[8] P. Erdős and A. Rényi, On two problems of information theory, Publ. Math. Inst. Hung. Acad. Sci. 8 (1963) 241–254.
[9] G. Katona, Combinatorial search problems, in: J.N. Srivastava et al., eds., A Survey of Combinatorial Theory (North-Holland, Amsterdam, 1973) 285–308.
[10] B. Lindström, Determination of two vectors from the sum, J. Combin. Theory 6 (1969) 402–407.
[11] B. Lindström, On B_2-sequences of vectors, J. Number Theory 4 (1972) 261–265.
[12] B. Lindström, Determining subsets by unramified experiments, in: J.N. Srivastava et al., eds., A Survey of Statistical Designs and Linear Models (North-Holland, Amsterdam, 1975) 407–418.
[13] M. Sobel, Binomial and hypergeometric group testing, Studia Sci. Math. Hung. 3 (1968) 19–42.
[14] R. Tošić, An optimal search procedure, J. Statist. Plann. Inference 4 (1980) 169–171.

Discrete Applied Mathematics 23 (1989) 11–24
North-Holland

LINEAR TIME ALGORITHMS FOR NP-HARD PROBLEMS RESTRICTED TO PARTIAL *k*-TREES

Stefan ARNBORG*
Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, S-100 44 Stockholm, Sweden

Andrzej PROSKUROWSKI**
Computer and Information Science Department, University of Oregon, Eugene, OR 97403, USA

Received 25 March 1986
Revised 1 March 1988

We present and illustrate by a sequence of examples an algorithm paradigm for solving NP-hard problems on graphs restricted to partial graphs of k-trees and given with an embedding in a k-tree. Such algorithms, linear in the size of the graph but exponential or superexponential in k, exist for most NP-hard problems that have linear time algorithms for trees. The examples used are optimization problems involving independent sets, dominating sets, graph coloring, Hamiltonian circuits, network reliability and minimum vertex deletion forbidden subgraphs. The results generalize previous results for series-parallel graphs, bandwidth-constrained graphs and nonserial dynamic programming.

1. Introduction

Partial k-trees are partial graphs of k-trees, i.e., graphs obtained by deleting edges from k-trees. k-trees are a generalization of trees, which are 1-trees. k-trees are built from the completely connected (complete) graph on k vertices by repeated addition of vertices, each of which is connected with k edges to k completely connected vertices. Our interest in such graphs stems from our belief that for many purposes the minimum k for which a graph G is a partial k-tree, $k(G)$, is a good measure of algorithmic properties of G. In this paper we give some credibility to this belief by displaying a design paradigm for algorithms on graphs embedded in k-trees. Algorithms in this family have time and space requirements linear in the number of vertices of the graph but exponential or even superexponential in k. We do not claim $k(G)$ to be a universal measure of graph complexity, since there are NP-complete problems on graphs which are easy for families of graphs which do not have bounded k (for example, CHROMATIC NUMBER restricted to cographs [8]) and there are also problems which are NP-complete on partial 1-trees (i.e., forests), such as BANDWIDTH.

* Research supported in part by the Swedish Board for Technical Development.
** Partially supported by the National Science Foundation, grant no. INT-8318441.

The algorithms given here compute a number, such as the minimum number of colors required to color vertices of a graph, or a yes/no answer. It is relatively straightforward to modify the algorithms to also compute a solution (such as an optimal coloring), e.g., by keeping trace information about intermediate stages during the problem reduction process, as shown in [2,7].

We present the algorithms in a form we believe to be easy to read and understand. Substantial improvements can be made by various types of optimizations, but the exponential cost dependence in k cannot be circumvented easily, because this would imply P = NP (since all problems considered below are NP-hard on arbitrary graphs and a graph on k vertices is a partial k-tree).

2. Definitions and outline of design paradigm

We use standard graph notation and terminology which, together with a definition of the problems we solve, can be found in [9]. We use G-adjacent and G-independent for vertices adjacent in G and vertex sets independent in G, respectively. A fuller account on partial k-trees can be found in [2–4].

A graph is a *k-tree* if and only if it satisfies either of the following conditions:

(i) It is the complete graph on k vertices, K_k.

(ii) It has a vertex v of degree k with completely connected neighbors, and the graph obtained by removing v and its incident edges is a k-tree.

The recursive definition above defines a *reduction process* for k-trees. It is well known that the reductions are confluent, i.e., one can test a graph for being a k-tree by repeatedly removing a vertex of degree k with completely connected neighbors (a *k-leaf*) until no such vertex remains, then the graph is a k-tree if and only if what remains is K_k [16]. The vertices eligible for removal at each step are the k-leaves of intermediate k-trees. A reduction sequence can be thought of as giving an orientation to the k-tree in the following sense. The removed vertices are said to succeed certain sets of k vertices (this ordering relation is between vertices and k-sets of vertices). First we define a *k-clique* to be a set of k pairwise adjacent vertices (and thus it is not a clique in standard graph terminology, where it denotes a maximal completely connected subgraph). If K is a k-clique, $v \notin K$ is a *descendant* of K in a given reduction sequence if and only if when v was being removed, each vertex to which it was adjacent was either a member of K or a descendant of K.

