Faces in the Crowd: On the Algorithmic Utility of Disks and Covers[⋆]

Faisal N. Abu-Khzam and Michael A. Langston

University of Tennessee, Knoxville, TN 37932, USA

Abstract. We study a pair of $\mathcal{NP}$-hard problems aimed at finding small sets of faces in planar graphs. In the disk dimension problem, we are given a planar graph G, and seek the least number k for which G embeds in the plane minus k open disks, with every vertex on the boundary of some disk. Useful properties of graphs with bounded disk dimension are derived. We show how to obtain an outerplanar subgraph of a graph of disk dimension k by removing at most $2k-2$ vertices. We use this reduction technique to obtain a tree-decomposition of width at most $2k$ and a linear-time 3-approximation algorithm for the pathwidth problem on graphs of fixed disk dimension. Such results were previously known only for outerplanar graphs (graphs of disk dimension one). In the related face cover problem, we are given a plane graph G, and seek the least number k of faces whose boundaries contain all the vertices of G. Although both problems are FPT, we are able to exploit the embedding that comes with face cover instances to derive a direct $O(5^k + N^2)$ algorithm, where N is the input size. This is a considerable improvement over the best previously-published face cover algorithms, which rely instead on reductions to planar dominating set.

1 The Disk Dimension Problem

Disk dimension was introduced in [6], where the celebrated Graph Minor Theorem was used to show that it is polynomial-time decidable for any fixed number of disks. The decision version of the general problem is $\mathcal{NP}$-complete [1]. We focus here on the optimization version of this problem. The disk dimension of a planar graph G, $dd(G)$, is defined as the least positive integer k for which G embeds in the plane minus k open disks, $\{d_i\}_{i=1}^k$, so that every vertex of G lies on the boundary, C_i, of some disk d_i.

1.1 Fundamentals

Let DD_k denote the family of planar graphs with disk dimension k. Given a DD_k graph G, the disk set $\{d_i\}_{i=1}^k$ will be called a DD_k layout of G. We use c_i to denote the center of disk d_i. The wheel graph of G with respect to $\{d_i\}_{i=1}^k$ is defined by: $G^w = (V \cup \{c_i\}_{i=1}^k, E \cup \bigcup_{i=1}^k \{\{c_i, u\} : u \in C_i\})$. Note that G^w is defined in terms of a particular layout and, therefore, is not necessarily unique. The planarity of G^w is useful, however, and will employ it in many of our proofs. Note also that, given a connected DD_k graph G with a DD_i layout ($i \leq k$), the diameter of the corresponding G^w is $\leq 3i - 1$.

A planar graph G is maximal planar if $\forall u, v \in V(G), \{u, v\} \notin E(G) \Rightarrow G' = (V(G), E \cup \{u, v\})$ is not planar. The disk dimension of such a graph is at least $n/3$, where n is the number of vertices. To see this, we note that in any DD_k layout of a maximal planar graph, none of the k disks can contain more than three vertices (but, counter-intuitively, a disk may have to contain fewer than three). In fact, $n/3$ is only a lower bound, since some graphs have disk dimension more than $n/3$. For example, it follows from Theorem 2 (to follow) that $dd(K_{2,11}) = 6 > \lceil (n/3) \rceil$.

A graph is outerplanar if it can be embedded in the plane minus a single open disk so that every vertex lies on the boundary of that disk. Thus outerplanar graphs are exactly the DD_1 graphs. Thanks to Euler's formula, we know that a planar graph of order n can have at most $3n - 6$ edges. And an outerplanar graph of order n has at most $2n - 3$ edges. The following result generalizes this to DD_k graphs.

[⋆] This research is supported in part by the National Science Foundation under grants EIA–9972889 and CCR–0075792, by the Office of Naval Research under grant N00014–01–1–0608, by the Department of Energy under contract DE–AC05–00OR22725, and by the Tennessee Center for Information Technology Research under award E01–0178–081.

