JOURNAL OF ALGORITHMS 7, 309–322 (1986)

Graph Minors. II. Algorithmic Aspects of Tree-Width

NEIL ROBERTSON AND P. D. SEYMOUR*

Ohio State University, Department of Mathematics, 231 West 18th Avenue, Columbus, Ohio 43210

Received January 15, 1983

We introduce an invariant of graphs called the *tree-width*, and use it to obtain a polynomially bounded algorithm to test if a graph has a subgraph contractible to H, where H is any fixed planar graph. We also nonconstructively prove the existence of a polynomial algorithm to test if a graph has tree-width $\leq w$, for fixed w. Neither of these is a practical algorithm, as the exponents of the polynomials are large. Both algorithms are derived from a polynomial algorithm for the DISJOINT CONNECTING PATHS problem (with the number of paths fixed), for graphs of bounded tree-width.

1. INTRODUCTION

All graphs in this paper are finite, and may have loops or multiple edges. $V(G)$ and $E(G)$ denote the sets of vertices and edges of the graph G, respectively. Let G be a graph. A *tree-decomposition* of G is a family $(X_i : i \in I)$ of subsets of $V(G)$, together with a tree T with $V(T) = I$, with the following properties.

(W1) $\bigcup(X_i : i \in I) = V(G)$.

(W2) *Every edge of G has both its ends in some X_i $(i \in I)$.*

(W3) *For $i, j, k \in I$, if j lies on the path of T from i to k then $X_i \cap X_k \subseteq X_j$.*

The *width* of the tree-decomposition is $\max(|X_i| - 1 : i \in I)$. The *tree-width* of G is the minimum $w \geq 0$ such that G has a tree-decomposition of width $\leq w$. (Equivalently, the tree-width of G is the minimum $w \geq 0$ such

*Research partially supported by NSF Grant MCS 8103440. Present address: Bell Communications Research. Inc., Morris Research and Engineering Center, 435 South Street, Morristown, New Jersey 07960.

0196-6774/86 $3.00

that G is a subgraph of a "chordal" graph with all cliques of size at most $w + 1$.)

Then, for example, trees and forests have tree-width 1 (or 0 in degenerate cases) and it can be shown that series-parallel graphs have tree-width ≤ 2. For $n \geq 1$, the complete graph K_n has tree-width $n - 1$, while the $n \times n$ grid (this has n^2 vertices—it is the adjacency graph of the chessboard with n^2 squares), for $n \geq 2$, has tree-width n.

A graph G is *contractible* to H when H can be obtained from G by contracting edges. H is a *minor* of G if G has a subgraph contractible to H. Other papers in this series [9, 10, 12, 14] establish a connection between the absence of a fixed graph H as a minor of an arbitrary graph G and the presence of a fixed type of structure on G. For example, in [12] it is shown that *if H is a planar graph there is a number w, depending only on H, such that any graph G with no minor isomorphic to H has tree-width $\leq w$.* Such structure can be exploited to obtain, by developments of standard methods, results which have defied solution for graphs in general. Thus, in [11] it is shown that *any set of graphs with bounded tree-width, no member of which is isomorphic to a minor of another member, must be finite.* This was proved by Kruskal [5] for graphs of tree-width ≤ 1, under the stronger inclusion relation of topological containment.

The main object of this paper is to use the result from [12] to obtain an algorithm for testing minor inclusion of a fixed planar graph H. The *size* of a graph G is $|V(G)| + |E(G)|$. If the running time of a graph algorithm can be bounded above by a polynomial in the size of the input graph, it is called a *polynomial* algorithm. Our main results are as follows.

(1.1) *For any fixed planar graph H, there is a polynomial algorithm to decide if the input graph has a minor isomorphic to H.*

(1.2) *For any fixed integer w, there is a polynomial algorithm to decide if the input graph has tree-width $\leq w$.*

(1.3) *For any fixed integers k, w there is a polynomial algorithm to solve the* DISJOINT CONNECTING PATHS *problem (stated below) for k paths, in graphs of tree-width $\leq w$.*

Our proofs for (1.1) and (1.3) are constructive (we find the algorithms), but for (1.2) the proof is nonconstructive. We should stress that these make no claim to be practical algorithms, for the exponent of the polynomial is very large, even for quite small H, k, w. Our concern is more theoretical, with the *NP*-completeness (or lack of it) of problems of this type. (We shall assume familiarity with this concept—see [2].)

