A New Graph Triconnectivity Algorithm and Its Parallelization*

Gary L. Miller†
University of Southern California, Los Angeles, CA 90089

Vijaya Ramachandran‡
University of Illinois, Urbana, IL 61801

August 9, 1988

We present a new algorithm for finding the triconnected components of an undirected graph. The algorithm is based on a method of searching graphs called 'ear decomposition'. A parallel implementation of the algorithm on a CRCW PRAM runs in $O(\log^2 n)$ parallel time using $O(n+m)$ processors, where n is the number of vertices and m is the number of edges in the graph.

*A preliminary version of this paper was presented at the *19th ACM Symposium on Theory of Computing*, New York, NY, May 1987.

† Supported by NSF Grant DCR 8514961.

‡ Supported by NSF Grant ECS 8404866 and the Semiconductor Research Corporation Grant 86-12-109.

1. Introduction

An *ear decomposition* [Wh, Lo] of an undirected graph is a partition of its edge set into an ordered collection of paths called *ears*. In this paper we present an efficient parallel algorithm based on ear decomposition for testing vertex triconnectivity of an undirected graph and for finding the triconnected components of the graph. Our algorithm runs in $O(\log^2(n+m))$ parallel time using $O(n+m)$ processors on a CRCW PRAM, where n is the number of vertices in the graph and m is the number of edges. A sequential linear-time algorithm for the problem is available [HoTa], but it is based on depth first search, and is not known to be efficiently parallelizable.

Parallel NC algorithms for testing triconnectivity and for finding triconnected components are reported in [JaKo, JaSi, MiRe], but none match our processor bound for general graphs. In particular, ours is the first efficient parallel algorithm for graph triconnectivity. More recently, building on the results we present, [RaVi] have obtained an efficient parallel triconnectivity algorithm that runs in logarithmic time. Also, [KanRa] have used the ear decomposition approach to obtain better sequential and parallel algorithms for graph four connectivity.

2. Parallel Model of Computation

The parallel model of computation that we will be using is the *PRAM* model, which consists of several independent sequential processors, each with its own private memory, communicating with one another through a global memory. In one unit of time, each processor can read one global or local memory location, execute a single RAM operation, and write into one global or local memory location.

PRAMs are classified according to restrictions on global memory access. An EREW PRAM is a PRAM for which simultaneous access to any memory location by different processors is forbidden for both reading and writing. In a CREW PRAM simultaneous reads are allowed but no simultaneous writes. A CRCW PRAM allows simultaneous reads and writes. In this case we have to specify how to resolve write conflicts. We will use the ARBITRARY model in which any one processor participating in a concurrent write may succeed, and the algorithm should work correctly regardless of which one succeeds. Of the three PRAM models we have listed, the EREW model is the most restrictive, and the ARBITRARY CRCW model is the most powerful. It is not difficult to see that any algorithm for the ARBITRARY CRCW PRAM that runs in parallel time T using P processors can be simulated by an EREW PRAM (and hence by a CREW PRAM) in parallel time $T\log P$ using the same number of processors, P.

Let S be a problem which, on an input of size n, can be solved on a PRAM by a parallel algorithm in parallel time $t(n)$ with $p(n)$ processors. The quantity $w(n)=t(n)\cdot p(n)$ represents the *work* done by the parallel algorithm. Any PRAM algorithm that performs work $w(n)$ can be converted into a sequential algorithm running in time $w(n)$ by having a single processor simulate each parallel step of the PRAM in $p(n)$ time units. More generally, a PRAM algorithm that runs in parallel time $t(n)$ with $p(n)$ processors also represents a PRAM algorithm performing $O(w(n))$ work for any processor count $P<p(n)$.

Define *polylog* $(n)=\bigcup_{k>0} O(\log^k n)$. Let S be a problem for which currently the best sequential algorithm runs in time $T(n)$. A PRAM algorithm A for S, running in parallel time $t(n)$ with $p(n)$ processors is *efficient* if

a) $t(n)=polylog(n)$; and

b) the work $w(n)=p(n)\cdot t(n)$ is $T(n)\cdot polylog(n)$.

