Fast Parallel Algorithms for the Subgraph Homeomorphism & the Subgraph Isomorphism Problems for Classes of Planar Graphs

Andrzej Lingas
Andrzej Proskurowski

CIS-TR-88-04
April 6, 1988

Department of Computer and Information Science
University of Oregon

Fast Parallel Algorithms for the Subgraph Homeomorphism and the Subgraph Isomorphism Problems for Classes of Planar Graphs

Andrzej Lingas
Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden

Andrzej Proskurowski
Department of Computer and Information Science
University of Oregon, Eugene, Oregon 97403, USA

Abstract

We consider the problems of subgraph homeomorphism with fixed pattern graph, recognition, and subgraph isomorphism for some classes of planar graphs. Following the results of Robertson and Seymour on forbidden minor characterization, we show that the problems of fixed subgraph homeomorphism and recognition for any family of planar graphs closed under minor taking are in $\mathcal{NC}$ (*i.e.*, they can be solved by an algorithm running in poly-log time using polynomial number of processors). We also show that the related subgraph isomorphism problem for biconnected outerplanar graphs is in $\mathcal{NC}$. This is the first example of a restriction of subgraph isomorphism to a non-trivial graph family admitting an $\mathcal{NC}$ algorithm.

1 Introduction

The subgraph homeomorphism problem is to determine whether a graph is homeomorphic to a subgraph of another graph. A graph H is homeomorphic to a graph G if the graph resulting from contracting all maximal paths in G with inner vertices of degree two to single edges is isomorphic to H. Thus, the subgraph homeomorphism problem can be viewed as a generalization of the subgraph isomorphism problem. The latter problem is to determine whether a graph is isomorphic to a subgraph of another graph. For instance, if H is an n-vertex circuit and G is an n-vertex planar graph of valence 3, $n \in N$, then determining whether H is homeomorphic

to a subgraph of G is equivalent to determining whether H is isomorphic to a subgraph of G which is in turn equivalent to the $\mathcal{NP}$-complete problem of determining whether a planar graph of valence 3 has a Hamiltonian circuit [GJ]. Thus, the subgraph homeomorphism and isomorphism problems are $\mathcal{NP}$-complete even if G and H range only over connected planar graphs of valence 3. Subgraph isomorphism also remains $\mathcal{NP}$-complete when the first input graph is a forest and the other input graph is a tree [GJ]. Analogous $\mathcal{NP}$-completeness results hold for the directed versions of the two problems [GJ]. However, in this paper we consider only undirected graphs.

If we fix the first graph H as a pattern graph then this fixed subgraph isomorphism problem is trivially solvable in polynomial time while the fixed subgraph homeomorphism problem remains open (in the undirected case) [As, FHW]. On the other hand, there are two known restrictions of the general subgraph isomorphism problem to non-trivial graph families that are solvable in polynomial sequential time. Trees constitute one of these families [Ma, Rey], the other is the class of biconnected outerplanar graphs [Li86]. (A graph is outerplanar if it can be embedded in the plane such that all its vertices lie on the outer face [H, Mi].)

Another view of the subgraph homeomorphism problem is as a generalization of the disjoint connected paths problem, DCP, defined as follows: Given a graph G and a set of vertex pairs $(s_i, t_i), 1 \leq i \leq k$, decide whether there exists a set of pairwise vertex-disjoint paths P_i in G connecting s_i with t_i. Also this problem is $\mathcal{NP}$-complete even if restricted to planar graphs [FHW]. However, if the number k of vertex pairs is bounded the problem seems to become more tractable. For $k = 2$, there are polynomial algorithms for DCP (see [GJ,RS85]). Recently, Robertson and Seymour have showed that for an arbitrary fixed k, the DCP problem restricted to planar graphs is solvable in polynomial time [RS85]. Since the fixed subgraph homeomorphism problem is trivially polynomial-time reducible to the DCP problem with appropriately chosen fixed k, the result of Robertson and Seymour yields also a polynomial time solution to the problem of fixed subgraph homeomorphism for planar graphs.

A well known application of the subgraph homeomorphism problem is the recognition problem for classes of graphs that can be characterized by the absence of some forbidden substructures. For instance, the recognition problem for planar graphs can be reduced to two fixed subgraph homeomorphism problems with K_5 and $K_{3,3}$ as the pattern graphs (see, for instance [H]). A similar characterization of planar graphs is provided by the operation of minor taking which can be viewed as a generalization of the path contraction operation used in the definition of subgraph homeomorphism. A graph H is a minor of another graph G if G can be reduced

to H by deletion and contraction of edges. A graph is planar if and only if it does not have a minor isomorphic to K_5 or $K_{3,3}$. Several other non-trivial classes of graphs can be also characterized by a finite list of forbidden minors, for instance outerplanar graphs, series-parallel graphs, partial 3-trees, bounded genus graphs [CH,APC,RS86]. For fixed H, the problem of testing whether G has a minor isomorphic to H can be be reduced to a finite number of fixed subgraph homeomorphism problems [RS85]. Thus, the problems of subgraph homeomorphism, DCP, minor containment, and recognition for several classes of graphs are intimately related. Often a solution to one of them yields solutions to the others.

As yet, there are no known efficient parallel algorithms even for the fixed versions of these problems (with the exception of the DCP problem with $k = 1$ [QD]). The situation is similar in the case of restricted subgraph isomorphism problems. It is only known that subgraph isomorphism for trees is solvable by a Las Vegas $\mathcal{NC}$ algorithm [LK].

In this paper, we consider the problems of DCP with fixed k, fixed subgraph homeomorphism, and recognition for any class of planar graphs closed under minor taking, and the subgraph isomorphism problem for biconnected outerplanar graphs. We show that all these problems can be solved in poly-log time using a polynomial number of processors, *i.e.*, they are in the class $\mathcal{NC}$ [Co,Ru,P]

Our parallel algorithms for the problems of DCP with fixed k, fixed subgraph homeomorphism, and recognition for any class of planar graphs closed under minor taking rely on the idea of a sequential method of Robertson and Seymour for the so restricted DCP problem [RS85]. In turn, the sequential method is implied by their interesting, bounded separator theorem for the above classes of planar graphs [RS85].

