SIAM J. COMPUT.
Vol. 17, No. 1, February 1988

ON THE COMPLEXITY OF COVERING VERTICES BY FACES IN A PLANAR GRAPH*

DANIEL BIENSTOCK† AND CLYDE L. MONMA†

Abstract. The pair (G, D) consisting of a planar graph $G=(V, E)$ with n vertices together with a subset of d special vertices $D \subseteq V$ is called k-planar if there is an embedding of G in the plane so that at most k faces of G are required to cover all of the vertices in D. Checking 1-planarity can be done in linear-time since it reduces to a problem of checking planarity of a related graph. We present an algorithm which given a graph G and a value k either determines that G is not k-planar or generates an appropriate embedding and associated minimum cover in $O(c^k n)$ time, where c is a constant. Hence, the algorithm runs in linear time for any fixed k. The fact that the time required by the algorithm grows exponentially in k is to be expected since we also show that for arbitrary k, the associated decision problem is strongly NP-complete, even when the planar graph has essentially a unique planar embedding, $d = \theta(n)$, and all facial cycles have bounded length. These results provide a polynomial-time recognition algorithm for special cases of Steiner tree problems in graphs which are solvable in polynomial time.

Key words. complexity, planar graphs, Steiner trees

AMS(MOS) subject classifications. 05, 68

1. Introduction.

Recently, there has been a great deal of interest in solving the Steiner tree problem in graphs. This problem is NP-complete even for planar grid graphs [GJ1]. (See [GJ2] for an excellent introduction to the area of computational complexity.) So recent work has centered on efficiently-solvable special cases and heuristic methods; see [Wi] for a survey of work on this problem.

Throughout this paper we deal with undirected graphs of the form $G=(V, E)$, where V is a set of n vertices and E is a set of edges connecting pairs of vertices. A graph is called *planar* if it can be embedded in the plane. A graph $G=(V, E)$ together with d special vertices $D \subseteq V$ is called *k-planar* if there is a planar embedding of G so that at most k faces of G are required to cover all of the vertices in D. Clearly, a planar graph is the same as an n-planar graph. The *planarity number* of G is the minimum k such that G is k-planar.

A recent paper by [EMV] presents an algorithm which solves the Steiner problem in an arbitrary graph; their algorithm runs in polynomial time for k-planar graphs, for any fixed k, with D being the vertices required to be in the Steiner tree. It is easy to see that checking 1-planarity of $G=(V, E)$ with special vertices $D \subseteq V$ is equivalent to testing the planarity of the associated graph $G^*=(V^*, E^*)$, where $V^* = V \cup \{\sigma\}$ and $E^* = E \cup \{(\sigma, v)\colon v \in D\}$, and so can be done in linear time [HT2]. They leave as an open question the complexity of testing k-planarity for fixed $k \geqq 2$.

In § 2, we present an algorithm which checks to see if a given (G, D) pair is k-planar given a *fixed* embedding of G and if so, determines the planarity number of G in $O(c^k n)$ time, when c is a constant. This is used in § 3 to generate an appropriate embedding of G and a cover of D by k or fewer faces, if possible, in $O(c^k n)$ time. Hence, the algorithm runs in linear time for any fixed k. The fact that the time required grows exponentially in k is to be expected as we show in § 4 that for arbitrary k, the associated decision problem is strongly NP-complete, even when the planar graph has essentially a unique planar embedding, $d = \theta(n)$, and all facial cycles have bounded length.

* Received by the editors September 22, 1986; accepted for publication (in revised form) April 16, 1987.

† Bell Communications Research, Morristown, New Jersey 07960.

In § 5, we present an optimization algorithm which finds the minimum number of faces required to cover all special vertices of a planar graph with a fixed embedding in $2^{O(\sqrt{d})}$ time. This exact algorithm is used to obtain a polynomial-time approximation algorithm which is asymptotically optimal (i.e., the relative error converges to zero), for the class of graphs we showed to be NP-complete in § 4.

2. Testing k-planarity for a fixed embedding. Consider a fixed embedding of the graph $G=(V, E)$. In this section we describe an algorithm that tests whether k faces are sufficient to cover all special vertices of D in this particular embedding and, if so, whether it determines the planarity number. This algorithm requires $O(c^k n)$ time for some constant c, and is used as a subroutine in our k-planarity testing algorithm for a variable embedding in the next section. We note that if G is three-vertex connected, then G has essentially a unique embedding and so the results of this section apply.

Throughout this section, we assume that the embedding of G is fixed. We transform the problem of covering D with faces into one of covering certain special faces as follows. We transform each vertex of D into a polygon, that is, if $v \in D$ has edges $e_1, e_2, \cdots, e_m$ incident on it, we replace v by a polygon with vertices $v_1, \cdots, v_m$ and edges $(v_1, v_2), \cdots, (v_m, v_1)$; such that for $1 \leqq i \leqq m$, e_i becomes incident to v_i (if the degree of v is 2, the polygon is a face of length two). We will refer to the new graph by $\hat{G}$, and the set of faces enclosed by the new polygons will be called $\hat{D}$.

Let $G'=(V', E')$ denote the dual graph of $\hat{G}$. The vertices of G' will be called points. The set of points of G' corresponding to $\hat{D}$ will be called D'. Now our problem becomes that of testing whether G' contains a set X of points such that

(i) $X \cap D' = \varnothing$,
(ii) X dominates D',
(iii) $|X| \leqq k$.

If such a set X exists, the corresponding set of faces $\hat{G}$ will be called a face cover.

Let S denote the subgraph of G' induced by all points in D' and all points adjacent to some point in D'. Now if S has more than k connected components, then certainly no set X satisfying (i)–(iii) exists. Hence assume otherwise. We have the following result.

LEMMA 1. *If a set X satisfying* (i)–(iii) *exists, the diameter of every connected component of S is at most $O(k)$.*

Proof. Aiming for a contradiction, let C be a connected component of S with diameter larger than $8k+6$, and let $p=f_1, f_2, \cdots, f_j$ be such a diameter. There are two cases.

(i) $|p \cap D'| \geqq 3(k+1)$. By construction, no point of D' is adjacent to another point of D'. Let $p \cap D' = \{s_0, s_1, \cdots, s_t\}$ with labeling to reflect the ordering of these points in p, and set $Z=\{s_0, s_3, s_6, \cdots, s_{3i}, s_{3(i+1)}, \cdots\}$. Then at least $|Z|$ points are needed to dominate Z. But this is a contradiction since $|Z| \geqq 3(k+1)/3 = k+1$.

(ii) $|p \cap D'| < 3(k+1)$. By construction, every point of $S-D'$ is adjacent to at least one point of D'. Let $p-D'=\{g_0, \cdots, g_r\}$ with labeling to reflect the ordering of these points in p, and set $Y=\{g_0, g_5, g_{10}, \cdots, g_{5i}, g_{5(i+1)}, \cdots\}$. For each i, set d_i to be an arbitrary point of D' adjacent to g_{5i}. Clearly, if $i \neq j$, then $d_i \neq d_j$. Further, a different point of $S-D'$ is required to dominate each point d_i. But the number of such points is at least $\frac{1}{5}(8k+7-3k-2)=k+1$, again a contradiction. □

Our algorithms will exploit this bounded dual diameter structure. The computations take place in the primal graph. Now, if a graph has dual diameter t, then every face is within distance t of an arbitrary face. Algorithm XTND given below will, when input an embedded planar graph H with n vertices, distinguished subset of faces E, and a constant L, test in linear time whether every face of H is within distance L of the outer face. If so, XTND will compute a minimum face cover of E in time bounded by $2^{O(L)}n$. In order to motivate XTND, we will describe it in three steps. First we consider a special type of planar graph called a Halin graph. Next, we analyze the structure of bounded diameter planar graphs. Finally, we describe XTND in the general case.

2.1. Halin graphs. An embedded planar graph H is called a *Halin graph* if its dual has a dominating set of cardinality one. Assume that the corresponding face of H is the outer face. Without loss of generality, the outer facial cycle C of H is simple. For if v is a multiple vertex of C, let $e_1=(u_1, v)$ and $e_2=(v, u_2)$ be two consecutive edges of C. Then we subdivide e_1 and e_2 by adding vertices w_1 and w_2, and add the edge (w_1, w_2) embedded in the outer face. Clearly the new graph is still Halin. This type of operation is called a *patching*. Patchings can also be used to ensure that all vertices in C have degree two or three. Similarly, if an interior vertex of H has degree one, we can shrink the corresponding edge. The final graph H' we obtain will be the union of a cycle C and forest T embedded inside C with all leaves in C. Moreover, from a face cover problem in H we obtain an equivalent problem in H'. Now we need some definitions adapted from [CNP].

(1) A *level* 1 *fan* of H' is a maximal set of paths $p_1, p_2, \cdots, p_n$ with a common endpoint $u \notin C$, which are otherwise disjoint, with opposite endpoint in C, and all interior vertices of degree 2 in H'. The vertex u will be called the *center* of the fan.

(2) To define *level t fans*, for $t>1$, we proceed as follows. As above, let $p_1, \cdots, p_m$ be a maximal set of paths with common endpoint $u \notin C$, otherwise disjoint, and with degree two interior vertices. Suppose that for $1 \leqq i \leqq m$, the endpoint of p_i different from u is the center of a level j_i fan, with $j_i \leqq t-1$, and that for some i, $j_i = t-1$. Then the collection of paths and fans is called a level t fan, and u its center. The fan whose center is the endpoint of p_i is called the *descendant fan* of p_i.

(3) If H' is the union of a level 1 fan and C, we say H' is a *wheel.*

THEOREM 2 (Adapted from [CNP]). *The Halin graph H' has at least one level* 1 *fan, and this fan can be constructed in linear time.* □

This theorem was used in [CNP] to construct a polynomial-time dynamic programming algorithm for the traveling salesman problem in Halin graphs. We will make a similar use here towards the face cover problem.