An efficient parallel algorithm is one that achieves a high degree of parallelism and comes to within a polylog factor of optimal speed-up with respect to the current best sequential algorithm. A major goal in the design of parallel algorithms is to find efficient algorithms with $t(n)$ as small as possible. The simulations between the various PRAM models make the notion of an efficient algorithm invariant with respect to the particular PRAM model used. For more on the PRAM model and PRAM algorithms, see [KarRa].

3. Graph-theoretic Definitions

An *undirected graph* $G=(V,E)$ consists of a *vertex set* V and an *edge set* E containing unordered pairs of distinct elements from V. A *path* P in G is a sequence of vertices $<v_0, \cdots, v_k>$ such that $(v_{i-1},v_i)\in E, i=1,\cdots,k$. The path P *contains* the vertices $v_0,\cdots,v_k$ and the edges $(v_0,v_1),\cdots,(v_{k-1},v_k)$ and has *endpoints* v_0, v_k, and *internal vertices* $v_1,\cdots,v_{k-1}$. The path P is a *simple path* if $v_0,\cdots,v_{k-1}$ are distinct and $v_1,\cdots,v_k$ are distinct. P is a *simple cycle* if it is a simple path and $v_0=v_k$. A single vertex is a trivial path with no edges. We denote by $|P|$, the number of vertices contained in path P.

We will sometimes specify a graph G structurally without explicitly defining its vertex and edge sets. In such cases, $V(G)$ will denote the vertex set of G and $E(G)$ will denote the edge set of G.

Let $P=<v_0,\cdots,v_{k-1}>$ be a simple path. The path $P(v_i,v_j), 0\le i,j\le k-1$ is the simple path connecting v_i and v_j in P, i.e., the path $<v_i,v_{i+1},\cdots,v_j>$, if $i\le j$ or the path $<v_j,v_{j+1},\cdots,v_i>$, if $j<i$. Analogously, $P[v_i,v_j]$ consists of the path segments obtained when the edges and internal vertices of $P(v_i,v_j)$ are deleted from P.

An *ear decomposition* [Lo,Wh] $D=[P_0,\cdots,P_{r-1}]$ of an undirected graph $G=(V,E)$ is a partition of E into an ordered collection of edge disjoint simple paths $P_0,\cdots,P_{r-1}$ called *ears*, such that P_0 is a simple cycle and for $i>0$, P_i is a simple path (possibly a simple cycle) with each endpoint belonging to a smaller numbered ear, and with no internal vertices belonging to smaller number ears. D is an *open ear decomposition* if none of the $P_i, i=1, \cdots, r-1$ is a simple cycle. A *trivial ear* is an ear consisting of a single edge. A graph has an open ear decomposition if and only if it is biconnected [Wh].

Let $D=[P_0, \cdots, P_{r-1}]$ be an ear decomposition for a graph $G=(V,E)$. For a vertex v in V, we denote by *earnumber*(v), the index of the lowest-numbered ear that contains v; for an edge e in E, we denote by *earnumber*(e), the index of the unique ear that contains e. We will say that v *belongs to* $P_{earnumber(v)}$.

Let $G=(V,E)$ be an undirected graph and let $V' \subseteq V$. A graph $G'=(V',E')$ is a *subgraph* of G if $E' \subseteq E \cap \{(v_i,v_j) \mid v_i,v_j \in V'\}$. The *subgraph of G induced by* V' is the graph $G''=(V',E'')$ where $E''=E \cap \{(v_i,v_j) \mid v_i,v_j \in V'\}$.

An undirected graph $G=(V,E)$ is connected if there exists a path between every pair of vertices in V. For a graph G that is not connected, a *connected component* of G is an induced subgraph of G which is maximally connected.

A vertex $v \in V$ is an *articulation point* of a connected undirected graph $G=(V,E)$ if the subgraph induced by $V-\{v\}$ is not connected. G is *biconnected* if it contains no articulation point. A *biconnected component* of G is an induced subgraph of G which is maximally biconnected.

