Processor Efficient Parallel Algorithms for the Two Disjoint Paths Problem, and for Finding a Kuratowski Homeomorph*

Samir Khuller
Stephen G. Mitchell
Vijay V. Vazirani

TR 89-960
April 1989

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Supported by NSF grant DCR 85-52938 and PYI matching funds from AT&T Bell Labs and Sun Microsystems, Inc.
Processor Efficient Parallel Algorithms
for the Two Disjoint Paths Problem,
and for Finding a Kuratowski Homeomorph*

Samir Khuller
Stephen G. Mitchell
Vijay V. Vazirani

Computer Science Department
Cornell University
Ithaca, NY 14853

Abstract

We give an NC algorithm for finding vertex disjoint $s_1, t_1$ and $s_2, t_2$ paths in an undirected graph $G$. An important step in solving the general problem is solving the planar case. A new structural property yields the parallelization, as well as a simpler linear time sequential algorithm for this case. We extend the algorithm to the non-planar case by giving an NC algorithm for finding a Kuratowski homeomorph, and, in particular, a homeomorph of $K_{3,3}$, in a non-planar graph. Our algorithms are processor efficient; in each case, the processor-time product of our algorithms is within a polylogarithmic factor of the best known sequential algorithm.

*Supported by NSF grant DCR 85-52938 and PYI matching funds from AT&T Bell Labs and Sun Microsystems, Inc.
1 Introduction

Given a graph $G$ and two pairs of vertices, $s_1,t_1$ and $s_2,t_2$, the two disjoint paths problem asks for vertex-disjoint paths connecting $s_i$ with $t_i$, $i = 1, 2$. This problem is well-studied from the point of view of sequential computation [PS 78], [Sh 80], [Se 80]; we give a fast parallel ($NC$) algorithm for it. In case $G$ is non-planar, our algorithm finds a Kuratowski homeomorph in $G$ (i.e., a subgraph homeomorphic to $K_{3,3}$ or $K_5$). This complements the known $NC$ planarity algorithms which give a planar embedding in the positive case; our algorithm provides a certificate of non-planarity in the negative case. Our algorithms are processor efficient; in each case, the processor-time product of our algorithms is within a polylogarithmic factor of the best known sequential algorithm.

An important step in solving the general problem is solving the planar case. A polynomial time algorithm for this case was given by Perl and Shiloach [PS 78]. Shiloach [Sh 80] showed how to solve the non-planar case as follows: Find a Kuratowski homeomorph— if it is a $K_{3,3}$ homeomorph, then use it as a high connectivity “switch” to find the two disjoint paths; if it is a $K_5$ homeomorph, use the algorithm of [Wa 68]. Independently, Seymour [Se 80] gave a polynomial time algorithm for the decision problem for general graphs; by self reducibility, this yields an algorithm for the search problem.

Using theorems from [PS 78] and [Sh 80] it is easy to show that the decision version of the two paths problem is in $NC$ (see Section 2). Our main contribution is to give an $NC$ algorithm for the search version, i.e., for actually finding the two disjoint paths. As pointed out in [KUW 85], an $NC$ algorithm for the decision version does not directly yield an $NC$ algorithm for the search version, since ordinary self-reducibility is a sequential process. We present algorithmically relevant structure (the $pq$-graph theorem) which makes the parallelization possible. Our algorithm requires $O(n)$ processors and $O(log^2 n)$ time. This structure also yields a simpler linear time sequential algorithm for the planar case.

Our proof of the $pq$-graph theorem is patterned on the familiar “crossing lemma”, which states that a cycle $C$ with four vertices $a, b, c$ and $d$ in this order cannot have $a$-$c$ and $b$-$d$ paths lying inside $C$. (This lemma follows from the Jordan Curve Theorem.) Both the crossing lemma and our $pq$-graph theorem relate the combinatorics of certain planar graphs to the geometry of their embeddings.

Our planar graph algorithm extends to $K_{3,3}$-free graphs using a theorem of Hall [Ha 43], as
in [Va 89]. The main difficulty in dealing with graphs containing a $K_{3,3}$ homeomorph is finding such a homeomorph in $NC$. The decision version of this problem was shown to be in $NC$ in [Va 89], and parallelizing the search version was left as an open problem. We solve this problem as follows: find any Kuratowski homeomorph in the graph, and in case this is a $K_5$, use it to find a $K_{3,3}$ homeomorph.

The Kuratowski homeomorph is found as follows: Given a non-planar graph $G$, use an $NC$ planarity testing algorithm to obtain a subgraph $G'$ of $G$ and an edge $(u, v)$, such that $G'$ is planar and $G' + (u, v)$ is non-planar. Obtain a maximal cycle containing vertices $u$ and $v$ in $G'$, and using ideas from Kuratowski’s theorem find a Kuratowski homeomorph in $G' + (u, v)$. This algorithm requires $O(n)$ processors and $O(\log^3 n)$ time; we leave open the problem of reducing this to the complexity of planarity testing, which is currently $O(n)$ processors and $O(\log^2 n)$ time [KIR 86].

Once the $K_{3,3}$ homeomorph is found, we also use it as a high connectivity “switch” to find the two disjoint paths; however, additional ideas are required to make this processor efficient. Our bounds are $O(n^2)$ processors and $O(\log^2 n)$ time, compared to the $O(n^2)$ sequential running time for the non-planar case (using the algorithm of [Sh 80], together with the $O(n^2)$ algorithm of [KR 87] for four-connected components).

The $k$-disjoint paths problem (where we need to find $k$ vertex-disjoint paths connecting specified pairs of vertices) is $NP$-complete if $k$ is part of the input [Ka 75]. For fixed $k \geq 3$, the problem was open for a long time. Recently, Robertson and Seymour have given polynomial time algorithms for this problem for any fixed $k$, derived from their extensive graph minor theory [RS 86a]. The following problem is central to the Robertson-Seymour theory: for a fixed graph $H$, decide whether the given graph $G$ contains $H$ as a minor. This problem has closely related structure to the $k$ disjoint paths problem. In fact the minor problem polynomial time reduces (even $NC$ reduces) to the $k$ disjoint paths problem [RS 85]. Robertson and Seymour’s polynomial time algorithm for minor testing [RS 86a] together with their proof of Wagner’s conjecture [RS 86b] has yielded non-constructive polynomial time algorithms for testing membership in any minor-closed family of graphs. For certain families (i.e., when the list of forbidden minors is known), this result gives explicit polynomial time algorithms as well (however, because of large multiplicative factors in the running time of the minor testing algorithm, these algorithms are not practical).
A natural question is whether the Robertson-Seymour theory can be used to obtain NC algorithms for the $k$-disjoint paths problem, for fixed $k$. Robertson and Seymour concentrate on the decision version of their problems, and rely on self-reducibility for solving the search version. One may need more structural properties in order to find parallel algorithms for the search versions. An NC algorithm for the $k$ disjoint paths problem will immediately imply that testing for membership in any minor-closed family of graphs is in NC, without actually producing such an algorithm. As above, if the list of forbidden minors is known, this result will give an explicit NC algorithm as well; several natural problems fall under this category (see e.g. [MRS 88]).

