[Va 87] V.V.Vazirani, 'NC Algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems', *Information and Computation* 80, No. 2 (1989), pp. 152–164.

the CRCW PRAM. This algorithm improves almost all the results presented in this paper, in terms of time and processor efficiency. This algorithm also improves the results in [KMV 88]. The parallel algorithm for the triangle problem can also be implemented in the stated time and processor bounds on the weaker CREW PRAM model (with no concurrent writes permitted).

Acknowledgements: Thanks to Vasant Shanbhogue for listening to my ideas in the middle of the night and especially to Estie Arkin for valuable comments on an earlier draft of the paper.

References

[As 85] T.Asano, 'An approach to the subgraph homeomorphism problem', *Theoretical Computer Science*, 38 (1985), pp. 249–267.

[BM 77] J.A.Bondy and U.S.R.Murty, 'Graph Theory with applications,' American Elsevier, New York (1977).

[FHW 80] S.Fortune, J.E.Hopcroft and J.Wyllie, 'The directed subgraph homeomorphism problem', *Theoretical Computer Science*, 10 (1980), pp. 111–121.

[FT 88] D.Fussel and R.Thurimella, 'Separation pair detection', *Proceedings of AWOC 88, LNCS 319*, (1988), pp. 149–159.

[GJ 78] M.R.Garey and D.S.Johnson, 'Computers and Intractability: A guide to the theory of NP-completeness', *Freeman, San Francisco.*

[Ha 43] D.W.Hall, 'A note on primitive skew curves', *Bull. Amer. Math. Soc.*, 49 (1943), pp. 935–937.

[KMV 88] S.Khuller, S.G.Mitchell, V.V.Vazirani, 'NC Algorithms for the two disjoint paths problem and for finding a Kuratowski homeomorph', *Technical Report TR 88-960, Computer Science Department, Cornell University*, (1988).

[KR 86] P.N.Klein and J.H.Reif, 'An efficient parallel algorithm for planarity', *Proceedings of FOCS conference*, (1986), pp. 465–477.

[KS 89] S.Khuller and B.Schieber, 'Efficient parallel algorithms for testing connectivity and finding disjoint s-t paths', *in preparation.*

[LR 80] A.S.LaPaugh and R.L.Rivest, 'The subgraph homeomorphism problem', *Journal of Computer and System Sciences*, 20, (1980), pp. 133– 149.

[MSV 86] Y.Maon, B.Schieber and U.Vishkin, 'Parallel ear decomposition search (EDS) and st-numbering in graphs', *Theoretical Computer Science*, 47 (1986), pp. 277–298.

[PS 78] Y.Perl and Y.Shiloach, 'Finding two disjoint paths between two pairs of vertices in a graph', *Journal of the Association for Computing Machinery*, 25, (1978), pp. 1–9.

[RS 86] N.Robertson and P.D.Seymour, 'The disjoint paths problem', *manuscript*, (1986).

[Sh 80] Y.Shiloach, 'A polynomial solution to the undirected two path problem', *Journal of the Association for Computing Machinery*, 27, (1980), pp. 445–456.

[SV 82] Y.Shiloach and U.Vishkin, 'An $O(log\ n)$ parallel connectivity algorithm', *Journal of Algorithms*, vol.3, (1982), pp 57-63.

[TV 85] R.E.Tarjan and U.Vishkin, 'An efficient parallel biconnectivity algorithm', *SIAM Journal on Computing*, 14 (1985), pp. 862–874.

Step 3: In $G_i - \{v\}$, find a cycle C of length ≥ 3. Since G is triconnected we can find three vertex disjoint paths from v to three distinct vertices on the cycle C. Consider the segments of these paths upto their first point of intersection with C. The graph obtained from the three disjoint paths together with C is homeomorphic to K_4. We may need to replace virtual edges by paths in the original graph using ideas from the proof of Lemma 5.

Analysis: We can find cycle C after obtaining a spanning tree of G and adding one non-tree edge. This takes $O(log\ n)$ time using $O(n)$ processors. The main bottleneck in the algorithm is finding the three disjoint paths, which takes $O(log^2 n)$ time using $M(n)$ processors.

Our algorithm to obtain a subgraph of G homeomorphic to $K_{2,3}$ is based on the proof of Lemma 7 in [As 85]. Again, we assume that G has $O(n)$ edges and a triconnected component G_i satisfying one of the conditions in Lemma 7. The bottleneck in the algorithm is in finding the three disjoint paths which takes $O(log^2 n)$ time using $M(n)$ processors.

Algorithm-Find-$K_{2,3}$:

Step 1: Consider G_i; we consider all the three cases which it may satisfy.