A *termination* $\mathscr{A}$ in a graph G is a set $\mathscr{A} = \{(s_i, t_i) : 1 \leq i \leq k\}$ of pairs of vertices of G, such that $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ are all distinct. The *support* $S(\mathscr{A})$ of $\mathscr{A}$ is $\{s_1, t_1, \ldots, s_k, t_k\}$. The pair $(G, \mathscr{A})$ is a *terminated graph* and it has *index* k. The terminated graph is *feasible* if there are paths

$P_1, \ldots, P_k$ of G, pairwise vertex-disjoint, such that P_i joins s_i and $t_i (1 \leq i \leq k)$.

The problem DISJOINT CONNECTING PATHS is: *Given a terminated graph, decide if it is feasible.* It is known [3] that DISJOINT CONNECTING PATHS is *NP*-complete. However, this depends on the index being a variable part of the input. The problem is not known to be *NP*-complete for any fixed index. Indeed, we conjecture that for fixed index there is a polynomial algorithm to solve it. This is so for index 2 [15, 16, 17] and for arbitrary fixed index if the graph is restricted to lie on a fixed surface [13]. (Incidentally, even for planar graphs the problem is *NP*-complete if we do not bound the index [6, 8].)

The key result of the present paper (essentially (1.3)) is that for any fixed numbers w, k there is a polynomial algorithm which, given as input a terminated graph with index $\leq k$, either decides if it is feasible or outputs that the graph has tree-width $> w$. The next section contains some preliminaries to the proof of this, and the proof itself is given in Section 3. In Sections 4 and 5 we derive (1.1) and (1.2), using (1.3) and the results of [11] and [12] stated above.

2. Preliminaries

Suppose we have a tree-decomposition of G, with notation as before. For each vertex $i \in I$ of the tree T, the connected components of $T \setminus i$ are called the *branches* of T at i. [$G \setminus X$ denotes the graph obtained by deleting X from G, where X may be a vertex or edge, or a set of vertices or edges.] The branches at i are in a natural correspondence with the edges of T incident with i.

(2.1) *For $i \in I$ and $v \in V(G)$, either $v \in X_i$, or there is a branch of T at i which contains all $j \in I$ with $v \in X_j$.*

This follows immediately from (W1), (W3). If $v \notin X_i$, let $T_i(v)$ denote this branch. (It is unique).

(2.2) *If $v, v' \notin X_i$ and v, v' are adjacent in G then $T_i(v) = T_i(v')$.*

Proof. By (W2), there exists $j \in I$ such that $v, v' \in X_j$. Then $j \neq i$, and so j is in some branch B of T at i. But then $B = T_i(v)$, because $v \in X_j$, and similarly $B = T_i(v')$.

(2.3) *If $v, v \notin X_i$ and are not separated in G by X_i, then $T_i(v) = T_i(v')$.*

[Y, Y' are *separated* by X in G if every path in G from Y to Y' meets X. Here X, Y, Y' are vertices or sets of vertices.]

Proof. Let $v_1, v_2, \ldots, v_r$ be a sequence of vertices of G not in X_i, each adjacent to the next, with $v_1 = v$ and $v_r = v'$. Then by (2.2),

$$T_i(v_1) = T_i(v_2) = \cdots = T_i(v_r)$$

and so $T_i(v) = T_i(v')$.

(2.4) *Let e be an edge of T with ends j, j' say, and N, N' be the vertex sets of the two components of $T \setminus e$. Then $X_j \cap X_{j'}$ separates $\bigcup(X_i : i \in N)$ and $\bigcup(X_i : i \in N')$.*

Proof. Suppose not. Then from (W1), there exist

$$v, v' \in V(G) - (X_j \cap X_{j'})$$

with $v \in X_i$ and $v' \in X_{i'}$ say, where $i \in N$ and $i' \in N'$, such that either $v = v'$ or v, v' are adjacent in G. From (W1), (W2) there exists $k \in I$ such that $v, v' \in X_k$. We assume without loss of generality that $k \in N'$. Then j, j' are both on the path of T between i and k, and so $X_i \cap X_k \subseteq X_j, X_{j'}$ by (W3). Hence $v \in X_j \cap X_{j'}$, a contradiction, as required.

(2.5) *If the tree-decomposition has width $< w$ and $Q \subseteq V(G)$, then there exists $X \subseteq V(G)$ with $|X| \leq w$ such that every component of $G \setminus X$ has at most $\frac{1}{2}|Q - X|$ vertices which are in Q.*

Proof. There are two cases.