Theorem 1. Let G be a planar graph having n vertices and e edges and satisfying $dd(G) = k$. Then $e \leq 2n + 3k - 6$

Proof Let n_i be the number of vertices in C_i. Then $\sum_{i=1}^{k} n_i = n$. G is not maximal planar since more edges can be drawn in the interior of each disk d_i between vertices that lie on the boundary. For each d_i, the maximum number of such edges is $n_i - 3$. This gives a total of $n - 3k$ edges that can be added to G to get another planar graph G' on n vertices satisfying $G \subseteq G'$. It follows that the number of edges of G' is bounded above by $3n - 6$ and is equal to $e + n - 3k$.■

Lemma 1. If $(A, B) = K_{2,3} \leq_t G$, and $i \geq dd(G)$, then the three vertices of B do not lie on the boundary of a single disk in any DD_i layout of G.

Proof Extending G to G^w allows us to get $K_{3,3} = (A \cup \{c_i\}, B) \leq_t G^w$. Which is impossible since G^w is planar.■

Theorem 2. If $K_{2,3m+1} \leq_t G$, then $dd(G) > m$.

Proof Let $(A, B) = K_{2.3m+1} \leq_t G$. If $dd(G) \leq m$, then at least 3 vertices of B must lie on the boundary of some disk d_i. This contradicts the assertion of lemma 1.■

Observation 1 Let $G = (V, E)$ be an element of DD_2. Then G has two vertices u and v such that $dd(G - \{u, v\}) = 1$. To see this, note that removing two adjacent vertices that lie on the two disks of a DD_2 layout of G would lead to unifying the two disks (It's like thickening edge (u, v) and opening a tunnel between the two disks.)

Lemma 2. Let G be a planar graph satisfying $dd(G) = k > m > 0$. Then G has at most $2m$ vertices $\{u_1, u_2, ...u_l\}, l \leq 2m$, such that $dd(G - \{u_1, u_2, ...u_l\}) \leq k - m$.

Proof There are at least two disks, d_i and d_j, in the layout of G that have an edge joining some u on d_i to a v on d_j. By Observation 1, removing u and v results in replacing d_i and d_j by one disk. This can be repeated m times or until we get a DD_{k-m} graph.

1.2 A Reduction Algorithm

According to Lemma 2, given a DD_k graph G, there are pairs of adjacent vertices whose removal reduces the disk dimension of G. The question is, how do we find such pair when G is not given by a DD_k layout.

Theorem 3. Let G be a DD_k graph satisfying $K_4 \leq_t G$. Let u be any of the 4 corners of the K_4 model in G. Then the 3 neighbors of u in the model can't all lie on the same disk as u.

Proof Note first that the four corners $\{u_i\}_{i=1}^{4}$ of a K_4-model cannot all lie on the boundary of the same disk. Let v_1, v_2, and v_3 be the neighbors of u_1 in the model. Assume also that, either $v_i = u_i$ or v_i is on the $u_1 - u_i$ path of the K_4 model. Let $\{d_i\}_{i=1}^{k}$ be a DD_k layout of G such that $u_1 \in C_1$. Assume $\{v_1, v_2, v_3\} \subset C_1$. If $u_2 \neq v_2$, then $(\{u_1, u_2\}, \{v_2, v_3, v_4\}) = K_{2,3} \leq_t G$. To see this, note that $u_2 - v_2, u_2 - u_3 - v_3$, and $u_2 - u_4 - v_4$ are vertex disjoint paths in the model of $K_4 \leq_t G$. Which is impossible by Lemma 1. ■

Theorem 4. Let G be a DD_k graph satisfying $(A, B) = K_{2,3} \leq_t G$. Let u be any of the 2 corners corresponding to A in the $K_{2,3}$ model. Then the 3 neighbors of u in the model can't all lie on the same disk as u.