The intuition behind the approach is the following: we can describe the properties of a fan using a bounded size list. This works because a level 1 fan has only two faces that share edges with other fans. When constructing the list for a level t fan, we only need to look separately at the lists for the descendant fans. Thus if F is a level t fan with m descendant fans, the list for F can be constructed in $O(m)$ time. Altogether this translates into a linear time algorithm for solving the minimum face cover in H'. Details are provided next.

If the forest T located inside C is not a tree, we reduce this to the single tree case as follows. Let e_1, e_2 be consecutive edges incident on vertices of C, that belong to different trees. Then subdivide e_i by introducing a new vertex w_i, for $i=1, 2$ and add the edge (w_1, w_2). This new edge, together with the position of e_i between w_i and C, $i=1, 2$, and a segment of C, bounds a "new" face of H'. We make this face forbidden (that is, we cannot use it towards a cover).

Clearly this procedure does not change the problem and repeatedly applying it will merge the forest into a single tree. Now if the outer face of H' is in E, then any internal face of H' will cover it. On the other hand, if the outer face of H' is not in

E, then it will cover every other face of H', and this is obviously optimal. Hence we may assume, without loss of generality, that the outer face is forbidden.

Suppose f is a fan of H' with center u and paths $p_1, \cdots, p_m$ in clockwise order. For $1 \leqq i \leqq m$ let $\hat{p}_i$ be the *continuation* of p_i; that is, a path of the form p_i, p'_i ending in C, such that p'_1 is obtained by following the most counterclockwise path out of p_1, and for $i > 1$, p'_i is obtained by following the most clockwise path out of p_i. The endpoints of $\hat{p}_1$ and $\hat{p}_m$ in C are called the *extremes* of f. For $2 \leqq i \leqq m$, denote by f_i the region bounded by $\hat{p}_1$, $\hat{p}_i$ and the corresponding section of C; and set E_i to be the subset of faces of E contained in f_i. For $i = 1$, f_1 consists of p_1 and its descendant fan f, while E_1 is the subset of E contained in f. Finally, define $l(f, i, a_1, a_i)$ to be the minimum number of internal faces needed to cover E_i, with the constraints that:

(1) If $a_1 = 0$, at least one face of E_i that has an edge on $\hat{p}_1$ has not been covered.

(2) If $a_1 = 1$, (1) does not apply and at least one face with an edge of $\hat{p}_1$ is used in the cover.

(3) If $a_1 = 2$, neither (1) nor (2) apply.

(4) Similar considerations apply for $a_i = 0$, 1 or 2.

Algorithm XTND takes as input a graph H' and keeps track of an auxiliary graph A. At the start, $A = H'$. In general, the vertices of A in its outer facial cycle will correspond to fans of H'; these vertices will be labeled by the corresponding fans. The output of Algorithm XTND is M, the minimum number of faces of H' needed to cover E. It is well known that repeatedly shrinking fans in a Halin graph eventually leads to a wheel.

Algorithm XTND (Halin Case).

(1) Find a level 1 fan g in A. Let f be the corresponding fan in H', with center u, defining paths $p_1, \cdots, p_m$ and continuations $\hat{p}_1, \cdots, \hat{p}_m$. For $1 \leqq i \leqq m$, let h_i be the descendant fan of p_i, with s_i defining paths (if h_i is a vertex, set $s_i = 0$).

(a) Set $l(f, 1, x, y) = l(h_1, s_1, x, y)$ for all x, y.

(b) For $i > 1$, suppose first that the face between p_{i-1} and p_i is not in E, and it is not forbidden. Then, for example,

$$l(f, i, 2, 2) = \min\{\min_{x,y}\{l(f, i-1, 2, x) + 1 + l(h_i, s_i, y, 2)\}, \min_{x>0, y>0}\{l(f, i-1, 2, x) + l(h_i, s_i, y, 2)\}\}.$$

Similar formulas are used to compute all other parameters $l(f, i, \cdot, \cdot)$, and also when the face between p_{i-1} and p_i is forbidden, or if it is in E.

(2) If g is a wheel in A, then let x be the face of H' between $\hat{p}_m$ and $\hat{p}_1$.

(a) If $x \notin D$, then $M = \min\left\{\min_{x,y}\{l(f, m, x, y) + 1\}, \min_{x>0, y>0}\{l(f, m, x, y)\}\right\}$.

(b) If $x \in E$, then $M = \min\left\{\min_{y}\{l(f, m, 1, y)\}, \min_{x}\{l(f, m, x, 1)\}\right\}$.

(c) Stop and output the value of M.

(3) Otherwise, shrink g into a single vertex in the outer face of A, and go to (1).

Algorithm XTND clearly works correctly, and since the workload in finding and shrinking fans in A is linear in the size of each fan, the total complexity is linear.

Notice that there is a last center r of a fan found in graph A. In fact, it is not difficult to see that this vertex may be prescribed before running XTND, by choosing fans with center $u \neq r$ whenever possible.

There are two observations concerning the Halin case that will be very useful later. First, let f be a fan of H' with center u, and consider a set of contiguous faces of f; that is, a set S of faces of f incident to u that appear consecutively as we travel around f. Then if we want to force all faces in S to take the same *value* (that is, all chosen or rejected), Algorithm XTND given above can still be used, almost verbatim, to compute a minimum face cover of E. Similar considerations apply if all faces of S are in E and covering any of them is interpreted as covering *all* of them. In both cases we will refer to the set S as a *split face.*

Second, suppose e_1 and e_2 are edges incident to C that appear consecutively as we travel around it. Then either at some point of the algorithm e_1 and e_2 will be part of consecutive paths in a fan, or they will be part of the first and last paths in the very last fan (a wheel) considered by the algorithm. In any case, let u be the center of the fan, and let x be the face of H' bounded by C and the paths containing e_1 and e_2. Then we can split x by adding arbitrary edges incident to u. By making the set of additional faces a split face, we obtain an equivalent problem. This new problem is easily solved by using the same sequence of fans as before. The vertex u will be called the ancestor of e_1 and e_2. This concludes the analysis of the Halin case.

2.2. Structure of bounded dual diameter graphs. We next investigate the structure of planar graphs of dual diameter bounded by a certain constant, as it pertains to our problem. Intuitively, our approach is as follows (this description is slightly incorrect as we describe later). Given a plane graph of dual diameter at most L, by consecutively "peeling" away layers of faces at a given distance from the outer face we will reach a "central" graph after at most L layers. This central graph must be Halin; we use a version of the algorithm in § 2.1 where we now consider fans that are extended with gridlike graphs with at most L rows, to solve the minimum face cover.

This description is incorrect in that there may be more than one "central" graph; as we peel layers the graph may decompose arbitrarily. We deal with this difficulty by using a special partial order on the components that we encounter recursively, and proceed essentially as outlined in the previous paragraph. [Ba] introduces a class of planar graphs called k-outerplanar. If a graph is k-outerplanar its dual diameter is at least k; both concepts are somewhat related (in turn, the radius r of the graph [RS] satisfies $k-1 \leqq r \leqq k$ and these two parameters are closely related).[1] [Ba] describes an algorithm for decomposing k-outerplanar graphs. Our procedure UNWRAP for peeling a bounded dual diameter graph is reminiscent of the one in [Ba], with some important differences which are necessary to make our face cover algorithm work.

Let H be an n-vertex plane graph of dual diameter L, with outer facial cycle C. Procedure UNWRAP proceeds as follows. First, any vertex of degree two and its two incident edges can be replaced by a single edge. Also, the patch operation allows us to assume that C contains no cutvertices. Further, if e is an edge incident to a vertex of C, we can assume that both endpoints have degree three (using patchings or expanding the endpoints into polygons, which will later be used as forbidden faces). Next, we delete C, together with all edges incident to it. Then H will be split into several connected components, each of which is a union of trees and maximal two-connected graphs, joined in a treelike manner. If two of the two-connected graphs share one vertex (a cutvertex) we can use the patch operation to obtain a larger graph. Assume we carry this out as many times as necessary. The final two-connected graphs

[1] In polynomial time, one can minimize over all embeddings the radius, the maximum dual distance to the outer face, and the outerplanarity. However, minimizing the dual diameter is NP-hard [BM].

will be called the *height*-1 *islands* (see Fig. 1). Clearly, if X is such an island, the distance from an internal face of X to the outer face of X is at most $L-1$. Finally, it is not difficult to see that a vertex in the outer face of X may be assumed to be adjacent to at most one vertex not in X. Now we can recursively use UNWRAP with each height-1 island to obtain height-2 islands. The procedure will terminate with at most $L-1$ recursive calls. The top islands found (all of height at most $L-1$) will be Halin graphs. The set of islands constitutes a partial order, which is constructed by UNWRAP in time $O(n)$. Having peeled away all of the layers we put them back together while preserving the modifications that were introduced (i.e., all of the subdivisions and new edges). It is in this graph H^*, rather than H, that we apply XTND, after a few more modifications described in the next section.

2.3. General case of XTND. We need one more piece of notation. The edges joining the outer face of a height-i island I to the outer face of the height-$(i-1)$ island enclosing I are called the *links* of I. Notice that if u is an endpoint of a link, then u has degree at most four, by the construction used. The edges of H^* that are not links are called layer edges. Let R be the maximum dual distance to the outerface.

We first consider the simplest possible case, which we call the *concentric case.* This arises when in every call to UNWRAP we discover precisely one island (and no trees). That is, there is exactly one height-i island for each $1 \leqq i \leqq R-1$ (see Fig. 2(a)). Let K denote the height-$(R-1)$ island.

We modify H^* as follows, if necessary. Let C^* be the outer facial cycle of H^*. Then by subdividing layer edges and introducing some new link edges and edges inside K, we can assume that every link edge is contained in a (unique) path from C^* to the interior of K, of length R, and similarly, every vertex in the outer facial cycle of K is contained in such a path. Notice that the new edges will split faces, but all members of a split face are "consecutive". For convenience, we still refer to the graph by H^*. H^* has $O(Rn)$ vertices. See Fig. 2(b), 2(c).