Case 1. *Some $i \in I$ has the property that for each branch B of T at i,*

$$|\{q \in Q - X_i : T_i(q) = B\}| \leq \tfrac{1}{2}|Q - X_i|.$$

Then we may take $X = X_i$. For $|X_i| \leq w$, and if C is a component of $G \setminus X_i$, all vertices v of C have $T_i(v)$ the same, by (2.3), and so C has at most $\frac{1}{2}|Q - X_i|$ vertices which are in Q, as required.

Case 2. *For each $i \in I$ there is a branch B_i of T at i such that*

$$|\{q \in Q - X_i : T_i(q) = B_i\}| > \tfrac{1}{2}|Q - X_i|.$$

Let e_i be the edge of T incident with i and with some vertex of B_i. Now T has fewer edges than vertices, and so there exist distinct $j, j' \in I$ such that $e_j = e_{j'}$. Then j, j' are adjacent in T, and $j' \in V(B_j)$, $j \in V(B_{j'})$. Put $N = V(B_{j'})$, $N' = V(B_j)$. Then $N \cup N' = I$, and $N \cap N' = \varnothing$. Put $X_j \cap X_{j'} = A$, and put

$$W = \bigcup_{i \in N} (X_i - A), \qquad W' = \bigcup_{i \in N'} (X_i - A).$$

Then $W \cup W' = V(G) - A$ and $W \cap W' = \varnothing$. Without loss of generality we assume that

$$|W \cap Q| \geq |W' \cap Q|. \tag{1}$$

Now since $N' = V(B_j)$ and

$$\{q \in Q - X_j : T_j(q) = B_j\} = W' \cap Q,$$

we have $|W' \cap Q| > \frac{1}{2}|Q - X_j|$. But

$$Q - X_j = (W' \cap Q) \cup ((W - X_j) \cap Q)$$

and so

$$|(W - X_j) \cap Q| < |W' \cap Q|. \tag{2}$$

From (1) and (2) we deduce that there exists $Y \subseteq X_j - A$ such that

$$|(W - Y) \cap Q| = |W' \cap Q|.$$

Then $(W - Y, W')$ is a partition of $V(G) - (Y \cup A)$, and so

$$|(W - Y) \cap Q| = |W' \cup Q| = \tfrac{1}{2}|Q - (Y \cup A)|.$$

But $W - Y$ and W' are separated in G by A, from (2.4), and so every component of $G \setminus (Y \cup A)$ has at most $\frac{1}{2}|Q - (Y \cup A)|$ vertices which are in Q. Also, $|Y \cup A| \leq w$, since $Y \cup A \subseteq X_j$. Thus we may take $X = Y \cup A$ and the theorem is satisfied.

(2.6) *If G has tree-width $< w$ and $Q \subseteq V(G)$, there exists a separation (H, H') of G such that $|V(H) \cap V(H')| \leq w$ and*

$$|(V(H) - V(H')) \cap Q|, |(V(H') - V(H)) \cap Q| \leq \tfrac{2}{3}|Q - (V(H) \cap V(H'))|.$$

[A *separation* of G is a pair of subgraphs (H, H') of G, such that $V(H) \cup V(H') = V(G)$, $E(H) \cup E(H') = E(G)$, and $E(H) \cap E(H') = \varnothing$.]

Proof. It is sufficient to find a partition (X, Y, Y') of $V(G)$ such that $|X| \leq w$ and X separates Y, Y' in G and such that

$$|Y \cap Q|, |Y' \cap Q| \leq \tfrac{2}{3}|Q - X|.$$

For then H, H' can be constructed with $V(H) = X \cup Y$, $V(H') = X \cup Y'$, and with the edges of G divided between H and H'. But since G has a

tree-decomposition with width $< w$, there exists $X \subseteq V(G)$ with $|X| \leq w$ such that every component of $G \setminus X$ has at most $\frac{1}{2}|Q - X|$ vertices which are in Q, by (2.5).

Let $G \setminus X$ have components $C_1, C_2, \ldots, C_r$, where

$$|V(C_i) \cap Q| = c_i \qquad (1 \leq i \leq r)$$

and $c_1 \geq c_2 \geq \cdots \geq c_r$. Then $c_1 + \cdots + c_r = |Q - X|$, and so $c_1 \leq \frac{1}{2}(c_1 + \cdots + c_r)$, and $r \geq 2$. If $c_1 \geq \frac{1}{3}(c_1 + \cdots + c_r)$ we may take $Y = V(C_1)$, $Y' = V(C_2) \cup \cdots \cup V(C_r)$. We assume then that $c_1 < \frac{1}{3}(c_1 + \cdots + c_r)$. Choose i minimum such that

$$c_1 + \cdots + c_i \geq \tfrac{1}{3}(c_1 + \cdots + c_r).$$

Then

$$c_1 + \cdots + c_{i-1} \leq \tfrac{1}{3}(c_1 + \cdots + c_r),$$

and $c_i \leq c_1 \leq \frac{1}{3}(c_1 + \cdots + c_r)$, and so

$$c_1 + \cdots + c_i \leq \tfrac{2}{3}(c_1 + \cdots + c_r).$$

Thus we may take $Y = V(C_1) \cup \cdots \cup V(C_i)$, $Y' = V(C_{i+1}) \cup \cdots \cup V(C_r)$, and the theorem is satisfied.