1. G_i has five or more vertices: Use the algorithm Find-K_4 to construct G', a subgraph homeomorphic to K_4. Let the vertices in G' of degree three be called v_1, v_2, v_3, v_4. If G' is exactly the graph K_4, then G_i must have another vertex $u \notin G'$. Find three disjoint paths from u to any three vertices of the K_4. From the three paths and the K_4 we can find a subgraph homeomorphic to $K_{2,3}$.
   If G' was a subdivision of K_4, then consider a vertex u on the path $P(v_1, v_2)$. Since G is triconnected, there must be a path from u to some other vertex w of G' in $G - \{v_1, v_2\}$. Using this path and the K_4 it becomes easy to extract the subgraph homeomorphic to $K_{2,3}$.
2. G_i is K_4 with a virtual edge: We can replace the virtual edge by a path of length ≥ 2 and obtain a subgraph homeomorphic to $K_{2,3}$.
3. G_i is a triple bond of virtual edges: We replace all three virtual edges by paths of length ≥ 2 and obtain a subgraph homeomorphic to $K_{2,3}$.

Thus we conclude:

Theorem 6 *For a given graph G, we can find a subgraph homeomorphic to $K_{2,3}$ or K_4, if G has such a subgraph in $O(log^2 n)$ time using $M(n)$ processors.*

Using similar ideas to the ones discussed above we can develop parallel algorithms for several other pattern graphs considered in [As 85]. There are a few cases which need to be considered and we leave these for the reader.

Theorem 7 *There is an $O(\log n)$ time algorithm using n processors to test whether a given graph G, contains a subgraph homeomorphic to C_4 or C_5 (where C_k is a simple cycle of length k) and to obtain the homeomorph.*

7 Conclusions

Recent work by Baruch Schieber and the author has resulted in a parallel algorithm to solve the k-vertex disjoint paths problem in $O(\log n)$ time using $(n+m)/\log n$ processors [KS 89] on

Theorem 4 *A graph is outer-planar if and only if it has no subgraph homeomorphic to K_4 or $K_{2,3}$.*

Lemma 6 (Asano) *A graph G has a subgraph homeomorphic to K_4 if and only if there is a triconnected component of G with four or more vertices.*

From Lemma 3 and Lemma 6 we are able to develop an efficient algorithm for testing whether a given graph G contains a subgraph homeomorphic to K_4.

Algorithm Test-for-K_4:

Step 1: For any simple graph G if $\mid V(G) \mid \geq 2$ and $\mid E(G) \mid \geq 2 \mid V(G) \mid -2$ then output 'yes'.

Step 2: Decompose G into its triconnected components; if G has no triconnected component with four or more vertices, then output 'no' otherwise output 'yes'.

Analysis: We use the $O(\log n)$ time parallel algorithm of [FT 88] to decompose the graph into its triconnected components using $O(n)$ processors.

Lemma 7 (Asano) *A simple graph G has a subgraph homeomorphic to $K_{2,3}$, if and only if there is a triconnected component of G satisfying one of the following:*
(i) It has five or more vertices.
(ii) It is the graph K_4 with at least one virtual edge.
(iii) It is a triple bond of three virtual edges.

This characterization enables us to efficiently test if a given graph G contains a subgraph homeomorphic to $K_{2,3}$ via an algorithm almost identical to Algorithm Test-for-K_4. Combining these two, we get an efficient parallel algorithm to test whether a given graph is outer-planar. Thus we conclude:

Theorem 5 *There is an $O(\log n)$ time algorithm using n processors to test if a given graph G is outerplanar.*

Remark: This does not yield an algorithm to obtain the planar embedding of the graph if it is outer-planar. To obtain a planar embedding, we add one artificial vertex v to G, and an edge from v to every vertex in the graph. The new graph is planar if and only if G was outerplanar and using the $O(\log^2 n)$ time algorithm of [KR 86] we can obtain a planar embedding for G. The addition of v ensures that all the vertices are on the same face in the obtained embedding.

Finding K_4 and $K_{2,3}$ Homeomorphs:

Our algorithm for obtaining a subgraph of G homeomorphic to K_4 is based on the proof of Lemma 6 in [As 85]. Assume G has $O(n)$ edges (if not, we can choose a subgraph of G having $O(n)$ edges and a K_4 homeomorph) and has a triconnected component with four or more vertices.

Algorithm Find-K_4:

Step 1: Consider the triconnected component of G having four or more vertices, and call it G_i.

Step 2: Choose any vertex v of G_i and consider the subgraph $G_i - \{v\}$ (which is biconnected).

(a) If there is a bridge B_x in the set $Outer\text{-}skew_C$ which is also skew to B_c and has attachment vertices x_1, x_2 skew to two attachment vertices of B_c (say c_1 and c_2 distinct from $\{x_1, x_2\}$) such that $\{c_1, x_1\}$ ($\{c_2, x_2\}$) belong to the upper (lower) chain. There are various cases to consider of which we illustrate only one (see Fig. 7). Here $c_1 = c_{ul}$ and $c_2 = c_{lr}$. We find a path from x_u to c_1 in B_{uc} and a path from c_1 to x_2 via a on C. We also find a path from x_2 to x_1 in B_x and complete the cycle by finding a path from x_1 to c_2 via b on C and a path from c_2 to c via x_l.