The subgraph isomorphism problem for biconnected outerplanar graphs can be sequentially solved by a recursive reduction to a monotone path finding problem in cubic time [Li86]. (Note that the reduction of PARTITION to this restriction of subgraph isomorphism given in [Sy82] is pseudo-polynomial and does not establish its $\mathcal{NP}$-completeness [Sy85].) Unfortunately, the recursive depth of the sequential algorithm in [Li86] is proportional to the size of the input graphs in the worst case. To obtain a recursive, efficient parallel algorithm, we need decrease the size of the input graphs by a constant factor in each recursive call. A straight forward way of doing it by using a two-vertex '$\frac{1}{3} - \frac{2}{3}$' separator [LT] in the first graph and guessing its image in the second graph can lead to hyper-polynomial number of considered components of the second graph.

We present a parallel algorithm for subgraph isomorphism restricted to biconnected outerplanar graphs, using a two-level, outerplanar graph cutting technique,

and show that it can be implemented by uniform circuits of poly-log depth and polynomial size. In this way, we establish the membership of this restriction of the subgraph isomorphism problem in the class $\mathcal{NC}$. No other restriction of the subgraph isomorphism problem to a non-trivial graph family is presently known to be in $\mathcal{NC}$.

We leave a more precise estimation of the parallel complexity of the presented algorithms to the final version. However, even not very sophisticated circuit implementations of our algorithms yield the membership of the considered problems in $\mathcal{NC}^3$. It is worth pointing out that analogous $\mathcal{NC}$ algorithms can be designed for the construction version of the considered decision problems.

The paper is organized as follows: In Section 2 we derive the $\mathcal{NC}$ algorithms for the problems of DCP with fixed k, fixed subgraph homeomorphism, and recognition for any class of planar graphs closed under minor taking. In Section 3 we reduce the subgraph isomorphism problem for biconnected outerplanar graphs to a more restricted problem of polygon imbedding. In Section 4 we present a parallel algorithm for the latter problem which implies the membership of the former problem in $\mathcal{NC}$. In Section 5 we discuss further potential applications of our techniques. In particular, they probably could be used to establish the membership of the subgraph isomorphism problem for three-connected planar graphs of bounded width in $\mathcal{NC}$.

In this paper, we use standard set and graph theoretic notation and definitions (for instance, see [AHU,H]). For the definitions of parallel random access machine, uniform circuit families, the classes $\mathcal{NC}^k, \mathcal{NC}$, and the corresponding notions of reducibility, the reader is referred to [C,P,Rei,Ru].

2 The recognition and subgraph homeomorphism problems for classes of planar graphs

To start with we need the following definition.

Definition 2.1 A class of graphs $\mathcal{F}$ is said to be *minor-closed* (closed under minor taking operation) if for every graph $G \in \mathcal{F}$ all its minors are also in $\mathcal{F}$. For a minor-closed class $\mathcal{F}$, a graph $H \notin \mathcal{F}$ is a *minimal forbidden minor* if every minor of H is in $\mathcal{F}$.

Robertson and Seymour [RS86] consider characterization of minor-closed classes of planar graphs through finite sets of minimal forbidden minors.

Fact 2.1[RS86]: For any minor-closed class of graphs there is a finite set of minimal forbidden minors. ▪

In [RS85], the problem of testing if a given graph G has a minor isomorphic to a fixed graph H is reduced to the problem of subgraph homeomorphism for G with respect to a set of *fixed* graphs. H is a minor of G if and only if G has a subgraph homeomorphic to any graph from a finite list of graphs derived from H.

Fact 2.2[RS85, Thm.4.1]: Let H be a graph. There is a finite list of graphs $H_1, \ldots, H_n$ such that for any graph G the following are equivalent:
(i) G has a minor isomorphic to H.
(ii) G contains a subgraph homeomorphic to one of $H_1, \ldots, H_n$. ▪

By the above fact, it suffices to solve the subgraph homeomorphism problem by an $\mathcal{NC}$ algorithm to show that the minor containment problem is in $\mathcal{NC}$. In their general form, the two problems can be probably solved by parallelization of a sequential algorithm outlined in a recent work of Robertson and Seymour [RS86]. In this paper, however, we consider only the case of planar graphs.

We follow Robertson and Seymour [RS85] in using their bounded separator theorem to 'divide and conquer' the complexity of the problem. A *separation* (V_1, V_2) of G is a pair of subsets of $V(G)$ such that $V_1 \cup V_2 = V(G)$ and no edge of G joins a vertex of $V_1 - V_2$ with a vertex of $V_2 - V_1$.

Fact 2.3[RS85, Thm.4.2]: For any planar graph H there is a number N with the following property. For every graph G with no minor isomorphic to H, and every subset X of $V(G)$, there is a separation (V_1, V_2) of G such that $\#((V_1 - V_2) \cap X), \#((V_2 - V_1) \cap X) \leq \frac{2}{3}\#(X)$ and $\#(V_1 \cap V_2) \leq N$. ▪

Below, we specify more formally the problem of disjoint connecting paths to which the subgraph homeomorphism problem easily reduces.

Definition 2.2 Given a graph G and two *terminal vertices* $s, t \in V(G)$, a *path of length k connecting* s and t is a sequence of vertices $v_0, v_1, ..., v_k$ such that $v_0 = s$, $v_k = t$, and $(v_{i-1}, v_i) \in E(G)$ for every $i, 1 \leq i \leq k$. Two paths are *disjoint* if they have no common non-terminal vertices. An instance of *disjoint connecting paths* problem is an undirected graph G and a *terminal set* $P = \{(s_1, t_1), ..., (s_n, t_n)\}$ which is a subset $V(G) \times V(G)$. We will denote the set of terminal vertices in P by $V(P)$. A set M of k disjoint paths in G is a *DCP* for P if there is a bijection $b : P \rightarrow M$ such that for $(s, t) \in P$, $b((s, t))$ connects s with t.