For the algorithms of the next section, the only facts we need about tree-width are (2.6) and the following elementary observation.

(2.7) *If G has tree-width $< w$, so does every minor of G.*

Our algorithm (1.3) to test feasibility of a terminated graph G takes advantage of the separation given by (2.6) with $Q = V(G)$ to reduce to two smaller problems, each with at most two-thirds (approximately) of the original vertices. There are basically two difficulties.

(i) This reduction can sometimes increase the index of the terminated graph, if the support of the original termination is not divided fairly evenly between the two sides of the separation, because the "separating" vertices are included in the supports of the new terminations. This is easily overcome; we simply apply (2.6) to both these smaller problems, this time with Q the support of the current termination, and reduce to four problems, each with index at most that of the original and each containing at most two-thirds of the original vertices.

(ii) The original terminated graph may be feasible only by means of paths which cross from H to H' and back several times, and our method of reduction to smaller problems must take account of this. For that reason we need the following technical device.

Let (H, H') be a separation of G, and let $\mathscr{A}_0 = \{(s_i, t_i) : 1 \le i \le k\}$ be a termination of G. Put $V(H) \cap V(H') = X$, $V(H) - V(H') = Y$, $V(H') - V(H) = Y'$. A *crossing scheme* with respect to (H, H') is a quadruple $(\mathscr{A}, \mathscr{A}', Z, Z')$, where $\mathscr{A}$, $\mathscr{A}'$ are terminations of G and $Z, Z' \subseteq X$, such that

(i) $S(\mathscr{A}) - X = S(\mathscr{A}_0) \cap Y$ and $S(\mathscr{A}') - X = S(\mathscr{A}_0) \cap Y'$,

(ii) every vertex of X is in an even number of $S(\mathscr{A}_0)$, $S(\mathscr{A})$, $S(\mathscr{A}')$,

(iii) $Z \cap Z' = \emptyset$, and $Z \cup Z' = X - (S(\mathscr{A}_0) \cup S(\mathscr{A}) \cup S(\mathscr{A}'))$,

(iv) if F is the graph with vertex set $S(\mathscr{A}) \cup S(\mathscr{A}')$ and edge set $\mathscr{A} \cup \mathscr{A}'$, then F has precisely k components $F_1, \ldots, F_k$ say, and for $1 \le i \le k$, F_i is a path joining s_i and t_i.

(2.8) *The number of different crossing schemes for $\mathscr{A}_0$ with respect to (H, H') is at most*

$$\max\bigl(2, (|X| + k)^{|X|}\bigr).$$

The proof is routine, and is left to the reader. (This is only a crude bound, which we have used for simplicity.) All that we shall need from (2.8) is that the number of crossing schemes is bounded by a function depending only on $|X|$ and k, and not on the size of G.

Now let $(\mathscr{A}, \mathscr{A}', Z, Z')$ be a crossing scheme for $\mathscr{A}_0$ with respect to (H, H'). Then $\mathscr{A}$ is a termination for $H \setminus Z$ and $\mathscr{A}'$ is a termination for $H' \setminus Z'$. We call $(H \setminus Z, \mathscr{A})$, $(H' \setminus Z', \mathscr{A}')$ the terminated graphs *induced* by the crossing scheme.

(2.9) *Let $(G, \mathscr{A}_0)$ be a terminated graph and let (H, H') be a separation of G. Then $(G, \mathscr{A}_0)$ is feasible if and only if there is a crossing scheme for $\mathscr{A}_0$ with respect to (H, H') such that both induced terminated graphs are feasible.*

This is clear once the various definitions have been digested, and it is left to the reader. The purpose of Z and Z' is to allow for the possibility that the desired paths may need to use the "separating" vertices without passing from one side of the separation to the other.