Let $G=(V,E)$ be a biconnected undirected graph. A pair of vertices $v_1,v_2 \in V$ is a *separating pair* for G if the induced subgraph on $V-\{v_1,v_2\}$ is not connected. G is *triconnected* if it contains no separating pair. We define *triconnected components* in Section 5.

Let $G=(V,E)$ be a biconnected graph, and let Q be a subgraph of G. Analogous to [Ev, p. 148], we define the *bridges of Q in G* as follows: Let V' be the vertices in $G-Q$, and consider the partition of V' into classes such that two vertices are in the same class if and only if there is a path connecting them which does not use any vertex of Q. Each such class K defines a *(nontrivial) bridge* $B=(V_B,E_B)$ of Q, where B is the subgraph of G with $V_B=K \cup$ {vertices of Q that are connected by an edge to a vertex in K}, and E_B containing the edges of G incident on a vertex in K. The vertices of Q which are connected by an edge to a vertex in K are called the *attachments* of B. An edge (u,v) in $G-Q$, with both u and v in Q, is a *trivial bridge* of Q, with attachments u and v.

Let $G=(V,E)$ be a biconnected graph, and let Q be a subgraph of G. We define the *bridge graph of* Q, $S=(V_S,E_S)$ as follows: Let the bridges of Q in G be $B_i, i=1, \cdots, k$. Then $V_S=V(Q) \cup \{B_1, \cdots, B_k\}$ and $E_S=E(Q) \cup \{(v,B_i) \mid v \in V(Q), 1 \le i \le k$, and v is an attachment of $B_i\}$.

Let $G=(V,E)$ be a biconnected graph with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$. We will denote the bridge graph of P_i by C_i. Let the bridges of P_i in G that contain vertices on ears numbered lower than i be $B_{r_1}, \cdots, B_{r_l}$: note that each such bridge has the two endpoints of P_i as attachments. We shall call these the *anchor bridges* of P_i. For any two vertices x,y on P_i, we denote by $V_i(x,y)$, the internal vertices of $P_i(x,y)$; we denote by $V_i[x,y]$, the vertices in $P_i[x,y]-\{x,y\}$ together with the vertices in the anchor bridges of P_i. In C_i, the bridge graph of P_i, $V_i[x,y]$ will represent the vertices in $P_i[x,y]-\{x,y\}$, together with the vertices on bridges in C_i corresponding to the anchor bridges. Note that either of $V_i(x,y)$ or $V_i[x,y]$ can be empty.

Let $G=(V,E)$ be a graph and let P be a simple path in G. If each bridge of P in G contains exactly one vertex not on P, then we call G the *star graph of* P and denote it by $G(P)$. We denote the bridges of $G(P)$ by *stars*. The unique vertex of a star that is not contained in P is called its *center*. Note that, in a connected graph G, the bridge graph of any simple path in G is a star graph.

Figure 1 illustrates some of our definitions relating to bridges.

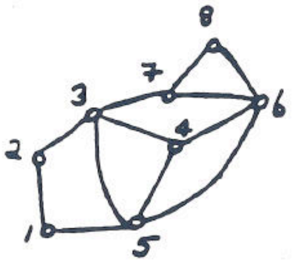
4. Ear Decomposition and Triconnectivity

Lemma 1 Let $G=(V,E)$ be an undirected graph with an ear decomposition containing r ears. Then $r=|E|-|V|+1$.
Proof Straightforward.

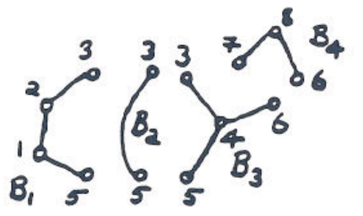
Lemma 2 [Wh] An undirected graph has an open ear decomposition if and only if it is biconnected.

