Subgraph Isomorphism in Planar Graphs and Related Problems

David Eppstein*

Department of Information and Computer Science
University of California, Irvine, CA 92717

Tech. Report 94-25

May 31, 1994

Abstract

We solve the subgraph isomorphism problem in planar graphs in linear time, for any pattern of constant size. Our results are based on a technique of partitioning the planar graph into pieces of small tree-width, and applying dynamic programming within each piece. The same methods can be used to solve other planar graph problems including diameter, girth, induced subgraph isomorphism, and shortest paths. We also extend our techniques to other families of graphs including the graphs of bounded genus.

*Work supported in part by NSF grant CCR-9258355.

1 Introduction

Subgraph isomorphism is an important and very general form of exact pattern matching. Theoretically, subgraph isomorphism is a common generalization of many important graph problems including finding Hamiltonian paths, cliques, matchings, girth, and shortest paths. Variations of subgraph isomorphism have also been used to model such varied practical problems as molecular structure comparison [2], integrated circuit testing [10], microprogrammed controller optimization [20], analysis of Chinese ideographs [21], robot motion planning [26], semantic network retrieval [28], and polyhedral object recognition [38].

In the general subgraph isomorphism problem, given a "text" G and a "pattern" H, one must either detect an occurrence of H as a subgraph of G, or list all occurrences. For certain choices of G and H there can be exponentially many occurrences, so listing all occurrences can not be solved in subexponential time. Because of reductions from Hamiltonian path and clique finding, the decision problem is NP-complete [19] so subexponential algorithms are unlikely. However for any fixed pattern H with ℓ vertices, both the enumeration and decision problems can easily be solved in polynomial $O(n^\ell)$ time, and for some patterns an even better bound might be possible. Thus one is led to the problem of determining the algorithmic complexity of subgraph isomorphism for a fixed pattern.

Here we consider the special case in which G and H are planar graphs, a restriction naturally occuring in many applications. We show that for any fixed pattern, planar subgraph isomorphism can be solved very efficiently, in time linear in $|G|$. This is the first known algorithm for this problem that is polynomial in G. Our results extend to some other problems including induced subgraph isomorphism and shortest paths.

Our algorithm uses a graph decomposition method similar to one used by Baker [5] to approximate various NP-complete problems on planar graphs. Her method involves removing vertices from the graph leaving a disjoint collection of subgraphs of small tree-width; in contrast we find a collection of non-disjoint subgraphs of small tree-width covering the neighborhood of every vertex.

2 New Results

We prove the following results. We assume here some constant bound on the size of the pattern H; the exact time dependence on H will be described later but is in general exponential.

- We can test whether any fixed pattern H is a subgraph of a planar graph G, or count the number of occurrences of H as a subgraph of G, in time $O(n)$.
- If connected pattern H has k occurrences as a subgraph of a planar graph G, we can list all occurrences in time $O(n + k)$. If H is 3-connected, $k = O(n)$ [15], and we can list all occurrences in time $O(n)$.
- We can count the number of induced subgraphs of a planar graph G isomorphic to any fixed connected pattern H in time $O(n)$, and if there are k occurrences we can list them in time $O(n + k)$.
- For any planar graph G for which we know a constant bound on the diameter, we can compute the exact diameter in time $O(n)$.
- For any constant h we can solve the h-clustering and connected h-clustering problems [23] in planar graphs in time $O(n)$.
- For any planar graph G for which we know a constant bound on the girth, we can compute the exact girth in time $O(n)$. The same bound holds if instead of girth we ask for the shortest nonfacial cycle or the shortest separating cycle.
- For any planar graph G and any constant ℓ, we construct in time $O(n)$ a compact routing data structure which can test for any pair of vertices whether their distance is at most ℓ, and if so find a shortest path between them, in time $O(\log n)$.

Finally, we extend our techniques to other families of graphs. We get linear or quadratic algorithms for any family having a certain relation between diameter and treewidth. In particular, we consider the minor-closed families studied extensively by Robertson and Seymour. We exactly characterize the minor-closed families with the relation needed to make our approach work: they are the families which do not include all apex graphs. We use our characterization to solve subgraph isomorphism in linear time for graphs of bounded genus, and for graphs with no $K_{3,a}$ minor.

3 Related Work

For the general subgraph isomorphism problem, nothing better than the naive $O(n^\ell)$ bound is known. Plehn and Voigt [33] give an algorithm for subgraph isomorphism which in planar graphs takes time $n^{O(\sqrt{\ell})}$, but this is still much larger than the linear bound we achieve.

Several papers have studied planar subgraph isomorphism with restricted patterns. It has long been known that if the pattern H is either K_3 or K_4, then there can be at most $O(n)$ instances of H as a subgraph of a planar graph G, and that these instances can be listed in linear time [6, 22, 32], a fact which has been used in algorithms to test connectivity [27], approximate maximum independent sets [6], and test inscribability [13]. Linear time and instance bounds for K_3 and K_4 can be shown to follow solely from the sparsity properties of planar graphs [11, 12], and similar methods also generalize to problems of finding $K_{2,2}$ and other complete bipartite subgraphs [11, 14]. In [15], we showed how to list all cycles of a given fixed length in *outerplanar* graphs, in linear time (see also [29, 30, 31, 39] for similar variants of outerplanar subgraph isomorphism). We used our outerplanar cycle result to find any *wheel* of a given fixed size in planar graphs, in linear time. Our results here generalize and unify this collection of previously isolated results, and also give improved dependence on the pattern size in certain cases.