The connected components of the subgraph induced by descendants of K are *branches* on K, and K is the *base* of a branch on K. The k-clique remaining after a completed reduction process is the *root* of the oriented k-tree. In the following, we will always consider a k-tree oriented by a given reduction process. For an oriented k-tree with vertex v not in the root clique, let $K(v)$ be the neighbors of v when it becomes a k-leaf during the reduction process. A vertex v, together with $K(v)$, will form a $(k+1)$-clique containing $(k+1)$ k-cliques. When v is removed, k of these cliques disappear and only $K(v)$ remains, see Fig. 1.

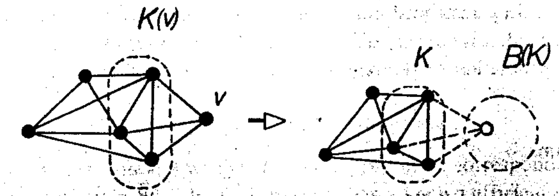

Fig. 1. Removing vertex v in 3-tree.

This process can also be described in terms of *combining removed branches* of the k-tree. Let $K' = K(v) \cup \{v\}$; the removal of v means that the branches on $K' - \{u\}$, $u \in K(v)$, are combined with v to a new branch on $K(v)$, see Fig. 2 (before removal of vertex 6, the branches are $\{1\}$ and $\{2,3\}$ on 2-clique $\{6,7\}$, $\{4,5\}$ on 2-clique $\{6,8\}$ and $\{9\}$ on 2-clique $\{7,8\}$; afterwards the branches on 2-clique $\{7,7\}$ are $\{1,2,3,4,5,6\}$ and $\{9\}$).

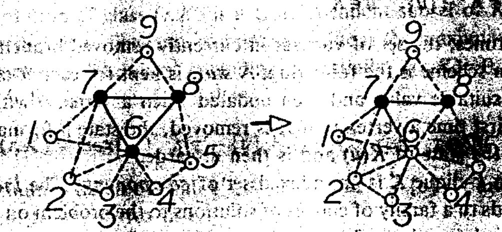

Fig. 2. Combining branches in 2-tree.

This view leads to our algorithm paradigm: For each k-clique of the embedding k-tree, we update successively a state information for the k-clique. This state information describes equivalence classes of solutions to a problem (usually a slight generalization of the original problem) restricted to the subgraph induced by the vertices of a k-clique and those removed vertices that are separated by the k-clique from all nonremoved vertices. The state information can be an extremal value for the class (e.g., the minimum number of edge covering vertices in a subgraph with a certain property, in the vertex cover problem), or some weighted sum (as in the network reliability problem where the weights are the probabilities of certain link set states and the sums are probabilities of classes of link set states) and is determined by three considerations: The number of classes must be bounded be a function of k, independently of the size of the problem graph; the information required for $K(v)$ must be computable from the information computed for the removed branches of $K' - \{u\}$, $u \in K'$, and from the problem data given for the subgraph induced by K'; and the answer to the problem must be deducible from the information computed for the root k-clique when all branches have been removed. A correctness proof is straightforward by structural induction on the branch structure. For the problems we have attacked, it is not very difficult to find the relevant information, but finding a suitable formulation of the required computation is often nontrivial.

3. Algorithms

Let $G=(V,E)$ be the graph of the problem, and let $T=(V,E')$ be its embedding k-tree, with $E \subset E'$. The solution to the general problem of finding an embedding when only the problem graph is given is discussed in Section 5. The root of the oriented T is a k-clique R. A reduction sequence is given with the embedding. In the algorithm descriptions we use the following notation: v is te vertex to be removed,

$$K=K(v), \quad K'=K\cup\{v\}$$

and

$$K^u=K'-\{u\}, \quad u\in K'.$$

$B(K)$ is, at all times, the set of vertices in currently removed branches on K.

The algorithm scheme is the following: A *state* is kept for each k-clique K of T, initialized to a suitable value and then updated when a vertex v with $K=K(v)$ is removed. The first time a vertex u of K is removed, the state information for K is used to update the state of $K(u)$ and is then discarded.

The state of a k-clique K is an indexed set of *components*. The *index set* $C(K)$ for K corresponds to a family of classes of solutions to the problem on the subgraph of G induced by the vertices of K and the vertices of the removed branches on K. These solutions will be called *partial solutions*, because our algorithms will obtain the desired solution to the given problem on G by implicity combining partial solutions. Thus, an index $c\in C(K)$ identifies all such partial solutions involving elements of K in the same manner (e.g., including into an edge-covering set exactly the same vertices of K.) The value of the component with index c, $s(c,K)$, is the optimum value of solutions represented by c. The final solution to the problem on G will be obtained by an iterative combination of these partial solutions in a *state update procedure*. This procedure is invoked when a vertex v is removed, and updates the values of all state components for the clique $K=K(v)$ with respect to the branch based on K that includes v. The update is based on the state components' values for all K^u ($u\in K'$) and—in general—reflects the ways in which the values of partial solutions on K^u influence the value of an optimal partial solution on K, subject to constraints expressed by the index c (for a particular state component).