In the following, we describe a procedure which for a given planar graph H and the corresponding integer N specified in Fact 2.3, takes as the input a graph G and a terminal set P in G with $\#V(P) \leq k$. The procedure either reports that G has a minor isomorphic to H or reports whether DCP for P in G exists. In the body of the procedure, called *DCP-or-Minor*, the instruction **halt** is interpreted as terminating the execution of the parallel procedure on all recursion levels. *DCP-or-Minor* uses an auxiliary procedure *Divide*. Given a separation (V_1, V_2) of a graph G and a terminal set P in G, *Divide* returns a family of pairs of terminal sets, P_1 and P_2, representing all possible reductions of the original DCP problem for P in G to DCP subproblems for P_1 in $G(V_1)$ and P_2 in $G(V_2)$.

procedure *Divide*(V_1, V_2, P);
begin
Construct the graph $L = (V(L), E(L))$ where $V(L) = L_1 \cup L_2 \cup L_3$ and L_1, L_2, L_3 are defined as follows: $L_1 = (V_1 - V_2) \cap V(P)$, $L_2 = (V_2 - V_1) \cap V(P)$, and $L_3 = V_1 \cap V_2$. $E(L)$ consists of all edges between L_1 and L_3, all edges between L_2 and L_3, and all edges between vertices of L_3;
for every set M of disjoint connecting paths for P in L
do in parallel
begin set Q to the set of edges of the paths in M;
return $P_1 = (V_1 \times V_1) \cap Q$, and $P_2 = (V_2 \times V_2) \cap Q$
end-do
end*(Divide)*;

DCP-or-Minor is now defined as follows.

procedure *DCP-or-Minor*(G, P, H, N);
$G = (V(G), E(G))$ is the input graph;
P is the input terminal set with at most k terminal pairs;
$H = (V(H), E(H))$ is the fixed planar minor;
N is the constant implied for H by Fact 2.3.
{Returns one of the following three answers:
(i) 'minor': G has a minor isomorphic to H. This follows from Fact 2.3 when no separation satisfying the theorem exists.
(ii) 'DCP exists': there exist disjoint connecting paths for P in G.
(iii) 'no DCP': there is no disjoint connecting paths for P in G }

begin {DCP-or-Minor}
if $\#V < 3N$

then return the answer through solving the DCP
problem for P in $G(V)$ by brute force
else begin $\{\#V \geq 3N\}$
for every subset S of N vertices of $V(G)$ **do in parallel**
begin (S)
find the connected components $C_1, ..., C_l$ of $G(V - S)$;
if there is a separation (V_1, V_2) such that
$V_1 \cap V_2 = S,\ V_1 \cup V_2 = V(G),\ \#(V_1 - V_2), \#(V_2 - V_1) \leq \frac{2}{3}\#V(G)$,
and V_1, V_2 are sums of $C_i \cup S$
then insert (V_1, V_2) into *separations*
end-do(S);
if *separations*$=\emptyset$
then return 'minor' and **halt**
else select (V_1, V_2) from *separations*;
for every pair (P_1, P_2) of terminal sets in *Divide*(V_1, V_2, P)
do in parallel
begin(P_1, P_2)
for $i = 1, 2$ **do in parallel**
begin(i)
if $\#V(P_i) \leq \max(5N, 2k)$ $\{k$ is the fixed bound on $\#P\}$
then
begin if *DCP-or-Minor*$(G(V_i), P_i, H, N)$ returns 'DCP exists'
then $DCP[V_i, P_i] :=$ **true**
end(then-clause)
else begin $\{\#V(P_i) > \max(5N, 2k)\}$
for every subset T of N vertices of V_i **do in parallel**
begin (T)
find the connected components $D_1, ..., D_l$ of $G(V_i - T)$;
if there is a separation (W_1, W_2) such that
$W_1 \cap W_2 = T,\ W_1 \cup W_2 = V_i$,
$\#((W_1 - T) \cap V(P_i)), \#((W_2 - T) \cap V(P_i)) \leq \frac{2}{3}\#V(P_i)$,
and W_1, W_2 are sums of $D_i \cup T$
then insert (W_1, W_2) into *separations*$[i]$
end-do(T);
if *separations*$[i]=\emptyset$
then return 'minor' and **halt**
else select (W_1, W_2) from *separations*$[i]$;
for every pair of terminal sets Q_1, Q_2 in *Divide*(W_1, W_2, P_i)

do in parallel
begin (Q_1, Q_2)
for $j = 1, 2$ **do in parallel**
begin (j)
set *answer*$[j]$=*DCP-or-Minor*$(G(W_j), Q_j, H, N)$
end-do(j);
if *answer*[1] and *answer*[2] report existence of DCP
then $DCP[V_i, P_i] :=$ **true**
end-do(Q_1, Q_2);
if $DCP[V_i, P_i] \neq$ **true then** $DCP[V_i, P_i] :=$ **false**
end {else-clause($\#V(P_i) > \max(5N, 2k)$)}
end-do(i);
if $DCP[V_1, P_1] \wedge DCP[V_2, P_2]$ **then** $DCP[V, P] :=$ **true**
end-do(P_1, P_2);
if $DCP[V, P]$ **then** return('DCP exists') **else** return('no DCP')
end*(DCP-or-Minor)*

In the following we develop two lemmas asserting the correctness of the above procedure and the possibility of its efficient parallel implementation.

Lemma 2.1: The procedure *DCP-or-Minor* is correct.

Proof: If the procedure returns 'minor', the correctness of the answer follows immediately from Fact 2.3, since no postulated separation was found, as either *separation* or *separation*$[i]$ was found empty. If the procedure returns 'DCP exists' or 'no DCP' in G for P then the correctness of the answer is implied by the following two claims:

(i) $DCP[V_1, P_1]$ and $DCP[V_2, P_2]$ are correctly evaluated for all pairs (P_1, P_2) of terminal sets produced by *Divide*(V_1, V_2, P).

(ii) For the graph G with $V(G) > 3N$ and a separation (V_1, V_2) fulfilling the conditions of Fact 2.3, there exist DCP in G for P if and only if there is a pair (P_1, P_2) of terminal sets produced by *Divide*(V_1, V_2, P) such that both DCP in $G(V_1)$ for P_1 and DCP in $G(V_2)$ for P_2 exist.

In turn, the correctness of evaluating $DCP[V_i, P_i]$ results from the inductive hypothesis asserting the correctness of *DCP-or-Minor* for smaller graphs and from the following: If $\#V(P_i) > \max(5N, 2k)$, and the required separation (W_1, W_2) of $G(V_i)$ exists then DCP in $G(V_i)$ for P_i exist if and only if there is a pair (Q_1, Q_2) of terminal sets produced by *Divide*(W_1, W_2, P_i) such that for $j = 1, 2$, *DCP-or-Minor*$(G(W_j), Q_j, H, N)$ returns 'DCP exists'.