(b) If there is no such bridge then there is no solution. The removal of vertices $\{c_{ul}, c_{ll}\}$ separate the graph into a connected component G_a containing a and G_{bc} containing b and c. Any path from c to a must enter G_a through c_{ul} or c_{ll}. Assume it is c_{ul}. Now we need to find a simple path from c_{ul} to b via a and avoiding c_{lr} (since any path from c to b will go through c_{lr}). Clearly this is impossible (see Fig. 8). There may be a bridge in the set $Outer\text{-}skew_C$ which is skew to B_c but does not have the vertices x_1 and x_2 as required. Again it can be seen that there is no solution to the problem (see Fig. 9).

Using the algorithms in [KR 86], [FT 88], [SV 82] and [TV 85] we can implement the above algorithm in $O(log^2 n)$ time using $O(n)$ processors. Thus we conclude:

Theorem 2 *In an undirected planar graph G, given three vertices a, b and c we can find a cycle (if it exists) containing these three vertices in $O(log^2 n)$ time using $O(n)$ processors.*

Using a lemma of Hall's [Ha 43] we can extend this algorithm to $K_{3,3}$-free graphs (the class of graphs which do not contain a subgraph homeomorphic to $K_{3,3}$) easily.

Theorem 3 (Hall) *Each triconnected component of a $K_{3,3}$-free graph is either planar or exactly the graph K_5.*

6 Finding other homeomorphs

In this section we consider the 'non-fixed vertex' version of the *SHP* for undirected graphs. We first state the following lemmas which prove some useful properties about graphs. The proofs may be found in [As 85].

Lemma 2 *For a triconnected graph H, a graph G has a subgraph homeomorphic to H if and only if there is a triconnected component of G that has a subgraph homeomorphic to H.*

Lemma 3 *If a simple graph G with two or more vertices has no subgraph homeomorphic to K_4, then $\mid E(G) \mid \leq 2 \mid V(G) \mid -3$.*

Lemma 4 *If a simple graph G with two or more vertices has no subgraph homeomorphic to $K_{2,3}$, then $\mid E(G) \mid \leq 2 \mid V(G) \mid -2$.*

Lemma 5 *Let D be a decomposition of a graph G into triconnected components. Let $\{e_1, e_2, ..., e_r\}$ be the set of all virtual edges of a triconnected component G' in D. Then there is a set of vertex-disjoint paths in G, say $\{P_1, P_2, ..., P_r\}$ such that each P_j $(j = 1, 2, ..., r)$ connects the two end vertices of e_j, and contains no edge of G'.*

A planar graph is *outer-planar* if it can be embedded in the plane so that all its vertices lie on the same face. The following theorem is an easy corollary to Kuratowski's theorem.

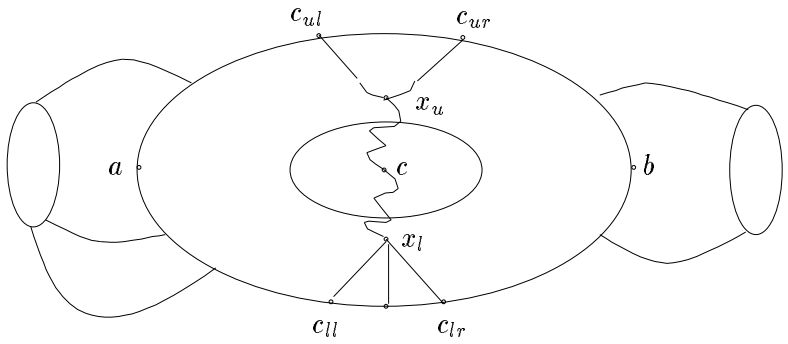

Figure 8: No solution possible

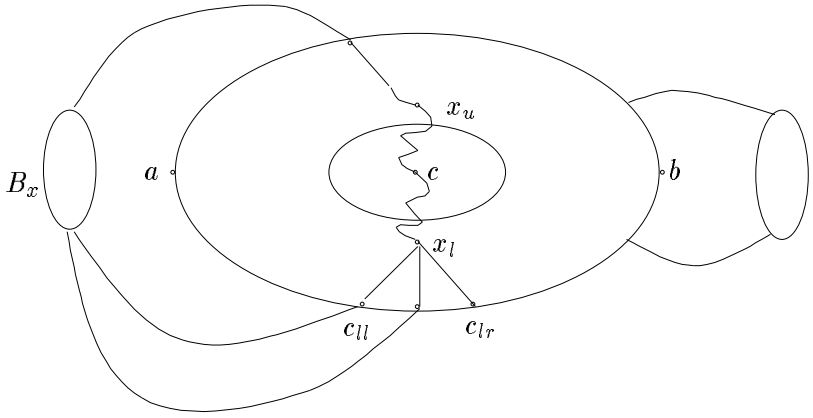

Figure 9: No solution possible

subgraph B_{lc} by adding the lower attachment vertices to the internal vertices of B_c. We now consider the following cases:

1. There are two disjoint paths from c to $\{c_{ul}, c_{ur}\}$ in B_{uc} (assuming $\{c_{ul}, c_{ur}\}$ are distinct). A cycle containing all three vertices a, b, c can now be found as shown in Fig. 6.