Proof Let v_1, v_2 and v_3 be the neighbors of u in the model. Then $(A, \{v_3, v_4, v_5\})$ is another $K_{2,3}$ model in G. Result follows by Lemma 1.■

We are ready now to present our reduction algorithm. Procedure $REDUCE$, below, uses the following assumptions and notations: We use the expression 2-corner of a $K_{2,3}$-model to denote any of the two elements of A when $(A, B) = K_{2,3} \leq_t G$. Function outerplanar is an outerplanarity test. If G is not outerplanar, and K_4 is a minor of G, a corner of the K_4-model is returned together with its three neighbors in the model. If K_4 is not a minor of G but $K_{2,3}$ is, a 2-corner is returned together with its three neighbors in the model. Note that Corners of a K_4-model can be found by the linear-time algorithm described in [9]. Corners of a $K_{2,3}$-model can also be found in linear time [13].

Procedure REDUCE

```
Input: A planar graph $G$ with $n$ vertices and $e$ edges, and an integer $k \geq 1$.
Output: A set $S$ of "at most" $2k-2$ vertices of $G$ such that $G-S$ is outerplanar. If no such set exists, return NULL.

Begin procedure
If($e > 2n-3k+6$)
    return NULL;
$S \leftarrow \phi$;
If ($k == 0$) return NULL;
If (outerplanar(G)) return $S$;
If ($K_4 \leq_t G$)
    $u \leftarrow$ corner of a $K_4$-model in $G$;
else
    $u \leftarrow$ 2-corner of a $K_{2,3}$-model in $G$;
$\{v_0, v_1, v_2\} \leftarrow$ neighborhood of $u$ in the model;
for ($i = 0; i < 3; i++$){
    $S' = REDUCE(G - \{u, v_i\}, k-1)$;
    if ($S' \neq 0$)
        return $S \cup S'$
    }
return NULL
end procedure
```

1.3 Tree Decompositions

Obtaining a tree decomposition of bounded width is important because NP-complete problems are frequently solvable in polynomial time (in fact in linear time) on graph of bounded treewidth. It's also worth noting that the famous Steiner tree problem is solvable in polynomial time on DD_k graphs.

Theorem 5. [11] For every planar graph H, there exists a constant C_H, such that every graph G that doesn't have an H-minor satisfies $tw(G) \leq C_H$.

Theorems 2 and 5 imply the following:

Corollary 1. For fixed k, graphs that belong to DD_k have bounded treewidth.

We have already observed that, if G is connected, the diameter of G^w is bounded above by $3k-1$. It is shown in [5], that, for a planar graph G, $tw(G)$ is $O(D)$ where D is the diameter of G. This provides an alternate proof of corollary 1. By Lemma 2, given a DD_k graph G, there are at most $2k-2$ vertices whose removal produces an outerplanar graph. And outerplanar graphs have treewidth two or less. We therefore have an explicit upper bound on the treewidth of DD_k graphs:

Theorem 6. Let G be a DD_k graph. Then $tw(G) \leq 2k$.

The following procedure uses the same technique as in $REDUCE$ to remove at most $2k-2$ vertices of the graph. It produces NULL only if $dd(G) > k$. There is no need, however, to go all the way until the resulting graph is outerplanar.

Procedure $\mathbf{ddk\_tw}$

```
Input: A planar graph $G$.
Output: A width $2k$ tree decomposition $(T, X)$ of $G$
Begin procedure
If (series-parallel($G$))$\{$
    $(T, X) \leftarrow sp\_tw(G)$;
    return $(T, X)$;
    $\}$
If ($k == 1$)
    return NULL; // G can't be outerplanar and not series-parallel
$u \leftarrow$ corner of a $K_4$-model in $G$;
$\{v_0, v_1, v_2\} \leftarrow$ neighborhood of $u$ in the model;
for $(i = 0; i < 3; i++)\{$
    $(T, X_1) \leftarrow ddk\_tw(G - \{u, v_i\}, k - 1)$
    if(width(T) $< 2k - 1)\{$
        $X \leftarrow \{X_i \cup \{u, v_i\} : X_i \in X_1\}$;
        return $(T, X)$;
        $\}$
    $\}$
return NULL;
end procedure
```

Theorem 7. Treewidth of DD_k graphs is $O(\sqrt{k})$.