Recently we were able to characterize the graphs occurring $O(n)$ times as subgraphs of planar graphs: they are exactly the 3-connected planar graphs [15]. However this result does not extend even to other 3-connected patterns, and our proof that general 3-connected planar graphs have few occurrences does not seem to lead to an efficient algorithm for their enumeration. In this paper we use different techniques which do not depend on high-order connectivity.

Itai and Rodeh [22] discuss the problem of finding the girth of a general graph, or equivalently that of finding short cycles. The special cases of finding $C_3 = K_3$ and $C_4 = K_{2,2}$ in planar graphs were discussed above. Richards [34] gives $O(n \log n)$ algorithms for finding C_5 and C_6 subgraphs, and leaves open the question for larger cycle lengths. Bodlaender [9] discusses the related problem of finding a path or cycle longer than some given length in a general graph, which he solves in linear time for a given fixed length bound. The planar dual to the shortest separating cycle problem has been related by Bayer and Eisenbud [7] to the Clifford index of certain algebraic curves. Again, we give linear time algorithms which unify all these cases.

Our shortest path data structure combines our methods of bounded tree-width decomposition with a separator-based divide and conquer technique due to Frederickson [16]. Obviously all pairs shortest paths can be computed in time $O(nm)$ after which the queries we describe can be answered in time $O(1)$, but some faster algorithms are known for approximate shortest paths: Frederickson and Janardan [17, 18] and Klein and Sairam [24] have described approximate shortest path data structures in planar graphs.

4 Diameter and Tree-Width

In this section we show a key structural property of planar graphs, that if they have low diameter they also have low tree-width. Such a result was implicit already in the work of Baker [5]. With a bound on tree-width we can use dynamic programming techniques to compute many graph properties in linear time [8, 40]. A result similar to the one in this section follows easily from the Robertson-Seymour "wall lemma" [36] (Lemma 5 below). However we give the following direct proof to make explicit the dependence on the diameter, and to show that the result does not introduce any of the scary constants ubiquitous in Robertson-Seymour theory.

We first define the concept of tree-width, introduced by Robertson and Seymour [35] and now standard in graph theory.

Definition 1. *A* tree decomposition *of a graph G is a representation of G as a subgraph of a chordal graph G'. The* width *of the tree decomposition is one less than the size of the largest clique in G'. The* tree-width *of G is the minimum width of any tree decomposition of G.*

The maximal cliques of a chordal graph can be arranged in a tree in such a way that the intersection of any two cliques is a subset of the cliques occurring along the corresponding path in the tree; this tree can be used for many efficient dynamic programming algorithms in treewidth-bounded graphs [8, 40].

The following lemma is the main result of this section.

Lemma 1. *Let planar graph G have diameter D. Then G has tree-width $O(D)$, and a tree-decomposition of G with width $O(D)$ can be found in time $O(Dn)$.*

Proof: We assume without loss of generality that G is maximal planar. We fix an embedding of G, and find in linear time a breadth first search

tree T (starting from any vertex) with depth at most D. We will find a representation of G as a subgraph of a chordal graph G' in which the maximal cliques are the subtrees in T connecting triples of vertices. Any such subtree consists of three paths in T meeting at a single vertex, and contains at most $3D$ vertices.

We form the tree decomposition recursively. Initially, we choose any edge $e = (u, v)$ in $G - T$, and form a clique connecting all vertices on the path from u to v in T. The cycle induced in G by e and this path separates G into two subgraphs, and we will form a decomposition of each subgraph independently.

In the general situation, we will be decomposing a subgraph G' separated from the rest of G by a cycle induced in T by some edge $e = (u, v)$. The cycle itself and edge e will already be represented by a clique in the tree decomposition. Since G is by assumption maximal planar, there will be a single triangular face in G' adjacent to e. Let e_1 and e_2 be the two other edges of G incident to that face. Without loss of generality e_1 is incident to u, e_2 is incident to v, and they are both incident to a third vertex w.

If both e_1 and e_2 are in T, the cycle we are decomposing is simply the triangle (e, e_1, e_2) and the recursion terminates. If one of the two is in T (say e_1 is in T), it is on the path from u to v in T and is already represented by the previously added clique. We continue recursively in the cycle induced by e_2. In the final case, neither e_1 nor e_2 is in T. We add a clique to our tree decomposition, formed by the subtree of T connecting u, v, and w. This clique represents the two cycles induced by edges e_1 and e_2, and we can recursively solve the subproblems within these two cycles.

In time $O(n)$ we can implicitly assign each edge of G to a triple (u, v, w) corresponding to a clique in which the edge is represented. In a further $O(Dn)$ time we can explicitly list the vertices involved in each clique. □

5 Subgraph Isomorphism with Fixed Tree-Width

We next show how to use dynamic programming in graphs of bounded tree-width to perform subgraph isomorphism testing. The exact statement of the problem we solve is complicated by the requirement that we count or list each subgraph isomorph exactly once. For simplicity, we state the lemma with one parameter measuring both the tree-width of the text and the size of the pattern.