The final problem we obtain will have split faces, but an extremely simple structure. This structure allows us to essentially use the same Algorithm XTND given before, with the fan structure of K driving the computation. The main difference lies in that, for every fan f of K, we simultaneously consider the entire "grid" of faces stretching

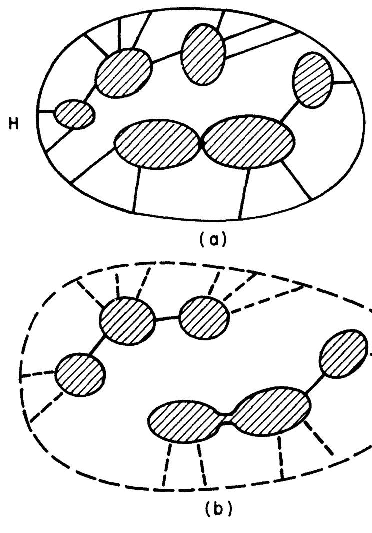


FIG. 1

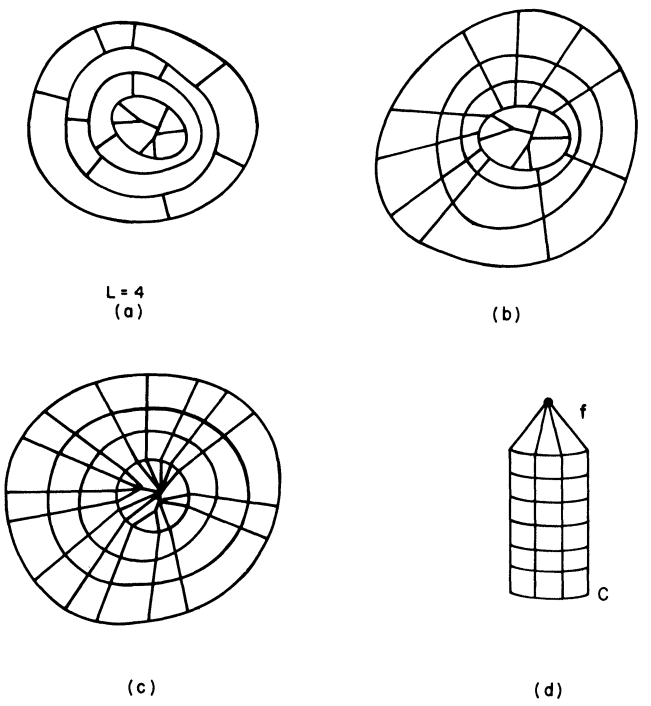


FIG. 2

from f to C (see Fig. 2(d)). Such a grid g will have arbitrarily many faces; however, the "left" and "right" paths of it are incident to at most $O(L)$ faces. The dynamic programming recursion will consider all possibilities for each such face (whether chosen or rejected for the cover, and in or not in the set E) in addition to the usual recursion for the fan f. Clearly there will be at most $2^{O(L)}$ possible states. Further, if a face f_1, adjacent to the left path in g, and a face f_2, incident to the right path in g, actually correspond to the same split face, then we need only consider states of the dynamic program where f_1 and f_2 take the same value. It is easy to see that the overall procedure takes time at most $2^{O(L)}n$. This ends the description of the concentric case.

The general case is only slightly more complex, but we need to develop a bit more machinery. We essentially construct a partial order on the various islands, and solve a sequence of problems moving upwards in the partial order. The last (i.e., topmost) problem to be solved will be of the concentric type described above.

As before, let H denote the overall graph, with outer facial cycle C. Choose an arbitrary height-$(L-1)$ island K, with outer facial cycle C'. Let e_1, e_2 be two consecutive links joining C' to the height-$(L-2)$ island enclosing K, then as in the concentric case, by splitting faces we can assume that for $i=1, 2$, e_i is contained in a length $R-1$ simple path from C' to C (notice that the length restriction says that p_i does not unnecessarily cut through islands). Then the wedge of e_1, e_2 is the subgraph of H bounded by p_1, p_2 and the corresponding segments of C', C. The wedges of K are constructed for all consecutive links (see Fig. 3(a), 3(b)).

Proceeding recursively, let W be a wedge of some island; and J be a highest island enclosed by W, say J of height i. The boundary of W will be made up of two paths p_l, p_γ, a segment of C and a segment x of at most two layer edges. Notice that at most one link edge e joins J to x. By face splitting we can guarantee that e exists.

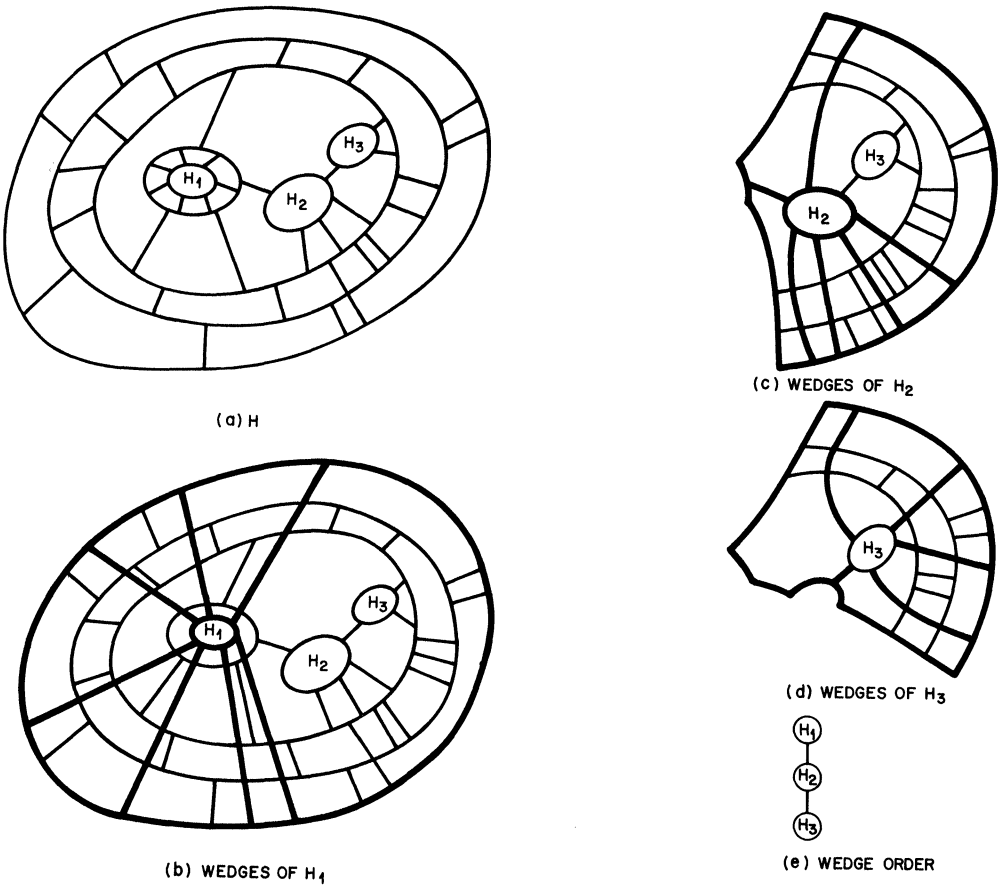


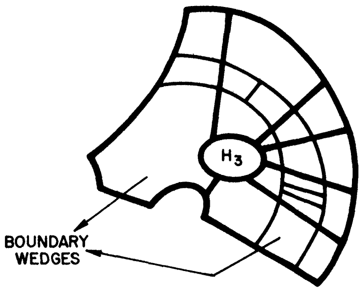
FIG. 3

Now we construct the wedges of J by including the link edges incident to vertices of the outer face of J in paths of length i to C, contained in W. The only exception is the edge e which will define two special wedges called the boundary wedges. This procedure is repeated recursively until we have computed wedges for all islands; trees are handled in a similar way (see Figs. 3(c) and 3(d)). Notice that this procedure constructs a partial order on a subset of islands and trees, with K at the top. We call this order the *wedge order* of H (see Fig. 3(e)).

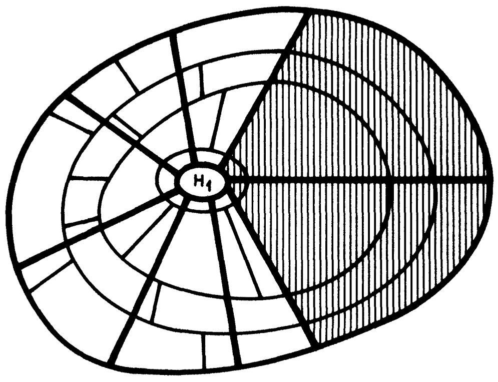
One fact is worth pointing out: through additional face splitting if necessary, for every wedge W of a height-i island, the number of faces in the two boundary wedges enclosed by W (if any) is altogether $2i$.

Let H^* denote the graph resulting from H after applying all the face splittings. Then H^* contains $O(Ln)$ vertices, since each set of face splitting is caused by some edge or island of H.

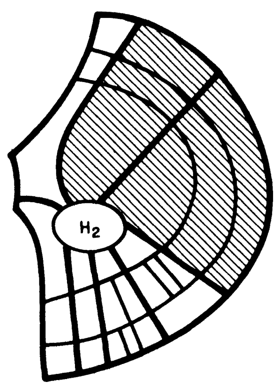
Our Algorithm XTND will operate on H^* by moving up the wedge order. As shown previously, E denotes the set of faces to be covered. Let I be an island at the bottom of the order. I is contained inside some wedge W that belongs to the father of I in the wedge order. Let I' denote the union of I and all its wedges except for the two boundary ones. I' has the structure of a Halin graph with a grid glued to part of its outer face. Then we compute the minimum number of internal faces of I' needed to cover all faces of $E \cap I'$; subject to each possible set of *constraints* corresponding



(f) ANALYZE H_3



(h) REPLACE WEDGE OF H_1 CONTAINING H_2 WITH GRID (SHADOWED); ANALYZE



(g) REPLACE WEDGE OF H_2 CONTAINING H_3 BY GRID (SHADOWED); ANALYZE H_2

FIG. 3. (*Continued.*)

to the patterns of the faces in the boundary wedges of I (i.e., whether chosen, left uncovered or rejected; taking split faces into account, there are $2^{O(L)}$ such patterns). Each of the computations is carried out much as in the case for Halin graphs.