Thus, the description of each algorithm consists of six parts:

(i) index set of state components for k-clique K;
(ii) corresponding family of classes of partial solutions (not explicitly present below);

(iii) value of state component in relation to corresponding class;
(iv) initial value of state;
(v) update procedure for the state of K when a vertex v such that $K(v)=K$ is removed;
(vi) computation of final answer from final state of the root.

We discuss optimization and weighted counting problems corresponding to decision problems as given in [9]. The first problem has a known efficient solution on partial k-trees using nonserial dynamic programming [2, 6], and it is mainly intended to introduce our notation.

Problem 1 (*Maximum independent set size* [9, GT20]).

For a given graph $G=(V,E)$ we want to find the maximum size of its independent sets. The state index set for K is the set of subsets τ of K such that no two G-adjacent vertices in K are both members of τ:

$$C(K)=\{\tau \mid \tau\subseteq K, (\{v_1,v_2\}\in E\wedge\{v_1,v_2\}\subset K)\rightarrow(v_1\notin\tau\vee v_2\notin\tau)\}.$$

A partial solution of class (τ,K) is a G-independent subset of $K\cup B(K)$ whose intersection with K is τ. The state component value is the maximum size of a solution in the corresponding class, excluding τ. Initially, $s(\tau,K)=0$, and when v is removed this value is updated as follows

$$s(\tau,K)=\max\left(\sum_{u\in K'} s(\tau'-\{u\},K^u)+|\tau'\cap\{v\}|\right), \quad \tau\in C(K).$$

(The maximum is taken over all $\tau'\subseteq K'$ such that $\tau'-\{v\}=\tau$ and $v\in\tau'$ only if v is not adjacent to any vertex in τ.)

Finally, the answer is

$$\max_{\tau\in C(R)} (s(\tau,R)+|\tau|).$$

Problem 2 (*Minimum dominating set size* [9, GT2]).

For a given graph $G=(V,E)$ we want to compute the minimum size of its dominating sets. A set $D\subset V$ is dominating in G if every vertex in V is either in D or G-adjacent to a vertex in D. The state index set for K is the set of pairs of vertex sets (τ,ω) of K, the "in" and "dominated" vertices, respectively, such that the set of "dominated" vertices contains the G-neighbors in K of "in" vertices, and the "in" and "dominated" sets are disjoint:

$$C(K)=\{(\tau,\omega)\mid \tau\cap\omega=\emptyset,\ \tau\cup\omega\subset K,\ \{v_1,v_2\}\in E\wedge\{v_1,v_2\}\subset K\rightarrow(v_1\in\tau\rightarrow v_2\in\omega\uplus\tau)\}.$$

The class of partial solutions for (τ,ω,K) is the family of subsets S of $B(K)$ such that every member of $B(K)\cup\omega$ is G-adjacent to a member of $S\cup\tau$. The value of the state component $s(\tau,\omega,K)$ is the minimum size of such a subset S from the corresponding family. Initially, $s(\tau,\omega,K)$ is 0 or ∞, depending on whether $S=\emptyset$

satisfies the requirement for membership in the class for (τ, ω, K) stated above (∞ indicating the infeasibility of a solution.)

When vertex v is removed, the values of state components for K are updated. A new solution is obtained from the solutions for $B(K^u)$, $u \in K'$, such that the classes (τ_u, ω_u, K^u) coincide on K and resolve the status of v by classifying it as either "in" or "dominated" i.e., in the set $\tau' = \bigcup_{u \in K'} \tau_u$ or in the set $\omega' = \bigcup_{u \in K'} \omega_u$. The union of the corresponding subsets S^u form a solution for $B(K)$ and $\bigcup_{u \in K'} S^u$ is placed in the equivalence class $(\tau' - \{v\}, \omega' - \{v\})$ of K. The component value for (τ, ω) is thus obtained by minimizing the size of the corresponding partial solution:

$$s(\tau, \omega, K) = \min\left(\sum_{u \in K'} s(\tau_u - \{u\}, \omega_u, K^u) + |\tau' - \tau| \right),$$

$$(\tau, \omega) \in C(K), \quad (\tau_u, \omega_u) \in C(K^u),$$

where the minimum is taken over all sets τ' and ω' such that $\tau = \tau' - \{v\}$ and $\omega = \omega' - \{v\}$ and $v \in \tau' \cup \omega'$. The answer will be

$$\min_{\tau \subset R} s(\tau, R - \tau, R) + |\tau|).$$

The update step looks rather complicated in this case. Observe, however, that $s(\tau, \omega, K)$ is nondecreasing in ω, and thus it is only necessary to consider minimal sets ω, for any choice of τ', ω'.