To complete the proof it remains to show that *DCP-or-Minor* always terminates. The latter follows from the fact that in any recursive call of *DCP-or-Minor* the size of the new input graph is smaller than than that of the original graph. ▪

Lemma 2.2. Let H be a fixed planar graph, N a fixed integer satisfying the thesis of Fact 2.3 for H, and let k be a fixed integer. For any graph G and any terminal set P in G with at most k terminal pairs, the procedure *DCP-or-Minor*(G, P, H, N) can be realized by an $\mathcal{NC}$ algorithm.

Proof: By [SV, Theorem 1] it suffices to show that the procedure will execute in poly-log time when carefully implemented on a parallel random access machine with concurrent read and concurrent write, using polynomial number of processors. The poly-log time performance and the polynomial upper bound on the number of processors rely on the following claims:

(i) We can check whether the required separation of V or V_i exists and if so, construct such a separation in poly-log time using a polynomial number of processors.

(ii) Let $k_0 = \max(5N, 2k)$. The family of terminal set pairs returned by *Divide* has never more than $2^{(k_0+2N)(k_0+2N-1)}$ members in any call of *Divide* during the execution of *DCP-or-Minor*(G, P, H, N).

(iii) The procedure *Divide* can be implemented to run in poly-log time on polynomial number of processors.

(iv) The recursion depth of *DCP-or-Minor*(G, P, H, N) is logarithmic.

The claims (i), (iii), and (iv) imply poly-log running time. The claims (ii) and (iv) imply that the recursion tree of *DCP-or-Minor*(G, P, H, N) has a polynomial number of nodes which combined with claims (i) and (iii) ensures a polynomial number of processors.

Let us prove the above four claims. According to the body of *DCP-or-Minor*, to implement (i) in the case of the V-separation, we proceed as follows. For all subsets S of $V(G)$ with at most N vertices, we test whether S induces the required separation by finding the connected components of $G(V - S)$. Note that for a fixed N the number of such subsets S is polynomial and the connected components can be found in poly-log time, using polynomial number of processors [QD]. Knowing the connected components, we can easily find their cardinalities in poly-log time using linear number of processors. Now, we can easily check whether it is possible to sum the connected components, say $C_1, ..., C_l$, and S into appropriate sets V_1 and V_2, and if so construct such a pair (V_1, V_2). This can be done by sorting the cardinalities in decreasing order, computing all their prefix sums in parallel, and applying binary search. By [Ak], these steps can be efficiently performed in parallel.

In the case of V_i-separation, we proceed analogously. The only difference is that instead of the cardinalities of the connected components of $G(V_i) - T$, we consider the cardinalities of their intersections with $V(P_i)$.

To prove (ii), we first observe that the cardinality of $V(P)$ (in recursive calls, $P = P_i$ or $P = Q_j$) never exceeds k_0 and the cardinality of $V(P_i)$ is bounded by k_0+N. Clearly, the former implies the latter as $V(P_1 \cup P_2) - V(P)$ consists of at most N vertices. The two statements can be proved by induction on the depth of recursive call of *DCP-or-Minor* in decreasing depth order. Whenever there are more than k_0 vertices in $V(P_i)$, either the procedure halts or a separation (W_1, W_2) is constructed such that $W_1 \cap W_2 = T$, $W_1 \cup W_2 = V_i$, $\#((W_1 - T) \cap V(P_i)), \#((W_2 - T) \cap V(P_i)) \leq \frac{2}{3}\#V(P_i)$. Hence, there are no more than $\frac{2}{3}(k_0 + N) + N$ vertices in $V(Q_j)$ which implies $\#V(Q_j) \leq k_0$ by $k_0 \geq 5N$. This completes the proof of the two statements. It follows now that the family of terminal set pairs returned by *Divide* during the execution of *DCP-or-Minor*(G, P, H, N) never exceeds $(2^{(k_0+2N)(k_0+2N-1)/2})^2$.

To show (iii), it is sufficient to observe that after sorting the vertices in V_1 and V_2 respectively, we can test a vertex in G for membership in V_1 and V_2 in poly-log time. This enables us to construct the graph L efficiently in parallel. Since L has never more than $k_0 + 2N$ vertices by the analysis in (ii), the remaining part of the body of the procedure can be executed in constant time.

To show (iv), it is sufficient to observe that in each recursive call in the body *DCP-or-Minor* the number of vertices of the new input graph is a constant fraction of that of the original graph. ∎

A class of planar graphs is non-trivial if it is non-empty and different from the class of all planar graphs. By using *DCP-or-Minor* and Lemmas 2.1 and 2.2, we can show that the DCP problem with a bounded number of terminal pairs for non-trivial classes of planar graphs is in $\mathcal{NC}$.

Theorem 2.1: Let $\mathcal{F}$ be a non-trivial minor-closed class of planar graphs and let k be a positive integer. The problem of testing for any graph G in $\mathcal{F}$, and any terminal set P in G of at most k pairs whether DCP for P in G exists is in $\mathcal{NC}$.

Proof: Since $\mathcal{F}$ is non-trivial and planarity is preserved under minor taking, $\mathcal{F}$ has at least one planar forbidden minor H by Fact 2.1. Now, it is sufficient to call *DCP-or-Minor*(G, P, H, N) where N is the integer constant specified by Fact 2.3. Since G is assumed to be in $\mathcal{F}$, it cannot have a minor isomorphic to H. Thus, we obtain as the answer either 'DCP exists' or 'no DCP'. Now, Lemmas 2.1 and 2.2 imply the thesis. ∎

Combining Theorem 2.1 with the obvious reduction of the fixed subgraph isomorphism problem to a polynomial number of DCP problems with fixed k, we

obtain the following theorem.

Theorem 2.2: Given a minor-closed class $\mathcal{F}$ of planar graphs, and a planar graph H, the subgraph homeomorphism problem for H and any graph G in $\mathcal{F}$ is in $\mathcal{NC}$.

Proof: To begin with, we need the following definition. A terminal set P in G corresponds to H if $(V(P), P)$ is a graph isomorphic to H. It is clear that H is homeomorphic to a subgraph of G if and only if there is a (terminal) subset P of $V(G) \times V(G)$ corresponding to H such that DCP for P in G exists. Note that since the graph H is fixed, the number of all such terminal sets P to test for DCP in G is polynomial in the size of G. Thus, by Theorem 2.1, we can perform all these tests and return the conjuction of their results in poly-log time using a polynomial number of processors. ∎

In turn, by combining Fact 2.1 and Theorem 2.2 with the reduction of the minor containment problem to that of subgraph homeomorphism (given in Fact 2.2), we obtain the following theorem.