Having carried out these computations, we replace W with a grid of width two, remove I from the wedge order and proceed. For the general step, we again pick an island from the bottom of the order and proceed as above. The only difference is that we may encounter wedges of the island containing width two grids that correspond to previously analyzed islands. But we have computed all relevant information for such grids, and it is easy to work this into the dynamic programming recursion. Details are left to the reader.

To analyze the complexity of the algorithm, notice that the total amount of work done on each wedge W is at most $2^{O(L)}|W|$. Hence, the complexity of the overall algorithm is at most $2^{O(L)}n$.

3. Testing k-planarity for a variable embedding. In this section, we return to the original problem of testing whether an n-vertex planar graph $G=(V, E)$ containing a distinguished subset of vertices D, admits an embedding in which D can be covered with k or fewer faces.

If G is three-connected, then we use Algorithm XTND of § 2, after expanding D into polygons. We show later how to reduce the one-connected case to that of two-connectivity. In what follows, we will therefore assume G is two-connected.

The basic approach consists of decomposing G into (roughly) its triconnected components [HT1]. These components are then inductively assembled into G, using dynamic programming. Next we will give some definitions.

(1) A *block* of G is a connected subgraph H of G with two distinguished vertices u_1 and u_2 with the property that either $G=H$, or all paths from $H-\{u_1, u_2\}$ to $G-H$ must pass through u_1 and u_2. The vertices u_1 and u_2 will be called the *extremes* of H. Notice that since G is two-connected, then in any embedding of G, u_1 and u_2 will be in a common face.

(2) Let H be a block of G with extremes u_1 and u_2. Let $xyzw$ be a binary vector of length four. Define $F(H, xyzw)$ to be the minimum number of internal faces of H needed to cover all vertices of D in H, taken over all embeddings of H with u_1 and u_2 in the outer face, with the following constraints:

(i) If $x=y=1$ then there exist vertices v_1 and v_2 in $D\cap H$ with $v_1 \neq v_2$ and $v_i \neq u_j$ for all i, j, that are *not* covered by the chosen internal faces of H, and such that v_1 is in one path from u_1 to u_2 in the outer face of H, and v_2 in the other (in an optimal embedding that attains $F(H, 11zw)$).

(ii) If $x+y=1$, exactly one of the paths from u_1 to u_2 in the outer face of H contains an uncovered vertex $v \in D\cap H$, with $v \neq u_1, u_2$.

(iii) If $x=y=0$, then all vertices of $D\cap H$, with the possible exception of u_1 or u_2, are covered.

(iv) If $z=0$ (resp., 1), then $u_1 \in D$ is (resp., is not) covered.

(v) If $w=0$ (resp., 1), then $u_2 \in D$ is (resp., is not) covered.

(If, $u_1 \notin D$, then we always set $z=0$; similarly for $u_2 \notin D$. An embedding that attains $F(H, xyzw)$ will be denoted by $E(H, xyzw)$.)

Notice that if in an optimal embedding of G, we use $E(H, xyzw)$ to embed H, where $x+y \geqq 1$ and $z+w \geqq 1$, then any face of $G-H$ used to cover the uncovered vertices of H different from u_1 and u_2 will also cover u_1 and u_2. Hence, for $x+y \geqq 1$, we reset $F(H, xy00) = \min_{z,w} F(H, xyzw)$. Therefore, the only 4-vectors we need to keep track of are (0000), (1000), (1100), (0010), (0001) and (0011).

Every planar graph admits a recursive decomposition into blocks. This decomposition can be represented by a rooted tree, where each vertex corresponding to some block of G, with the root representing G, and the leaves (essentially) correspond to the triconnected components of G. For each block appearing in the tree, one of three canonical ways of decomposing it occurs: a "Series" Case, a "Parallel" Case and a "Messy" Case. Our Algorithm DYN for testing k-planarity proceeds upwards from the leaves in the decomposition tree by computing the F parameters. Once a block B has been analyzed, it is replaced in its parent by a small (bounded size) gadget (and we keep track of the F parameters of B to compute those of its parent).

In order to simplify the description, we will present together Algorithm DYN and the recursive decomposition structure. Further, the algorithm will be described recursively (i.e., proceeding from the root down). The complexity is analyzed later.

Now suppose $\{u_1^*, u_2^*\}$ is an arbitrary cutset of G. Then in any embedding of G, u_1^* and u_2^* will be in at least one common face, without loss of generality, the outer face of G. Then testing k-planarity of G is achieved by regarding G as a block with extremes u_1^* and u_2^*, and computing

$$\min\{F(G, 0000), 1+\min\{F(G, xyzw)\colon x+y+z+w \geqq 1\}\}.$$

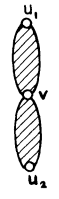
ALGORITHM DYN.
Input: a block H with extremes u_1, u_2
Output: all quantities $F(H, xyzw)$

There are three cases.

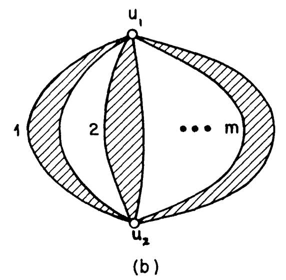
(1) *Series Case.* H has a cutvertex v. Then $v \neq u_i$, $i = 1, 2$, and we write $H = H_1 \cup H_2$, where $u_i \in H_i$, $i = 1, 2$, and $H_1 \cap H_2 = v$ (see Fig. 4(a)). It is easy to compute the F parameters for H in constant time from those of H_1, H_2. Refer to Appendix A(1) for details.

(2) *Parallel Case.* Fix z and w. H is two-connected and $\{u_1, u_2\}$ is a cutset of H. Write $H = \bigcup_{j=1}^{m} H_j$, where $H_j \cap H_k = \{u_1, u_2\}$ for each $j \neq k$, and each H_j is a block with extremes $\{u_1, u_2\}$ (see Fig. 4(b)).

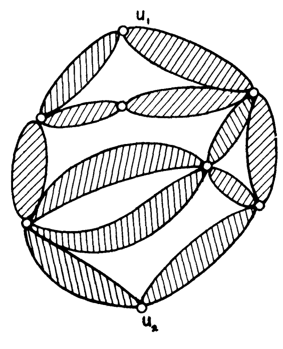
Intuitively, we proceed as follows. Suppose we select two blocks to act as "leftmost" and "rightmost." In Appendix A(2) we show that it is possible to select either a "best" embedding for each of the remaining "internal" blocks, or else a simple tie situation may arise. In either case we are allowed to essentially ignore the detailed structure of the internal blocks, and there is an efficient (linear time) procedure to pair them up and obtain an optimal embedding (permutation and rotations of the internal blocks). Further, we are always able to select a "best" leftmost and rightmost block. The overall procedure runs in time linear in m. Details are provided in Appendix A(2).



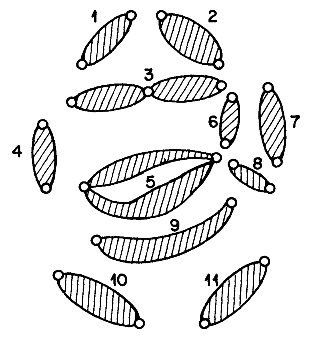
(a)



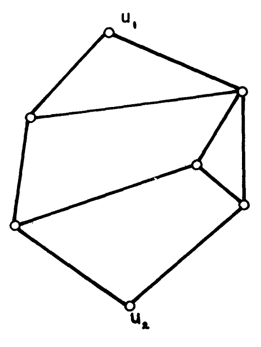
(b)



(C.1) SHADOWED GRAPHS ARE THREE CONNECTED



(C.2) THE 11 M.M.B.'S CORRESPONDING TO (C.1)



(C.3) GRAPH OBTAINED BY REPLACING EACH M.M.B. WITH AN EDGE

FIG. 4

(3) *Messy Case.* Cases (1) and (2) do not apply (see Fig. 4(c.1)). Then either (i) there is a cutset $\{v, w\}(\neq \{u_1, u_2\})$ of H, or (ii) H is three connected. In the latter case, we can apply the results of § 2 (by using forbidden faces, if necessary). So assume (i) occurs.

Now for every cutset $\{v, w\}$ of H we can define the main block of $\{v, w\}$ as the union of all connected components of $H-\{v, w\}$ not containing u_1 or u_2, together with $\{v, w\}$, if any such components exist. Now if for all cutsets $\{v, w\}$ the main block is empty, it is easy to see that H has a unique embedding with $\{u_1, u_2\}$ restricted to lie in the outer face, and therefore we are essentially in alternative (ii). Otherwise, the main blocks can be ordered by inclusion; call those at the top the maximal main blocks (see Fig. 4(c.2)), or m.m.b.'s for short. If we replace every m.m.b. by an edge, the resulting graph H' has a unique embedding with $\{u_1, u_2\}$ restricted to lie in the outer face (Fig. 4(c.3)).

Our strategy is to analyze each m.m.b. W recursively first. Next we replace W by an appropriate small gadget, attached to the rest of H by the extremes of W. We want the gadget to "retain" all the relevant properties of W, while having essentially a unique embedding (it will turn out that either we can find a "best" embedding for W, or there are a few ties; the gadgets we use will take care of either case). In this manner, we will reduce the problem on H to one where the embedding has been prescribed, and we can use the results of § 2. In summary, the approach is as follows:

(a) Analyze every m.m.b.

(b) For every m.m.b. W, compute its best embedding(s) and replace W with a small gadget. We will keep track of a *weight* corresponding to W that describes the cost of "internal" faces of W.

(c) Compute each parameter F for the resulting graph $\hat{H}$, and to each, add the sum of the weights of the m.m.b.'s to obtain the F parameters of H.

Details are provided in Appendix A(3). The overall complexity of this case will be, at most, $2^{O(k)}|\hat{H}|$.