Proof Define: $PDS_k = \{G : G$ is planar and has a dominating set of size $\leq k\}$, and $DD_k^w = \{G^w : G \in DD_k\}$. Clearly, $DD_k^w \subset PDS_k$. And, By the work of Alber et al., treewidth of PDS_k graphs is $O(\sqrt{k})$. Let G be a DD_k graph, then $G \subset G^w$ corresponding to a suitable DD_k layout.

1.4 Pathwidth Approximation

Optimal path decompositions of DD_k graphs can be obtained in principle in polynomial time. This is due to [3], which asserts that optimal path decomposition can be found in polynomial time for graphs of bounded treewidth. The algorithm suggested is not practical, however, because it operates on sets of size $O(n^{11})$ in its first step.

Motivated by fast approximation algorithms for the pathwidth of DD_1 graphs, we show how to get similar algorithms for DD_k graphs In fact, we rely on Lemma 2 (by using procedure $REDUCE$) to show how to obtain a linear time algorithm whose performance ratio is 3 on general DD_k graphs and 2 on some biconnected ones.

Take, for example, a DD_2 graph G. Delete a (suitable) pair $\{u, v\}$ of vertices. The resulting graph, H, is outerplanar. Using the work of [7], we can find a path decomposition (P, X) that is not more than $3pw(H)$ in $O(n)$ time. (The bound stated in [7] is only O($nlogn$), because the algorithm described there relies on obtaining optimal path decompositions of trees. It has more recently been shown that such decompositions can be computed in linear time [12]. Thus the algorithm of [7] runs in linear time as well.) Adding the two vertices to every element of X gives a path decomposition of G of width $\leq 3pw(H) + 2 \leq 3pw(G) + 2$ because $H \subseteq G$. If H were biconnected, we could use the algorithm of [2] to obtain a path decomposition of width $\leq 2pw(H) + 1$.

Procedure ddk_pw, below, assumes that a path decomposition of a given outerplanar graph, H, can be obtained by function $outpl\_pw$ in linear time. The width of the path decomposition returned by $outpl\_pw$ is not more than $3pw(H) + 2$. Thus, given graph G as input, ddk_pw returns a path decomposition of width not exceeding $3pw(G) + 2K$. It produces NULL only if $dd(G) > k$.

Procedure $\mathbf{ddk\_pw}$

```
Input: A planar graph $G$.
Output: A path decomposition $(P, X)$ of $G$

Begin procedure
If (outerplanar($G$)){
        $(P, X) \leftarrow outpl\_pw(G)$;
        return $(P, X)$;
        }
$S \leftarrow REDUCE(G, k)$
if($|S| \leq 2k - 2$){
        $(P, X_1) \leftarrow outpl\_pw(G \backslash S)$;
        $X \leftarrow \{X_i \cup S : X_i \in X_1\}$;
        return $(P, X)$;
        }
return NULL;
end procedure
```

2 The Face Cover Problem

In the (optimization version of) the face cover problem, we are given a plane graph G, and seek the least number k of faces whose boundaries contain all the vertices of G. The corresponding decision version is $\mathcal{NP}$-complete [1]. Of course a face of a plane graph can be treated as a disk with vertices on its boundary. Thus the disk dimension of a planar graph is equal to its minimum face cover taken over the set of all possible planar embeddings. Both problems are FPT [4].

When k is fixed and a face cover of size at most k exists, finding such a cover can be accomplished in linear time. Both the $O(12^k n)$ method of Downey-Fellows and the $O(36^{\sqrt{(34k)}} n + n^2)$ method of Alber et al rely on reductions to planar dominating set. We present a direct $O(5^k + N^2)$ algorithm.

The notation $G = (V, F)$ is used when referring to a plane graph. The boundary of a face, $f \in F$, is the set of vertices appearing in the ordered tuple associated with f. The order being that of the particular anti-clockwise drawing of the vertices of f.

