is a separator of  $G$ . To be able to access the subgraphs  $C_i^j$  in the increasing order of their sizes, they should be bucket-sorted, in time proportional to the number of them, at most  $\mathcal{O}(n^{k+1})$ . The exit conditions for the algorithm can be checked in constant time per examined subgraph, by incrementally maintaining counts of partial  $k$ -tree components for each separator, and of incorrectly guessed separators for the whole graph. There are less than  $n$  vertices in a subgraph  $C_i^j$ , and the access to a 'related' separator  $C_m$  (in the innermost loop) can be made in constant time. Checking the union of the relevant partial  $k$ -tree components is again of order of the size of  $C_i^j$ , and thus, the overall time complexity is  $\mathcal{O}(n^{k+2})$ .  $\square$

To prove the correctness of our algorithm we state and prove two lemmas. The first one reflects the fact that partial  $k$ -trees are  $k$ -decomposable (cf. Arnborg and Proskurowski [3, Thm. 2.7]).

**LEMMA 4.3.** *A given graph  $G$  of size at least  $k + 2$  is a partial  $k$ -tree if and only if there exists a  $k$ -vertex separator  $C_i$  such that all subgraphs  $C_i^j$  (as defined in the algorithm) are partial  $k$ -trees.*

*Proof.* (By induction on the size  $n$  of  $G$ ). Obviously true for  $n = k + 2$ , since of graphs with  $k + 2$  vertices only  $K_{k+2}$  is not a partial  $k$ -tree. Assuming the hypothesis true for all smaller graphs, consider a graph  $G$  of size  $n \geq k + 3$ . If  $G$  is a partial  $k$ -tree, then it has a vertex  $v$  which in some  $k$ -tree embedding has a completely connected neighborhood  $S$ . The graph  $G_1 = G - \{v\} \cup S$  is also a partial  $k$ -tree with a postulated separator  $C_i$  (by the inductive assumption). If this  $C_i$  is identical with  $S$ , then it fulfills the requirements for  $G$  since  $S \cup \{v\}$  (the new  $C_i^j$  containing  $v$ ) is a partial  $k$ -tree. Otherwise, the  $C_i^j$  (of the embedding of  $G_1$ ) that contains  $S$  can be extended by  $v$  to a partial  $k$ -tree. Hence, the necessity is proved. The sufficiency follows immediately from the constructive definition of  $k$ -trees:  $G$  can be constructed using  $C_i$  as a base, and independently attaching an embedding for each  $C_i^j$ .  $\square$

The second lemma addresses the operation of the algorithm when computing the partial answers. Note that  $C_i^j$  has a complete subgraph induced by  $C_i$ .

**LEMMA 4.4.** *A graph  $C_i^j$ , as defined in the algorithm, is a partial  $k$ -tree iff there exists a vertex  $v$  in  $C_i^j$  and a set  $F$  of  $k$ -vertex separators  $C_m \neq C_i$  contained in  $C_i \cup \{v\}$  such that graphs  $C_m^l - C_m$  (for all  $l$  such that  $C_m^l \subset C_i^j$  is a partial  $k$ -tree) partition  $C_i^j - C_i - \{v\}$ .*

*Proof.* We recall that a  $k$ -tree can be constructed with any  $k$ -complete subgraph as its base. If  $C_i^j$  is a partial  $k$ -tree, then any  $k$ -tree  $T$  embedding it can be constructed from  $C_i$  by first adding a vertex,  $v$ , adjacent to all vertices of  $C_i$ , and then constructing the remainder of  $T$  as  $k$ -trees  $T_m^l$  based on some  $k$ -complete subgraphs of  $C_i \cup \{v\}$ . The  $k$ -trees  $T_m^l$  overlap only on  $C_i \cup \{v\}$ , and thus their subgraphs  $C_m^l$  partition  $C_i^j - C_i - \{v\}$ . This proves the necessity. If such a family  $F$  of separators exists, then a  $k$ -tree  $T$  embedding  $C_i^j$  can be constructed from  $C_i$  by first adding  $v$  adjacent to all the vertices of  $C_i$  and then building up the remainder of  $T$  as the union of embeddings of the partial  $k$ -trees  $C_m^l$ , each constructed with  $C_m$  as its base.  $\square$