## 2 NC algorithm for the decision problem

**Lemma 1** Solving the two disjoint paths (search or decision) problem for a graph $G$ NC-reduces to solving it for the triconnected components of $G$.

**Proof:** Decompose $G$ into its “tree” $T'$ of triconnected components [MR 87]. The lemma is obvious if $s_i, t_i$ ($i = 1, 2$) are all in the same triconnected component. Suppose $s_i, t_i$ ($i = 1, 2$) are in components $S_i$ and $T_i$ of $T'$ respectively (see Fig. 1). Let $N_i$ be the path from $S_i$ to $T_i$ in $T'$. If $N_1$ and $N_2$ do not intersect the disjoint paths are easy to obtain. Suppose $N_1$ and $N_2$ intersect in a path $P_0, P_1, .., P_k$ in $T'$. Let $(a_i, b_i)$ be the separating pair between $P_i$ and $P_{i+1}$. In each triconnected component $P_i$ find “parallel” disjoint paths $P(a_{i-1}, a_i), Q(b_{i-1}, b_i)$ and “cross” disjoint paths $P'(a_{i-1}, b_i), Q'(b_{i-1}, a_i)$. (We use the notation $P(v_i, v_j)$ for a path from vertex $v_i$ to $v_j$.) If any component allows both parallel and cross paths, then using one of these we can always find the disjoint paths. Otherwise, there is a unique way of connecting $\{a_0, b_0\}$ with $\{a_{k-1}, b_{k-1}\}$. We can now determine whether this allows disjoint $s_i, t_i$ paths.

Henceforth we will assume that $G$ is triconnected.

**The decision algorithm:**

**Step 1:** Find the separating triples in $G$ [KR 87]. For each triple $\{a, b, c\}$ discard the components of $G - \{a, b, c\}$ which do not contain a vertex from $\{s_1, s_2, t_1, t_2\}$. Add virtual edges between the vertices in each triple. Let $G'$ be the new graph.
Figure 1: Decomposition tree $T'$ of triconnected components

Step 2: If $G'$ is planar then the answer to the decision version is "no" iff there is a face containing all four vertices $s_1, s_2, t_1, t_2$ in this order (by theorem 1 below and the fact that a triconnected graph has a unique planar embedding [Wh 33]).

Step 3: If $G'$ is non-planar then the answer is "yes" [Sh 80].

**Theorem 1 (Perl-Shiloach)** Let $G$ be a triconnected planar graph, and let $s_1, t_1, s_2, t_2$ be four vertices of $G$. Two disjoint paths from $s_1$ to $t_1$ and from $s_2$ to $t_2$ exist if and only if the vertices $s_1, s_2, t_1, t_2$ are not on a common face in this order.

3 Finding $k$ disjoint $u$-$v$ paths

There is a straightforward algorithm for finding $k$ disjoint $u$-$v$ paths in $O(\log^2 n)$ time using $O(M(n))$ processors by using flow techniques. A more sophisticated algorithm requiring $O(\log^2 n)$ time and $O(m)$ processors is developed in [KS 89]. For completeness, we sketch the simple algorithm below.
Given the graph \(G(V, E)\), we construct the directed graph \(G'(V', E')\) as follows:

\[
V' = \{x', x'' \mid x \in V - \{u, v\}\} \cup \{u, v\};
\]

\[
E' = \{(x', x'')\}
\cup \{(y'', x'), (x'', y') \mid (x, y) \in E, \{x, y\} \subseteq V - \{u, v\}\}
\cup \{(u, x') \mid (u, x) \in E\}
\cup \{(x'', v) \mid (x, v) \in E\}.
\]

In the directed graph \(G'\) we treat the node \(u\) as a "source" and the node \(v\) 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 \(u\)-\(v\) paths in \(G\). The paths in \(G\) will be vertex disjoint, since corresponding to each vertex \(x \in G\) we have two vertices \((x'\) and \(x'')\) in \(G'\) connected with 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 \(u\)-\(v\) paths in \(G\). A flow of size \(k\) in \(G'\) is obtained by \(k\) successive augmentations; each augmentation corresponds to finding a \(u\)-\(v\) path in \(G\). We can find the flow augmenting paths efficiently in parallel using matrix multiplication.

4 Two disjoint paths—the planar case

Our algorithm for finding disjoint paths in case \(G\) is planar and triconnected relies on the structure of "pq-graphs", described below. Since \(G\) is triconnected, using the algorithm in [KS 89] we can find three vertex disjoint paths \(P_a, P_b,\) and \(P_c\) from \(s_1\) to \(t_1\), and similarly three vertex disjoint paths \(Q_a, Q_b,\) and \(Q_c\) from \(s_2\) to \(t_2\). We denote by \(G_{pq}\) the subgraph of \(G\) consisting of the \(P_i\) and \(Q_j\) paths. We denote by \(P[v_i; v_j]\) the segment of the path \(P\) from \(v_i\) to \(v_j\).

Definition 1 \(G_{pq}\) will be called a pq-graph if there is a numbering for the \(P_i\) and \(Q_j\) paths such that when the \(P_i\)'s are ordered from \(s_1\) to \(t_1\) and the \(Q_j\)'s from \(s_2\) to \(t_2\), the following six conditions are satisfied.

1. The first intersection of each \(Q_j\) with any \(P\)-path is a vertex of \(P_1\).
2. The last intersection of each \(Q_j\) with any \(P\)-path is a vertex of \(P_3\).
3. Let \(v_1 \in Q_j \cap P_1, v_3 \in Q_j \cap P_3 (v_1, v_3 \neq s_1, t_1)\). Then \(Q_j[v_1; v_3]\) intersects \(P_2\).
4. The first intersection of each $P_i$ with any $Q$-path is a vertex of $Q_1$.

5. The last intersection of each $P_i$ with any $Q$-path is a vertex of $Q_3$.

6. Let $v_1 \in Q_1 \cap P_i$, $v_3 \in Q_3 \cap P_i$ ($v_1, v_3 \neq s_2, t_2$). Then $P_i[v_1; v_3]$ intersects $Q_2$.

Lemma 2 If $G_{pq}$ is not a pq-graph, then there exist disjoint $s_1, t_1$ and $s_2, t_2$ paths in $G_{pq}$. Moreover, these paths can be obtained in $O(\log^2 n)$ time using $O(n)$ processors.

The proof builds on ideas from [PS 78], where they impose conditions 1–3.

Theorem 2 If $G$ contains vertex disjoint $s_i, t_i$ paths, $i = 1, 2$, and if $G_{pq}$ is a pq-graph, then one of these paths can be taken to be either $P_2$ or $Q_2$.

A complete proof is given in the next section.

The search algorithm for planar graphs:

Step 1: Find a planar embedding of $G$. If in the planar embedding $s_1, s_2, t_1$ and $t_2$ are vertices on some face $F$ of $G$ in this order then there is no solution.

Step 2: Find three vertex disjoint $P$-paths from $s_1$ to $t_1$ and three vertex disjoint $Q$-paths from $s_2$ to $t_2$. Let $G_{pq}$ be the graph consisting of these six paths.