We also have

(2.10) *The index of any terminated graph induced by a crossing scheme for $\mathscr{A}_0$ with respect to (H, H') is at most*

$$\tfrac{1}{2}|V(H) \cap V(H')| + \tfrac{1}{2}\max\bigl(|(V(H) - V(H')) \cap S(\mathscr{A}_0)|,$$
$$|(V(H') - V(H)) \cap S(\mathscr{A}_0)|\bigr).$$

3. The Basic Algorithm

Now we give the details of our main algorithm. We require first

(3.1) *For any fixed integer w, there is a polynomial algorithm which has input a simple graph G and a subset Q of $V(G)$, and which has output either*

(i) *a separation (H, H') of G with $|V(H) \cap V(H')| \leq w$, such that*

$$|(V(H) - V(H')) \cap Q|, |(V(H') - V(H)) \cap Q| \leq \tfrac{2}{3}|Q - (V(H) \cap V(H'))|,$$

or

(ii) *a (valid) statement that no such separation exists.*

Proof. We examine all subsets $X \subseteq V(G)$ of cardinality $\leq w$, and check if some component of $G \setminus X$ has more than $\frac{2}{3}|Q - X|$ vertices which are in Q. If this is true for all such X, then no separation of the required kind exists; and if not we can produce H, H', as in the proof of (2.6). Clearly this is a polynomial algorithm, since w is fixed. Moreover, the running time is bounded by a polynomial in $|V(G)|$, since G is simple. This concludes the proof.

We shall inductively define two algorithms $A_1(n)$ and $A_2(n)$, for $n \geq 1$. Throughout, w is a fixed integer. $A_1(n)$ has as input a terminated graph $(G, \mathscr{A})$ with index $\leq \frac{5}{2}w$, where G is simple and has at most n vertices. It either decides whether $(G, \mathscr{A})$ is feasible or outputs that G has tree-width $\geq w$. $A_2(n)$ has as input a terminated graph $(G, \mathscr{A})$ with index $\leq 3w$, where G is simple and has at most n vertices. It either decides whether $(G, \mathscr{A})$ is feasible or outputs that G has tree-width $\geq w$.

Definition of $A_1(n)$. If $n \leq 4w$ proceed by exhaustion. If $n > 4w$ (this restriction is only of importance later when analyzing running time) take two steps.

Step 1. Use the algorithm of (3.1) with $Q = V(G)$ to find a separation (H, H') of G where $|V(H) \cap V(H')| \leq w$ and

$$|V(H) - V(H')|, |V(H') - V(H)| \leq \tfrac{2}{3}|V(G) - (V(H) \cap V(H'))|.$$

If no such separation exists, announce (by (2.6)) that G has tree-width $\geq w$, and stop. If the separation has been found, go to step 2.

Step 2. Generate all crossing schemes for $\mathscr{A}$ with respect to (H, H'), and for each apply $A_2(\frac{2}{3}n + \frac{1}{3}w)$ to the two induced terminated graphs. If in one of these applications it is found that the graph has tree-width $\geq w$, announce that G has tree-width $\geq w$. If not, then the feasibility of all these

induced terminated graphs has been found. Then decide if $(G, \mathscr{A})$ is feasible using (2.9).

We observe that $A_2(\frac{2}{3}n + \frac{1}{3}w)$ is applicable here, since

$$|V(H)|, |V(H')| \leq \tfrac{2}{3}n + \tfrac{1}{3}w$$

and the induced terminated graphs all have index $\leq \frac{5}{2}w + \frac{1}{2}w = 3w$, by (2.10).

DEFINITION OF $A_2(n)$. This is identical with the definition of $A_1(n)$, except that in step 1 we use (3.1) with $Q = S(\mathscr{A})$ instead of with $Q = V(G)$, and in step 2 we apply $A_1(n)$ instead of $A_2(\frac{2}{3}n + \frac{1}{3}w)$.

We observe that $A_1(n)$ is applicable here, since

$$|V(H)|, |V(H')| \leq n$$

and, by (3.1) and (2.10), the induced terminated graphs all have index at most

$$\tfrac{1}{2}|V(H) \cap V(H')| + \tfrac{1}{2} \cdot \tfrac{2}{3} \cdot 6w \leq \tfrac{5}{2}w.$$

This completes the definition of $A_1(n)$ and $A_2(n)$.