This concludes the description of Algorithm DYN. An efficient implementation of DYN would first compute the block decomposition. This is accomplished by using the algorithm in [HT1] to decompose a graph into its "split components". This algorithm is easily modified to output the decomposition tree in linear time, with each block labeled series, parallel or messy (in the messy case the decomposition is into m.m.b.'s). If v is a vertex of the tree that represents a block W, at v we store the graph W' resulting from W by replacing each of its children with an edge. The overall space required is linear. Finally, we "force" the original extremes u_1^*, u_2^* chosen for G to lie in the outer face by adding at the start the edge (u_1^*, u_2^*) if necessary.

Having computed the decomposition, we proceed upwards with DYN. If W' is the graph stored at some vertex of the tree, the work involved in analyzing W' (when we reach it) is $O(|W'|)$ in the series and parallel cases, and at most $2^{O(k)}|W'|$ in the messy case. Thus we conclude as follows.

LEMMA 3. *The complexity of testing whether G is k-planar with* DYN *is at most $O(c^k n)$, where c is independent of n.* □

Next we pass to the one-connected case.

A one-connected graph G is the union of *bricks*; that is, edges or 2-connected subgraphs that intersect at cutvertices only. G will have a "treelike" structure; namely, if a path leaves a brick B through a cutvertex v, the only way the path can return to B is by crossing v again. Now suppose $G=(V, E)$ has connectivity one; let v be an arbitrary cutvertex, and write $G=\bigcup_{i=0}^{m} G_i$, where $G_j \cap G_k = v$, for all $j \neq k$, and each $G_i - v$ is nonempty and connected. Let $D_i = D \cap G_i$.

If $v \in D$, then v will be covered by some face f. Now we can always embed every G_i simultaneously so that f is a face of it without requiring more than the minimum number of faces to cover all of D. Thus, if

$$\hat{k} = \text{planarity number of } G$$

and

$$\hat{k}_i = \text{planarity number of } G_i, \text{ for } i = 0, \cdots, m \text{ (where we cover } D_i\text{)},$$

then

$$\hat{k} = \sum_{i=0}^{m} \hat{k}_i - (m-1),$$

and the problem decomposes with v in each D_i.

Now suppose $v \notin D$. By not including v in D, we can compute $\hat{k}_i$ and use $\sum_{i=0}^{m} \hat{k}_i$ as the number of faces to cover D. If we use $\bar{D}_i = \{v\} \cup D_i$ instead of D_i for a given graph G_i, we might increase the planarity number of G_i by 1. However, if we add v to D and compute the planarity number of G as before, we might actually decrease the number of faces required to cover D because the graphs G_i will be able to share the additional face. Hence, if we try both possibilities (i.e., whether or not to add v to D), and decompose into a problem for each G_i, we will certainly solve the original problem. However, this may create too much work if a brick of G contains too many cutvertices. Instead we will use an approach inspired by the next lemma.

LEMMA 4.

$$\hat{k}_{i,1} = \textit{planarity number of } G_i \textit{ to cover } D_i.$$

$$\hat{k}_{i,2} = \textit{planarity number of } G_i \textit{ to cover } \bar{D}_i.$$

Then

(a) *If, for at least one* $i \geqq 1$, $\hat{k}_{i,2} = \hat{k}_{i,1}$, *then it is optimal to add* v *to* D.

(b) *If, for all* $i \geqq 1$, $\hat{k}_{i,2} = \hat{k}_{i,1} + 1$, *then it is optimal not to add* v *to* D.

Proof. The proof is clear. □

Notice that G_0 is not considered in the lemma.

Next we proceed as follows. Let G be a graph whose k-planarity we want to test; exactly one brick B of G has been painted red. Let $v \in B$ be a cutvertex of G. If $v \in D$, then proceed as outlined above. If $v \notin D$, then (as previously) we form the graphs G_i, so that B is contained in G_0, and we paint red the brick of G_i for $i \geqq 1$, that contains v. Next, we solve, in each G_i for $i \geqq 1$, the two problems to cover D_i, and to cover $\bar{D}_i$; we then use Lemma 4 to decide whether or not to add v to D. Notice that each graph G_i contains exactly one red brick. Clearly, repeating this procedure will eventually reduce the problem in G to at most two separate problems in each brick. Thus the complexity is again linear.

4. NP-completeness for arbitrary k.

In this section we show that the problem of determining whether a (G, D) pair is k-planar or not is strongly NP-complete when k is variable. Hence, it is unlikely that a polynomial-time algorithm exists for recognizing a k-planar graph for arbitrary k.

In order to do this, we define the following decision problem which we call FACE COVER: "Given a planar graph $G = (V, E)$ together with a subset of vertices $D \subseteq V$ and an integer K, can G be embedded in the plane so that at most K faces are required to cover all of the vertices in D?" We will show that FACE COVER is strongly NP-complete even when G is three-connected and so has essentially a unique planar embedding, when $d = \theta(n)$, and when all of the faces of G have bounded length. A graph G is 3-connected if the removal of any two vertices of G leaves the rest of G connected. It is well known [Wh] that a 3-connected planar graph has an embedding which is essentially unique; i.e., all embedding have the same facial structure and differ only in which face is the outer face.

The reduction will be from VERTEX COVER: "Given a graph $G = (V, E)$ and integer L, is there a subset of vertices $W \subseteq V$ with $|W| \leqq L$, such that for every edge $(u, v) \in E$ at least one of u or v belongs to W?" VERTEX COVER is strongly NP-complete for cubic planar graphs [GJS].

THEOREM 5. FACE COVER *is strongly* NP-*complete even when* G *is* 3-*connected*, $d = \theta(n)$, *and all faces of* G *have bounded length.*

Proof. Consider an instance of VERTEX COVER given by a cubic planar graph $G = (V, E)$ and integer K, where G has no loops or parallel edges and every edge is contained in exactly two faces. This problem is known to be strongly NP-complete [GJS].

We obtain an instance of FACE COVER by setting $K = L$ and $\bar{G} = (\bar{V}, \bar{E})$ being the planar dual of G, i.e., place a vertex $v \in \bar{V}$ in every face f_v of G, and an edge $(u, v) \in \bar{E}$ if faces f_u and f_v of G share an edge. Subdivide each edge $(u, v) \in \bar{E}$ by adding a vertex $x(u, v)$ and denote the new graph by $\hat{G} = (\hat{V}, \hat{E})$. Let D be the set of vertices $x(u, v)$ for $(u, v) \in \bar{E}$. Given the particular embeddings of G and $\hat{G}$, there is clearly a one-to-one correspondence between vertices in D and edges of G, and similarly, between faces of $\hat{G}$ and vertices of G. Thus, there is a one-to-one correspondence between a face cover of $(\hat{G}, D)$ and a vertex cover of G of the same cardinality.

In general, $\hat{G}$ might have other planar embeddings where a face cover in this new embedding of $\hat{G}$ does not correspond to any vertex cover in G. To remedy this, we form the graph $G^* = (V^*, E^*)$ from $\hat{G}$ by adding, inside each face of $\hat{G}$, edges to form a cycle containing all vertices of D in that face, and embed the cycle inside the face. It is easy to see that the face cover problem on G^* is equivalent to the face cover problem in $\hat{G}$. We will show that G^* is 3-connected and so this embedding is essentially unique [Wh] which will complete the proof. To see that G^* is three-connected, notice that in $\bar{G}$, all facial cycles consist of precisely three edges. However, these cycles may not be simple (i.e., triangles) if G has bridges. Nevertheless, in G^* all facial cycles have length three *and* are simple, in other words G^* is triangulated and, thus, three-connected, since G^* is a simple graph.

Finally, note that $|D|$ equals the number of edges of $\bar{G}$, which is $\theta(n)$. □

(Note: An alternative proof of NP-completeness for the case $D = V$ appears in [FHS].)

5. An exact and an approximate algorithm for fixed embeddings. Consider the variation of FACE COVER in which we are forced to use a given embedding. That is, FIXED EMBEDDING FACE COVER (FEFC): "Given an embedded planar graph $G = (V, E)$, an integer k, and a subset $D \subseteq V$, can D be covered with at most k faces?" Clearly FEFC is NP-complete.

In this section we will present:

(a) A set of transformations for FEFC that allow us to assume that $d = \theta(n)$, while not increasing the maximum facial length.

(b) An exact algorithm for FEFC that runs in time $2^{O(\sqrt{n})}$.

(c) A polynomial-time approximation algorithm for FEFC that is asymptotically optimal if G has bounded length facial cycles, as is the case in our NP-completeness proof. (Our basic approach is similar to that of [LT] in that we use the planar separator theorem.)

The following set of transformations can be visualized as simplifying the problem. Their main objective is to allow us to assume that $d = \theta(n)$ without increasing the facial cycle length and preserving the value of the optimal solution. These transformations are crucial for the proof of Theorem 6 presented later.

(1) Every vertex of $V - D$ of degree one is shrunk into its neighbor.

(2) Every facial cycle of length two (resp. one) is shrunk into a single edge (resp. vertex).

(3) In every face g with at most one vertex in D, all vertices in $V - D$ are shrunk into a single vertex.

(4) Every vertex v in $V - D$ of degree two, adjacent to vertices u and w, is deleted, and edges (u, v) and (v, w) are replaced by the edge (u, w).

(5) Any two vertices of $V - D$ that are adjacent are shrunk into a single vertex.

(6) For every loop $e = (v, v)$ such that (say) the subgraph contained in the interior of e has an outer facial cycle consisting of v and vertices of $V - D$ only, e is deleted.

(7) If a vertex $v \in D$ is contained in a unique face of f, then f must appear in any cover; hence, we delete v, and remove from D all members in f.

This concludes the list of transformations. Clearly, Transformations (1)-(7) can be applied until no longer possible, in polynomial time, to obtain an equivalent problem.

Theorem 6 given below places a lower bound on the size of D in a loop-free graph where none of Transformations (1)-(7) can be applied. However, if we apply (1)-(7) to a graph G, the resulting graph G' may contain loops. In order to count vertices of D in G', we modify it as follows:

(a) Every loop (v, v) with $v \in D$ or v adjacent to at least three vertices of D is deleted.