Problem 3 (*Chromatic number* [9, GT4]).

For a given graph $G = (V, E)$ we want to compute the size of a smallest set L of colors such that V can be colored with elements from L in such a way that no two G-adjacent vertices have the same color.

For a partition π, let π/u be the partition obtained by removing u from its block in π and then removing the block if it became empty.

The index set for the state of K, $C(K)$, is the set of partitions of K such that no two G-adjacent vertices are in the same block.

The partial solutions in class (π, K) are colorings of the subgraph of G induced by $K \cup B(K)$, such that π describes the coloring of K. The state component value for a class is the minimum total number of colors used in a partial solution of the class. Initially, $B(K)$ is empty and thus $s(\pi, K) = |\pi|$.

Given a coloring π' of K', consider colorings S_u of K^u $(u \in K')$ taken from the class $(\pi'/u, K^u)$. Since these colorings are compatible on K' for all $u \in K'$ (no two blocks overlap unless identical on K'), $\bigcup_{u \in K'} S_u$, a partition of $K' \cup \bigcup_{u \in K'} B(K^u)$ is a coloring of $K \cup B(K)$.

This coloring is assigned the new equivalence class $(\pi'/v, K)$, and the number of colors it uses is the largest of $|\pi'|$, the number of colors required for K', and the largest of the number of colors required for any S_u. Thus, upon removal of vertex v, the state should be updated as follows:

$$s(\pi, K) = \min_{\substack{\pi' \in C(K') \\ \pi'/v = \pi}} \max(|\pi'|, \max_{u \in K'} s(\pi'/u, K' - \{u\})), \quad \pi \in C(K).$$

The final answer is

$$\min_{\pi \in C(R)} s(\pi, R).$$

Problem 4 (*Hamiltonian circuit* [9, GT37]).

For a given graph $G = (V, E)$, has it a cycle passing through every vertex?

We solve the decision problem. An important variation which occurs in synchronization of distributed systems [12] asks for a shortest closed walk which covers all vertices, and can be solved with a straightforward but not trivial modification of the method we describe here.

The state of K is indexed by a set of pairs (H, I), where H is a set of mutually disjoint (unordered) pairs of different vertices in K, and I is a set of "touched" vertices in K, disjoint from H. A partial solution of class (H, I) is a set of $|H|$ vertex disjoint paths in the subgraph of G induced by $K \cup B(K)$, each with endpoints in a pair of H, such that no two consecutive internal vertices of a path are both in K, and where the path set covers $I \cup B(K) \cup \bigcup_{h \in H} h$.

The state component for (H, I) is 0 or 1, depending on whether or not the class is empty. The state $s(K)$ is thus a subset of the index set, and it is initially $\{(\emptyset, \emptyset)\}$, indicating that no paths are possible and that no vertices of K are touched.

We must now consider how to update $s(K)$. Upon removal of v, $s(K)$ is replaced by the set S, where $(H, I) \in S$ arises from a set of $|K'|$ pairs $(H_u, I_u) \in s(K^u)$, one for each $u \in K'$, such that each of the following conditions (i)–(iii) are satisfied. (Removal of the last nonroot vertex, the *root case*, is different and is described separately.)

(i) $I_u \cap I_w = \emptyset$ if $u \neq w$; $\bigcup_{h \in H_u} h \cap I_w = \emptyset$, $u, w \in K'$.

(ii) No pair occurs in more than one H_u.

(iii) The graph $F = (K', \bigcup_{u \in K'} H_u)$ (i.e., with edges over K' given by elements of H_u) has no cycle or vertex of degree 3 or higher, i.e., F is a set of paths and isolated vertices.

We now define $(H', I') \in C(K')$ as follows: H' is the set of endpoint pairs of paths in F, I' is the union of the I_u $(u \in K')$ and the interior points of the paths of F. $(H, I) \in S$ is obtained from (H, I') in one of the following ways:

(i) If $v \in I'$, then $I = I' - \{v\}$ and $H = H'$.

(ii) If $\{v, v_1\} \in H'$, for some v_1, and there exists a vertex $v_2 \in K'$ which is G-adjacent to v, and such that $v_2 \neq v_1$ and $v_2 \notin I'$, then either

(a) there is a vertex $v_3 \in K'$ such that $\{v_2, v_3\} \in H'$, $H = H' - \{\{v, v_1\}, \{v_2, v_3\}\} \cup \{\{v_1, v_3\}\}$ and $I = I' \cup \{v_2\}$; or

(b) v_2 is member of no pair in H', $H = H' - \{\{v, v_1\}\} \cup \{\{v_1, v_2\}\}$ and $I = I'$.

(iii) if v is neither in I' nor in any pair of H', but is G-adjacent to v_1 and v_2 which are in $K' - I'$, augment the graph F given above with $\{v_1, v_2\}$ and then define (H, I) in exactly the same way as (H', I') was defined but on the augmented graph.