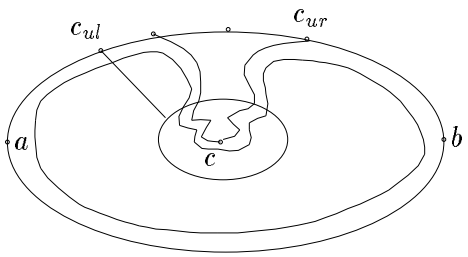

Figure 6: Construction of required cycle using c_{ul} and c_{ur}

2. There are two disjoint paths from c to $\{c_{ll}, c_{lr}\}$ in B_{lc}. This case is identical to the previous case.

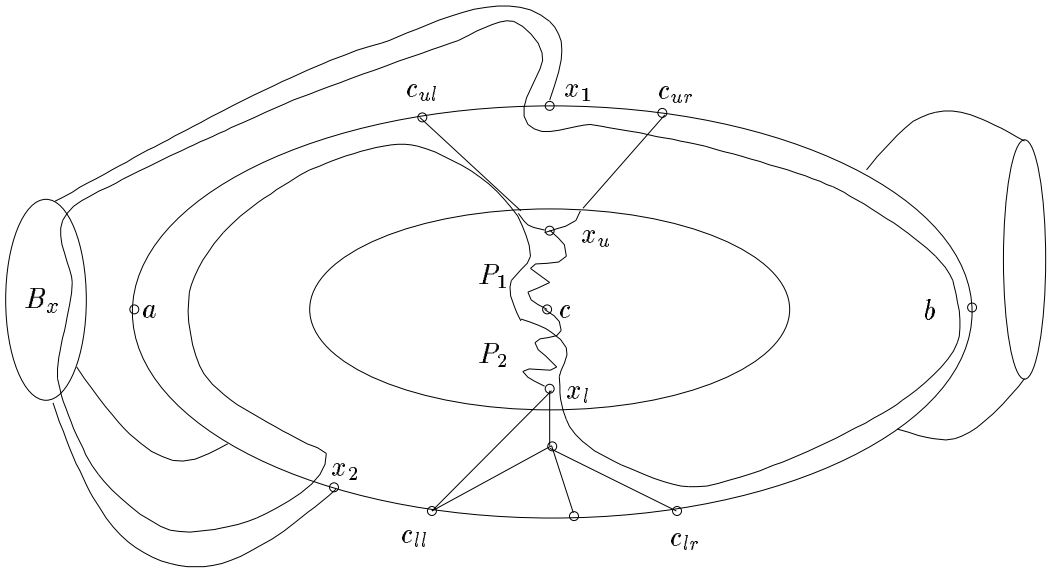

Figure 7: Construction of required cycle using B_x

3. If neither of the above two cases apply, then there are cut-vertices x_u and x_l separating c from the attachment vertices. (If c_{ll} and c_{lr} are identical then x_l may be the lower attachment vertex.) Decompose B_c into its biconnected components and obtain the cut-vertices closest to c (x_u and x_l). Construct two disjoint paths $P_1[c; x_u]$ and $P_2[c; x_l]$ from c to $\{x_u, x_l\}$. There are two cases to check:

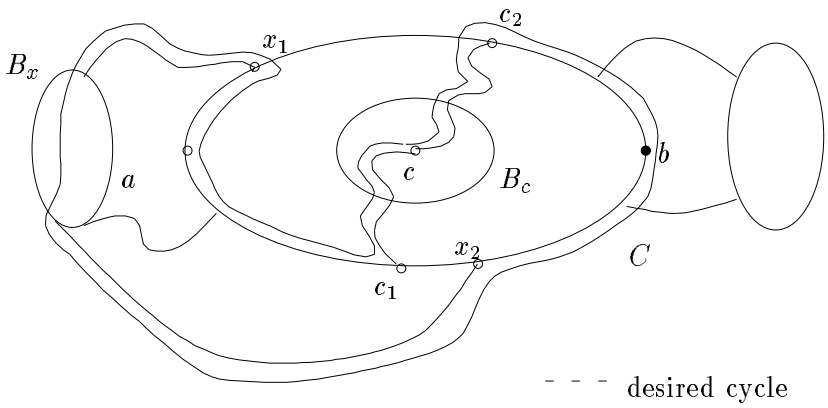

Figure 4: Bridge B_x is skew to $\{c_1, c_2\}$

(b) Bridge B_x is skew to $\{a, b\}$ as well as to $\{c_1, c_2\}$ then a solution easily follows by constructing two disjoint paths P_1 and P_2 from c to $\{c_1, c_2\}$ (see Fig. 4). Let B_x have attachment vertices x_1 and x_2 skew to (c_1, c_2) and (a, b). We illustrate one of the cases in Fig. 4. The cycle is now easy to construct by splicing in the appropriate segments as shown.