Lemma 2. *Assume we are given graph G with n vertices along with a tree decomposition of G with width w. Let S be a subset of the vertices of G, and let H be a fixed graph with at most w vertices. Then in time $O(c^{w \log w} n)$ for some constant c, we can count all isomorphs of H in G that include some vertex in S. In time $O(c^{w \log w} n + kw)$ we can list all such isomorphs.*

Proof: We perform dynamic programming in a tree coming from the tree representation of G. Each node in the tree corresponds to a clique in the tree decomposition of G, and the subtree rooted at that node corresponds to a subgraph separated from the rest of G by the vertices in that clique.

Let a *partial isomorph* at a node N of the tree be an isomorphism between an induced subgraph H' of the pattern H and a subgraph of the portion of G corresponding to the subtree rooted at N.

We let G' be the graph induced in G by the vertices in N, together with two additional vertices, each connected to all vertices in N. Each of the two additional vertices also is given a self-loop. Then from any partial isomorph at N we can derive a graph homomorphism from all of H to G', which is one-to-one on vertices of N, maps the rest of H' to the first additional vertex, and maps $H - H'$ to the second additional vertex in G'. Let a *partial isomorph boundary* be such a map.

There are $O(c_1^{w \log w}$ possible partial isomorph boundaries for a given node, for some constant c_1. For each partial isomorph boundary, in each node, we compute the number of partial isomorphs which give rise to that boundary. We also compute a similar count of those partial isomorphs involving a vertex of S. These numbers can be computed in a straightforward way from the same information at the node's children, by combining the $O(c_1^{w \log w})$ counts from each children in pairs of children at a time, resulting in $O(c_2^{w \log w})$ work per combined pair and $O(c^{\ell \log \ell} n)$ overall work.

At the root node of the tree, we simply sum the number of isomorphs involving S among those partial isomorph boundaries for which none of H is mapped to the second additional vertex. To recover the isomorphs themselves we simply return back through the tree using the already computed counts to determine which portions of the total sum came from which partial isomorphs at each level. □

The same techniques also lead to the same result for counting or listing induced subgraphs isomorphic to H. As a corollary to Lemma 2, we could perform planar subgraph isomorphism for connected patterns in $O(n^2)$ time, by letting $S = \{v\}$ for each vertex v in turn, and by only searching in the

subgraph of G within distance w of v; by Lemma 1 this subgraph has tree-width $O(w)$. We will see later how to use techniques similar to those of neighborhood covers to improve this bound, and how to extend this idea to disconnected patterns.

The following theorem on computing diameter improves the naive $O(n^2)$ bound for all pairs shortest paths when the diameter is small. Note that diameter is not a subgraph isomorphism problem but it succumbs to similar techniques.

Theorem 1. *We can compute the diameter $D = D(G)$ of a planar graph G, in time $O(c^{D \log D} n)$ for some constant c.*

Proof: We can compute an approximation to the diameter by breadth first search from any particular vertex, after which by Lemma 1 we can perform dynamic programming in a tree decomposition of width $O(D)$. We first sweep the tree decomposition and compute for every node the distances in the subtree rooted at that node between every vertex associated with the node. There are $O(D^2)$ distances per node, and two matrices of distances can be combined in $O(D^3)$ time, so this phase takes time $O(D^3 n)$. We then perform a similar sweep to compute distances in G between the same pairs of vertices, in the same time bound. We finally sweep through the tree decomposition a third time, keeping at each node N a set of candidates to be endpoints of the diametral pair. If two candidate vertices have the same set of distances to all vertices in N, we only need to keep one of the two, so $O(c^{D \log D})$ candidates need be kept. At each stage we merge lists of candidates for adjacent nodes in the tree, using the distances computed in the first two sweeps to find the true shortest paths between every pair of candidates. □

6 Neighborhood Covers

We have seen that we can perform subgraph isomorphism quickly in graphs of bounded tree-width, and that the subgraph of any planar graph G induced by the vertices near some particular vertex has bounded tree-width. Therefore we can *cover* G by the collection of all such subgraphs; such a cover has the property that the neighborhood of every vertex is contained in some subgraph of the cover, and that every subgraph of the cover has small tree-width. However the cover is not *efficient*: the total size of all subgraphs is $O(n^2)$, larger than we want.

Awerbuch et al. [3, 4] have introduced the very similar concept of a *neighborhood cover*, which is a covering of a graph by a collection of subgraphs, with the properties that the neighborhood of every vertex is containined in some subgraph, and that every subgraph has small diameter. They showed that any (nonplanar) graph has a neighborhood cover in which the diameter of each subgraph is $O(w \log n)$, and in which the total size of all subgraphs is $O(m \log n)$; such a cover can be computed in time $O(m \log n + n \log^2 n)$ [3].

Neighborhood covers were introduced by Awerbuch and Peleg [4] who used them for distributed computation: one can perform local computations in each cover rather than in the whole graph, since each neighborhood is covered, and the computations terminate quickly since each subgraph has small diameter. Because of the relation between diameter and tree-width in planar graphs, such a neighborhood cover is also almost exactly what we want to speed up our subgraph isomorphism algorithm. However there are two problems. First, the size and construction time of neighborhood covers are higher than we want (albeit only by logarithmic factors). Second, and more importantly, the diameter is sufficiently high that we are unable to use dynamic programming directly in the subgraphs of the cover. We would be forced to use some additional techniques such as separator-based divide and conquer, introducing more unwanted logarithmic factors.