Let $f_1(n)$ and $f_2(n)$ denote the respective worst-case running times of $A_1(n)$ and $A_2(n)$. Let the algorithm of (3.1) have running time $\leq c_1 n^{d_1}$ for $m \geq 1$, when $|V(G)| = n$. Then from (2.8) we have

$$\begin{aligned}
\text{if } n > 4w \quad \text{then } f_1(n) &\leq c_1 n^{d_1} + c_2 + c_3 f_2\left(\tfrac{2}{3}n + \tfrac{1}{3}w\right) \\
&\leq c_1 n^{d_1} + c_2 + c_3 f_2\left(\tfrac{3}{4}n\right) \\
&\leq c\left(n^{d_1} + f_2\left(\tfrac{3}{4}n\right)\right), \\
\text{if } n > 4w \quad \text{then } f_2(n) &\leq c_1 n^{d_1} + c_4 + c_5 f_1(n) \\
&\leq c\left(n^{d_1} + f_1(n)\right), \\
\text{if } n \leq 4w \quad \text{then } f_1(n), f_2(n) &\leq c,
\end{aligned}$$

where $c_2, \ldots, c_5, c$ are appropriate constants depending only on w. Then for $n > 4w$,

$$f_2(n) \leq (c + c^2)n^{d_1} + c^2 f_2\left(\tfrac{3}{4}n\right)$$

and so $f_2(n)$ is bounded by a polynomial in n. The algorithms $A_2(n)$ $(n > 0)$ therefore provide a polynomial algorithm A_2 (say), which, given as input a terminated graph $(G, \mathscr{A})$ of index $\leq 3w$ where G is simple, either decides if $(G, \mathscr{A})$ is feasible or announces that G has tree-width $\geq w$.

It follows that

(3.2) *For any two fixed integers w, k there is a polynomial algorithm which has input a terminated graph $(G, \mathscr{A})$ of index $\leq k$, and outputs either*

(i) *$(G, \mathscr{A})$ is feasible, or*

(ii) *$(G, \mathscr{A})$ is not feasible, or*

(iii) *G has tree-width $> w$.*

Proof. Define $w' = 1 + \max(w, k/3)$, and apply algorithm A_2, with w' replacing w, to a pair $(G', \mathscr{A})$ where G' is the underlying simple graph of G.

4. Applications of the Basic Algorithm

In this section we derive (1.1), which is the main result of this paper, and we restate it for the reader's convenience.

(4.1) *For any fixed planar graph H, there is a polynomial algorithm to decide if the input graph has a minor isomorphic to H.*

Let v be a vertex of a graph G which is incident with only two edges e_1, e_2, and suppose e_1, e_2 are not loops. Let e_i have ends v, v_i $(i = 1, 2)$, so that $v_1, v_2 \neq v$. The result of *suppressing* v is the graph obtained from G by deleting v and adding a new edge with ends v_1, v_2.

We say that G *topologically contains* H if H can be obtained from a subgraph of G by repeated suppression. We observe that if G topologically contains H then $V(H) \subseteq V(G)$. There is a connection between topological containment and minors, for clearly if G topologically contains H then G has a minor isomorphic to H. The converse does not hold, however—it is possible for G to have a minor isomorphic to H and yet not to topologically contain any graph isomorphic to H.

The following two results are clear.

(4.2) *If $V(H) \subseteq V(G)$ and e is a loop of G incident with $v \in V(H)$, and no loop of H is incident with v, then G topologically contains H if and only if $G \setminus e$ does.*

(4.3) *If $V(H) \subseteq V(G)$, and e is a loop of G and f is a loop of H, both incident with the same vertex, then G topologically contains H if and only if $G \setminus e$ topologically contains $H \setminus f$.*

Now let $V(H) \subseteq V(G)$, and suppose that no loop of G is incident with a vertex of H. We construct a terminated graph $(G', \mathscr{A})$ as follows. Let the edges of H be $e_1, \ldots, e_k$, and let e_i have ends $u_i, v_i \in V(G)$ $(1 \leq i \leq k)$.

Take a set X of $2k$ new vertices $\{s_1, t_1, \ldots, s_k, t_k\}$. Let $Y = V(G) - V(H)$, and for $v \in X \cup Y$, define

$$\begin{aligned} g(v) &= v \text{ if } v \in Y \\ &= u_i \text{ if } v = s_i \ (1 \le i \le k) \\ &= v_i \text{ if } v = t_i \ (1 \le i \le k). \end{aligned}$$

Define a simple graph G' with a vertex set $X \cup Y$, in which distinct $a, b \in X \cup Y$ are adjacent if and only if $g(a), g(b)$ are adjacent in G. Let $\mathscr{A} = \{(s_i, t_i) : 1 \le i \le k\}$. Then $(G', \mathscr{A})$ is a terminated graph. We evidently have

(4.4) *G' has tree-width at most $2k$ more than the tree-width of G.*

This follows because we may obtain a subgraph of G by deleting $2k$ vertices of G'. Moreover, it is easy to see the following.