Theorem 2.3: Given a minor-closed class $\mathcal{F}$ of planar graphs, the recognition problem for $\mathcal{F}$ is in $\mathcal{NC}$.

Proof: Our efficient parallel algorithm for the recognition problem consists of two major steps. First, we test the input graph for planarity using Miller-Reif's algorithm [MR]. Their algorithm runs on a concurrent read, concurrent write PRAM with $n^{O(1)}$ processors in time $O(\log n)$ ([MR, Theorem 26]). Hence, it can be implemented by $\mathcal{NC}$ circuits [SV, Theorem 1]. In the second step, we assume the input graph G to be planar and test whether it is in $\mathcal{F}$ by checking if it has a minor isomorphic to at least one of minimal forbidden minors defining $\mathcal{F}$ (see Fact 2.1). Since planarity is preserved under minor taking operation it is enough to perform these tests only for such planar minors for $\mathcal{F}$ to know whether G is in $\mathcal{F}$. Our method of performing the test relies on Fact 2.2 and Theorem 2.2. For each planar minimal forbidden minor K for $\mathcal{F}$, we use (by Fact 2.2) the finite list of planar graphs $H_1(K), ..., H_l(K)$ such that G has a minor isomorphic to K if and only if at least one of the graphs on the list is homeomorphic to a subgraph of K. Thus, we test each such a planar graph $H_i(K)$ and G for subgraph homeomorphism using the parallel algorithm described in the proof of Theorem 2.2. By finiteness of the list of minimal forbidden minors K of $\mathcal{F}$ and finiteness of the lists $H_1(K), ..., H_l(K)$, and Theorem 2.2, we obtain the thesis. ∎

3 A reduction of subgraph isomorphism for biconnected outerplanar graphs to polygon imbedding

In this and the next sections, we consider the subgraph isomorphism problem for *outerplanar graphs*. An outerplanar graph is a graph which can be embedded in the plane in such a way that all its vertices lie on the exterior face [Mi]. We shall call such an embedding of a graph in the plane, an *outerplanar embedding*. By [MR], we can easily deduce the following lemma:

Lemma 3.1: Given a biconnected outerplanar graph, we can find the cycle bounding the exterior face of its outerplanar embedding using $\mathcal{NC}$ circuits.

Proof sketch: Extend the input graph by a single vertex w adjacent to all original vertices. Note that the resulting graph is still planar. Find a planar embedding of the new graph. It is easy to see that the vertices adjacent to w in the clockwise order around w form the sought cycle. A planar embedding of the new graph can be constructed by a concurrent read, concurrent write PRAM with $n^{\mathcal{O}(1)}$ processors in time $\mathcal{O}(\log n)$ (Theorem 26 in [MR]). Hence, it can be constructed by $\mathcal{NC}$ circuits by Theorem 1 in [SV], and consequently, the whole procedure can be performed by $\mathcal{NC}$ circuits. ∎

Using the following definitions of planar figures in terms of standard geometric notation (see [PS]), we will be able to specify outerplanar embeddings of biconnected outerplanar graphs more precisely.

Definition 3.1 A *partial triangulation* of a simple polygon is a set of non-intersecting diagonals of the polygon. A *partially triangulated polygon* (PTP for short) Q is a union of a simple polygon and a partial triangulation of the simple polygon. The vertices of the simple polygon are vertices of Q, whereas the edges of the simple polygon and the diagonals from the partial triangulation of the simple polygon are edges of Q. The former edges of Q are called *boundary edges* of Q, the latter edges of Q are called *diagonal edges* of Q.

Mitchell observes in [Mi] that a biconnected outerplanar graph is in fact a partially triangulated polygon. By Lemma 3.1, we have:

Lemma 3.2: Given a biconnected outerplanar graph, we can find its outerplanar embedding in the form of a partially triangulated (convex) polygon using $\mathcal{NC}$ circuits.

It follows that a biconnected outerplanar graph has a unique outerplanar embedding (in the topological sense) up to the mirror image (see also [Sy82]).

Definition 3.2. A partially triangulated polygon with a distinguished boundary edge is called a *rooted, partially triangulated polygon* (RPTP for short). The distinguished edge is called the *root* of the RPTP. Given a RPTP P, the graph induced by P is denoted by $G(P)$. Now, given two RPTP, P and Q, we say that P can be *root-imbedded* into Q if and only if there is an isomorphism between $G(P)$ and a subgraph of $G(Q)$ that maps the root of P on the root of Q, and preserves the clockwise ordering of the vertices on the perimeter of P. Such an isomorphism is called a *root-imbedding* of P into Q.

In the following lemma, we show that the problem of subgraph isomorphism for biconnected outerplanar graphs is efficiently reducible (in parallel) to the problem of testing two RPTP's for root-imbedding.

Lemma 3.3: The problem of subgraph isomorphism for biconnected outerplanar graphs is $\mathcal{NC}$ reducible to the problem of testing whether an RPTP can be root-imbedded in another RPTP.

Proof sketch: Let G and H be biconnected outerplanar graphs. By Lemma 3.2, we can find outerplanar embeddings P and Q of G and H, respectively, in the form of partially triangulated convex polygons, using $\mathcal{NC}$ circuits. Let Q' be the mirror image of Q. Let us root P at its arbitrary boundary edge e. It is clear that G is isomorphic to a subgraph of H if and only if there is a subfigure R of Q (or Q'), and an edge d of Q (Q') such that R is a partially triangulated polygon consisting of all edges of Q (Q') on a given side of d and of the edge d on its boundary, and P can be root-imbedded in the RPTP R rooted at d. Note that there is only a linear number of candidates for such subfigures R. Hence, the reduction can be done by $\mathcal{NC}$ circuits. ▪

To specify and analyze our parallel algorithm for root-imbedding for RPTP in the next section, we need also the following definitions and lemmas.

Definition 3.3. Let P be a RPTP with n vertices. The *diagonal separator* of P is a diagonal or a diagonal edge of P that partitions P into two RPTP, each of no more than $\frac{2}{3}n + 2$ vertices.