(b) Otherwise, let $e = (v, v)$ be a loop, and let x be the only neighbor of v in the interior of e, with $x \in D$. Now, (7) or (2) cannot be applied; hence, the interior of e contains vertices other than x. But there must be at least one such vertex $w \in D$ such that (w, v) can be added while preserving planarity; this is true because of (a), and the fact that neither (3) nor (6) can be applied. In that case, (w, v) is added. Now v has three neighbors in D.

It is easy to verify that after applying (a) and (b) to G', the resulting graph G'' is such that none of the transformations (1)-(7) can be applied, and that G'' is loop-free.

THEOREM 6. *Let $G = (V, E)$ be a loop-free planar graph with a fixed embedding, where $|V| = n$ and $|D| = d$, and suppose that none of the Transformations* (1)-(7) *can be applied. Then $d \geqq (n+4)/3$.*

Proof. Notice that every vertex of $V - D$ has degree at least three and is only adjacent to vertices of D. Suppose that G contains a pair of parallel edges $e = (u, v) = e'$, with say, $v \in V - D$. Then the interior region bounded by e and e' must contain a vertex $w \in D$ with $w \neq u$ such that either v and w are adjacent, or the edge (v, w) can be added to the embedding (in which case we do so). Similar considerations apply to the exterior of e'.

Now, delete all edges with both endpoints in D, let C be an arbitrary connected component with $\hat{n}$ vertices and $\hat{d}$ elements of D. Clearly, C has no facial cycles of length one or two. The latter follows because if $e = (u, v) = e'$ are a pair of parallel edges with $u \in D$, $v \in V - D$, then there exist vertices w_1, $w_2 \in D$ adjacent to v, and located in the interior and exterior of $\hat{e}'$; this is guaranteed by the previous paragraph.

Consequently, the number of $\hat{e}$ of edges of C satisfies $\hat{e} \leqq 3\hat{n} - 6$. On the other hand, if $\hat{f}$ denotes the number of faces of C, $\hat{f} = \sum_{k \geqq 3} \hat{f}_k$, where $\hat{f}_k$ is the number of faces of length k. Since C is bipartite, $\hat{f}_3=0$. Thus, $\hat{e}=\frac{1}{2}\sum_k k\hat{f}_k \geqq 2\sum_{k\geqq 4}\hat{f}_k=2\hat{f}$. Moreover, $\hat{e}\geqq 3(\hat{n}-\hat{d})$, since for each vertex of C not in D, we count at least three edges. Hence,

$$\hat{e}=3\hat{n}-x \quad \text{where } 6\leqq x\leqq 3\hat{d}.$$

By Euler's formula,

$$\hat{f}=2\hat{n}-x+2\leqq\frac{3\hat{n}-x}{2},$$

or

$$\hat{n}+4\leqq x\leqq 3\hat{d},$$

i.e.,

$$\hat{d}\geqq\frac{\hat{n}}{3}+\frac{4}{3},$$

which concludes the proof. □

We note that the bound in Theorem 6 is best possible.

We now present an exact algorithm for finding the minimum number of faces required to cover all special vertices of a planar graph given a fixed embedding in $2^{O(\sqrt{n})}$ time. As in Theorem 7, we may take $d=\theta(n)$ by the application of appropriate transformations in the embedded graph. Let $G=(V,E)$, $D\subseteq V$, $d=|D|$, and $n=|V|$ be the input. The algorithm proceeds as follows:

(1) Let S be an $O(\sqrt{n})$-separator of G. Write $G=G_1\cup G_2$, where $G_1\cap G_2=G(S)$, the subgraph of G induced by S. Let D_i be the subset of S contained in G_i. Write $G_i=(V_i,E_i)$, and embed G_i as it appears in G.

(2) A face f of G that contains vertices of G_1-S and of G_2-S will be called a boundary face. Ideally, we would like to proceed independently with G_1 and G_2. However, these two graphs interact on the boundary faces. Thus, we modify G_1 (and similarly, G_2) so that the boundary face structure is "preserved" in a "legal" way. This is attained in two steps.

(i) For every boundary face f, we replace each path of f that intersects V_1 at its endpoints with an edge. The resulting graph contains a face f_1 that corresponds to f; we call f_1 an inherited face. Select an arbitrary added edge (u,v) of f_1 and subdivide it by introducing a grey vertex $w(f)$. Carry out this transformation for all boundary faces of G_1; call the resulting graph G_1'.

Now we are essentially ready to proceed independently with graph G_1'. The idea is to use the grey vertices to force boundary faces to be used in a cover: if f is such a face and we change the color of $w(f)$ from grey to black, this should force us to use f. However, there is a problem. G_2 may have several connected components and the removal of each component could introduce in G_1' a (possibly large) new face that does not correspond to any face of G. We call such a face a gap face. Each gap face is made up of edges added in (i). We handle this problem as follows.

(ii) In each gap face subdivide every edge by introducing a new white vertex. Connect all such vertices consecutively as we travel around the face with red edges. Call the resulting graph $\hat{G}_1$. Similarly, define $\hat{G}_2$.

(3) Write $\hat{S} = S \cap D$. Next, for every partition $\hat{S} = \hat{S}_1 \cup \hat{S}_2$ with $\hat{S}_1 \cap \hat{S}_2 = \emptyset$, and every subset X of grey vertices, we color X black, the remaining grey vertices white, and solve the following two face cover problems.
   (a) In $\hat{G}_1$, cover $(D_1 - S) \cup \hat{S}_1 \cup X$, with minimum value $k_1(\hat{S}_1, X)$.
   (b) In $\hat{G}_2$, cover $(D_2 - S) \cup \hat{S}_2 \cup X$, with minimum value $k_2(\hat{S}_2, X)$; set $f(\hat{S}_1, \hat{S}_2, X) = k_1(\hat{S}_1, X) + k_2(\hat{S}_2, X) - |Y|$, where Y is the set of inherited faces used in both covers. Then the minimum cover of D has value $\min_{\hat{S}_1, \hat{S}_2, X} f(\hat{S}_1, \hat{S}_2, X)$.

The proof of correctness of the algorithm proceeds as follows.

(1) Consider any face cover of D in G. Then an arbitrary subset of the boundary faces will be used, and if any vertex of $S \cap D$ is not covered by a boundary face, then it is either covered by a face of G_1 that contains no vertices of $G_2 - S$ (an internal face of G_1), or it is covered by a face of G_2 that has no vertices of $G_1 - S$ (an internal face of G_2).

(2) Consider any of the problems on graph $\hat{G}_1$, with $\hat{S}_1$ and X as above. Then, without loss of generality, *none* of the faces containing red edges is ever used in an optimal solution, since the black vertices that any such face may cover are also covered by an inherited or an internal face. Thus, all vertices of X are covered by inherited faces; and all vertices of $\hat{S}_1$ are either covered by inherited faces, or by faces of $\hat{G}_1$ that are copies of internal faces of G_1 (similarly for $\hat{G}_2$). Consequently, for each $\hat{S}_1$, $\hat{S}_2$ and X, we can take the two optimal solutions on $\hat{G}_1$ and $\hat{G}_2$, respectively, and obtain a face cover of D in G of cardinality precisely $f(\hat{S}_1, \hat{S}_2, X)$.

(3) Finally, consider an arbitrary face cover F of D in G. Let F_i be the set of internal faces of G_i used in F for $i = 1, 2$. Let Z be the set of boundary faces used in F. Notice that $|F| = |F_1 \cup Z| + |F_2 \cup Z| - |Z|$. Also, set $\hat{S}_1$ is the set of vertices of $S \cap D$ covered by internal faces of G_1, and $\hat{S}_2 = (S \cap D) - \hat{S}_1$. Let $X(Z)$be the set of grey vertices of $\hat{G}_1$ contained in those inherited faces corresponding to Z. Then $k_i(\hat{S}_i, X(Z)) \leqq |F_i \cup Z|$ for $i = 1, 2$, and thus, $k_1(\hat{S}_1, X(Z)) + k_2(\hat{S}_2, X(Z)) \leqq |F| + |Z|$ which implies that $f(\hat{S}_1, \hat{S}_2, X(Z)) \leqq |F|$.

This concludes the proof of the correctness of the algorithm.

To derive the complexity of the algorithm, the number of edges and vertices added to each graph G_i to obtain $\hat{G}_i$ is $O(|S|)$. Consequently, G_i has at most $\frac{2}{3}n + O(\sqrt{n})$ vertices. Furthermore, the total number of triples $(\hat{S}_1, \hat{S}_2, X)$ is at most $2^{O(|S|)}$. As a result, if $T(n)$ is the worst-case complexity of the algorithm, we have $T(n) \leqq 2^{O(\sqrt{n})} T(\frac{2}{3}n + O(\sqrt{n}))$; from which $T(n) \leqq 2^{O(\sqrt{n})}$ is straightforward.

The approximation algorithm is somewhat similar to the exact algorithm, with the exception that we will use planar separators that produce "equal" size subgraphs. Let $G = (V, E)$ be the input with $D \subseteq V$. Let S be a "50/50" separator of G. Write $G = G_1 \cup G_2$, where each $G_i = (V_i, E_i)$ is defined as in the exact algorithm. Next, add edges to each G_i to obtain the inherited faces, and call the resulting graph $\bar{G}_i$. Notice that the faces of $\bar{G}_1$ correspond to the internal faces of G_1, boundary faces, and also (possibly) to a third type of face that corresponds to connected components of G_2. (The vertices on these faces are all contained in S.) Proceed similarly with G_2. Set $D_i = (D \cap V_i) - S$. Now, suppose we solve the following problems with $i = 1, 2$:

(*) In $\bar{G}_i$, cover D_i; let the optimum cover have size $\bar{k}_i$.