Face cover has a highly practical linear-time algorithm when $k = 1$, since this is just recognition of outerplane graphs. However, even for $k = 2$, none of the known algorithms seem to be encouraging.

For a planar graph G, $\delta(G) \leq 5$. Which implies that we can always find a vertex that belongs to the boundaries of at most five faces. So, if we use the search tree technique, that is commonly used in exact exponential algorithms, branching at such vertex introduces at most five ”smaller” instances of the problem. The question now is, after performing this branch operation, is there any guarantee that another vertex of degree ≤ 5 is present in the resulting graph? The answer is usually no. However, since what is needed here is the number of faces to which a vertex belongs, we shall see in subsection 5 that some reduction rules can be used to modify the graph and always guarantee (in the worst case) the existence of a vertex belonging to no more than five faces that qualify for membership in the cover.

From this point on, we use the term k-plane to denote a plane graph that has a face cover of size k. k-plane graphs are particular embeddings of DD_k (planar) graphs. When k is fixed, such graphs have bounded treewidth and we showed in a previous subsection that a tree decomposition of width not exceeding $2k$ can be constructed in linear time ($O(3^k n)$) without the need for a k-plane embedding. If the embedding is given, a face cover of size k will help obtaining such tree decomposition (easily) in $O(n)$.

Obtaining a tree decomposition of bounded width is very important since most $\mathcal{NP}$-complete problems are solvable in polynomial time (often linear) on graphs of bounded treewidth.

We shall obtain a face cover algorithm that runs in $O(5^k + N^2)$. In addition to being an improvement of the best known previous algorithms, experiments showed that our algorithm is practical in general. Moreover, we present a reduction technique that reduces the input size to $2k^3$ in linear time. We are aware of the work of [8] that results in reduction of input to a kernel of size $O(k^2)$. Our reduction technique is (much simpler and ...)

We denote the input size by N. It is (obviously) quadratic in $|V|$ (thus also quadratic in $|F|$ due to Euler's formula). Face cover is a special case of the hitting set problem. In this problem, we are given a collection, F, of subsets of a given set X, a positive integer k. The question posed is whether X has a subset, C, of k or fewer elements such that each element of F has a non-empty intersection with C?

To see why face cover is a special case of hitting set, note that a face cover of $G = (V, F)$ is a hitting set of the sets $\{F(u) : u \in V\}$. HS is $W[2]$ hard. However d-hitting-set, the case where the input consists of sets of size $\leq d$, is FPT. There are algorithms that solve the fixed-parameter d-HS efficiently when d is small enough. In fact 3-HS and 4-HS have, respectively, $O(2.270^k + N)$ and $O(3.3^k + N)$ algorithms [10]. A d-HS algorithm can be used to solve the face cover problem when the degree of any vertex of the input plane graph is bounded by d. We will deal with the general face cover problem. Moreover, preprocessing techniques used for HS apply well to FC instances.

We shall assume that our input is like the input of a HS algorithm. In fact, this is why we chose to characterize plane graphs by vertices and faces. The data structure used for a plane graph consists, essentially, of two lists corresponding to vertices and faces of the graph. Each element of the vertex list is a pointer to a vertex structure, which contains the list of faces containing the vertex.

We use the search tree technique in our FC algorithm. During the search process, the vertex set is partially covered by the already selected faces. We shall, then, reduce the graph at each step by eliminating covered vertices. While this action is easy (and safe) when dealing with a general HS instance, it must be performed carefully in our case. Especially because we need to be assured that every node in the search tree has at most 5 children. In fact, deleting a covered vertex from a plane graph might change the size of an optimal face cover and produce incorrect results (see Figure 1 for an example). Before deleting a covered vertex, v, we modify all faces containing it by, simply, deleting v from their (ordered) lists. This action is called face compaction and is depicted in Figure 2.