For a PTP P, the tree $T(P)$ *dual* to P consists of vertices in one-to-one correspondence to the inner faces of P and of edges connecting vertices corresponding to adjacent faces in P.

Lemma 3.4: Given a RPTP P, we can find a diagonal separator of P using $\mathcal{NC}$ circuits.

Proof sketch: First, we triangulate P to obtain a completely triangulated polygon P'. It can be done by a concurrent read, exclusive write PRAM using a polynomial number of processors in poly-log time [ACGOY], and hence it can be implemented by $\mathcal{NC}$ circuits [SV]. Given P', we can construct the tree $T(P')$ dual to P' in constant time, in parallel. Now, to find a diagonal separator of P, it is sufficient to find a $\frac{1}{3} - \frac{2}{3}$ vertex separator of $T(P')$ with vertex weights appropriately defined (such a separator always exists [LT], see [Li85] for the details). This can be done by communicating the total weight of $T(P')$ to each vertex v of $T(P')$, and finding the total weight of descendants of v for each vertex v of $T(P')$. The latter can be done by using Euler's path techniques by $\mathcal{NC}$ circuits (combine [TV] with [SV]). ∎

Definition 3.4. Let P be a PTP. Given two edges e and d of P, let γ be the path in $T(P)$ between the two closest vertices in $T(P)$ corresponding to the faces adjacent to e and d respectively. Then, the *dual path* between e and d is the sequence of diagonal edges of P that separate the faces in the sequence of faces in P corresponding to γ.

Lemma 3.5: Given two edges e and d of a PTP, we can find the dual path between e and d and its middle element, using $\mathcal{NC}$-circuits.

Proof sketch: As in the proof of Lemma 3.4, we first build the tree T dual to the completely triangulated polygon P'. Next, the dual path can be easily constructed from the corresponding vertex path in T. The latter path can be found by using a standard $\mathcal{O}(\log n)$ method on a concurrent read, exclusive write parallel RAM with $\mathcal{O}(n^2)$ processors. In the j-th iteration of the method, we find, for each vertex v in the tree, the path from v to its ancestor at distance 2^j by concatenating the path from v to its ancestor at distance 2^{j-1} with the copied path between the two ancestors of v. By [SV], the method can be implemented by (uniform) circuits of unbounded fan-in, $\mathcal{O}(\log n)$ depth and polynomial size. Hence, it can be implemented by $\mathcal{NC}^2$ circuits. Given the dual path, we can find its median by finding for each its element the number of preceding and following elements which can be easily implemented by $\mathcal{NC}$ circuits.∎

4 A parallel algorithm for imbedding of partially triangulated polygons

Our parallel algorithm for the problem of root-imbedding for PTP consists of two recursive procedures $RI1$ and $RI2$. The first procedure tests whether the input RPTP P can be root-imbedded in the input RPTP Q. First, it finds a diagonal separator of Q and then it guesses its image in Q by trying all possible pairs of vertices of Q in parallel. To check whether the bottom part of P cut off by the diagonal separator (*i.e.*, the part not containing the root) can be root-imbedded in the corresponding part of Q cut off by the guessed image of the separator, the procedure calls recursively itself. To check whether the upper part of Q (the part containing the root) can be imbedded in the upper part of Q such that the root of P is mapped on the root of Q and the diagonal separator on its guessed image, the procedure $RI1$ calls $RI2$. The latter procedure solves the above problem as follows. First, it finds the dual path from the diagonal separator to the root of P. If the dual path is empty then it cuts off the left and right part of the upper part of P along the diagonals connecting the left and right endpoints of the diagonal separator and the root of P respectively (if for instance, the left endpoints overlap, the left part is empty). Analogously, it cuts off the corresponding left and right part from the upper part of Q. Next, $IR2$ tests whether the left and the right upper part of P can be respectively root-imbedded in those of the upper part of Q by calling the procedure $RI1$, twice in parallel. If both tests are positive, it returns YES. If the dual path contains more edges, the procedure $RI2$ finds the median and then guesses an image of the median in the upper part of Q by trying all possible pairs of vertices of the upper part of Q, in parallel. To check whether the upper and bottom part of the upper part of P divided by the median can be respectively imbedded in the corresponding parts of the upper part of Q divided by the guessed image of the median, it calls recursively itself twice in parallel.

The use of the diagonal separator and the path median ensures an $\mathcal{O}(\log^2 n)$ recursive depth of the algorithm composed of the two procedures. Since by Lemmas 3.4 and 3.5, the problems of finding the diagonal separator, the dual path, and its median can be solved by $\mathcal{NC}$ circuits, we can conclude that the procedure can be implemented by uniform circuits of poly-log depth. The crux is to observe that the circuits need only a polynomial number of processors. This follows from the fact that each figure occurring as a parameter in the recursive calls of $RI1$ can be obtained from P or Q by cutting along a single diagonal/edge in P or Q, or a straight line segment between two vertices in Q. In the latter case, all edges intersecting

this segment are deleted from Q. The above fact holds inductively for the bottom parts. For the upper parts, it is sufficient to observe that neither the root of P or Q nor the 'horizontal' cutting edges (*i.e.*, diagonal separators and the medians) occur in the final RPTP's that are produced by $RI2$ and become parameters of $RI1$ (see Fig. 4.1). It follows that the number of figures that are parameters in the recursive calls of $RI1$ is polynomial. Hence, the number of different figures occurring as parameters in the recursive calls of $RI2$ is also polynomial, since they are obtained from the figures being parameters of $RI1$ by cutting along at most two single diagonals/edges. We conclude that the number of distinct, potential recursive calls of $RI1$ and $RI2$ is polynomial. Therefore, we can implement all these potential recursive calls using a bottom-up method of poly-log depth taking a polynomial number of processors.

Thus, we have the following theorem.

Theorem 4.1: The problem of root-imbedding for RPTP is in $\mathcal{NC}$.

Combining Lemma 3.3 with Theorem 4.1, we obtain the main result of this section.

Theorem 4.2: The problem of subgraph isomorphism for biconnected outerplanar graphs is in $\mathcal{NC}$.

We conclude this section with a more formal description of the procedures $RI1$ and $RI2$. To simplify the notation, we assume that the input RPTP's are convex and no vertical line passes through any pair of their vertices. In the body of the procedure, $X(p)$ denotes the X coordinate of p.