Step 3: If $G_{pq}$ is not a pq-graph then use lemma 2 to obtain the two disjoint paths.

Step 4: If $G_{pq}$ is a pq-graph then by theorem 2 either

1. $s_2, t_2$ are in the same connected component in $G - P_2$, or
2. $s_1, t_1$ are in the same connected component in $G - Q_2$.

Theorem 3 For planar graphs, there is an $O(\log^2 n)$ time algorithm for the two disjoint paths search problem using $O(n)$ processors.

Using the NC algorithms of [TV 85], [MR 87], and [KIR 86] and [KS 89] we can implement the above algorithm in the stated time and processor bounds. The sequential complexity of the above algorithm matches that of [PS 78] and is $O(n)$. 

6
5 Proof of the pq-graph theorem

The proof of theorem 2 uses the Jordan regions of a cycle in a plane embedded graph to derive results about the ordering of vertices on the cycle. The "crossing lemma", stated below, is prototypical of such an argument.

Let $G = (V, E)$ be a plane embedded graph. A cycle $C$ partitions the plane into itself and two connected open Jordan regions (disjoint from $C$). A face of $G$ is a cycle one of whose Jordan regions is empty, that is, contains no vertices or edges of $G$.

Theorem 2 will be presented here as a corollary to the following theorem.

**Theorem 2.1** In a pq-graph, vertices $s_1, s_2, t_1,$ and $t_2$ appear in that order on a face which is vertex disjoint from $p_2$ and $q_2$ (except at the endpoints).

**Corollary 2.2** If $G$ contains vertex disjoint $s_1, t_1$ and $s_2, t_2$ paths and if $G_{pq}$ is a pq-graph, then one of these paths can be taken to be either $p_2$ or $q_2$.

**Proof:** By theorem 2.1 the vertices $s_1, s_2, t_1$, and $t_2$ appear in order on a face $C$ of $G_{pq}$, where $C$ is disjoint from $p_2$ and $q_2$ (except at $s_1, s_2, t_1,$ and $t_2$). Define the four sections of $C$ to be the subpaths from $s_1$ to $s_2$, from $s_2$ to $t_1$, from $t_1$ to $t_2$, and from $t_2$ to $s_1$. Call the empty region of $C$ the inside. Then if $G$ has no path inside of $C$ between two different sections, then the vertices $s_1, s_2, t_1$, and $t_2$ lie on a face of $G$.

Therefore, if the two disjoint paths exist, there is some path $r$ in $G - G_{pq}$ inside $C$ connecting two different sections of $C$. Then we can take one path to include $r$ and edges of $C$, and the other can be taken to be either $p_2$ or $q_2$.

Some geometrical lemmas

Before we prove theorem 2.1, we present a few useful lemmas. The first is an old friend, and needs no proof.

**Lemma 2.3** (Crossing lemma) If $C$ is a cycle in the plane containing distinct vertices $a, b, c,$ and $d$ in that order, then there do not exist disjoint paths $p$ joining $a$ and $c$ and $q$ joining $b$ and $d$ which both lie inside $C$.
(We have already used the crossing lemma implicitly in the proof of corollary 2.2.)

**Lemma 2.4 (Endpoint lemma)** In a pq-graph,

1. \( s_1 \not\in q_2 \); \( t_1 \not\in q_1 \); \( s_2 \not\in p_2 \); \( t_2 \not\in p_1 \);
2. \( s_1 \not\in q_3 \); \( t_1 \not\in q_3 \); \( s_2 \not\in p_3 \); \( t_2 \not\in p_2 \).

**Proof:** Vertex \( s_1 \) cannot lie on \( q_2 \), since any path \( p_i \) must hit \( q_1 \) first before hitting \( q_2 \). The other assertions follow similarly.

\[ \square \]

Let \( C \) be a cycle in a plane embedded graph. If vertices \( s \) and \( t \) do not lie on \( C \), then they are separated by \( C \) if they lie in distinct Jordan regions of \( C \).

Given a cycle \( C \) and a path \( p \), a **touching** of \( p \) and \( C \) is a maximal common subpath (with one or more vertices). A **crossing** of \( p \) and \( C \) is a touching by which \( p \) crosses from one Jordan region of \( C \) to the other.

Given a path \( p \) and vertex disjoint paths \( q_1, q_2, \) and \( q_3 \), a **segment** of \( p \) is a maximal subpath \( s \) which is disjoint from the \( q_i \) paths, except possibly at the endpoints of \( s \). A \( q_i-q_j \) **hit** is a segment of \( p \) whose endpoints lie in \( q_i \) and \( q_j \).

**Lemma 2.5 (Cycle lemma)** Let \( G \) be a plane embedded graph, and suppose that vertices \( s \), \( t \) are joined by vertex disjoint paths \( q_1 \), \( q_2 \), and \( q_3 \). Suppose that \( C \) is a cycle separating \( s \) and \( t \). Then \( C \) has a \( q_1-q_3 \) hit.

**Remark:** Symmetrically, \( C \) must also have a \( q_1-q_2 \) hit and a \( q_2-q_3 \) hit.

**Proof:** Since \( s \) and \( t \) lie in opposite Jordan regions of \( C \), each path \( q_i \) must contain an odd number of \( C \) crossings. Since there are three \( q \) paths, \( C \) must contain an odd number of \( q \) path crossings.

Suppose that \( C \) has no \( q_1-q_3 \) hit. Denote by \( Q_{13} \) the region bounded by the cycle \( q_1q_3 \), not containing \( q_2 \). Similarly define \( Q_{12} \) and \( Q_{23} \). Denote

\[
\hat{Q}_{12} = Q_{12} \cup q_1 \\
\hat{Q}_{23} = Q_{23} \cup q_2.
\]

Orient the cycle \( C \). Let \( v \) be a vertex on \( C \), and let \( w \) be the vertex preceding \( v \) in the orientation of \( C \). Assign to \( v \) a state \( Q(v) \) as follows:
Then it is easy to verify that, traveling along \( C \) from an arbitrary initial vertex \( c_0 \), the possible state transitions are exactly those indicated by the diagram in Fig. 2.

![Diagram](image)

Figure 2: Possible state transitions for \( C \).

Also, it is easy to see that a cycle whose state transitions obey this diagram must have an even number of \( q \) path crossings. But we have argued that if \( C \) separates \( s \) and \( t \) then \( C \) has an odd number of crossings. Hence \( C \) must have a \( q_1-q_3 \) hit.

\[ \square \]

Suppose that \( G \) is a \( pq \)-graph. Define the regions \( Q_{ij} \) and \( P_{ij} \) as in the proof of the cycle lemma. Note that \( Q_{12}, Q_{23}, \) and \( Q_{13} \) are pairwise disjoint, as are \( P_{12}, P_{23}, \) and \( P_{13} \).

**Lemma 2.6 (Region lemma)** In a \( pq \)-graph

(i) \( s_1 \in Q_{13} \cup q_1 \);
(ii) \( s_2 \in P_{13} \cup p_1 \);
(iii) \( t_1 \in Q_{13} \cup q_3 \);
(iv) \( t_2 \in P_{13} \cup p_3 \).