Note that not every element of $C(K')$ yields an element of $C(K)$ (when none of the cases above apply).

The root case, i.e., the last reduction made, follows the same rules, but two additional cases must be considered. Assume that v is the last vertex removed and $R' = R \cup \{v\}$:

(i) If there are exactly two nonempty branches on the k-cliques in R' then we may combine states (H, I_1) and (H, I_2) for these branches if $I_1 \cap I_2 = \emptyset$, $H = \{\{v_1, v_2\}\}$ and $I_1 \cup I_2 \cup \{v_1, v_2\} = R'$, in which case we finish at once with answer yes.

(ii) If the graph F above consists of one cycle, the sets I_u are disjoint and together with the vertices of F they cover R', then we also finish with answer yes.

If the root case did not provide an answer, the final answer is obtained from $s(R)$ as follows: The answer is yes if and only if there is a $(H, I) \in s(R)$ such that the subgraph of G induced by $R - I$, and augmented with edges corresponding to H, has a Hamiltonian circuit using all the edges from H.

Problem 5 (*Network reliability* [9, ND20]).

We assume that a graph $G = (V, E)$ is given together with a reliability p_e (between 0 and 1) for each edge (link in network) $e \in E$, and a subset $V' \in V$ of vertices. The probability that a link e works (its state is "up") is p_e, and the links fail (are in the "down" state) independently of each other. The problem is to compute the probability that the set V' is spanned by working links. Two interesting special cases are $V' = V$ and $|V'| = 2$. We solve the case $V' = V$ and indicate how the solution can be generalized.

For partitions π_1, π_2, let $\pi_1 \vee \pi_2$ denote their join, obtained by replacing pairs of intersecting blocks (one from π_1 and one from π_2) by their union until a partition remains.

If π_1 and π_2 represent partitions of a set of vertices into connected components of graphs with respect to two sets of edges, then their join is the resulting partition into connected components of a graph whose edge set is the union of the two edge sets. The operator "/" is defined as in Problem 3.

Let $P_1(K, v, \pi)$ be the probability that the links between a k-clique K and vertex v are in states ("up", "down") such that "up" links form a graph with connected components π on $K \cup \{v\}$.

Let $\Pi(K, v)$ denote the set of partitions of $K \cup \{v\}$ that have only singleton blocks except for possibly the block containing v. Let likewise $P_2(K, \pi\}$ be the probability that links in state "up" between vertices in K induce partition π.

The index set $C(K)$ for the state of K is the set of partitions of K. Partial solutions considered are partial graphs (where the included edges are "up") of the subgraph of G induced by $K \cup B(K)$, with no edges between vertices of K.

A partial solution belongs to class (π, K) if and only if the corresponding partial graph consists of $|\pi|$ connected components such that the intersection of each component with K is a block of π. The value for a component of the state is the sum of probabilities of partial solutions in the corresponding class. The probability of a partial solution is the product of reliabilities of "up" edges and unreliabilities $(1 - p_e)$ of "down" edges. Initially, the state components describe networks without links, i.e., $s(\pi, K) = 1$ or 0 depending on whether π has only singleton blocks, or not.

Upon removal of a vertex v, a partial solution S' for K' is a partial graph of the subgraph of G induced by $K' \cup \bigcup_{u \in K'} B(K^u)$, with edges in K' removed. Such an S' is simply the union of some partial solutions S_u associated with K^u, $u \in K'$. If each S_u belongs to class (π_u, K^u), these solutions will induces a partition $\pi' = \bigvee_{u \in K'} \pi_u$ on K'. Since the solutions S_u are in independent parts of G, the probability of class (π', K') is given by

$$r(\pi', K') = \sum_{\substack{\pi_u \in C(K^u) \\ \bigvee_{u \in K'} \pi_u = \pi'}} \prod_{u \in K'} s(\pi_u, K^u), \quad \pi' \in C(K).$$

When computing the new probabilities of partial solution classes of K, we must also take in account the effect of the edges of G connecting v to members of K: if these edges are in states inducing partition $\pi'' \in \Pi(K, v)$, then the resulting partition of K will be $\pi = (\pi'' \vee \pi')/v$, but the corresponding solution belongs to (π, K) only if v is not isolated from K. A contribution to $s(\pi, K)$ will thus result from π', π'' if $(\pi', \pi'') \in \Pi'(\pi, K)$, where

$$\Pi'(\pi, K) = \{(\pi', \pi'') \mid \pi' \in C(K') \wedge \pi'' \in \Pi(K, v) \wedge (\pi' \vee \pi'')/v = \pi \wedge |\pi| = |\pi' \vee \pi''|\},$$

and the new state probabilities can be defined as:

$$s(\pi, K) = \sum_{(\pi', \pi'') \in \Pi'(\pi, K)} r(\pi', K') P_1(K, v, \pi''), \quad \pi \in C(K).$$

When all reductions have been made (resulting in the root clique R), the answer to the problem is obtained by considering the influence of the existing edges of R on the partial solutions (involving only edges not in R). We consider therefore all partition pairs that have as the only block of their join the whole clique R. Thus, the network reliability is given by:

$$\sum_{\substack{\pi_1, \pi_2 \in C(R) \\ \pi_1 \vee \pi_2 = \{R\}}} s(\pi_1, R) P_2(R, \pi_2).$$

For the case V' being a proper subset of V, the classes of states must be refined: For each partition π of K we define one subclass for every subset of not more than $|V'|$ blocks. A partial solution is assigned a subclass if the selected blocks correspond to connected components of the branches with at least one member of V', and we allow connected components isolated from K as long as they contain all or no members of V'.