Now, by taking the union of the optimum covers we can obtain a cover of D in G by adding at most $|S|$ faces. Hence, if we denote by k the size of an optimum cover of D in G, we have $k \leqq \bar{k}_1 + \bar{k}_2 + |S|$. On the other hand, given a cover F for G we can obtain a cover for $\bar{G}_1$ and one for $\bar{G}_2$ by restricting F appropriately. Hence, $|F| \geqq \bar{k}_1 + \bar{k}_2 - O(|S|)$, or $|k - \bar{k}_1 + \bar{k}_2| = O(|S|) = O(\sqrt{n})$.

Our algorithm will not solve the problems on $\bar{G}_i$ exactly. Rather, we keep subdividing each graph and modifying the resulting subgraphs until we obtain (on each branch of the recursion) graphs of size $O(\log^2 n)$. We solve these problems using the exact algorithm, and by piecing together their solutions, we will obtain a face cover of D in G. It is not difficult to verify that the problems produced at the ith recursive step have at most $O(n2^{-i})$ vertices. Hence, the total number of recursive levels is at most $T = \log n - 2 \log \log n + O(1)$. Consequently, the total error is, up to a constant, at most

$$\sum_{i=0}^{T} 2^i \sqrt{\frac{n}{2^i}} = O(\sqrt{n} 2^{T/2}) = O\left(\frac{n}{\log n}\right).$$

Hence, if G has bounded length facial cycles, and since $d = \theta(n)$, the relative error of our approximation algorithm is $O(1/\log n)$.

To estimate the complexity of this procedure, notice that the number of problems to be solved exactly is $O(2^T) = O(n/\log n)$, and each such problem takes time at most $2^{O(\sqrt{\log^2 n})} = n^{O(1)}$. The total number of vertices in the recursion tree is also $O(2^T)$, and we conclude that the algorithm runs in polynomial time.

6. Concluding remarks. We have shown that checking k-planarity of graph G with n vertices $D \subseteq V$ can be done in linear time for any fixed k. This provides an efficient recognition algorithm for this class of graphs for which the Steiner tree problem can be solved in polynomial time [EMV]. We have also shown that if k is not fixed, the associated decision problem is NP-Complete even if G has essentially a unique embedding, $d = \theta(n)$, and all facial cycles have bounded length. We obtain a polynomial-time algorithm for this latter case which is asymptotically optimal.

We note that the work of Robertson and Seymour on Wagner's conjecture could be used to check k-planarity for any fixed k in $O(n^4)$ time [Se]. However, their algorithm would *not* provide an embedding and covering as our algorithm does. It might be possible to specialize their result to our problem. We leave this as an open problem.

Appendix A—The three cases of DYN.

(1) *Series Case.* The formulas for this case are: Fix z and w. Then

$$F(H, 00zw) = \min\left\{F(H_1, 00z0) + \min_t \{F(H_2, 00tw)\}, \min_t \{F(H_1, 00zt)\} + F(H_2, 000w)\right\}.$$

Similarly,

$$F(H, 10zw) = \min\left\{\min_{w'} \{F(H_1, x'y'zw') \colon x' + y' = 1\} + \min_{z''} \{F(H_2, x''y''z''w) \colon x'' + y'' \leqq 1\},\right.$$

$$\min_{w'} \{F(H_1, x'y'zw') \colon x'+y' \leqq 1\}$$

$$+\min_{z''} \{F(H_2, x''y''z''w) \colon x''+y''=1\},$$

$$\left.\begin{aligned}&\min \{F(H_1, x'y'z1) \colon x'+y' \leqq 1\}\\ &+\min \{F(H_2, x''y''1w) \colon x''+y'' \leqq 1\}\end{aligned}\right\}.$$

All other parameters are computed analogously.

(2) *Parallel Case.* Fix z and w. Suppose we choose two candidates H_l and H_r as the "left" and "right" blocks in $E(H, xyzw)$, where, for example, we would use embeddings $E(H_l, xb_lzw)$ and $E(H_r, b_ryzw)$ for some $b_l, b_r \in \{0, 1\}$. Subject to this specific choice (i.e., l, r, b_l and b_r), we show how to compute the best embedding of H. Now suppose $k \neq l, r$. Which embedding should be used for H_k? Now, if $z+w \geqq 1$, we must always use $E(H_k, x'y'00)$ for some x', y' (and also, $b_l = b_r = 0$). Assume $z = w = 0$.

(a) If $F(H_k, 0100) \leqq F(H_k, 0000) - 1$, then $E(H_k, 0100)$ is preferred over $E(H_k, 0000)$; if the reverse inequality holds, then the opposite choice is preferred.

(b) $E(H_k, 0100)$ and $E(H_k, 1100)$ are compared in a similar way.

(c) If $F(H_k, 1100) \leqq F(H_k, 0000) - 2$ then $E(H_k, 1100)$ is preferred; if $F(H_k, 0000) \leqq F(H_k, 1100)$ then $E(H_k, 0000)$ is preferred. The only possible tie occurs precisely when $F(H_k, 0000) = F(H_k, 1100) + 1$. We will represent the better embedding, in this case, by $E(H_k, {*}{*}00)$.

Running through all cases (a)–(c) yields the best embedding for H_k, with a possible tie between (0000) and (1100). We still have to decide how to permute the H_k's, and how to rotate each individual H_k. This is done as follows. Assume first that no ties occurred, and for each H_k with $k \neq l$, r, we create a binary vector of two entries corresponding to the best embedding, together with an additional vector (b_l, b_r), where b_l and b_r specify the status of the inside face of H_l and H_r, respectively. Now we have to order these vectors cyclically and rotate them so that the total number of consecutive vectors (α, β) followed by (γ, λ) with $\beta + \gamma \geqq 1$ is minimum. For example, if the vectors are $(0,0)(0,1)(0,1)(0,0)(0,1)(0,1)(1,1)(1,1)$, the best ordering is $(0,1)(1,0)(0,0)(0,0)(0,1)(1,1)(1,1)(1,0)$ of value 4. In general, is it easy to see that an optimal arrangement is obtained by putting all (1,1)'s in a string; if there is at least one $(0,1)$ put it at one end of the string; if there is another, put it at the other end; next pair up all remaining $(0,1)$'s (at most one will not be paired up), and pair up all $(0,0)$'s. Let n_{ij} equal the number of (i,j)'s; the value will be

$$n_{11} + 1 + \left\lceil \frac{n_{01}}{2} \right\rceil - 1 = n_{11} + \left\lceil \frac{n_{01}}{2} \right\rceil \quad \text{if } n_{01} > 0,$$

$$n_{11} + 1 \quad \text{if } n_{01} = 0,\ n_{00} > 0,$$

$$n_{11} \quad \text{if } n_{01} = n_{00} = 0.$$

Now, if a tie occurred in at least one H_k, i.e., $n_{**} > 0$, using a $(1,1)$ embedding increases the n_{11} count by one but saves one internal face. Therefore, we either make all $(*,*)$'s into $(1,1)$'s, or we make all of them into $(0,0)$'s. That is, if $n_{**} > 0$ and

$n_{01} > 0$ or $n_{00} > 0$, then make all $(*, *)$'s into $(0, 0)$'s, and if $n_{01} = n_{00} = 0$, then make them all into $(1, 1)$'s.

In this way, we compute the value $m(b_l, b_r)$ of an optimal arrangement to account for faces between the H_i's. Let $f_k(b_l, b_r)$ be the number of internal faces used by each H_k in the optimal embedding. Thus,

$$F(H, xy00) = \min_{l,r}\left\{\min_{b_l,b_r}\left\{F(H_l, xb_l00) + F(H_r, b_ry00) + \sum_{k \neq l,r} f_k(b_l, b_r) + m(b_l, b_r)\right\}\right\}.$$

The above procedure can be implemented easily in a quadratic amount of work. We next sketch how to improve on it so that the $F(\cdot)$ parameters are obtained in linear time. Assume that we want to compute $F(H, xy00)$ (the general case is simpler), and fix b_l and b_r (there are four cases), without yet fixing H_l or H_r. We first consider the case where there are *no* ties in any of the blocks. Then the quantity $m(b_l, b_r)$ is well defined. We can also select a "best" block to act as a left end, namely, select k such that

$$F(H_k, xb_l00) - f_k(b_l, b_r)$$

is minimized, where we use the notation given above. Similarly, we can choose a "best" right block. It is not difficult to show that these blocks should indeed occupy the ends.

We now pass to the case of ties. If either

(i) $b_l + b_r \leqq 1$, or

(ii) $n_{01} + n_{00}$ is 0 or at least 3,

Then the ties will always be resolved, and we proceed as in the previous paragraph. Otherwise, we also try all possible ways of using, as end blocks, the (at most two) blocks whose best embedding is of type 01 or 00. This only adds a constant number of additional cases. Hence, this case can indeed be computed in linear time.

(3) *Messy Case.* Assume we have computed the m.m.b.'s. Let W be an m.m.b., with extremes v, w; we want to describe the behavior of W with a small gadget.

Suppose W is an interior m.m.b.; that is, the extremes of W are contained in the interior of H'. We can compare the possible embeddings of W as follows:

(a) For fixed z and w, we compare each pair $(xyzw)$ and $(x'y'zw)$ precisely as in case (2) of DYN. For example, if $F(W, 1000) \leqq F(W, 1100)$, then $E(W, 1000)$ is preferred over $E(W, 1100)$ (and if the inequalities are reversed, otherwise). Recall that we have a tie between $E(W, 0000)$ and $E(W, 1100)$ if $F(W, 0000) = 1 + F(W, 1100)$.

(b) If $F(W, 1000) \geqq F(W, 00zw)$ with $z + w \geqq 1$, then $E(W, 00zw)$ is preferred over $E(W, 1000)$. The reason for this is that if in an optimal embedding of H we use $E(W, 1000)$, we can instead use $E(W, 00zw)$ and keep everything else fixed to obtain a feasible embedding that must also be optimal.

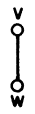
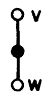
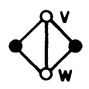
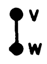
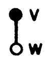
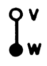
(c) If $F(W, 0000) \leqq F(W, 0011)$ then $E(W, 0000)$ is preferred; if $F(W, 0011) + 1 \leqq F(W, 0000)$ then $E(W, 0011)$ is preferred.