9
Proof: We will show that \( s_2 \in \mathbf{P}_{13} \cup p_1 \). The other assertions follow similarly.

By the endpoint lemma, if \( s_2 \) lies on some \( p \) path, it lies on \( p_1 \). Suppose, therefore, that \( s_2 \) does not lie on any \( p \) path.

Since the first segment of \( q_1 \) is a path from \( s_2 \) to \( p_1 \), \( s_2 \) cannot lie in \( \mathbf{P}_{23} \). Similarly, \( t_2 \not\in \mathbf{P}_{12} \); by the endpoint lemma, \( t_2 \not\in p_1 \cup p_2 \). Suppose \( s_2 \in \mathbf{P}_{12} \). Then \( p_1p_2 \) is a cycle separating \( s_2 \) and \( t_2 \). By the cycle lemma, \( p_1p_2 \) must have a \( q_1-q_3 \) hit, contradicting the definition of \( pq \)-graphs. Therefore, \( s_2 \in \mathbf{P}_{13} \).

Let \( a \) be a vertex on \( p_1 \). Then we say that a path \( q_i \) straddles \( a \) if \( q_i \) has a \( p_1-p_1 \) segment lying in \( \mathbf{P}_{13} \) whose endpoints lie on \( p_1 \) on either side of \( a \). If, furthermore, \( a \) is the first hit of some path \( q_j \) with \( p_1 \), then we say also that \( q_i \) straddles \( s_2 \). We make similar definitions for \( s_1 \), \( t_1 \), and \( t_2 \).

**Lemma 2.7 (Straddle lemma)** No \( q \) path straddles \( s_2 \) or \( t_2 \). No \( p \) path straddles \( s_1 \) or \( t_1 \).

**Proof:** Refer to Fig. 3. Suppose that \( q_i \) straddles \( a \), the first hit of \( q_j \) on \( p_1 \). Let \( r \) be a segment of \( q_i \) lying in \( \mathbf{P}_{13} \) with endpoints \( x, y \neq a \) lying on \( p_1 \) on either side of \( a \). Let \( C \) be the cycle formed by \( r \) and the section of \( p_1p_2 \) containing \( p_2 \). Then from the definition of a \( pq \) graph it follows that \( C \) cannot have a \( q_1-q_3 \) hit.

Let \( C' \) be the cycle formed by \( r \) and the \( p_1 \) subpath \( xy \). Then one region of \( C' \) is \( R' \subseteq \mathbf{P}_{13} \), and one region of \( C \) is \( R = R' \cup \mathbf{P}_{12} \cup xy \). Because \( a \in R \) and \( q_j \) does not hit \( C \) before hitting \( a, s_2 \in R \).
By the region lemma \( t_2 \not\in \textbf{P}_{12} \); by the endpoint lemma \( t_2 \not\in xy \). Also, \( t_2 \not\in \textbf{R}' \), since the last hit of any \( q \) path is with \( p_3 \), not \( p_1 \). Therefore \( t_2 \not\in \textbf{R} \), and \( C \) separates \( s_2 \) and \( t_2 \), contrary to the cycle lemma.

Therefore \( q \) cannot straddle \( s_2 \). The other assertions follow similarly.

\[ \square \]

**The proof of theorem 2.1**

Let \( G \) be a \( pq \) graph. We start by constructing a *shell* \( C \) around \( G - q_2 \). Eventually, we show that \( C \) is the face required by the theorem.

**Proposition 2.8** \( G - q_2 \) has a directed cycle \( C \) (the shell) so that

(i) \( G - q_2 \) lies on the inside (right side) of \( C \);
(ii) the outside (left side) of \( C \) lies within \( \textbf{P}_{13} \);
(iii) \( C \) is disjoint from \( p_2 \) (except at \( s_1 \) and \( t_1 \)); and
(iv) \( C \) contains \( s_1, s_2, t_1, \) and \( t_2 \) in that order.

**Proof:** We give the construction in three stages. (See Fig. 4.)

Stage I:

Consider first the subgraph consisting of paths \( p_1, p_2, \) and \( p_3 \), embedded so that these paths appear in clockwise order around \( s_1 \). Take initially for \( C \) the cycle \( p_1p_3 \), oriented so that \( p_1 \) is directed from \( s_1 \) to \( t_1 \). Then \( p_2 \) lies on the right of \( C \), the outside of \( C \) is \( \textbf{P}_{13} \), and \( C \) is disjoint from \( p_2 \) except for the vertices \( s_1 \) and \( t_1 \), which appear on \( C \).

Stage II:

If \( s_2 \not\in p_1 \), then by the region lemma \( s_2 \in \textbf{P}_{13} \). In this case, add to our subgraph the segments of \( q_1 \) and \( q_3 \) from \( s_2 \) to their first hit on \( p_1 \), and reroute \( C \) to take these \( q \) segments. Do similarly if \( t_2 \not\in p_3 \). Then the enlarged subgraph still lies inside \( C \), the outside of \( C \) lies within \( \textbf{P}_{13} \), \( C \) is disjoint from \( p_2 \), and the vertices \( s_2, t_1, t_2, \) and \( s_1 \) appear on \( C \) in that order.

Stage III:

Since every \( q \) path now begins and ends on \( C \), any \( q \) subpath which escapes \( C \) must return to \( C \). Add \( q_1 \) to our subgraph segment by segment, being careful to reroute \( C \) along every segment which would otherwise escape. Let \( r \) be such a segment. The endpoints of \( r \) must lie
Figure 4: The shell $C$.

on a $p$ path. Since $r$ cannot have a $p_1$-$p_3$ hit, $r$ does not straddle $s_1$ or $t_1$. By the straddle lemma, $r$ does not straddle $s_2$ or $t_2$. Therefore $C$ still contains $s_2$, $t_1$, $t_2$, and $s_1$ in order. Also, $C$ continues to avoid $p_2$ and enclose the entire subgraph. In a similar manner, add $q_3$.

The remainder of the proof is devoted to showing that $q_2$ also lies inside $C$. The next two propositions show that $C$ is well behaved.

Denote by $C^+$ the oriented subpath of $C$ from $s_1$ to $s_2$ to $t_1$. Similarly define $C^-$ from $t_1$ to $t_2$ to $s_1$.

**Proposition 2.9**

(i) $C^+$ contains the first hit of $p_1$ with $q_1$;
(ii) $C^+$ contains the last hit of $p_1$ with $q_3$;
(iii) $C^-$ contains the first hit of $p_3$ with $q_1$;
(iv) $C^-$ contains the last hit of $p_3$ with $q_3$;
(v) $C^+$ contains the first hit of $q_1$ with $p_1$;
(vi) $C^+$ contains the first hit of $q_3$ with $p_1$;
(vii) $C^-$ contains the last hit of $q_1$ with $p_3$;
(viii) $C^-$ contains the last hit of $q_3$ with $p_3$;

**Proof:** (i) If $s_1 \in q_1$ then we need only observe that $s_1 \in C^+$. Otherwise, observe that the first hit $c$ of $p_1$ with $q_1$ lies on $C^+$ in stage I of the above construction, and cannot be straddled by any $q_1$ or $q_3$ segment in stages II or III. (ii)–(iv) follow similarly.