A naive implementation of this algorithm would be clearly infeasible for $k = 4$, but a careful implementation of the sum of products evaluation above makes at least $k = 6$ feasible.

Problem 6 (*Minimum vertex removal forbidden subgraph F*).

For a fixed graph F the problem is: Given a graph $G(V,E)$, which is the smallest size of a vertex set such that its removal leaves a graph with no subgraph isomorphic to F.

This problem is a generalization of INDEPENDENT SET (where we ask for the minimum number of vertices that must be removed from G so that the subgraph induced by the remaining vertices has no subgraph isomorphic to K_2, the complete graph on two vertices). The algorithm we present is impractical unless a much more efficient implementation is found, but it shows a large family of problems that can be solved in linear time on partial k-trees.

Let $F=(V_F,E_F)$. A member of the index set for K is a set of pairs, $\{(m,c)\}$, where m is a bijection between a subset K^m of K and a subset V^m of V_F, and c is the set of vertices of the union of some connected components of $F(V_F-V^m)$, the subgraph of F induced by V_F-V^m (the number of these components vary over the full range of choices from the set of all connected components of $F(V_F-V^m)$). The value s for a state component indexed by $\{(m,c)\}$ means that at least s vertices must be removed from $B(K)$ so that $G(B(K))$ contains no subgraph isomorphic to F and $G(K\cup B(K))$ contains no such subgraph intersecting K and isomorphic to an induced subgraph of F, except as described by the pairs (m,c). By the latter we mean that $G(K^m\cup B(K))$ has a subgraph isomorphic to $F(V^m\cup c)$ and the corresponding bijection is an extension of m. Initially, all isomorphisms between induced subgraphs of F and subgraphs of $G(K)$ are described in one state component with c empty, and the state component value zero.

The update procedure consists of combining a set of at most one state component for each K^u, $u\in K'$ and combining compatible mappings. We do not describe the procedure in detail, because it is straightforward, clearly independent of the size of $B(K)$ and hyperexponential in both k and the size of F.

When the final state of the root has been computed, delete from the indices all pairs (m,c) such that $V^m\cup c$ is a proper subset of V_F. Should this make two indices equal, select the smallest of the two state component values. For each remaining state component, find the sum of the state component value and the size of the smallest set hitting all V^m in the index, and take the minimum of these sums as the answer to the problem.

4. Crude performance estimates

For each of the discussed problems we claim performance linear in $|G|$. This is obvious once we notice that the embedding is given with a vertex removal order, and we have (up to now tacitly) assumed that the next vertex for removal, and the information required for the corresponding state update, can be found in constant time. The work required for the update step is constant, except for Problem 5 where the cost is constant under the uniform cost measure (charging unit cost to arithmetic operations). The performance dependence on k is essentially given by the number of terms in the update expression, but this number can be trimmed by careful analysis of the computation and is therefore an upper bound. A lower bound for the worst-case update cost is proportional to the maximum number of state components of a k-clique, since it is possible to construct problems where all state components are nontrivial. In the INDEPENDENT SET problem we have $|C(K)|\le 2^k$, and every state component from K^u occurs twice in the update expression, giving the upper bound $(k+1)2^{k+1}$. In the DOMINATING SET problem, $|C(K)|$ is bounded by 3^k. The number of minimal elements of $\Omega(\tau',\omega')$ can be crudely bounded by the number of injections $\omega'\to 1,\ldots,k+1$, which has the sum over (τ',ω') given by $(k+1)^{k+1}$. In the CHROMATIC NUMBER problem, the lower and upper bounds will be $p(k)$ and $p(k+1)$, respectively, where $p(n)$ is the number of partitions of n elements. The remaining problems have exponential index sets for the states and the update procedure is, in a straightforward implementation, two-exponential because all combinations of one index from each of the $(k+1)$ involved k-cliques must be considered. We do not know if a one-exponential implementation of solutions for these problems is possible. Since the asymptotic exponential behavior does not say anything about the complexity of our algorithms for small values of k, we have calculated exactly the worst-case number of indices and steps in the update procedure for Problems 1–5. These upper bounds may help determine the feasibility of a VLSI implementation of our algorithms. Assume that such an implementation is considered feasible when the number of operations per vertex removal is less than 10^4 (this depends, of course, on technology and application). Then our algorithms are feasible for values of k as follows:

(1) INDEPENDENT SET: 9 to 13;
(2) DOMINATING SET: 4 to 8;
(3) GRAPH K-COLORABILITY: 7 to 8;
(4) HAMILTONIAN CIRCUIT: 4 to 7;
(5) NETWORK RELIABILITY: 3 to 8.