Instead, we use a technique similar to that of Baker [5] to form a cover that has the properties we want directly: the subgraph within distance w of every vertex is included in some covering subgraph, each covering subgraph has tree-width $O(w)$, and each vertex of G is included in $O(1)$ subgraphs (so the total size of all subgraphs is $O(n)$). One also wants a third property that the collection of subgraphs is not much larger than the original graph G. For the distributed computing applications this is expressed in terms of the maximum number of subgraphs any vertex is contained in, but for our purposes we will only need a bound on the total size of all subgraphs (or equivalently on the average number of subgraphs the vertices are contained in).

Lemma 3. *Let G be a planar graph. Then we can find a collection of subgraphs G_i with the following properties:*

- *For every vertex v of G, the subgraph G' induced by the vertices of G within distance w of v is a subgraph of one of the graphs G_i.*
- *Every vertex of G is included in at most three subgraphs G_i.*

- *Every subgraph G_i has tree-width $O(w)$.*

Proof: We choose any vertex v, and form a breadth first search tree from v. This partitions G into layers, so that each edge connects either a pair of vertices in a single layer, or a pair of vertices in adjacent layers. The layers can be numbered by their distance from v. We let the graphs G_i be the induced subgraphs formed by vertices in layers iw to $(i+3)w-1$. Each such graph covers the neighborhoods of the points in layers $(i+1)w$ to $(i+2)w-1$, so every neighborhood is covered. A point in layer j will be covered only by the three graphs $G_{\lfloor j/w \rfloor + k}$ for k in the set $\{-2,-1,0\}$. And every graph G_i is a subgraph of the graph G' formed by removing all layers higher than $(i+3)w-1$, and collapsing into v all layers below iw; the breadth first search tree in G induces a breadth first search tree in G' with radius $3w$. Hence by Lemma 1 G' and its subgraph G_i have tree-width $O(w)$. □

The lemma could be strengthened so that each vertex of G is included in at most two subgraphs, by taking groups of $4w$ layers in the breadth first search tree, but this would increase the constant factor in the $O(w)$ tree-width bound. In fact for our subgraph isomorphism algorithm we could take groups of $2w$ layers, and reduce both the tree-width of each G_i and the total size of all graphs G_i.

7 The Subgraph Isomorphism Algorithm

Theorem 2. *We can count the isomorphs or induced isomorphs of a given connected pattern H, having w vertices, in a planar text graph G with n vertices, in time $O(c^{w \log w} n)$. If there are k such isomorphs we can list them all in time $O(c^{w \log w} n + wk)$.*

Proof: We apply Lemma 3, with $S = V(G)$, to find in time $O(n)$ a set of disjoint subgraphs G_i with tree-width $O(w)$, covering the radius w neighborhoods of all vertices in G. We choose one such subgraph G_i, let S be the vertices in G_i with covered neighborhoods, and find all subgraph isomorphs involving vertices in S using the algorithm of Lemma 2. We then remove S from all other covering subgraphs G_j so that the resulting graphs form a cover of $G-S$, and we continue to use that cover to find all remaining subgraph isomorphs in $G-S$. □

Corollary 1. *We can compute the girth $g = g(G)$ of a planar graph, in time $O(c^{g \log g} n)$.*

Proof: This is equivalent to searching for a pattern H consisting of a cycle of length at most g. We perform binary search among the set of such patterns, increasing the total time by a factor of $O(\log g)$ which is swamped by the $c^{g \log g}$ factor in the time bound. □

We note that instead of girth we can find the shortest nonfacial cycle, in a similar bound, by counting the number of cycles of a given size and comparing that number to the number of faces of the same size.

8 Disconnected Patterns

The methods we have described so far require that the pattern be connected. We now describe a general method for handling disconnected patterns. The technique will enable us to count the number of matching patterns, after which some sort of separator-based divide and conquer can likely be used to find an instance of a matching pattern, but we have been unable to extend this technique to the problem of listing all subgraph isomorphs of a disconnected pattern.

We illustrate our method for graphs with two components. Suppose H has two connected components H_1 and H_2. We can use our algorithm to count separately the number of occurrences of H_1 and H_2; say these numbers are h_1 and h_2. Then there are $h_1 h_2$ ways of embedding H in G such that both H_1 and H_2 are isomorphically mapped but their instances may overlap. There are $O(1)$ planar graphs that could be formed by overlapping H_1 and H_2, each of which is connected, and we may count the occurrences of each by our subgraph isomorphism algorithm. The number of occurrences of H is then simply $h_1 h_2 - \sum k_i$, where the numbers k_i count the number of ways each overlapping graph occurs in G. If some overlapping graph could be formed in multiple ways from H_1 and H_2 we have to count it with an appropriate multiplicity.

The result extends easily to higher numbers of components using a simple inclusion-exclusion principle.

Lemma 4. *Let H have as connected components a collection of subgraphs H_i, and let connected graphs K_j be formed by overlapping sets of the graphs H_i. Then there is a polynomial $p(V)$ such that if for any graph G, k_j denotes*

the number of occurrences of K_j in G and V is the vector $(k_1, k_2, \ldots)$, then $p(V)$ is equal to the number of occurrences of H as a subgraph of G.