Now we consider the case when B_c has three or more vertices of attachment. We orient the cycle clockwise and call $C[a; b]$ the *upper chain* of C. We call $C[b; a]$ the *lower chain* of C. By $C(a; b)$ we refer to the path $C[a; b] - \{a, b\}$. Similarly we define $C(b; a)$. We need to consider the various ways in which B_c could be attached to C. We define *upper attachment* vertices and *lower attachment* vertices to be the attachment vertices of B_c on the upper and lower chains of C respectively.

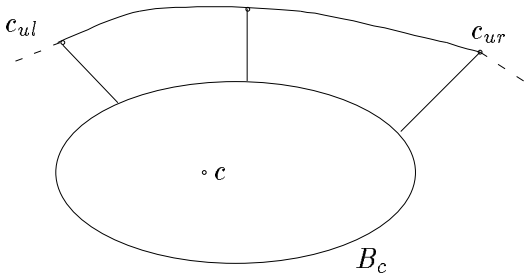

Figure 5: The subgraph B_{uc}

The extreme vertices of attachment on the upper chain are called *upper-left* and *upper-right* vertices respectively. These are denoted by c_{ul} and c_{ur} (see Fig. 5). (When B_c has only one attachment vertex they are not distinct.) Similarly define c_{ll} and c_{lr} to be the *lower-left* and *lower-right* vertices. We denote by B_{uc} the subgraph of B_c induced by the internal vertices of B_c together with the upper attachment vertices and the chain $C[c_{ul}; c_{ur}]$. Similarly define the

simple cycle C_i'. Let $P_i(x_i^1, x_i^k)$ be the subpath of C_i' from x_i^1 to $x_i^{k_i}$ avoiding the the attachment bar of B_i.

In C', replace all the segments which are attachment bars of some bridge $B_i \in Outer\text{-}non\text{-}skew_{C'}$ by the path $P_i[x_i^1; x_i^{k_i}]$. The new cycle C contains both a and b, and all its outer bridges are skew to (a, b).

The above steps can be implemented efficiently in parallel using the parallel algorithms in [KR 86], and [SV 82], and standard pointer doubling techniques. The parallel algorithm takes $O(log^2 n)$ time using $O(n)$ processors. □

Now consider the cycle C as constructed in Lemma 1. If vertex c is on C, then C is the required cycle. Assume that c is an interior vertex of some bridge B_c. Since G is biconnected, B_c has at least two vertices of attachment on C.

If B_c has only two vertices of attachment (c_1, c_2) on C then there are two cases to consider.

1. If the two vertices of attachment are on one of $C[a; b]$ or $C[b; a]$ then in B_c we can find two disjoint paths from c to $\{c_1, c_2\}$ and a solution easily follows using these two paths along with the appropriate segment of C (see Fig. 2).

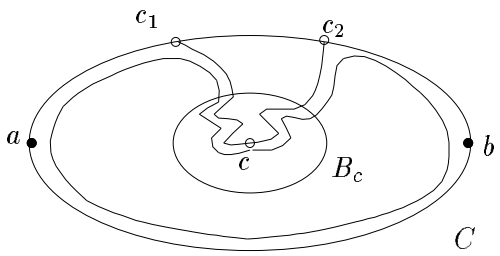

Figure 2: Two attachment vertices on $C[a; b]$

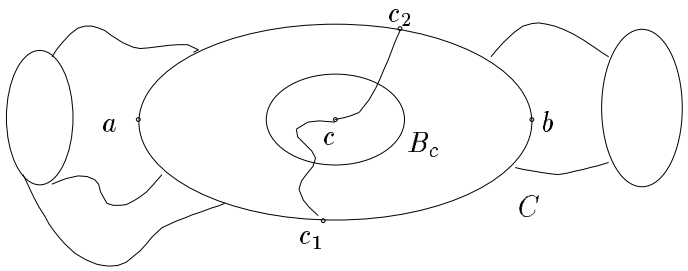

Figure 3: No solution to problem

2. If c_1 is on $C(a; b)$ and c_2 on $C(b; a)$ then there are two cases to consider.

   (a) If there is no bridge in the set $Outer\text{-}skew_C$ which is also skew to $\{c_1, c_2\}$ then there is no solution since $\{c_1, c_2\}$ separate all three of a, b, c into three separate connected components (see Fig. 3).

by $IntC$ and $ExtC$ respectively. In a plane graph G, each bridge of G relative to C is entirely contained in $IntC$ or $ExtC$. A bridge in $IntC$ ($ExtC$) is called an *inner* (*outer*) bridge.

Consider a planar embedding of G. Since G is biconnected, we can find a cycle C' in G containing a and b by finding two disjoint a-b paths. The disjoint a-b paths are found in parallel by using the s,t-numbering of the graph [MSV 86]. (This technique is only useful for constructing 2 disjoint paths, and does not extend to finding k paths.) Now consider the set of bridges of G with respect to the cycle C'. Given a planar embedding of G, the bridges may be partitioned into two sets:
$Outer_{C'} = \{B_i \mid B_i$ is embedded in $ExtC'\}$
$Inner_{C'} = \{B_j \mid B_j$ is embedded in $IntC'\}$

A bridge $B_i \in Outer_{C'}$ is in the set $Outer\text{-}skew_{C'}$ if B_i is skew to (a,b); B_i is in the set $Outer\text{-}non\text{-}skew_{C'}$ if it is not skew to (a,b). Ensure that in the embedding, vertex c is in the interior of C'.