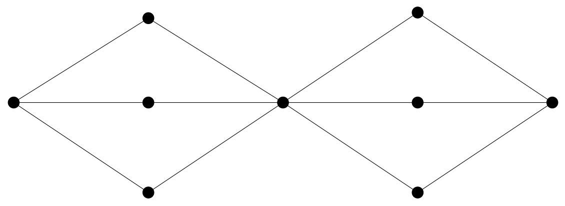

Fig. 1. Deleting the vertices that are covered by selecting the outer face produces a wrong answer.

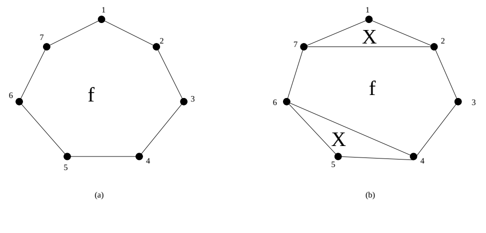

Fig. 2. Removing (covered) vertices 1 and 5 from face f = (1,2,3,4,5,6,7) produces the face f = (2,3,4,6,7). The regions marked with X are marked faces of the graph.

2.1 Preprocessing

Preprocessing techniques proved useful in many FPT algorithms, as it sometimes quickly reduces the input instance (often in linear time) to one of a bounded size. If such is the case, preprocessing is termed kernelization and the resulting instance is called a kernel.

For a given plane graph $G = (V, F)$, Vertices and faces are of two types: active and marked. Active vertices are those to be covered (i.e., not covered yet) and active faces are those that can be used to cover active vertices.

Marking a face is easy since marked faces are just faces that are not to be considered in the cover. So we simply delete the marked face from all lists corresponding to its vertices.

Marking a vertex, v, is equivalent to deleting its list of adjacent faces, which we denote by $F(v)$, and deleting its index from the list, $V(f)$, of each face, f, that contains it. To show that such simple operation is sound, we prove it to be equivalent to a surgical operation on the graph.

Removing Marked Vertices Two neighbors, u and w, of a vertex, v, are consecutive if some face, f, contains the three vertices such that one if the two ordered tuples (u, v, w) and (w, v, u) is a sub-tuple of the ordered tuple of f.

Since pendant vertices are covered by only one face, removing a marked pendant vertex does not affect the solution.

If a vertex, v, is of degree at least 2, and is marked, then active faces that are adjacent to it will not be needed to cover it. Deleting v could lead to wrong answers as shown in Figure 1. So, prior to removing v, we make sure that the marking operation avoids such cases. The marking operation simply consists of adding edges between consecutive neighbors of v and marking all faces that contain v in the resulting graph. This action is safe in the following sense: every face that is adjacent to v and is used to cover a neighbor of v, must contain two consecutive neighbors of v. We refer to this operation on the graph by the surgical operation. In our implementation of the algorithm, we didn't need to perform the surgical operation. It is used only to show that the algorithm has the claimed performance and the operations of marking vertices and faces are sound.

We draw the attention of the reader that a general version of the face cover problem was presented in [1], where not all vertices of the graph are to be covered. Our algorithm will be ready to deal with this version as well. In fact, if a vertex is not to be covered, or has been covered earlier during the process of finding the face cover, then it will be marked.

Dealing with Dominated Faces and Vertices The Dominated Face Rule. If two faces, f and f', of G are such that $V(f) \subset V(f')$, then f is said to be dominated by f'. In such case, f can be marked since every face cover of G that contains f can be replaced by a (better) cover that contains f'.
The Dominated Vertex Rule. If two vertices, u and v, are such that $F(u) \subset F(v)$, then v is said to be dominated by u, and v is marked since any face that covers u will cover v.

Checking if there is a face or a vertex that is dominated takes $O(N^2)$ time: To check if a given vertex is dominated by another, we look at $F(v)$. Then for any pair of faces in $F(v)$ (there are $O(N)$ of these), checking if the vertex set of one is a subset of the other takes $O(\sqrt{N})$. This is done for all $O(\sqrt{N})$ vertices of G. (Thus the $O(N^2)$ time claimed).