(v) If $s_2 \in p_1$ then we need only observe that $s_2 \in C^+$. Otherwise, observe that the first hit $d$ of $q_1$ with $p_1$ lies on $C^+$ in stage II of the above construction, and cannot be straddled by any
prop segment in stage III. (vi)-(viii) follow similarly.

**Proposition 2.10** Suppose \( s_2 \not\in p_1 \), and denote by \( a \) and \( b \) the first vertices of \( q_1 \) and \( q_3 \) upon \( p_1 \). Then these vertices appear on \( C^+ \) in the order \( s_1, a, s_2, b, t_1 \). Analogous statements hold at \( t_1, t_2, \) and \( s_1 \).

**Proof:** Let \( c \) denote the first hit of \( p_1 \) on \( q_1 \), and let \( d \) denote the last hit of \( p_1 \) on \( q_3 \). By the previous proposition \( a, b, c, \) and \( d \) lie on \( C^+ \), with \( c \) being the first, and \( d \) being the last. By the crossing lemma applied to \( C \) and \( c, b, a, \) and \( d, b \) cannot precede \( a \). By another application of the crossing lemma to \( c, b, s_2, \) and \( d, b \) cannot precede \( s_2 \). Similarly, \( s_2 \) cannot precede \( a \).

We now consider a cycle inside \( C \), made up mostly of \( q_1 \) and \( q_3 \), which we will show contains \( q_2 \) on its inside.

Let \( g \) be the path inside \( C \) from \( s_1 \) to \( t_1 \) constructed by following \( p_1 \) from \( s_1 \) to \( q_1 \), \( q_1 \) to \( s_2 \), \( q_3 \) from \( s_2 \) to the last hit of \( q_3 \) along \( p_1 \), and \( p_1 \) to \( t_1 \). Similarly define \( g' \) from \( t_1 \) to \( t_2 \) to \( s_1 \). Then by the preceding proposition and an application of the crossing lemma \( g \) and \( g' \) are vertex disjoint (except for \( s_1 \) and \( t_1 \)). We will show that \( q_2 \) lies inside \( gg' \). First we prove the following.

**Proposition 2.11** Consecutive hits of \( g \) with \( C^+ \) appear in forward order along \( C^+ \). Similarly, consecutive hits of \( g' \) with \( C^- \) appear in forward order along \( C^- \).

**Proof:** We show that \( g \) cannot pass through a vertex \( a \) on \( C^+ \) and subsequently hit a vertex \( b \) which precedes \( a \) on \( C^+ \). We proceed by induction on \( a \) along \( C^+ \). Initially, we take \( a = s_1 \). (See Fig. 5.)

If \( a \neq t_1 \) then \( g \) must proceed to hit \( C^+ \) at some vertex \( b \) which, by the induction hypothesis, succeeds \( a \) along \( C^+ \). If \( b = t_1 \), then \( g \) does not proceed beyond \( b \). Suppose that \( b \neq t_1 \), and let \( r \) denote the \( a-b \) subpath of \( g \). Since \( g \) does not self-intersect, \( g \) will hit neither \( a \) nor \( b \) again. Suppose that \( g \) hits some vertex \( c \in C^+ \) lying strictly between \( a \) and \( b \). Let \( r' \) denote the subpath of \( g \) from \( c \) to \( t_1 \). Then \( r \) and \( r' \) violate the crossing lemma applied to \( C \). Therefore \( g \) subsequently never hits any vertex preceding \( b \) on \( C^+ \).
So far we have shown that the shell $C$ is a face of the graph $G - q_2$ on which the vertices $s_2, t_1, t_2,$ and $s_1$ appear in the given order, and that $C$ is disjoint from $p_2$ (except for $s_1$ and $t_1$). We note that $q_2$ is disjoint from the cycle $gg'$ (except at $s_2$ and $t_2$). All that remains to be shown is that $q_2$ lies inside the cycle $gg'$, implying that $q_2$ (and hence all of $G$) lies inside $C$, and that $q_2$ is vertex disjoint from $C$ (except at $s_2$ and $t_2$).

We shall need the following. We say that $gg'$ has four sections, meaning the $t_2$-$s_1$ subpath (section I), the $s_1$-$s_2$ subpath (section II), the $s_2$-$t_1$ subpath (section III), and the $t_1$-$t_2$ subpath (section IV).

**Proposition 2.12** No $gg'$ segment of $p_2$ lying outside (to the left of) $gg'$ has endpoints in two different sections.

**Proof:** Suppose that $p_2$ has a $gg'$ segment $xy$ lying to the left of $gg'$, with $x$ and $y$ in different sections of $gg'$. By the endpoint lemma $s_2, t_2 \notin p_2$, so neither $x$ nor $y$ is equal to $s_2$ or $t_2$. Therefore both $gg'$ paths from $x$ to $y$ contain an $s$ or $t$ vertex, and in particular both paths hit $C$ at some point strictly between $x$ and $y$. If $x \neq s_1$ or $t_1$ let $a$ be the last vertex of $gg'$ (strictly) before $x$ which lies on $C$, and let $b$ be the next vertex of $gg'$ (strictly) after $x$ which lies on $C$. (See Fig. 6.) If $x = s_1$ or $t_1$, the choice of $a$ and $b$ depends on the direction of the $p_2$ segment. $gg'$ hits $y$ strictly after hitting $b$ (and before returning again to $a$), so the cycle formed by the $ab$ subpath of $gg'$ and $C$ encloses $y$ in the region to its left. But $gg'$ hits $C$ after hitting $y$, so that $gg'$ hits $C$ after hitting $b$ at a vertex between $a$ and $b$. This contradicts the previous proposition. □
Proposition 2.13 Path $q_2$ lies inside (to the right of) $gg'$. 

Proof: Partition $p_2$ into its $gg'$ segments. Since $s_1$ lies in sections I and II, and $t_1$ lies in sections III and IV, some $gg'$ segment $r$ of $p_2$ crosses from one of sections I or II to one of III or IV. By the previous proposition, $r$ lies inside $gg'$. Segment $r$ must hit $q_1$ first, $q_3$ last, and in between $r$ must hit an internal vertex of $q_2$. Since $q_2$ is disjoint from $gg'$, all of $q_2$ lies inside $gg'$. 

This concludes the proof of theorem 2.1.

6 Extracting a Kuratowski homeomorph

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 $V(B) \cap V(C)$.

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 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 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.

**Lemma 3** If $G$ is planar, and $G + e$ is non-planar and triconnected, then we can find a Kuratowski homeomorph in $G + e$ in $O(\log^2 n)$ time using $O(n)$ processors.