Theorem 3. *We can count the isomorphs of any (possibly disconnected) pattern H having a constant number of vertices, in a planar text graph G with n vertices, in time $O(c^{w \log w} n)$*

Proof: Each graph K_j is formed by identifying sets of vertices in H, so there can be at most $c^{w \log w}$ such graphs. For each such graph, we perform the algorithm of Theorem 2, then plug the results into the polynomial p of Lemma 4. Each term of p corresponds to a (possibly disconnected) graph formed by identifying parts of H, so there are $c^{w \log w}$ terms and p can be constructed and evaluated in time $O(c^{w \log w})$. □

The *h-clustering problem* is that of approximating the maximum clique by finding a set of h vertices inducing as many edges as possible. The *connected h-clustering* problem adds the restriction that the induced subgraph be connected. Keil and Brecht [23] study these problems, and show that even though cliques are easy to find in planar graphs [32], the connected h-clustering problem is NP-complete for planar graphs. See [25] for approximate h-clustering algorithms in general graphs. One method for exact solution to the h-clustering problem is simply to test subgraph isomorphism for all possible planar graphs on h vertices.

Corollary 2. *For any h we can solve the planar h-clustering and connected h-clustering problems in time $O(c^{h \log h} n)$.*

9 Improvement for Certain Patterns

For certain patterns, such as the wheels, our results can be further improved to reduce the time dependence on $|H|$. Note that if the diameter $\mathrm{diam}(H)$ is small, we can use that value instead of $|H|$ in our neighborhood cover of G, reducing the tree-width of the subgraphs G_i to $O(\mathrm{diam}(H))$. Lemma 2 can then be improved to have time $O(c^{|H|+\mathrm{diam}(H)\log|H|} n)$. The $c^{|H|}$ term in this bound comes from the fact that in the dynamic programming algorithm we need to keep track not only of how the vertices in a tree-decomposition node of G_i map to H, but also of the connected components of the subgraph of H induced by the unmapped vertices. If the removal of $O(\mathrm{diam}(H))$ vertices from H cannot partition H into many components, this term will vanish.

Theorem 4. *If a given pattern H is Hamiltonian or 3-connected, or if it has bounded degree, we can count the isomorphs of H in a planar text graph G with n vertices in time $O(c^{\mathrm{diam}(H)\log|H|}n)$.*

Proof: We cover G by graphs of treewidth $O(\mathrm{diam}(H))$, and perform dynamic programming within each graph.

At each node of each tree decomposition we store the set of ways a subgraph of H could be mapped to that node and its descendents. Each such map consists of a relation between vertices of the node and vertices of H, together with a set of those components of the remaining vertices of H that are covered by nodes lower in the tree decomposition. There are $(H+1)^{O(\mathrm{diam}(H))}$ possible relations, multiplied by 2^k sets of components where k counts the number of components formed by removing $O(\mathrm{diam(H)})$ vertices from H. In the classes of graphs stated in the lemma, $k = O(\mathrm{diam}(H))$. □

For instance we can count the isomorphs of a wheel W_k in a planar text graph G with n vertices, in time $O(nk^c)$ for some constant c. In fact in this case it is not difficult to come up with an $O(nk^2)$ algorithm directly.

Theorem 5. *We can count the isomorphs of any wheel W_k in a planar text graph G with n vertices in time $O(nk^2)$.*

Proof: For each vertex v, we count the number of cycles of length k in the neighbors of v. The sum of the sizes of all neighborhoods in G is $O(n)$. Each neighborhood is outerplanar and therefore has treewidth 2. We use standard dynamic programming techniques in a tree decomposition of each neighborhood, storing for each length $\ell \leq k$ the number of paths of length ℓ connecting the two vertices in each node. □

10 Shortest Path Data Structure

We next describe a technique for finding shortest paths in planar graphs. Let a parameter ℓ be given (typically, a fixed constant). We wish to test, for any two vertices u and v, whether there is a path from u to v of distance at most ℓ, and if so return the shortest such path.

Theorem 6. *For any planar graph G, and any value of ℓ, we can in time $O(\ell^2 n)$ build a data structure of size $O(\ell n)$, with which we can perform the queries described above in time $O(\ell^2 \log n)$ each.*

Proof: We first apply the decomposition of Lemma 3. This provides a cover of G by subgraphs G_i with total size $O(n)$, with tree-width $O(\ell)$ each, having the property that one such graph contains the radius-ℓ neighborhood around each vertex u. Then any query (u, v) need be asked only within that one graph.

Since each G_i has tree-width $O(\ell)$, there is some set of $O(\ell)$ vertices which separate G_i into components of fewer than $n_i/2$ vertices each. Repeating this separation recursively we can find a separator tree for G_i in which each separator has size $O(\ell)$.

We then construct the following data structure [16] using this separator tree. For each separator of size s we store a dense $s \times s$ matrix of the distances between every pair of separator vertices. These matrices can be computed in time $O(s^3)$ per separator by a two-pass dynamic programming algorithm. Since each s is $O(\ell)$, and each vertex of G_i is in one separator, the total time for this construction is $O(\ell^2 n)$ and the matrices take space $O(\ell n)$ to store.