We noticed that many preprocessing actions described in our previous version of the algorithm (as well as subsequent face cover kernelization techniques by other authors) were just particular cases of the above two rules. For example, if a path of length more than two whose interior vertices are all of degree two, then we used to contract all interior edges (edges between interior vertices). Thus keeping one (necessary) interior vertex of degree two. Isn't this just the dominated vertex rule?

From this point on, we call a preprocessed plane graph an annotated plane graph. The reason being that such graph might have marked vertices and faces. We will only be concerned with active and marked vertices, so we denote by $G = (A, M, F)$ an annotated plane graph G.

2.2 Kernelization

This subsection is devoted to the proof of the following theorem.

Theorem 8. Let (G, k) be an instance of face cover. There is an algorithm that runs in $O(N^2)$ and produces an, (G', k'), of face cover such that $k' \leq k$ and $|V(G')| \leq 2k^3$.

Lemma 3. Three or more vertices of a plane graph, G, may not be common to more than two faces of G.

Proof Assume vertices u_1, u_2, and u_3 belong to three faces, f_1, f_2, and f_3. Construct another plane graph, G', by drawing three vertices, v_1, v_2, and v_3 such that vertex v_i is placed inside face f_i and joined to all vertices that lie on the boundary of f_i. The subgraph of G' induced by vertices $\{u_i\}_{i=1}^3$ and $\{v_i\}_{i=1}^3$ is isomorphic to $K_{3,3}$. This is a contradiction. ∎

Lemma 4. Let G be a k-plane graph. If two faces of G have more than $2k$ common vertices, then any face cover of size k contains at least one of them.

Proof Let C be a face cover of size k. Assume faces f_1 and f_2 contain $2k + 1$ common vertices and are not elements of F. Then, by the pigeon hole principle, some element of F must cover at least three of the $2k + 1$ common vertices of f_1 and f_2. This is impossible by lemma 3. ∎

Corollary 2. Let G be a k-plane graph. If no pair of faces of G have $2k + 1$ common vertices, then every face whose length exceeds $2k^2$ is in any optimal face cover of G.

To obtain our kernel, we simply search for a face, f, of length $> 2k^2$. This is easy since the length of a face is captured when reading input. The number of common vertices with all other faces is then computed. (In fact we only consider faces that contain at least one common vertex with f). If a face f' has more than $2k$ common vertices with f then their common vertices are all marked, and a virtual new vertex, v, is added to the list of vertices such that $F(v) = \{f, f'\}$. If no such face f' exists, then f is added to the cover and marked. Moreover, if the number of faces that share more than $2k$ common vertices with f exceeds k, then f is also selected in the cover and marked, otherwise, more has k faces will have to be in the cover. This operation is repeated until no more pairs of faces having such large number of common vertices are detected. Note that we stop the search if more than k disjoint pairs of faces are found. This proves Theorem 8. The process just described runs in $O(kN)$: it takes $O(\sqrt{N})$ to find f. Then it takes $O(N)$ to find all faces that have more than $2k$ common vertices with f.

2.3 A Direct Face Cover Algorithm

Our direct algorithm is represented by the procedure $FaceCover$ shown below. Subroutines $KERNELIZE$ and $MARKFACE$ correspond (obviously) to processes previously described in details.

Procedure FaceCover

```
Input: A plane graph $G = (V, F)$ given by $\{F(u) : u \in V\}$ and $\{V(f) : f \in F\}$, and an
integer $k \geq 1$.
Output: A face cover, $C$ of size $\leq k$ of $G$ if one exists. NULL otherwise.

Begin procedure
$(G, C, k_1) \leftarrow KERNELIZE(G, k)$
Select an active vertex $v \in V$ such that $|F(v)| = min\{F(u)|u$ is active in $V\}$.
For every $f \in F(v)$ do
    $C_1 \leftarrow C \cup \{f\}$;
    $G_1 \leftarrow MARKFACE(f)$;
    $(G_1, C_1, k_2) \leftarrow KERNELIZE(G_1, k_1 - 1)$;
    $C_2 \leftarrow FACECOVER(G_1, k_2)$;
    if($C_2 \neq NULL$)
        return $C_1 \cup C_2$
return NULL
end procedure
```

We shall prove that, at every call to $FACECOVER$, the selected vertex, v, has no more than five active faces in its $F(v)$ list. We know that the first call is guaranteed to select such vertex. As a remark, we note that, at least three such vertices are present in the graph. This is guaranteed by virtue of Euler's formula and the following lemma.