**Proof:** Let $u$ and $v$ be the endpoints of $e$. The proof of Kuratowski's theorem in [BM 77] relies on finding a cycle $C$ of $G$ that contains $u$ and $v$ and is such that the set $IntC$ is maximal. We give a parallel algorithm for finding such a maximal cycle. The rest of the proof follows by a case analysis given in [BM 77]. Since $G$ is biconnected, we can find a cycle $C$ in $G$ containing $u$ and $v$ by finding two disjoint $u$-$v$ paths (see section 3). 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:

$Outer_C = \{ B_i \mid B_i \text{ is embedded in } ExtC \}$

$Inner_C = \{ B_j \mid B_j \text{ is embedded in } IntC \}$

A bridge $B_i \in Outer_C$ is in the set $Outer$-skew$C$ if $B_i$ is skew to $(u, v)$; $B_i$ is in the set $Outer$-non-skew$C$ if it is not skew to $(u, v)$. Since $G + e$ is non-planar, there is at least one bridge in the set $Outer$-skew$C$ (otherwise $e$ can be embedded in $ExtC$ in a planar embedding of $G$).

We first modify the cycle $C$ to obtain a cycle $C^1$ containing $u$ and $v$ so that there are no bridges in the set $Outer$-non-skew$C^1$ with respect to the cycle $C^1$. We then show how to modify $C^1$ to obtain cycle $C^3$ containing $u$ and $v$ such that the bridges in set $Outer$-skew$C^3$ are single edges skew to $(u, v)$.

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 $x_i^1, x_i^2, \ldots, x_i^{k_i}$ (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 was shown in [BM 77] 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 $Outer-non-skew_C$ has all of its attachment vertices (and thus its attachment bar) on the segment $C[v; u]$ or $C[u; v]$.

Consider the planar embedding of $B_i$ and its attachment bar, and call the resultant graph $B_i'$ (see Fig. 7). 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 simple cycle $C_i'$. Let $P_i(x_i^1, x_i^{k_i})$ be the subpath of $C_i'$ from $x_i^1$ to $x_i^{k_i}$ avoiding the attachment bar of $B_i$.

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

We perform the transformation from $C^1$ to $C^3$ in two stages. Consider the cycle $C^1$ and its bridges $B_i$ in $Eztc^1$ which are skew to $(u, v)$. If $B_i$ has only two attachment vertices then clearly the bridge is only a path, and in fact, since the graph is triconnected, it is only a single edge. Assume $B_i$ has more than two attachment vertices on $C^1$. Let the attachment vertices on $C^1[u; v]$ be $x_1^1, x_2^1, ..., x_{k_i}^1$ (considered in order). The attachment vertices on $C^1[v; u]$ are $y_1^1, ..., y_{k_i}^1$ (see Fig. 8). We will replace $C^1[x_i^1; x_i^{k_i}]$ by a path $P'$ in $B_i$ to obtain $C^2$; and subsequently replace $C^2[y_i^1; y_i^{k_i}]$ by a path $P''$ in $B_i$ to obtain $C^3$, the maximal cycle.

Obtain the subgraph $B_i^1$ of $B_i$ by deleting the attachment vertices $y_i$ on $C^1[v; u]$ (see Fig. 9).
Consider the external face in a planar embedding of $B^1_i$. Find a path $P$ from $x^1_i$ to $x^k_i$ along the external face. This path may not be simple as shown in Fig. 10. Find the articulation vertex closest to $x^1_i$ on $P$, and obtain a simple path $P'$ by "short-circuiting" $P$. Replace the segment $C^1[x^1_i; x^k_i]$ by the path $P'$ to obtain the cycle $C^2$. Do this step in parallel for every bridge in $EktC^3$ skew to $(u, v)$. Now identify the subgraphs $B_i$ of $G$ which form bridges with respect to the new cycle $C^2$. Let $x^2_i$ and $y^2_i$ be the attachment vertices of $B_i$ on the upper and lower chains $C^2[u; v]$ and $C^2[v; u]$. Obtain the subgraph $B_i^2$ of $B_i$ by deleting the attachment vertices $x^2_i$ of $B_i$ on $C^2[u; v]$. In a planar embedding of $B_i^2$ replace the segment $C^2[y^1_i; y^1_i]$ by the path $P''$ from $y^1_i$ to $y^2_i$ in the external face. If $G$ is triconnected it is easy to see that the path $P''$ is simple (see Fig. 11). After the replacement we obtain cycle $C^3$; the only bridges in $EktC^3$ are single edges skew to $(u, v)$.

Consider the set $Inner_{C^3}$ of inner bridges of $C^3$. Since $G + e$ is non-planar, the set $Inner_{skew_{C^3}}$ is non-empty (else a planar embedding for $G + e$ can be obtained). If each bridge in $Inner_{skew_{C^3}}$ avoids every bridge in $Outer_{C^3}$, then each such bridge can be transferred to $EktC^3$, yielding a planar embedding for $G + e$. Hence, there must be an inner bridge $B_1$ which overlaps an outer bridge $B_2$ and which is skew to $(u, v)$.

![Figure 8: Bridge with attachment vertices skew to $(u, v)$](image)

At this point, by considering various cases for the possible configurations of attachment vertices of $B_1$ and $B_2$ on $C^3$, we can obtain a Kuratowski homeomorph as in [BM 77]. The
Figure 9: Subgraph $B_i^1$

Figure 10: General form of path $P'$

Figure 11: Path $P_i'$ is simple
parallel algorithm takes $O(\log^2 n)$ time using $O(n)$ processors using [KIR 86] and [SV 82].

From the proof of Lemma 3, we also conclude the following:

**Lemma 4** In a planar graph $G$, we can find a maximal cycle $C$ that contains $u$ and $v$ in $O(\log^2 n)$ time using $O(n)$ processors.

Lemma 3 yields the parallel algorithm described below.

**Algorithm for finding a Kuratowski homeomorph:**

Step 1: Order the edges in $E$ arbitrarily as $e_1, e_2, \ldots, e_m$. Let $G_i$ be the subgraph consisting of the first $i$ edges. Clearly, $G_1$ is planar, and $G_m$ is not.

Step 2: Find the smallest index $k$ such that $G_1, G_2, \ldots, G_{k-1}$ are all planar, but $G_k (= G_{k-1} + e_k)$ is not, using the planarity testing algorithm of [KIR 86].

Step 3: Find a triconnected non-planar component of $G_k$; call it $G_k'$. Note that $e_k \in G_k'$ and that $G_k' - e_k$ is planar and biconnected. Apply lemma 3 to the graph $G_k' - e_k$ and edge $e_k$ to find a Kuratowski homeomorph.

**Remarks:** We can assume that $G$ has $O(n)$ edges; if not, we can select $3n - 5$ of them, keeping $G$ non-planar. Step 2 can be implemented either by checking all the graphs $G_i$ in parallel for planarity, or by doing a binary search on the graphs $G_i$. Using the algorithm of [KIR 86], which takes $O(\log^2 n)$ time using $O(n)$ processors, we can implement the above algorithm to run in $O(\log^2 n)$ time with $O(n^2)$ processors, or in $O(\log^3 n)$ time using $O(n)$ processors. We leave open the problem of finding a Kuratowski homeomorph in the same time and processor bounds as for planarity testing, i.e., $O(n)$ processors and $O(\log^2 n)$ time.