To answer a query, we combine the $O(\log n)$ matrices from the path in the separator tree connecting the two vertices. This combination can be viewed as a weighted shortest path problem in a graph with $O(\ell \log n)$ vertices and $O(\ell^2 \log n)$ edges, each with a weight between 1 and ℓ, which therefore takes time $O(\ell^2 \log n)$. □

11 General Families of Graphs

We next consider other families of graphs than the planar ones. For which families does our subgraph isomorphism technique work?

Definition 2. *Family $\mathcal{F}$ of graphs has the* diameter-treewidth property *if there is some function $f(D)$ such that every graph in $\mathcal{F}$ with diameter at most D has tree-width $f(D)$.*

Then Lemma 1 can be rephrased as showing that the planar graphs have the diameter-treewidth property with $f(D) = O(D)$. With such a property, Lemma 2 can be used to solve subgraph isomorphism in $\mathcal{F}$ for any fixed connected pattern in time $O(n^2)$. Lemma 4 applies without regard for $\mathcal{F}$, and shows that subgraph isomorphism can always be solved for disconnected patterns as quickly as it can for connected patterns.

For planar graphs, we were able to use the decomposition into pieces of low tree-width proved in Lemma 3 to speed up the time from quadratic to

linear. The proof of Lemma 3 relies on the diameter-treewidth property, and on another key property of planar graphs: any *minor* (subgraph of a contraction) of a planar graph is also planar. Thus we are led to the study of families closed under minors. These minor-closed families have been studied extensively by Robertson, Seymour, and others, and include such familiar graph families as the planar graphs, outerplanar graphs, graphs of bounded genus, graphs of bounded tree-width, and graphs embeddable in $\mathbb{R}^3$ without any linked or knotted cycles. In this section we exactly characterize those minor-closed families of graphs having the diameter-treewidth property, in a manner similar to Robertson and Seymour's characterization of the minor-closed families with bounded treewidth as being those families that do not include all planar graphs [36].

Definition 3. *An* apex graph *[42] is a graph G such that for some vertex v (the* apex*), $G - v$ is planar.*

Apex graphs are also known as nearly-planar graphs, and have been introduced to study linkless 3-dimensional embeddings of graphs [37]. The significance of apex graphs for us is that they provide examples of graphs without the diameter-treewidth property: let G be an $n \times n$ planar grid, and let G' be the apex graph formed by connecting some vertex v to all vertices of G; then G' has treewidth $n+1$ and diameter 2. Apex graphs will figure prominently in our characterization of families having the diameter-treewidth property.

Definition 4 (Robertson and Seymour [36]). *A* wall *is a subdivision of the hexagonal tiling of a region of the plane. The* size *of a wall is the number of tiles on the shortest path from some central tile to the boundary of the tiled region.*

Walls are very similar to planar grid graphs but have a slight advantage of having degree three. Thus we can hope to find them as subgraphs rather than as minors in other graphs.

Lemma 5 (Robertson and Seymour [36]). *For any s there is a number $w = W(s)$ such that any graph of treewidth w or larger contains as a subgraph a wall of size s.*

Lemma 6 (Robertson and Seymour [36]). *Let G be a planar graph. Then there is some $s = s(G)$ such that any wall of size s has G as a minor.*

Theorem 7. *Let $\mathcal{F}$ be a minor-closed family of graphs. Then $\mathcal{F}$ has the diameter-treewidth property iff $\mathcal{F}$ does not contain all apex graphs.*

Proof: One direction is easy: we have seen that the apex graphs do not have the diameter-treewidth property, so no family containing all apex graphs can have the property.

In the other direction, we wish to show that if $\mathcal{F}$ does not have the diameter-treewidth property, then it contains all apex graphs. By Lemma 6 it will suffice to find a in $\mathcal{F}$ formed by connecting some vertex v to all the vertices of a wall of size n, for any given n. If $\mathcal{F}$ does not have the diameter-treewidth property, there is some D such that $\mathcal{F}$ contains graphs with diameter D and with arbitrarily large tree-width.

Let G be a graph in $\mathcal{F}$ with diameter D and tree-width $W(N_1)$ for some large N_1 and for the function $W(N)$ shown to exist in Lemma 5. Then G contains a wall of size N_1. We partition the wall into smaller regions, themselves walls of size N_2 and arranged in the form of a wall of size N_3. Thus there are $\Theta(N_3^2)$ regions. Choose any vertex $v \in G$ and find a tree of shortest paths from v to each of the regions. Since G has diameter D, the tree will have height D and there must be some level ℓ of the tree for which the number N_4 of regions reached is larger by a factor of $N_3^{2/D}$ than the number of regions represented by vertices of all previous tree levels combined.

We then contract levels 1 through $\ell - 1$ of the tree to a single vertex v. This gives a minor of G in which v is connected to N_4 distinct regions of our original wall, and in which $N_4/N_3^{2/D}$ other regions are "damaged" by having a vertex included in the contracted portion of the tree. We find a subset S of $\Theta(N_4)$ of the regions connected to v, so that no two regions are adjacent, and so that no region is adjacent to a damaged region. Thus each region in S is surrounded by a larger wall, and the edge between v and the region has its endpoint near the center of the larger wall.