Here the lower value has been demonstrated feasible in this paper, whereas the higher value is a possibility that may be approached after careful analysis and implementation of the update step.

5. Applicability and related work

Many graph optimization problems, NP-hard on general graphs, can be solved in polynomial time when restricted to a special class of graphs. We have described a design paradigm yielding linear time algorithms when the graphs are embeddable in k-trees with a fixed value of k. The examples cover a range of problems chosen so as to suggest that the approach is generally applicable.

We want to relate our work to three different previous approaches. First, it is well known that many problems, NP-hard on arbitrary graphs, are polynomial when the graph is restricted to be a forest or a tree [9] or a series-parallel graph [11]. Since these are the first two members of the hierarchy of partial k-trees, our results are a natural generalization of those results. A general characterization of a family of problems solvable in linear time on two-terminal series-parallel graphs was attempted by Takamizawa, Nishizeki and Saito [20]. They consider graph properties defined by means of a finite set of graphs which are forbidden configurations in graphs with the property. Three different cases arise when a configuration means a subgraph, an induced subgraph, or a homeomorphic subgraph. For any graph property P defined in one of these ways they show that the decision problem for P, the minimum vertex deletion problem for P, and the minimum edge deletion problem for P can all be solved in linear time on series-parallel graphs. The result carries over to partial k-trees, although their proof and notation do not. As an example, one of the cases was generalized to partial k-trees in Section 3, Problem 6. Another related result is Gavril's [10] polynomial time algorithms for certain problems restricted to chordal graphs.

The second related work is by Monien and Sudborough [14]. They show that a number of problems have polynomial time algorithms on bandwidth-constrained graphs. Since a graph with bandwidth not greater than k is always a partial k-tree but not vice versa, our results can also be seen as a generalization of this technique from linear structure to tree structure. At the same time, some problems easy on bandwidth-constrained graphs are difficult on partial k-trees and even on trees. BANDWIDTH itself is one such problem.

Third, nonserial dynamic programming is a method for minimizing a function of a set of variables which is a sum of terms, each being a function of a subset of the variables, see Bertele and Brioschi [6]. If we construct the interaction graph whose vertices represent the variables and which has an edge (x_i, x_j) if and only if x_i and x_j occur in the same term, the standard nonserial dynamic programming algorithm is clearly a small modification of a member of our algorithm family. Finding an embedding and vertex reduction order is a problem equivalent to what is known as the secondary optimization problem of nonserial dynamic programming. We have shown, with Corneil [4], that the embedding problem is NP-hard for arbitrary k but can be solved in time $O(n^{k+2})$ for fixed k. There may be room for improvement, since an embedding of a partial 3-tree can be found in time $O(n \log n)$ [3] as opposed to $O(n^5)$ indicated by the general algorithm. This means that if the embedding is not given with the graph our algorithms are no longer linear but still polynomial. Our method can be seen as an extension of nonserial dynamic programming to a wider class of objective functions. Note also that the independent set problem can be directly translated to an instance of nonserial dynamic programming [2], but not the other problems solved in Section 3. Rosenthal [17] has developed an algorithm in the nonserial dynamic programming paradigm for NETWORK RELIABILITY which is probably closely related to our algorithm for the same problem, although he does not describe it in detail. He also showed nonserial dynamic programming optimal when the terms are arbitrary functions of variables over a finite domain, in the decision tree computation model and for chordal interaction graphs [18].

Finally, an important motivation for our study was provided by some "real life" problems for which solutions similar to ours were developed with great effort in application oriented environments. As an example, the reliability graph of a battle ship, constructed for weapons effect simulations, had several thousand vertices but was found to be a partial 4-tree [1]. The application examples used in the engineering literature of algorithms for communication networks are usually partial 4-trees and often series-parallel graphs. The application does not generate the embedding, but for $k \leq 4$ it is easy to find even on large graphs. On the other hand, some applications generate large square grids, and these cannot be embedded in k-trees for small values of k. As an example, the reliability problems for square grids, which has been studied extensively in percolation physics, can presently be solved only by Monte Carlo and other approximative methods.

Since the distribution in 1984 of the preliminary version of this paper, we have become aware of several studies (some inspired by our work, others independent of it) of efficient computations on graphs very much resembling partial k-trees (Bern et al. [5], Wimer et al. [21], Lingas [13], Seese [19].) Investigating a family of problems associated with *graph minors*, Robertson and Seymour [15] used the parameter $k(G)$ (which they call *tree-width of G*) to prove the existence (and, as of this revision, to provide an actual construction) of an efficient algorithm solving a restricted version of the NP-complete problem DISJOINT CONNECT PATHS.