Lemma 1 *In a planar graph G we can find a cycle C through two specified vertices a and b such that there is no bridge in the set $Outer\text{-}non\text{-}skew_C$. Moreover, this cycle may be found in $O(\log^2 n)$ time using $O(n)$ processors.*

Proof: Start by finding a cycle C' through a and b as described above. Ensure that vertex c is in the interior of cycle C' in the planar embedding (else modify the embedding). We now modify the cycle C' to obtain a cycle C containing a and b such that there are no bridges in the set $Outer\text{-}non\text{-}skew_C$ with respect to the cycle C.

Since G is biconnected, each bridge has at least two vertices of attachment on C'. Let the attachment vertices of B_i on C' be $x_i^1, x_i^2, ..., x_i^{k_i}$ (considered in clockwise order on C'). We call $x_i^1 (x_i^{k_i})$ the first (last) attachment vertex of B_i on C'. The segment $C'[x_i^1; x_i^{k_i}]$ is called the *attachment bar* of B_i on C'. It is easy to see that outer bridges avoid one another. Hence, the attachment bar of each outer bridge can overlap with the attachment bar of another outer bridge only at an end-vertex, and not at any internal vertex.

Each bridge in the set $Outer\text{-}non\text{-}skew_{C'}$ has all of its attachment vertices (and thus its attachment bar) on the segment $C'[a;b]$ or $C'[b;a]$.

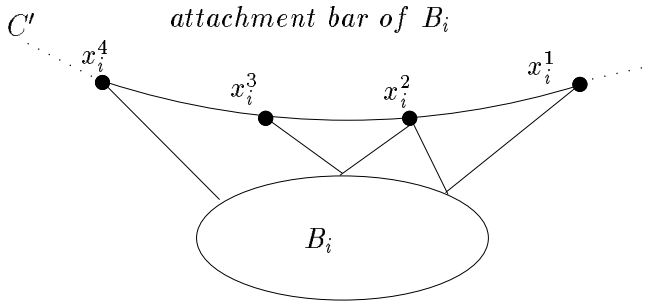

Figure 1: The graph B_i'

Consider the planar embedding of B_i and its attachment bar, and call the resultant graph B_i' (see Fig. 1). Note that B_i' is biconnected and planar. In B_i' the vertices x_i^1 and $x_i^{k_i}$ are on the outermost face in the planar embedding. Since B_i' is biconnected, the outer face of B_i' is a

4 A parallel algorithm for the triangle problem

We give a parallel algorithm for the following problem: given an undirected graph G, and three specified vertices a, b, c of G, determine whether the three vertices lie on a common simple cycle, and construct such a cycle if one exists.

Our parallel algorithm is a parallelization of the sequential algorithm developed by LaPaugh and Rivest [LR 80] to solve the problem. We assume that G is biconnected, since a, b and c must be in the same biconnected component if the cycle exists.

The main idea is to decompose G into pieces and to look for paths which must exist in these pieces if the cycle is to exist in G. Using the k-disjoint paths algorithm developed in section 2, we can either construct the desired cycle from these paths, or conclude that such a cycle does not exist.

The bottleneck in the algorithm is obtaining three vertex disjoint paths which can be done in $O(log^2 n)$ time using $M(n)$ processors. All the other steps can be implemented in parallel using the algorithms in [SV 82], [TV 85] and [FT 88]. Thus we conclude:

Theorem 1 *In an undirected graph G, given three vertices a, b and c we can find a cycle containing these three vertices in $O(log^2 n)$ time using $M(n)$ processors.*

5 The triangle problem for planar graphs

In the previous section we developed a parallel algorithm for solving the triangle problem in an arbitrary graph G. Unfortunately the algorithm is inefficient in the number of processors it uses. In this section we concentrate our attention on planar graphs and present an $O(n)$ time sequential algorithm for the problem. We are also able to provide a very efficient parallel algorithm for the problem that uses n processors and $O(log^2 n)$ time.

We assume that G is biconnected since a, b and c must be in the same biconnected component if the cycle exists. Note that if any of the edges $(a, b), (a, c), (b, c)$ are in G then the problem can be solved easily. Say, (a, c) is in G, now the problem is reduced to finding a path from a to c containing b which can be done by finding disjoint paths from b to $\{a, c\}$. Henceforth, we assume that none of the edges $(a, b), (a, c), (b, c)$ are in G. Before we present the algorithm we review some definitions.

Definitions: Let C be a cycle in G, and let e and f be edges of G not in C. Define the equivalence relation $=_C$ by $e =_C f$ if and only if there is a path in G that includes e and f and has no internal vertices in common with C. The subgraphs induced by the edges of the equivalence classes of $E(G) - E(C)$ under $=_C$ are called the *bridges* of G relative to C. The *vertices of attachment* of bridge B to cycle C are the vertices in $V(B) \cap V(C)$. The remaining vertices of bridge B are called the *interior* vertices of B.