Since S still has many more regions than were damaged, using an isoperimetric inequality for grid graphs we can find a subset S' of at least $\Omega(N_3^{2/D})$ regions such that all of S' can be connected by chains of undamaged regions. If $N_2 = \Omega(n)$ and $|S'| = \Omega(n^2)$, we can use this connected series of wall regions to find a minor M of G consisting of a wall of size n with each vertex connected to v. These conditions can both be assured by letting $N_1 = \Omega(n)^{D+1}$. We can carry out this construction for any n, and since by Lemma 6 every apex graph can be found as a minor of graphs of the form of M, all apex graphs are minors of graphs in $\mathcal{F}$ and are therefore themselves

graphs of $\mathcal{F}$. □

We next discuss applications of this characterization to standard families of graphs, including graphs of bounded genus.

Lemma 7. *For any g there is an apex graph with genus more than g.*

Proof: The graphs of genus g have by Euler's formula at most $3n + O(g)$ edges. Any apex graph formed by connecting the apex to every vertex of a maximal planar graph will have $4n - 10$ edges. By choosing n large enough one can find an apex graph with too many edges to have genus g. □

The next family of graphs we consider are those having no $K_{3,a}$ minor for some fixed a. These are interesting as a generalization of planar graphs (which are those without a $K_{3,3}$ or K_5 minor) and because our previous characterization of the subgraphs occurring linearly many times in planar graphs has the following generalization:

Theorem 8 (Eppstein [15]). *Let $\mathcal{F}_a$ be the family of graphs having no $K_{3,a}$ minor, and let pattern H be a graph in $\mathcal{F}_a$. Then there is a bound of $O(n)$ on the number of times H can occur as subgraphs of graphs graphs in $\mathcal{F}_a$, iff H is 3-connected.*

Lemma 8. *There is an apex graph G that is not in $\mathcal{F}_a$.*

Proof: Let $G = K_{3,a}$. □

Corollary 3. *For any fixed pattern H, we can test subgraph isomorphism for H in graphs with any fixed bound on the genus, or in graphs with no $K_{3,a}$ minor for any fixed a, in time $O(n)$.*

12 Conclusions and Open Problems

We have shown how to solve planar subgraph isomorphism for any pattern in time $O(n)$. We have also solved certain related problems in similar time bounds. A number of generalizations of the problem remain open:

- We have shown that we can solve planar subgraph isomorphism even for disconnected patterns in time $O(n)$. Can we list all occurrences of a disconnected pattern in time $O(n + k)$?

- Bui and Peck [41] describe an algorithm for finding the smallest set of edges partitioning a planar graph into two sets of vertices with specified sizes; if the edge set has bounded size their algorithm has cubic running time. Can we use our methods to find such a partition more quickly?

- We have generalized our technique to certain minor-closed families of graphs, and characterized those minor-closed families for which it applies. However the relation we showed between diameter and tree-width was not as strong as for planar graphs: for planar graphs $w = O(d)$ while for other minor-closed families our proof only shows that $w = W(c^{d+1}))$ for some constant c, where $W(x)$ represents the rapidly-growing function used by Robertson and Seymour to prove Lemma 5. Can we prove tighter bounds on tree-width for general minor-closed families?

- Are there natural families of graphs that are not minor-closed and that have the diameter-treewidth property?

- Our previous results on subgraph multiplicity [15] included the fact that in any family of graphs with no $K_{a,b}$ minor, the a-connected subgraphs could only have $O(n)$ subgraph isomorphs. How quickly can we list all such isomorphs? Our results on minor-closed families cover the case that $a = 3$, and show that different techniques will be needed for larger values of a.

- It seems possible that the recently discovered randomized coloring technique of Alon et al. [1] can improve the dependence on the size of the pattern from $O(c^{w \log w})$ to $O(c^w)$, but only for the decision problem of subgraph isomorphism. Can we achieve similar improvements for the counting and listing versions of the subgraph isomorphism problem?

Acknowledgements

This work was supported in part by NSF grant CCR-9258355. I thank Sandy Irani and George Lueker for helpful comments on a draft of this paper.

References

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs. In *Proc. 26th ACM Symp. Theory of Computing*, 1994. To appear.

[2] P. J. Artymiuk, P. A. Bath, H. M. Grindley, C. A. Pepperrell, A. R. Poirrette, D. W. Rice, D. A. Thorner, D. J. Wild, P. Willett, F. H. Allen, and R. Taylor. Similarity searching in databases of three-dimensional molecules and macromolecules. *J. Chemical Information and Computer Sciences*, 32:617–630, 1992.

[3] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. Near-linear cost sequential and distributed constructions of sparse neighborhood covers. In *Proc. 34th IEEE Symp. Foundations of Computer Science*, pages 638–647, 1993.

[4] B. Awerbuch and D. Peleg. Sparse partitions. In *Proc. 31st IEEE Symp. Foundations of Computer Science*, pages 503–513, 1990.

[5] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. In *Proc. 24th IEEE Symp. Foundations of Computer Science*, pages 265–273, 1983.

[6] R. Bar-Yehuda and S. Even. On approximating a vertex cover for planar graphs. In *Proc. 14th ACM Symp. Theory of Computing*, pages 303–309, 1982.

[7] D. Bayer and D. Eisenbud. Graph curves. *Advances in Mathematics*, 86:1–40, 1991.