**Theorem 4** In a non-planar graph $G$, a Kuratowski homeomorph can be obtained in $O(\log^2 n)$ time using $O(n^2)$ processors, or in $O(\log^3 n)$ time using $O(n)$ processors.
7 Finding a $K_{3,3}$ homeomorph

Our algorithm for finding a $K_{3,3}$ homeomorph is a parallelization of the sequential algorithm given in [As 85]. This algorithm is based on the following theorem of Hall [Ha 43].

**Theorem 5 (Hall)** Each triconnected component of a $K_{3,3}$-free graph (i.e., a graph not containing a $K_{3,3}$ homeomorph) is either planar or exactly the graph $K_5$.

**Theorem 6** There is an $O(\log^2 n)$ time algorithm using $O(n^2)$ processors, or an $O(\log^3 n)$ algorithm using $O(n)$ processors which finds a $K_{3,3}$ homeomorph (if one exists).

**Proof:** First decompose $G$ into its triconnected components. By Hall’s theorem, one of the components is non-planar and contains at least six vertices. Use the algorithm of lemma 4 to find any Kuratowski homeomorph $G'$. In case it is a homeomorph of $K_5$, call the vertices in $G'$ of degree four $v_1, v_2, v_3, v_4$, and $v_5$. If $G'$ is exactly the graph $K_5$, then the triconnected component containing $G'$ must contain another vertex $v$ not in $G'$. Obtain three disjoint paths from $v$ to any three vertices of the $K_5$, say $v_1, v_2$, and $v_3$. These can be found by introducing an artificial sink vertex $u$ and three edges from $u$ to $v_1, v_2$ and $v_3$, and then finding three vertex-disjoint $v-u$ paths. From the three paths and the $K_5$ it is easy to find a subgraph homeomorphic to $K_{3,3}$ by putting $v_1, v_2$, and $v_3$ in one partition of the $K_{3,3}$ and $v, v_4$, and $v_5$ in the other.

If the homeomorph $G'$ is a subdivision of $K_5$, then consider a vertex $u$ on the path $P(v_1, v_2)$, where $v_1$ and $v_2$ are vertices of degree four in $G'$. ($P$ is a sub-division of an edge of the $K_5$.) Since $G$ is triconnected, there must be a path in $G - \{v_1, v_2\}$ from $u$ to some other vertex $w$ of $G'$. Use this path to extract a $K_{3,3}$ homeomorph by considering where the path first hits $G'$ (details in [As 85]). \(\Box\)

8 Two disjoint paths—the non-planar case

The two paths problem for $K_{3,3}$-free graphs reduces to the planar case via Hall’s theorem as in [Va 89]. For the remaining case when $G$ contains a subgraph $G_{3,3}$ homeomorphic to $K_{3,3}$, we modify Shiloach’s algorithm [Sh 80]. Call the nine paths of $G_{3,3}$ representing the edges of $K_{3,3}$ $p$-edges, and call the six vertices of $G_{3,3}$ representing the six vertices of $K_{3,3}$ $p$-vertices.
Algorithm for triconnected graphs having a $K_{3,3}$ homeomorph:

Step 1: Find the separating triples in $G$ [KR 87]. For each triple $\{a, b, c\}$ discard the components of $G - \{a, b, c\}$ which do not contain a vertex from $\{s_1, s_2, t_1, t_2\}$. Add virtual edges between the vertices in each triple. Let $G'$ be the new graph.

Step 2: Find a $K_{3,3}$ homeomorph in $G'$ (else $G'$ is $K_{3,3}$-free); call it $G'_{3,3}$.

Step 3: Modify $G'_{3,3}$ to include $s_1$ as one of the "corner" vertices. This is done by finding three disjoint paths from $s_1$ to $G'_{3,3}$. Call the modified homeomorph $G''_{3,3}$.

Step 4: Modify $G''_{3,3}$ further to include $t_1$ either as a corner vertex or on a $p$-edge incident to $s_1$. Call this homeomorph $G'''_{3,3}$.

Step 5: Find four disjoint paths $\pi_1, \pi_2, \pi_3$ and $\pi_4$ connecting $s_1, t_1, s_2$ and $t_2$ with four $p$-vertices of $G'''_{3,3}$ different from $s_1$ and $t_1$. These paths together with the $K_{3,3}$ homeomorph yield the two disjoint $s_i, t_i$, $i = 1, 2$, paths.

Step 1 can be parallelized using the algorithm in [KR 87]. Theorem 4 yields a parallel algorithm for step 2. For steps 3 and 4, Shiloach gives a sequential path extension algorithm which modifies $G'_{3,3}$. We parallelize this by working with prefixes. We describe the procedure in detail for step 3; the idea for step 4 is similar.

We make $s_1$ a $p$-vertex by constructing three disjoint paths $P_1, P_2$, and $P_3$ from $s_1$ to three $p$-vertices on the same side of $G'_{3,3}$. Let $P_i$ have $v_1^i, v_2^i, ..., v_{k_i}$ as its vertices of intersection with the $p$-edges (considered in order from $s_1$). Let $P_1^i, ..., P_3^i$ denote the segments $P_i[s_1 ; v_1^i], ..., P_i[s_1 ; v_{k_i}]$ respectively. We define the set of prefixes $(i, j, k)$ of $P$ as $\{P_1^i, P_2^j, P_3^k\}$.

Lemma 5 The subgraph $G'_{3,3}$ and some set of prefixes yields a subgraph $G''_{3,3}$ of $G$ homeomorphic to $K_{3,3}$, with $s_1$ as a $p$-vertex.

Proof: The proof is based on an exhaustive case analysis, as in [Sh 80]. First consider the set $(1, 1, 1)$ of $P$. In case the prefixes are incident on different $p$-edges or on $p$-vertices, use one of the base cases from [Sh 80], for which no extension of any prefix is required to get $G'_{3,3}$. The non-trivial case is that in which all three prefixes are incident on the same $p$-edge, say $e_1$, with endpoints $v_1$ and $v_2$. In this case extend the middle prefix to its next intersection with a $p$-edge (or a $p$-vertex). There are several cases to consider:
1. The extended segment hits $e_1$.

2. The extended segment hits a $p$-edge different from $e_1$.

3. The extended segment hits a $p$-vertex different from $v_1$ and $v_2$.

![Diagram](image)

Figure 12: Extend the middle prefix $P_2$

![Diagram](image)

Figure 13: Extending $P_3$ yields a base case

In the first case, continue extending the segments by extending the middle prefix at each step. Since the $P_1$-paths end at different $p$-vertices, this construction eventually falls into case
two or three. By extending only the middle prefix, we ensure that the middle prefix does not intersect the two end segments \([v_1; x_1]\) and \([v_2; x_2]\) of \(e_1\), where \(x_1\) and \(x_2\) are the end vertices of the non-middle prefixes. With this condition, in the last two cases the solution follows from Shiloach's base cases.

We illustrate this construction with an example. In Fig. 12 the middle prefix is of path \(P_2\), which we extend to its second hit with a \(p\)-edge. If it hits a \(p\)-edge other than \(e_1\), we immediately apply case two. Assume that it is \(e_1\). Now the prefix of \(P_3\) becomes the middle prefix, which we extend to its second hit with a \(p\)-edge. In this case it is a \(p\)-edge different from \(e_1\); apply a base case to obtain \(G_{3,3}'\) (see Fig. 13).