(4.5) *G topologically contains H if and only if $(G', \mathscr{A})$ is feasible.*

We use (4.2)–(4.5) to give an algorithm as follows.

(4.6) *For any fixed graph H_0 and fixed integer w, there is a polynomial algorithm which has input graphs G, H with $V(H) \subseteq V(G)$ such that H is isomorphic to a subgraph of H_0, and which has output either*

(i) *G topologically contains H, or*

(ii) *G does not topologically contain H, or*

(iii) *G has tree-width $> w$.*

Proof. If some vertex of H is incident with a loop of G, we use (4.2) or (4.3) to eliminate the loop. If there is no such vertex, we construct $(G', \mathscr{A})$ as for (4.4) and (4.5), and apply (3.2) to $(G', \mathscr{A})$ with w replaced by $w + 2k$. We find either

(i) $(G', \mathscr{A})$ is feasible, or

(ii) $(G', \mathscr{A})$ is not feasible, or

(iii) G' has tree-width $> w + 2k$.

For the first two alternatives we use (4.5) to decide whether or not G topologically contains H, and for the third we deduce from (4.4) that G has tree-width $> w$.

(4.7) *For any fixed graph H and fixed integer w, there is a polynomial algorithm which has input a graph G, and which has output one of*

(i) *there is a graph isomorphic to H topologically contained by G, or*

(ii) *there is no such graph, or*

(iii) *G has tree-width $> w$.*

Proof. We use (4.6) to try the $n(n-1)\cdots(n-p+1)$ injections of $V(H)$ into $V(G)$, one by one (where $|V(G)| = n$ and $|V(H)| = p$).

(4.8) *For any graph H there is a finite set $H_1, \ldots, H_r$ of graphs such that for any graph G, G has a minor isomorphic to H if and only if G topologically contains a graph isomorphic to one of $H_1, \ldots, H_r$.*

This is elementary. (We "split" every vertex of H with valency ≥ 4 into two adjacent vertices of valency ≥ 3, repeatedly, in all possible ways.)

(4.9) *For any fixed graph H and fixed number w, there is a polynomial algorithm which has input a graph G, and has output either*

(i) *G has a minor isomorphic to H, or*

(ii) *G has no minor isomorphic to H, or*

(iii) *G has tree-width $> w$.*

Proof. We apply (4.7) to G, testing for each of the graphs $H_1, \ldots, H_r$ of (4.8) in turn.

Theorem (4.1) follows from (4.8) and the following.

(4.10) [12]. *For every planar graph H there is a number w such that every graph with no minor isomorphic to H has tree-width $\leq w$.*

The algorithm for (4.1) is as follows. Given a graph G, we apply the algorithm of (4.9) to it, with w as in (4.10). Each of the three possible outcomes determines whether or not G has a minor isomorphic to H.

5. An Algorithm for Tree-Width

Now we prove theorem (1.2). The following is shown in [11].

(5.1) *Let w be a number, and let $\mathscr{G}$ be a set of graphs each with tree-width $\leq w$. Suppose that no member of $\mathscr{G}$ is isomorphic to a minor of another member. Then $\mathscr{G}$ is finite.*

We deduce

(5.2) *For any fixed number w, there is a finite set of graphs $H_1, \ldots, H_r$ such that for any graph G, G has tree-width $> w$ if and only if G has a minor isomorphic to one of $H_1, \ldots, H_r$.*

Proof. Let $\mathscr{G}'$ be the set of all graphs H such that H has tree-width $> w$ but every minor of H distinct from H has tree-width $\leq w$. Then every

member of $\mathscr{C}'$ has tree-width $w + 1$, as is easily seen. By (2.7), a graph has tree-width $> w$ if and only if it has a minor in $\mathscr{C}'$. Choose a representative from each isomorphism class of $\mathscr{C}'$, and let the set of these representatives be $\mathscr{C}$. Now no member of $\mathscr{C}$ is isomorphic to a minor of another, and so $\mathscr{C}$ is finite by (5.1) (with $w + 1$ replacing w). Moreover, a graph has tree-width $> w$ if and only if it has a minor isomorphic to a member of $\mathscr{C}$. This completes the proof.