A bridge with k vertices of attachment is called a k-bridge. Two k-bridges with the same vertices of attachment are *equivalent* k-bridges. The vertices of attachment of a k-bridge B with $k \geq 2$ effect a partition of C into edge-disjoint paths, called the *segments* of B. Two bridges *avoid* one another if all the vertices of attachment of one bridge lie in a single segment of the other bridge; otherwise they *overlap*. Two bridges B and B' are *skew* if there are four distinct vertices u, v, u', v' of C such that u and v are vertices of attachment of B, u' and v' are vertices of attachment of B', and the four vertices appear in the order u, u', v, v' on C. It is shown in [BM 77] that if two bridges overlap, then they are either skew or equivalent 3-bridges.

If C is a closed Jordan curve in the plane, then the rest of the plane is partitioned into two disjoint open sets called the *interior* and *exterior* of C. We denote the closures of the regions

shared memory. In each time unit, a processor can read from a memory cell, perform an arithmetic or logical computation, and write into a memory cell. Concurrent reads and concurrent writes into the same memory cell by different processors are permitted. If a write conflict occurs, an arbitrary processor succeeds.

2 Finding k disjoint s-t paths

We present a parallel algorithm for finding k-vertex disjoint paths (for any constant k) between a pair of vertices in an undirected graph G, using flow techniques.

Given the graph $G(V, E)$ we construct the directed graph $G'(V', E')$ as follows:

$$V' = \{v', v'' \mid v \in V - \{s, t\}\} \cup \{s, t\}$$

$$E' = \{(v', v'')\} \cup \{(u'', v'), (v'', u') \mid (u, v) \in E, \{u, v\} \subseteq V - \{s, t\}\} \cup$$
$$\{(s, v') \mid (s, v) \in E\} \cup \{(v'', t) \mid (v, t) \in E\}$$

In the directed graph G' we treat the node s as a 'source' and the node t as a 'sink'. All the edges in the directed graph are treated as unit capacity edges.

It is easy to see that a flow of size k in G' corresponds to k-vertex disjoint s-t paths in G. The paths in G are vertex disjoint since corresponding to each vertex $v \in G$ we have two vertices (v' and v'') in G' connected by an edge of unit capacity, ensuring that only a single unit of flow is pushed through each vertex of G. Moreover, the maximum flow in G' is equal to the number of vertex disjoint s-t paths in G. A flow of size k in G' is obtained by k successive augmentations; each augmentation corresponding to finding a s-t path in G. Menger's theorem guarantees the existence of k-vertex disjoint s-t paths in a k-vertex connected graph. Each flow augmenting path can be found by matrix multiplication, and since we only need a constant number of flow augmenting paths this can be efficiently implemented in parallel.

Thus the k paths may be found in $O(log^2 n)$ time using $M(n)$ processors. We first used this technique in [KMV 88].

3 SHP for directed graphs

In this section we deal with the 'fixed vertex' version of the SHP for directed graphs. If the pattern graph H has the property that all the arcs share a common tail (head), then we can obtain a parallel algorithm to obtain a subgraph of G homeomorphic to a subgraph of H, by a parallelization of the algorithm in [FHW 80] using the flow augmenting path ideas (see section 2).

When G is a directed acyclic graph and H is any fixed pattern graph, Fortune, Hopcroft and Wyllie [FHW 80] give a polynomial time algorithm via a 'pebbling game'. Assuming that H has k arcs; they show how to construct another graph G' with $O(n^k)$ vertices that encodes the configurations of the pebbling game. In G', a path finding algorithm is used to find a path from the starting configuration to any winning configuration. We observe that the graph may be constructed in parallel and the path finding can also be implemented in parallel using matrix multiplication.

1. H is a graph, all of whose arcs share a common head or tail,

2. G is acyclic.

For the case of undirected graphs, the problem was known to be in polynomial time for only a few non-trivial graphs until recently, when Robertson and Seymour in [RS 86] showed that the problem is solvable in polynomial time for all fixed graphs H. Unfortunately, the constants involved in their algorithm are forbiddingly large (larger than a "tower of 2's" whose height (number of levels of exponentiation) is worse than exponential in $| V(H) |$). For special pattern graphs the only non-trivial results known were for the cases when H is a cycle of length three [LR 80], or when H consists of two independent edges [Sh 80], [PS 78]. These problems can also be stated as follows: determine, if there exists a simple cycle containing three given vertices in an undirected graph G; or, given pairs of vertices $\{(s_i, t_i)\}$ $(i = 1, 2)$ in an undirected graph, find two vertex-disjoint paths connecting s_i with $t_i, (i = 1, 2)$. Both these problems are basic problems for all fixed *SHP*'s for undirected graphs, since any undirected graph which has at least two edges and is not a tree of height one, contains a cycle of length three, or two disjoint edges. Thus, the fixed *SHP* contains one of the above two problems as a subproblem unless it is a tree of height one. Khuller, Mitchell and Vazirani [KMV 88] developed an NC algorithm that solves the two disjoint paths problem. In this paper, we present an NC algorithm for the problem of finding a simple cycle containing three fixed vertices in the graph. For the case of planar graphs we develop a linear time algorithm which yields a more efficient parallel algorithm.