We need also the following definition.

Let R be a PTP. Given a sequence $\alpha = v_1, v_2, ..., v_k$ of vertices of R, and a fragment β of R, we denote by $R(\alpha, \beta)$ the largest PTP such that:

(i) if there is no more than one vertex in α then it is empty;

(ii) it consists of some edges of R and the edges $(v_1, v_2), ..., (v_{k-1}, v_k)$;

(iii) the edges $(v_1, v_2), ..., (v_{k-1}, v_k)$ form a continuous fragment of its perimeter;

(iv) it is disjoint from β.

See Fig. 4.2 for an example.

The two procedures are specified as follows.

procedure $RI1(P, Q)$
begin

```
if P has three vertices then
  begin
    if Q contains a triangle then return YES else return NO
  end
else
  begin
    find a diagonal separator (v1, v2) of P where X(v1) < X(v2);
    P1 ← P((v1, v2), {root(P)}) rooted at (v1, v2);
    P2 ← P((v1, v2), P1 − {(v1, v2)});
    for all vertices w1, w2 of Q where (w1, w2) is not the root of Q
    do in parallel
    begin
      if (v1, v2) is an edge of P and (w1, w2) is not an edge of Q
      then return NO and halt;
      Q1 ← Q((w1, w2), {root(Q)}) rooted at (w1, w2);
      Q2 ← Q((w1, w2), Q1 − {(w1, w2)});
      if RI1(P1, Q1) ∧ RI2(P2, Q2, (v1, v2), (w1, w2), root(P), root(Q))
      then return YES else return NO
    end
  end
end


procedure RI2(P, Q, (v1, v2), (w1, w2), (v1', v2'), (w1', w2'))
begin
D ← the dual path from (v1, v2) to (v1', v2') in P;
if D = ∅ then
  begin
    if (v1, v1') is an edge of P and (w1, w1') is not an edge of Q
    or (v2, v2') is an edge of P and (w2, w2') is not an edge of Q
    then return NO and halt;
    PL ← P((v1, v1'), {v2, v2'}) rooted at (v1, v1');
    QL ← Q((w1, w1'), {w2, w2'}) rooted at (w1, w1');
    PR ← P((v2, v2'), {v1, v1'}) rooted at (v2, v2');
    QR ← Q((w2, w2'), {w1, w1'}) rooted at (w1, w1');
    if RI1(PL, QL) ∧ RI1(PR, QR) then return YES else return NO
  end
else
```

```
begin
  (v_3, v_4) ← the middle edge in D where X(v_3) < X(v_4);
  PU ← P((v_3, v_4), {(v_1, v_2)});
  PD ← P((v_3, v_4), {(v'_1, v'_2)});
  for all vertices w_3 and w_4 of Q
    where (w_3, w_4) is different from (w_1, w_2) and (w'_1, w'_2)
  do in parallel
  begin
    QU ← Q((w_3, w_4), {(w_1, w_2)});
    QD ← Q((w_3, w_4), {(w'_1, w'_2)});
    if RI2(PU, QU, (v_3, v_4), (w_3, w_4), (v'_1, v'_2), (w'_1, w'_2)) ∧
      RI2(PD, QD, (v_1, v_2), (w_1, w_2), (v_3, v_4), (w_3, w_4))
    then return YES else return NO
  end
 end
end
```

5 Extensions

The problem of subgraph isomorphism for biconnected outerplanar graphs can be seen as an abstraction of two following geometric problems for partially triangulated polygons P, Q :

(i) decide whether P is similar to a subfigure of Q;

(ii) decide whether P is congruent to a subfigure of Q.

The two problems can be respectively termed as the problems of sub-similarity and sub-congruency for partially triangulated polygons. Both have potential applications in pattern recognition.

The problems of sub-similarity and sub-congruency for partially triangulated polygons can be solved analogously to the problem of subgraph isomorphism for biconnected outerplanar graphs. First, we reduce both problems to their rooted versions (where the mapping on a distinguished boundary edge is fixed), using $\mathcal{NC}$-circuits. Then, we solve the rooted versions by subsequently modifying the parallel algorithm for root-imbedding for RPTP. In the case of the sub-similarity problem, we appropriately add tests for the congruency of angles formed by the roots and cutting edges of P and Q with the adjacent edges. In the case of the sub-congruency problem, we add also tests for edge length equality for the roots and cutting edges, respectively. The above modifications of the procedures $RI1$ and $RI2$ do not affect their asymptotic, worst-case circuit complexity. Hence, in analogy to Theorem

3.2, we have that the problems of sub-similarity and sub-congruency for partially triangulated polygons are in $\mathcal{NC}$.

It seems also possible to generalize our $\mathcal{NC}$ algorithm for subgraph isomorphism restricted to biconnected outerplanar graphs to include three-connected planar graphs of bounded width. We say that a planar graph G has width $\leq k$ if for any vertex v in any planar embedding of G there is a path composed of at most k edges/diagonals connecting v with the outer face.

The idea of a generalization of our $\mathcal{NC}$ algorithm would rely on the following insights:

(i) Any three-connected planar graph has at most two different embeddings on the sphere. These can be constructed by an $\mathcal{NC}$ algorithm [MR].

(ii) Three-connected planar graphs of width $\leq k$ have a $\frac{1}{3} - \frac{2}{3}$ separator in the form of edge/diagonal path of constantly bounded length. Such a separator can be found using an $\mathcal{NC}$-algorithm.

(iii) The above edge/diagonal paths would be used in the analogous manner to that of diagonal separators, median separators, *etc.*. Note that there is a polynomial number of such paths.

(iv) In order to keep the total length of cuts on the perimeter of each considered subfigure constantly bounded, it would be necessary to use yet another cutting procedure, resembling the V_i-separation from the procedure *DCP-or-Minor* (section 2). This would ensure a polynomial number of subfigures that could ever be considered. Hence, we would again obtain only a polynomial number of potential recursive calls.

As for the methods of Section 2, we suspect that similar methods can be used to design efficient parallel algorithm for the discussed problems restricted to families of not necessarily planar graphs (for instance, partial k-trees [AP]).

6 References

[ACGOY] A. Aggarwal, B. Chazelle, L. Guibas, C. O'Dunlaing, and C. Yap, Parallel Computational Geometry, in *Proc. 25th Annual IEEE Symposium on Foundations of Computer Science 1985*, 468-477.