References

[1] S. Arnborg, Reduced state enumeration: Another algorithm for reliability evaluation, IEEE Trans. Reliability 27 (1978) 101–105.

[2] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability: A survey, BIT 25 (1985) 2–23.

[3] S. Arnborg and A. Proskurowski, Characterization and recognition of partial 3-trees, SIAM J. Algebraic Discrete Methods 7 (1986) 305–314.

[4] S. Arnborg, D.G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, SIAM J. Algebraic Discrete Methods 8 (1987) 277–284.

[5] M.W. Bern, E.L. Lawler and A.L. Wong, Linear time computation of optimal subgraphs of decomposable graphs, J. Algorithms 8 (1987) 216–235.

[6] U. Bertele and F. Brioschi, Nonserial Dynamic Programming (Academic Press, New York, 1972).

[7] D.G. Corneil and J.M. Keil, A dynamic programming approach to the dominating set problem on k-trees, SIAM J. Algebraic Discrete Methods 8 (1987) 535–543.

[8] D.G. Corneil, H. Lerchs and L. Stewart Burlingham, Complement reducible graphs, Discrete Appl. Math. 3 (1981) 163–174.

[9] M.R. Garey and D.S. Johnson, Computers and Intractability (Freeman, San Francisco, CA 1979).

[10] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, SIAM J. Comput. 1 (1972) 180–187.

[11] D.S. Johnson, The NP-completeness column: An ongoing guide, J Algorithms 4 (1984).

[12] G. Lelann, Distributed Systems: Towards a Formal Approach (IFIP, 1977) 155–160.

[13] A. Lingas, Subgraph isomorphism for easily separable graphs of bounded valence, in: Proceedings Workshop on Graph-Theoretic Concepts in Computer Science (Trauner, 1985).

[14] B. Monien and I.H. Sudborough, Bandwidth-constrained NP-complete problems, in: Proceedings 13th ACM STOC (1981) 207–217.

[15] N. Robertson and P.D. Seymour, Graph minors II: Algorithmic aspects of treewidth, J. Algorithms 7 (1986) 309–322.

[16] D. Rose, Triangulated graphs and the elimination process, J. Math. Anal. Appl. 32 (1970) 597–609.

[17] A. Rosenthal, Computing the reliability of a complex network, SIAM J. Appl. Math. 32 (1977) 384–393.

[18] A. Rosenthal, Dynamic programming is optimal for nonserial optimization problems, SIAM J. Comput. 11 (1982) 47–59.

[19] D. Seese, Tree-partite graphs and the complexity of algorithms, Preprint, P-Math 08/86, Akademie der Wissenschaften der DDR, Karl Weierstrass Institut für Mathematik (1986).

[20] K. Takamizawa, T. Nishizeki and N. Saito, Linear-time computability of combinatorial problems on seriesparallel graphs, J. ACM 29 (1982) 623–641.

[21] T.V. Wimer, S.T. Hedetniemi, and R. Laskar, A methodology for constructing linear graph algorithms, DCS, Clemson University, Clemson, SC (1985).

Discrete Applied Mathematics 23 (1989) 25–31
North-Holland

POLYNOMIAL ALGORITHMS FOR RESTRICTED EUCLIDEAN p-CENTRE PROBLEMS

Larry AUPPERLE and J. Mark KEIL

Department of Computational Science, University of Saskatchewan, Saskatoon, Canada S7N 0W0

Received 4 August 1987
Revised 5 April 1988

Given n demand points in the plane, the p-centre problem is to locate p supply points so as to minimize the maximum distance from a demand point to its nearest supply point. Megiddo and Supowit have recently shown that not only is this problem NP-hard, but even finding a close approximate solution to the problem is NP-hard. In this paper we present a polynomial time algorithm for the Euclidean p-centre problem when the demand points are restricted to lie on a fixed number of parallel lines.

1. Introduction

Given n demand points in the plane, the p-centre problem is to locate p supply points (anywhere in the plane) so as to minimize the maximum Euclidean distance from a demand point to its nearest supply point. Megiddo and Supowit [3] have shown that not only is this problem NP-hard, but it is NP-hard to even approximate a solution to within about 15%. The best approximation algorithm available is guaranteed only to find a solution within two times the optimal solution [2].

In practice, the positions of the demand points will not be arbitrary and it may be possible to obtain an exact solution to the problem, in polynomial time, if appropriate restrictions on the positions of the demand points are realized. For example, if the demand points are restricted to lie on a single line then the p-centre problem is solvable in $O(n \log n)$ time [1,4].

In this paper we consider the p-center problem where the demand points are restricted to lie on a fixed number of parallel lines.

2. Preliminaries

When dealing with any Euclidean p-centre problem, it is natural to reduce the problem to that of covering the demand points with p circles of equal radius. To solve the p-centre problem one must find the minimum radius R so that p circles of radius R can be located so as to cover the demand points. In any such solution at least one of the circles will be the minimum spanning circle of the set S of points

0166-218X/89/$3.50