(5.3) *For any fixed number w, there is a polynomial algorithm which, given a graph G, decides if G has tree-width $> w$.*

Proof. Let $H_1, \ldots, H_r$ be as in (5.2). We apply (4.9) to G with $H_1, \ldots, H_r$ in turn. If one of the applications yields that G has tree-width $> w$, we are done. Otherwise we decide whether or not G has a minor isomorphic to H_i, for $1 \le i \le r$, and hence decide whether or not G has tree-width $> w$, by (5.2).

We cannot claim to have constructed an algorithm here, because we do not know any way to find the required graphs $H_1, \ldots, H_r$. Thus this is merely a nonconstructive proof of the existence of the algorithm.

The argument for (5.3) can be extended to yield the following.

(5.4) *Let $\mathscr{F}$ be a set of graphs, such that for any two graphs G, G', if $G \in \mathscr{F}$ and G' is isomorphic to a minor of G then $G' \in \mathscr{F}$. Suppose that there is a number w such that all members of $\mathscr{F}$ have tree-width $\le w$. Then there is a polynomial algorithm to test if an arbitrary graph is in $\mathscr{F}$.*

The proof is left to the reader. Alternatively formulated, we have

(5.5) *Let $\mathscr{F}$ be any set of graphs, membership of which is preserved under isomorphism and under taking minors. Suppose that some planar graph is not in $\mathscr{F}$. Then there is a polynomial algorithm to test if an arbitrary graph is in $\mathscr{F}$.*

This follows easily from (4.10) and (5.4). We conjecture that the second sentence can be removed from (5.5).

Independently of this paper, Arnborg, Corneil, and Proskurowski [1] have given a polynomial algorithm to test if a graph has tree-width $\le k$, for fixed k, and have shown the problem of determining tree-width to be *NP*-complete. See also [4, 7] for related results.

Note added in proof. March 1986: We have recently proved both the conjectures of this paper. The general DISJOINT CONNECTING PATHS problem (for any fixed index) and the problem of deciding membership of an arbitrary minor-closed class (generalizing (5.5)) can both be solved by polynomial algorithms with running times $O(|V|^2 \cdot |E|)$. We have linear time algorithms for (3.2) and (4.1).

REFERENCES

1. S. ARNBORG, D. G. CORNEIL, AND A. PROSKUROWSKI, Complexity of finding embeddings in a k-tree, preprint, 1984.
2. M. R. GAREY AND D. S. JOHNSON, "Computers and Intractability," Freeman, San Francisco, 1979.
3. R. M. KARP, On the complexity of combinatorial problems, *Networks* **5** (1975), 45–68.
4. L. M. KIROUSIS AND C. H. PAPADIMITRIOU, Searching and pebbling, preprint, 1984.
5. J. B. KRUSKAL, Well-quasi-ordering, the tree theorem, and Vázsonyi's conjecture, *Trans. Amer. Math. Soc.* **95** (1960), 210–225.
6. J. F. LYNCH, The equivalence of theorem proving and the interconnection problem, *ACM SIGDA Newsletter* **5** (3) (1975), 31–65.
7. N. MEGIDDO, S. L. HAKIMI, M. R. GAREY, D. S. JOHNSON AND C. H. PAPADIMITRIOU, The complexity of searching a graph (preliminary version), *in* "Proc. 22nd IEEE Found. Of Comp. Sci. Symp.," 1981, pp. 376–381.
8. D. RICHARDS, Complexity of single-layer routing, unpublished manuscript, 1981.
9. N. ROBERTSON AND P. D. SEYMOUR, Graph minors. I. Excluding a forest, *J. Combin. Theory Ser. B* **35** (1983), 39–61.
10. N. ROBERTSON AND P. D. SEYMOUR, Graph minors. III. Planar tree-width, *J. Combin. Theory Ser. B* **36** (1984), 49–64.
11. N. ROBERTSON AND P. D. SEYMOUR, Graph minors. IV. Tree-width and well-quasi-ordering, submitted.
12. N. ROBERTSON AND P. D. SEYMOUR, Graph minors. V. Excluding a planar graph, *J. Combin. Theory Ser. B*, to appear.
13. N. ROBERTSON AND P. D. SEYMOUR, Graph minors. VII. Disjoint paths on a surface, submitted.
14. N. ROBERTSON AND P. D. SEYMOUR, Graph minors. XIV. Excluding a nonplanar graph, in preparation.
15. P. D. SEYMOUR, Disjoint paths in graphs, *Discrete Math.* **29** (1980), 293–309.
16. Y. SHILOACH, A polynomial solution to the undirected two paths problem, *J. Assoc. Comput. Mach.* **27** (1980), 445–456.
17. C. THOMASSEN, 2-linked graphs, *European J. Combin.* **1** (1980), 371–378.