**THEOREM 4.5.** *The algorithm correctly determines whether a given graph  $G$  is a partial  $k$ -tree.*

*Proof.* The termination criteria of the algorithm's main loop correspond to the view of partial  $k$ -trees as  $k$ -decomposable graphs. By Lemma 4.4, every subgraph  $C_i^j$  of size at most  $h$  is correctly classified. If  $G$  is a partial  $k$ -tree then the final decision (return YES) is reached when the size of the subgraph  $C_i^j$  increases to at most  $(n+k)/2$  (this corresponds to  $G$  having only  $k$ -path embeddings).  $\square$

The algorithm discussed above answers the embeddability decision problem, but does not produce an embedding when one exists. It is obvious that this can be achieved by storing an embedding for every  $C_i^j$  if and when it has been classified as a partial  $k$ -tree.

*Note added in proof.* In a recent paper (*Graph minors XIII: The disjoint path problem*, manuscript, September 1986), Robertson and Seymour show—nonconstructively—the existence of an  $\mathcal{O}(n^2)$  algorithm for recognizing partial  $k$ -trees. Such an algorithm would require the knowledge of the set of all minimal forbidden minors for the class of partial  $k$ -trees.

**Acknowledgments.** Derek G. Corneil wishes to express his appreciation of the hospitality of IMAG at the Université de Grenoble. Andrzej Proskurowski gratefully acknowledges the hospitality of the Royal Institute of Technology. The referees have provided helpful criticism, and in particular suggested an improved presentation of the results in § 3.

#### REFERENCES

- [1] S. ARNBORG, *Reduced state enumeration—Another algorithm for reliability evaluation*, IEEE Trans. Reliability, R-27 (1978), pp. 101–105.
- [2] ———, *Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey*, BIT, 25 (1985), pp. 2–33.
- [3] S. ARNBORG AND A. PROSKUROWSKI, *Characterization and recognition of partial 3-trees*, this Journal, 7 (1986), pp. 305–314.
- [4] ———, *Linear time algorithms for NP-hard problems on graphs embedded in  $k$ -trees*, TRITA-NA-8404, The Royal Institute of Technology, 1984.
- [5] J. A. BONDY AND U. S. R. MURTY, *Graph Theory with Applications*, North-Holland, Amsterdam, 1976.
- [6] C. J. COLBOURN AND A. PROSKUROWSKI, *Concurrent transmissions in broadcast networks*, Proc. Int. Conf. Automata, Languages, Programming, Springer-Verlag, Berlin-Heidelberg-New York, Lecture Notes in Computer Science, 172 (1984), pp. 128–136.
- [7] D. G. CORNEIL AND J. M. KEIL, *A dynamic programming approach to the dominating set problem on  $k$ -trees*, this Journal, 8 (1987), to appear.
- [8] A. M. FARLEY, *Networks immune to isolated failures*, Networks, 11 (1981), pp. 255–268.
- [9] A. M. FARLEY AND A. PROSKUROWSKI, *Networks immune to isolated line failures*, Networks, 12 (1982), pp. 393–403.
- [10] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability*, W. H. Freeman, San Francisco, CA, 1979.
- [11] P. C. GILMORE AND A. J. HOFFMAN, *A characterization of comparability graphs and of interval graphs*, Canad. J. Math., 16 (1964), pp. 539–548.
- [12] E. M. NEUFELDT AND C. J. COLBOURN, *The most reliable series-parallel networks*, Dept. of Computing Science, University of Saskatchewan, TR 83-7, 1983.
- [13] A. PROSKUROWSKI, *Separating subgraphs in  $k$ -trees: Cables and caterpillars*, Discrete Math., 49 (1984), pp. 275–285.
- [14] D. J. ROSE, *Triangulated graphs and the elimination process*, J. Math. Anal. Appl., 32 (1970), pp. 597–609.
- [15] ———, *On simple characterization of  $k$ -trees*, Discrete Math., 7 (1974), pp. 317–322.
- [16] A. WALD AND C. J. COLBOURN, *Steiner trees, partial 2-trees, and minimum IFI networks*, Networks, 13 (1983), pp. 159–167.
- [17] M. YANNAKAKIS, *Computing the minimum fill-in is NP-complete*, this Journal, 2 (1981), pp. 77–79.