[AHU] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, (Addison-Wesley, Reading, Massachusetts, 1974).

[AP] S. Arnborg and A. Proskurowski, Characterization and recognition of partial

3-trees, *SIAM Journal of Algebraic and Discrete Methods 7*, 2(1986), 305-314.

[APC] S. Arnborg, A. Proskurowski, and D.G. Corneil, Forbidden minors characterization of partial 3-trees, *UO-CIS-TR-86-07*, University of Oregon (1986);

[Ak] S.G. Akl, *Parallel Sorting Algorithms*, (Academic Press, New York, 1985).

[As] T. Asano, An approach to the subgraph homeomorphism problem, *Theoretical Computer Science*, 38 (1985), 249-267.

[C] S.A. Cook, The Classification of Problems which have Fast Parallel Algorithm, in *Proc. Foundations of Computation Theory, Borgholm, Sweden 1983*, Springer-Verlag LNCS 158.

[CH] G. Chartrand and F. Harary, Planar permutation graphs, *Ann. Inst. Henri Poincare Sec. B* 3(1967), 433-438.

[FHW] S. Fortune, J. Hopcroft, and J. Wyllie, The directed subgraph homeomorphism problem, *Theoretical Computer Science 10* (1980), 111-121.

[GJ] M.R. Garey, D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-completeness* (Freeman, San Francisco, 1979).

[H] F. Harary, *Graph Theory* (Addison-Wesley, Reading Massachusetts, 1969).

[Li85] A. Lingas, On partitioning polygons, in *Proc. of 1st ACM Symposium on Computational Geometry, Baltimore 1985.*

[Li86] A. Lingas, Subgraph Isomorphism for Biconnected Outerplanar Graphs in Cubic Time, in *Proc. 3rd STACS, Orsay 1986*, Springer-Verlag, LNCS 210.

[LK] A. Lingas, M. Karpinski, Subtree isomorphism and bipartite perfect matching are mutually NC reducible, *manuscript* (1986).

[LT] R.J. Lipton and R.E. Tarjan, Applications of a planar separator theorem, *SIAM J. Computing 9*, 3(1980), 513-524.

[Ma] D.W. Matula, Subtree isomorphism in $O(n^{\frac{5}{2}})$, *Annals of Discrete Mathematics 2* (1978), 91-406.

[Mi] S.L. Mitchell, Linear algorithms to recognize outerplanar and maximal outerplanar graphs, *Information Processing Letters 9*, 5(1979), 229-232.

[MR] G. Miller and J.H. Reif, Parallel Tree Contraction and its Applications, in *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science 1985*, 478-489.

[P] N. Pippenger, On simultaneous resource bounds, in *Proc. 20th Annual IEEE Symposium on Foundations of Computer Science 1979*, 307-311.

[PS] F.P. Preparata and M.I. Shamos, *Computational Geometry, An Introduction*, Texts and Monographs in Computer Science, Springer-Verlag, New York.

[Rei] K.R. Reischuk, Parallel Machines and their Communication Theoretical Limits, in *Proc. 3rd STACS, Orsay 1986*, Springer-Verlag, LNCS 210.

[QD] M.J. Quinn and N. Deo, Parallel Graph Algorithms, *Computing Surveys, Vol. 16, No. 3* (1984), 320-348.

[Rey] S.W. Reyner, An Analysis of a good algorithm for the subtree problem, *SIAM J. Computing 6* (1977), 730-732.

[Ru] W.L. Ruzzo, On uniform circuit complexity, *J. Computer and System Science 22* (1981), 365-383.

[SV] L. Stockmeyer and U. Vishkin, Simulation of Parallel Random Access Machines by Circuits, *SIAM J. Computing 13* (1984), 409-422.

[Sy82] M.M. Sysło, The subgraph isomorphism problem for outerplanar graphs, *Theoretical Computer Science 17* (1982), 91-97.

[Sy85] M.M.Sysło, private communication.

[TV] R.E. Tarjan and U. Vishkin, Finding Bi-connected Components and Computing Tree Functions in Logarithmic Parallel Time, in *Proc. 25th Annual IEEE Symposium on Foundations of Computer Science 1984*, 12-20.

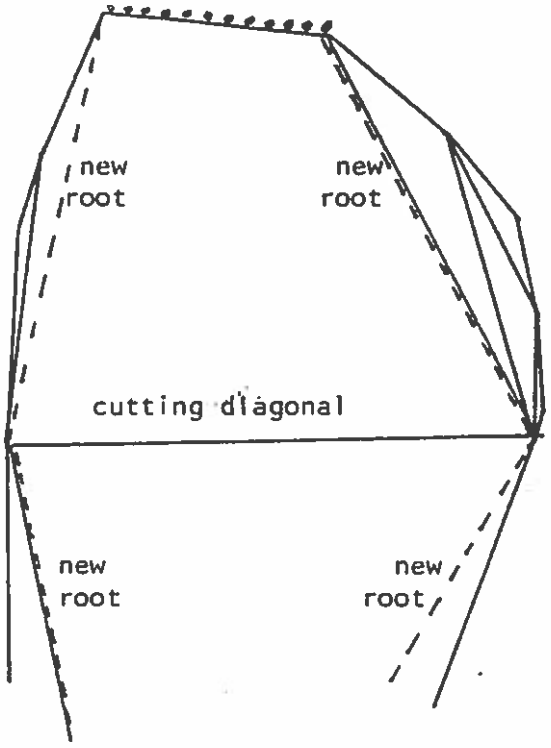

Fig. 4.1.

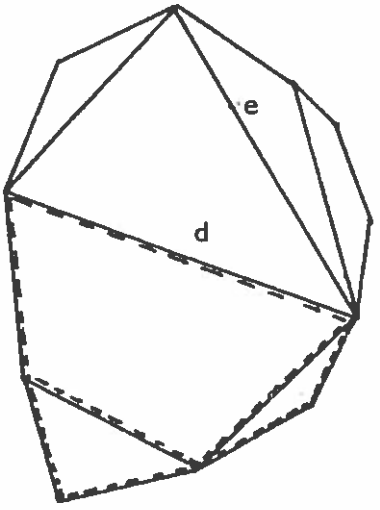

Fig. 4.2. The PTP P(d,e) is marked with dashed lines.