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

[9] H. L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14:1–23, 1993.

[10] A. D. Brown and P. R. Thomas. Goal-oriented subgraph isomorphism technique for IC device recognition. *IEE Proceedings I (Solid-State and Electron Devices)*, 135:141–150, 1988.

[11] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Computing*, 14:210–223, 1985.

[12] M. Chrobak and D. Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theoretical Computer Science*, 86:243–266, 1991.

[13] M. B. Dillencourt and W. D. Smith. A linear-time algorithm for testing the inscribability of trivalent polyhedra. In *Proc. 8th ACM Symp. Computational Geometry*, pages 177–185, 1992.

[14] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. Manuscript.

[15] D. Eppstein. Connectivity, graph minors, and subgraph multiplicity. *J. Graph Theory*, 17:409–416, 1993.

[16] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Computing*, 16:1004–1022, 1987.

[17] G. N. Frederickson and R. Janardan. Efficient message routing in planar networks. *SIAM J. Computing*, 18:843–857, 1989.

[18] G. N. Frederickson and R. Janardan. Space-efficient message routing in c-decomposable networks. *SIAM J. Computing*, 19:14–30, 1990.

[19] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman, 1979.

[20] A. Guha. Optimizing codes for concurrent fault detection in microprogrammed controllers. In *Proc. IEEE Intl. Conf. Computer Design: VLSI in Computers and Processors (ICCD '87)*, pages 486–489, 1987.

[21] Hong Dong, Youshou Wu, and Xiaoqiag Ding. An ARG representation for Chinese characters and a radical extraction based on the representation. In *9th IEEE Intl. Conf. Pattern Recognition*, volume 2, pages 920–922, 1988.

[22] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM J. Computing*, 7:413–423, 1978.

[23] J. M. Keil and T. B. Brecht. The complexity of clustering in planar graphs. *J. Combinatorial Mathematics and Combinatorial Computing*, 9:155–159, 1991.

[24] P. N. Klein and S. Sairam. Fully dynamic approximation schemes for shortest path problems in planar graphs. In *Proc. 3rd Worksh. Algorithms and Data Structures*, pages 442–451. Lecture Notes in Computer Science 709, Springer-Verlag, Berlin, 1993.

[25] G. Kortsarz and D. Peleg. On choosing a dense subgraph. In *34th IEEE Symp. Foundations of Computer Science*, pages 692–703, 1993.

[26] S. Y. T. Lang and A. K. C. Wong. A sensor model registration technique for mobile robot localization. In *Proc. 1991 IEEE Intl. Symp. Intelligent Control*, pages 298–305, 1991.

[27] J.-P. Laumond. Connectivity of plane triangulations. *Information Processing Letters*, 34:87–96, 1990.

[28] R. Levinson. Pattern associativity and the retrieval of semantic networks. *Computers & Mathematics with Applications*, 23:573–600, 1992.

[29] A. Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science*, 63:295–302, 1989.

[30] A. Lingas and A. Proskurowski. On parallel complexity of the subgraph homeomorphism and the subgraph isomorphism problem for classes of planar graphs. *Theoretical Computer Science*, 68:155–173, 1989.

[31] A. Lingas and M. M. Syslo. A polynomial-time algorithm for subgraph isomorphism of two-connected series-parallel graphs. In *Proc. 15th Int. Colloq. Automata, Languages and Programming*, pages 394–409. Lecture Notes in Computer Science 317, Springer-Verlag, 1988.

[32] C. H. Papadimitriou and M. Yannakakis. The clique problem for planar graphs. *Information Processing Letters*, 13:131–133, 1981.

[33] J. Plehn and B. Voigt. Finding minimally weighted subgraphs. In *Proc. 16th Intl. Worksh. WG90, Graph-Theoretic Concepts in Computer Science*, pages 18–29. Lecture Notes in Computer Science 484, Springer-Verlag, 1991.

[34] D. Richards. Finding short cycles in planar graphs using separators. *J. Algorithms*, 7:382–394, 1986.

[35] N. Robertson and P. D. Seymour. Graph minors II: algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.

[36] N. Robertson and P. D. Seymour. Graph minors V: excluding a planar graph. *J. Combinatorial Theory B*, 41:92–114, 1986.

[37] N. Robertson, P. D. Seymour, and R. Thomas. A survey of linkless embeddings. In *Graph Structure Theory: Proc. Joint Summer Conf. Graph Minors*, pages 125–136. Contemporary Mathematics 147, Amer. Math. Soc., 1991.

[38] T. Stahs and F. Wahl. Recognition of polyhedral objects under perspective views. *Computers and Artificial Intelligence*, 11:155–172, 1992.

[39] M. M. Syslo. The subgraph isomorphism problem for outerplanar graphs. *Theoretical Computer Science*, 17:91–97, 1982.

[40] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *J. Assoc. Computing Machinery*, 29:623–641, 1982.

[41] Thang Nguyen Bui and A. Peck. Partitioning planar graphs. *SIAM J. Computing*, 21:203–215, 1992.

[42] D. J. A. Welsh. Knots and braids: some algorithmic questions. In *Graph Structure Theory: Proc. Joint Summer Conf. Graph Minors*, pages 109–124. Contemporary Mathematics 147, Amer. Math. Soc., 1991.