(d) If $F(W, 1100) + 2 \leqq F(W, 0011)$ then $E(W, 1100)$ is preferred; if $F(W, 0011) \leqq F(W, 1100)$ then $E(W, 0011)$ is preferred, with a tie if $F(W, 1100) + 1 = F(W, 0011)$.

(e) In a similar way, we compare $E(W, 1100)$ with $E(W, 00zw)$ where $z + w = 1$, with a tie if $F(W, 1100) + 1 = F(W, 00zw)$. We may even have a three way tie between $E(W, 1100)$, $E(W, 0010)$ and $E(W, 0001)$.

Now suppose we are able to select a best embedding $E(W, x^*y^*z^*w^*)$; that is, there are no ties for best. Then we replace W with the appropriate graph in Table 1

TABLE 1
Replacement graphs with no ties.

(0000)	
(1000)	
(1100)	
(0011)	
(0010)	
(0001)	

(where black vertices represent vertices of D). Let $\hat{H}$ be the new graph. It is easy to prove that $F(\hat{H}, xyzw) + F(W, x^*y^*z^*w^*) = F(H, xyzw)$. Thus, we solve the problem on $\hat{H}$ first and then add the weight $F(W, x^*y^*z^*w^*)$.

By induction, we can replace any set of interior m.m.b.'s (with no ties for best embedding) whenever they have no common extremes. If, on the other hand, say $W_1, \cdots, W_k$ have a common extreme $v \in D$, then: (i) if the best embedding for each W_k prescribes that v not be covered, then leave v uncovered in each replacement graph, and (ii) if, in at least one W_j the best embedding covers v, then use the same replacement for each W_i, except that v is removed from D.

A few complications arise if a particular interior m.m.b. W has ties for best embeddings. The list of all possible ties is given here:

(T1) $\quad F(W, 0000) = F(W, 1100) + 1,$

(T2) $\quad F(W, 0011) = F(W, 1100) + 1,$

(T3) $\quad F(W, 1100) + 1 = F(W, 0010),$

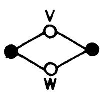
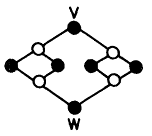
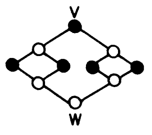
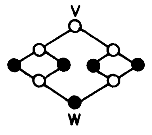
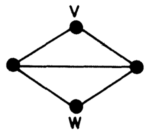
(T4) $\quad F(W, 1100) + 1 = F(W, 0001),$

(T5) $\quad F(W, 1100) + 1 = F(W, 0010) = F(W, 0001),$

(T6) $\quad F(W, 0010) = F(W, 0001).$

Table 2 contains the replacement graph to be used for each tie, together with the appropriate weight to be added after solving. It is not difficult to verify that these are all correct; here we will do so for the most complicated case (T2). Let $\hat{H}$ be obtained from H by replacing W with the corresponding graph in case (T2). Now in any optimal embedding $E(H, xyzw)$, at least one internal face of the replacement graph will be

TABLE 2
Replacement graphs with ties.

CASE	GRAPH	WEIGHT
T1		F(W, 1100)
T2		F(W, 1100) − 1
T3		F(W, 1100) − 1
T4		F(W, 1100) − 1
T5		F(W, 1100)
T6	● $v \equiv w$	F(W, 0010)

NOTICE THAT ALL WEIGHTS ARE NONNEGATIVE

used. It is easy to see that, without loss of generality, it is either the face containing v and w (but none of the other faces), or the two other faces (but not the one containing v and w). Given our choice for the weight, $F(W, 1100) - 1$, this gives the correct result.

Thus, we can replace interior m.m.b.'s with small graphs, with the small caveat concerning extremes that are common to several m.m.b.'s as given before. Noninterior m.m.b.'s are dealt with similarly:

(i) Suppose we want to compute $F(H, 0000)$; then we replace every m.m.b. using the above rules.

(ii) Suppose we are trying to compute $F(H, 00zw)$ where $z + w \geqq 1$; say, $F(H, 0010)$ (i.e., u_1 is not covered). Then we proceed exactly as in (i), except that now if an m.m.b. W contains u_1 as an extreme, we use the best embedding for W of the form $E(W, xy1w)$, and all faces of the resulting graph $\hat{H}$ that are incident on u_1 are rejected. (There is a small wrinkle if there is an m.m.b. W_1 with extremes u_1 and v, $v \in D$, and such that v is an extreme of only one other m.m.b., W_2. In this case we may have to force v to be covered by W_1 or W_2. Since there are at most two vertices such as v, this yields only four or fewer problems.)

In cases (i) and (ii), we use Algorithm XTND in the resulting graph $\hat{H}$ after the replacement. If XTND finds a cover of cardinality k or less, then we add the sum of the weights of the replacement graphs to the cardinality of a minimum cover of $\hat{H}$. If, on the other hand, XTND finds that $\hat{H}$ has no covers of cardinality k or less, then the corresponding embedding $E(H, 0000)$ or $E(H, 0010)$ should not be considered. (Similarly with $F(H, 0001)$ and $F(H, 0011)$.)

(iii) The final case occurs for $F(H, xyzw)$ where $x+z \geqq 1$; say, we want to compute $F(H, 10zw)$. We proceed in two stages. Let p_1 and p_2 be the two paths from u_1 to u_2 in the outer face of H'. Then we obtain $\hat{H}$ as for computing $F(H, 00zw)$, and we remove from D all vertices in the segment of the outer face of $\hat{H}$ corresponding to p_1. We then apply XTND to this graph, add the weights of the replacement graphs to the returned cardinality, and denote the obtained quantity by $m(p_1)$. Similarly compute $m(p_2)$. Then $m(H, z, w) = \min(m(p_1), m(p_2))$ is the minimum number of faces needed to cover $D \cap H$ (with the exception of u_1 and u_2 as indicated by z and w) where we do not require that we cover one of the segments of the outer face.

Now if $m(H, z, w) < F(H, 00zw)$, then we are done; set $m(H, z, w) = F(H, 10zw)$. If $m(H, z, w) = F(H, 00zw)$, then we proceed to the second stage. Let W be an arbitrary m.m.b. whose corresponding edge in H' is contained in p or p'. Let $\hat{H}(W)$ be the graph obtained from H by replacing W with the best embedding of the form $E(W, 1y00)$, and all other m.m.b's replaced with graphs as for cases (i) and (ii).

Notice that the replacement graph for W will contain a vertex of D in the outer face of $\hat{H}(W)$ (see Table 1); call this vertex w', and set $f(W)$ to be the min cardinality of face cover of $D \cap \hat{H}(W)$, with all faces incident to w' rejected plus weight of replacement graphs. Then $\min_W f(W)$ finds the best embedding of H that does not cover at least one vertex (which is not an extreme of an m.m.b.) in the outer face. In a similar way, we compute all quantities $g(v)$, where $v \in D$ is extreme of an m.m.b. in the outer face of H' that we want not to cover. Then $\min\{\min_W f(W), \min_v g(v)\}$ is the quantity $F(H, 1000)$ we seek.

There is one shortcut that we can take to improve the algorithm above. Suppose there are at least k possible graphs W for which $F(W, 1y00) \leqq k$ that can be used to compute $\min_W f(W)$. Now, the fact that we have reached Stage 2 implies that $F(H, 10zw) \geqq F(H, 00zw)$. However, any internal face of H can cover at most one of the vertices of D counted above; otherwise, H' would contain a cutset of two vertices in one of the segments p_1 and p_2, which is impossible. Therefore, $F(H, 00zw) > k$, and we can avoid Stage 2 altogether. Similar considerations apply towards computing $\min_v g(v)$. Consequently, we can assume that each of the minimums in Stage 2 at most has $O(k)$ arguments.

REFERENCES

[Ba] B. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, Proc. 24th Annual Symposium on the Foundations of Computer Science, (1983), pp. 265-273.

[BM] D. Bienstock and C. C. Monma, *On the complexity of embedding planar graphs to minimize certain distance measures*, submitted.

[CNP] G. Cornuejols, D. Naddef and W. R. Pulleyblank, *Halin graphs and the traveling salesman problem*, Math. Programming, 26 (1983), pp. 287-294.

[EMV] R. E. Erickson, C. L. Monma and A. F. Veinott, Jr., *Send-and-split method for minimum-concave-cost network flows*, Math. Oper. Res., to appear.

[FHS] M. Fellows, F. Hickling and M. Syslo, *A topological characterization and hard graph problems*, Extended Abstract, University of New Mexico, Albuquerque, NM.

[GJ1] M. R. Garey and D. S. Johnson, *The rectilinear Steiner tree problem is NP-complete*, SIAM J. Appl. Math., 32 (1977), pp. 835-859.

[GJ2] ———, *Computers And Intractability: A Guide To The Theory Of NP-Completeness*, W. H. Freeman, San Francisco, CA, 1979.

[GJS] M. R. Garey, D. S. Johnson and L. Stockmeyer, *Some simplified NP-complete graph problems*, Theoret. Comput. Sci., 1 (1976), pp. 237-267.

[HT1] J. E. Hopcroft and R. E. Tarjan, *Dividing a graph into triconnected components*, this Journal (1973), pp. 135-158.

[HT2] ———, *Efficient planarity testing*, J. Assoc. Comput. Mach., 21 (1974), pp. 549-568.

[LT] R. J. Lipton and R. E. Tarjan, *Applications of the planar separator theorem*, this Journal, 9 (1980), 615-627.

[Pr] J. S. Provan, *Convexity and the Steiner tree problem*, Technical Report, TR-85/3, University of North Carolina, January 1985.

[RS] N. Robertson and P. D. Seymour, *Graph minors.* III. *Planar tree-width*, J. Combin. Theory Ser. B, 36 (1984), pp. 49-64.

[Se] P. D. Seymour, personal communication, August 1986.

[Wh] H. Whitney, *Two-isomorphic graphs*, Amer. J. of Math. Soc., 34 (1932), pp. 339-362.

[Wi] P. Winter, *Steiner problems in networks: a survey*, Networks, to appear.