Now consider the 'non-fixed vertex' version of the *SHP*. This problem not only polynomially reduces, but also NC reduces to the 'fixed vertex' version of the problem. For the case of undirected graphs, for special pattern graphs H, Asano [As 85] gives efficient algorithms to test whether a graph G contains a subgraph homeomorphic to H, and also provides algorithms to obtain these homeomorphs. In particular, a sequential algorithm is given for the fixed graph $K_{3,3}$ (also called the Thomsen graph). The parallel complexity of this problem was first studied by Vazirani [Va 87], and he developed an $O(\log^2 n)$ time algorithm using n processors for checking whether a given graph G contains a subgraph homeomorphic to $K_{3,3}$, but the question of obtaining the homeomorph was left open. In [KMV 88] the problem of obtaining a subgraph homeomorphic to $K_{3,3}$ was solved by presenting an $O(\log^2 n)$ time algorithm using $M(n)$ processors (where $M(n)$ is the number of arithmetic operations required to multiply two $n \times n$ matrices).

We consider several other pattern graphs H, for each of which we give an $O(\log n)$ time algorithm using n processors to determine whether an input graph G has a subgraph homeomorphic to H. Included among these are the graphs K_4 and $K_{2,3}$ which are used to characterize outerplanar graphs. Thus, we provide an $O(\log n)$ time algorithm using n processors to test, if a given graph G is outerplanar (which is faster than the $O(\log^2 n)$ time algorithm of [KR 86] for planarity testing). We can also find subgraphs of G that are homeomorphic to these graphs, if G has such a homeomorph, in $O(\log^2 n)$ time using $M(n)$ processors.

In this paper we restrict our attention to the *SHP* for which, the images of edges of H, are paths which are required to be vertex disjoint. In [LR 80] a reduction from the edge-disjoint problem to the vertex-disjoint problem was given; we observe that the reduction can be carried out efficiently in parallel. Thus a parallel algorithm for the vertex-disjoint version implies a parallel algorithm for the edge-disjoint version. Most of our algorithms are based on a parallel algorithm to find k-vertex disjoint paths between a pair of vertices in a graph G.

The model of computation assumed is the CRCW (Concurrent Read Concurrent Write) PRAM model. The model consists of a number of identical processors and a common globally

Parallel Algorithms for the Subgraph Homeomorphism Problem

Samir Khuller *
Computer Science Department
Cornell University
Ithaca, NY 14853

Abstract

The subgraph homeomorphism problem for a fixed graph H is stated as follows: given a graph G, determine whether G has a subgraph homeomorphic to H, and obtain it. We study the parallel complexity of this problem for various pattern graphs H, and present fast NC algorithms for various versions of this problem. We also present an efficient NC algorithm to check if a given graph is outer-planar and to obtain its forbidden homeomorphs K_4 or $K_{2,3}$, if it is not.

1 Introduction

The subgraph homeomorphism problem (*SHP*) for a fixed graph H (the pattern graph) is stated as follows: given a graph $G(V, E)$, determine whether G has a subgraph homeomorphic to H; i.e., a pair of one-to-one mappings, (v, a), the first from vertices of H to vertices of G; the second from edges of H to simple paths of G. We require a path in G which corresponds to edge (x, y) in H, to go from $v(x)$ to $v(y)$ in G. The images of vertices of H, which are vertices of G, may be specified *a priori* and the images of edges of H (which are paths in G), may be required to be vertex-disjoint or edge-disjoint. Graphs G and H are either both directed or both undirected.

If G and H are both part of the input, then the *SHP* is NP-complete [GJ 78]. In this paper we restrict our attention to various *SHP*'s derived by fixing H. The main question for such *SHP*'s is, "For a given pattern graph H, is there a polynomial time algorithm, which given an input graph G, will determine whether there is a homeomorphic image of H occurring in G ?" The complexity of this problem was left open in [GJ 78]. Recently, Robertson and Seymour [RS 86] have shown that for undirected graphs this problem is solvable in polynomial time for all fixed graphs H. In this paper, we are interested in the parallel complexity of the *SHP* for some fixed pattern graphs.

First consider the 'fixed vertex' version of the problem (the input specifies which vertex of H corresponds to which vertex of G). For the case of directed graphs, the sequential complexity of the problem has been characterized by Fortune, Hopcroft and Wyllie [FHW 80]. They show that if H is a fixed graph, all of whose arcs share a common tail, or all of whose arcs share a common head, it is solvable in polynomial time and is NP-complete for all other fixed graphs H. They also show that the problem is solvable in polynomial time when G is a directed acyclic graph. In this paper, we develop NC algorithms for the following cases:

*supported by NSF grant DCR 85-52938 and PYI matching funds from AT&T Bell Labs.