Lemma 5. If G is a planar graph, then G has at least three vertices of degree ≤ 5.

Proof Let $m = |M = \{v \in V(G) : d(v) > 5\}|$ and $l = |L = \{v \in V(G) : d(v) \leq 5\}|$. Then: $3n - 6 \geq e(G) \geq \frac{\sum_{v \in L} d(v) + \sum_{v \in M} d(v)}{2} \geq \frac{\sum_{v \in L} d(v) + 6m}{2} \geq \frac{\sum_{v \in L} d(v) + 6(n-l)}{2} \geq \frac{\sum_{v \in L} d(v)}{2} + 3n - 3l$. Therefore $l \geq 2 + \frac{\sum_{v \in L} d(v)}{6} > 2$ ■

If a vertex, v, is the only active vertex of face f, then f will only be selected (and marked) if v doesn't belong to any other face. Otherwise, it would be dominated (thus marked). We can, therefore, assume that every active face has at least two active vertices.

Faces of length two may exist due to the surgical operation which could introduce multiple edges between two vertices. This case is easily handled by $KERNELIZE$ since two vertices cannot belong to more than one face of length two (by the dominated face rule).

Lemma 6. Let v be an active vertex of an annotated plane graph, G. Then no marked neighbor of v belongs to an active face of v.

Proof The lemma follows immediately from the surgical operation.

Theorem 9. $FACECOVER$ runs in $O(5^k + N^2)$ time.

Proof At each call to $FACECOVER$, the subgraph induced by active vertices of G_1 must have a vertex, v, of degree ≤ 5. By Lemma 6, the active faces in $F(v)$ are faces that are common to v and its active neighbors. Thus the number of such active faces would exceed five only if v has multiple edges with at least one of its neighbors. Which means that v belongs to faces of length two. However, a face of length two will only be active if it's the unique face that is common to its two vertices. This completes the proof. ■

References

1. D. Bienstock and C. L. Monma. On the complexity of covering vertices by faces in a planar graph. *SIAM Journal on Computing*, 17:53–76, 1988.

2. H. L. Bodlaender and F. V. Fomin. Approximation of pathwidth of outerplanar graphs. *Journal of Algorithms*, 43(2):190–200, 2002.
3. H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21:358–402, 1996.
4. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
5. D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 1999.
6. M. R. Fellows and M. A. Langston. Nonconstructive advances in polynomial-time complexity. *Information Processing Letters*, 26:157–162, 1987.
7. R. Govindan, M. A. Langston, and X. Yan. Approximating the pathwidth of outerplanar graphs. *Information Processing Letters*, 68:17–23, 1998.
8. T. Kloks, C. M. Lee, and J Liu. New algorithms for k-face cover, k-feedback vertex set, and k-disjoint cycles on plane and planar graphs. *Proceeding of 28th International Workshop on Graph-Theoretic Concepts in Computer Science*, to appear.
9. P. C. Lui and R. C. Geldmacher. An O(max(m, n)) algorithm for finding a subgraph homeomorphic to K_4. *Congressius Numeratium*, 29:597–609, 1980.
10. R. Niedermeier and P. Rossmanith. An efficient fixed parameter algorithm for 3-hitting set. *Journal of Discrete Algorithms*, 2001.
11. N. Robertson and P. D. Seymour. Graph minors xiii. the disjoint paths problem. *J. Combin. Theory Ser. B*, 63:65–110, 1995.
12. K. Skodinis. Computing optimal linear layouts of trees in linear time. Technical Report MIP-9904, Department of Computer Science, University of Passau, Passau, Germany, 1999.
13. R. Thomas. Private communication, 2001.