We illustrate a sequence of extensions of the middle prefixes in Fig. 14; the sequence of extensions terminates when \(P_2\) hits a \(p\)-edge different from \(e_1\).

\[ \square \]

\[ \begin{array}{c}
\text{extend } P_2 \\
\text{prefix } (1,2,1) \\
\end{array} \quad \begin{array}{c}
\text{extend } P_2 \\
\text{prefix } (1,3,1) \\
\end{array} \]

\[ \begin{array}{c}
\text{extend } P_3 \\
\text{prefix } (1,3,2) \\
\end{array} \quad \begin{array}{c}
\text{extend } P_3 \\
\text{prefix } (1,4,2) \\
\end{array} \]

Figure 14: A sequence of extensions of prefixes of the \(P\)-paths
We now develop a parallel algorithm for finding the subgraph $G'_{3,3}$. First check all the base cases (when the first hits of the $P_i$ paths yield $G'_{3,3}$). The only non-trivial case is when the first hits of all three $P_i$ paths are on $p$-edge $e_1$. Now “guess” the ending configuration (there are only 6 of them) of the prefixes of the paths $P_1$, $P_2$ and $P_3$ on $e_1$. Extend the middle path (say $P_2$) of the guessed configuration until it first hits a $p$-vertex or an edge different from $e_1$. This can be done in $O(\log n)$ time using $O(n)$ processors. After extending $P_2$, compute the leftmost and rightmost hits ($x_l$ and $x_r$) of $P_2$ with $p$-edge $e_1$. The segments $e_1[v_1; x_l]$ and $e_1[x_r; v_2]$ are free of intersection with $P_2$. Extend paths $P_1$ and $P_3$ independently until they land on these segments in the same order as in the guessed configuration. We check to ensure that they do not have intersections with any $p$-edge other than $e_1$. (If they do, then we guessed the configuration incorrectly.)

Shiloach showed further that we can obtain $G''_{3,3}$ with $s_1$ as a $p$-vertex and with either:

1. $t_1$ also as a $p$-vertex, or

2. $t_1$ lying on a $p$-edge incident with $s_1$.

The construction is similar to that given above. Find disjoint paths $Q_1$, $Q_2$, and $Q_3$ connecting $t_1$ with three distinct $p$-vertices of $G'_{3,3}$; now there are more cases to consider, since the prefixes of the $Q_i$ paths may land on $p$-edges incident to $s_1$ or on $p$-edges not incident to $s_1$.

**Lemma 6** The subgraph $G'_{3,3}$ and some set of prefixes yields $G''_{3,3}$ satisfying the required conditions.

This construction is similar to the above construction and can be carried out in NC.

For Step 5 we introduce artificial source and sink vertices to construct the four $\pi_i$-paths connecting $s_1$, $t_1$, $s_2$ and $t_2$ with four $p$-vertices on $G''_{3,3}$ which are different from $s_1$ and $t_1$ (these are guaranteed to exist by Step 1). Shiloach showed how $G''_{3,3}$ could be used to make two disjoint connections, one between $\pi_1$ and $\pi_2$ and the other between $\pi_3$ and $\pi_4$, to yield the desired disjoint $s_1$, $t_1$ paths.

There are three cases to consider:

1. $s_1$ and $t_1$ are $p$-vertices not connected by a $p$-edge.

2. $s_1$ and $t_1$ are $p$-vertices connected by a $p$-edge.
3. \( t_1 \) is a \( p \)-edge incident with \( s_1 \).

Shiloach showed that by considering only the first hits of \( \pi_3 \) and \( \pi_4 \) we can obtain a solution for case 1.

The solution for case 2 is obtained as follows: the only non-trivial subcase (the others are symmetric) is as shown in Fig. 15. We describe first an \( O(n) \) sequential "game" played by paths \( \pi_1, \pi_3 \), and \( \pi_4 \) on \( p \)-edges \( e_2 \) and \( e_3 \). (Our game is a modification of Shiloach's sequential algorithm.) We say that a prefix of path \( \pi_i \) covers a prefix of \( \pi_j \) if they land on the same \( p \)-edge with the endpoint of \( \pi_j \) closer to \( s_1 \) than the endpoint of \( \pi_i \). The game starts with \( \pi_3 \). Inductively, we extend the current \( \pi \) path until it lands uncovered on \( p \)-edge \( e_2 \) or \( e_3 \), or until it leaves the game altogether by landing on a \( p \)-edge different from \( e_1 \), \( e_2 \), or \( e_3 \). Ultimately, one of the three paths leaves the game and we obtain prefixes allowing disjoint \( \pi_1-\pi_2 \) and \( \pi_3-\pi_4 \) connections.

Figure 15: The only non-trivial subcase for case 2

We give a parallel algorithm to obtain such a prefix. First guess which \( p \)-edge each path ends on, and which path has left the game. There are only six distinct configurations and we check all of them. Suppose we guess that the first path to leave is \( \pi_1 \). Extend \( \pi_1 \) to its first hit on some edge other than \( e_1, e_2 \) or \( e_3 \). (The appropriate prefix of \( \pi_1 \) can be obtained by a prefix computation in \( O(\log n) \) time using \( O(n) \) processors.) Extend \( \pi_3 \) and \( \pi_4 \) to their shortest prefixes on edges \( e_2 \) and \( e_3 \) which are not covered by the extended prefix of \( \pi_1 \). If our guess was correct, then from the success of the sequential game we know that the extensions of \( \pi_3 \) and \( \pi_4 \) hit no \( p \)-edges besides \( e_1, e_2, \) and \( e_3 \). It is now easy to obtain the required disjoint connections.
Figure 16: Paths $\pi_3$ and $\pi_4$ cover $\pi_1$

Figure 17: Path $\pi_1$ covers $\pi_4$
The solution for case 3 is very similar to the solution for case 2. The only difference is that all four πi paths participate in the game on all three edges e. Ultimately one of the four paths leaves the game to hit another edge. We guess which path is the first to leave and extend the other paths until they are not covered by the extended prefix of the first one. A solution can now be obtained (see [Sh 80] for details).

**Lemma 7** One of the sets of prefixes of π along with the modified G3,3 provides the disjoint π1-π2 and π3-π4 connections.

**Remark:** Our lemmas 5, 6, and 7 avoid Shiloach’s “W-assumption”, which relies on a sequential process, at the expense of more complicated case analyses.

The bottlenecks in the algorithm are finding four-connected components [KR 87] and finding a K3,3 homeomorph.

**Theorem 7** Given a graph G and vertices s1, t1, s2 and t2, we can solve the two paths problem in NC2 using O(n2) processors.

9 Acknowledgements

We wish to thank Dan Bienstock for sharing his knowledge of the graph minor theory, and Umesh Vazirani for valuable comments on the writeup. I, Samir Khuller, would like to thank the people at the IBM T. J. Watson Research Center for the pleasant environment provided during the summer of 1988.

References