Lemma 3 Let $G=(V,E)$ be a biconnected undirected graph for which vertices x and y form a separating pair. Let D be an open ear decomposition for G. Then there exists an ear P_i in D that contains both x and y, such that every path from a vertex in $V_i(x,y)$ to a vertex in $V_i[x,y]$ in G_i passes through either x or y. Further ear P_i uniquely determines a connected component C in $V-\{x,y\}$, in the sense that no other ear P_j in G that contains x and y, and has vertices in C, has the property that every path from a vertex in $V_j(x,y)$ to a vertex in $V_j[x,y]$ in G_j passes through either x or y.
Proof Since x and y form a separating pair, the subgraph of G induced by $V-\{x,y\}$ contains at least two connected components. Let C_1 and C_2 be two such connected components.

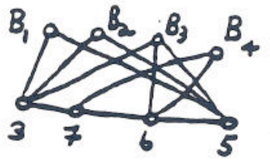
Case 1 The first ear P_0 contains no vertex in C_2 (see figure 2):



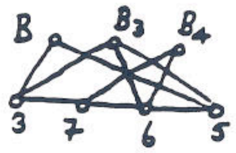
G with open ear decomposition $D = [P_0, P_1, P_2, P_3, P_4]$; $P_0 = <1,2,3,4,5,1>$, $P_1 = <3,7,6,5>$, $P_2 = <6,4>$, $P_3 = <7,8,6>$, $P_4 = <3,5>$.



Bridges of P_1.



Bridge graph of P_1.



Ear graph G_1.

figure 1
Illustrating the definitions

Consider the lowest-numbered ear, P_i, that passes through a vertex v in C_2. Since its endpoints are distinct and must be contained in earlier ears, P_i must contain x and y. Further, all vertices in $V_i(x,y)$ lie in C_2, and none of the vertices in $V_i[x,y]$ lie in C_2. Hence every bridge of P_i in G has its attachments all in $V_i(x,y) \cup \{x,y\}$, or all in $V_i[x,y] \cup \{x,y\}$. Hence every path from a

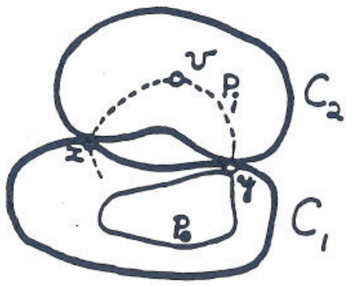


figure 2
Case 1 in the proof of Lemma 3

vertex in $V_i(x,y)$ to a vertex in $V_i[x,y]$ in G_i passes through either x or y.

To prove the second claim of the lemma for this case, suppose P_j is an ear which contains x and y and also a vertex, say u, in C_2. Then $j>i$, since P_i is the lowest-numbered ear to contain a vertex in C_2. Since P_i contains x and y, x and y must be the endpoints of P_j, and all of the other vertices in it lie in C_2. Further, since $i<j$ and vertex v is contained in P_i, the vertices in the bridge of P_j containing v (call it B') are in $V_j[x,y]$, and since C_2 is a connected component in the subgraph induced by $V-\{x,y\}$, there is a path from B' to the vertex u in $V_j(x,y)$ that avoids both x and y. This establishes the second claim of the theorem for this case.

Case 2 P_0 contains a vertex in C_2:

If P_0 contains no vertex in C_1, then case 1 applies to C_1. Otherwise P_0 contains at least one vertex from C_1, and one vertex from C_2. But then, since P_0 is a simple cycle, it must contain x and y, and again (by the argument of Case 1), every path from a vertex in $V_0(x,y)$ to a vertex in $V_0[x,y]$ must contain either x or y, and P_0 is the unique ear with this property, which has a vertex in C_2.[]

We will say that a separating pair x,y *separates* ear P_i if x and y lie on P_i, and the vertices in $V_i(x,y)$ are disconnected from the vertices in $V_i[x,y]$ in the subgraph of G induced by $V-\{x,y\}$. By Lemma 3, every separating pair in G separates some ear. (Note that a separating pair may separate more than one ear; for instance, in the graph G in figure 1, the pair 3,5 is a pair separating ears P_0 and P_4).

Lemma 4 Let $G=(V,E)$ be a biconnected graph with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$, and let x,y be a pair of vertices in G. Let the ears that contain both x and y be $P_{i_1}, \cdots, P_{i_k}$. Then x,y is not a separating pair for G if and only if, for each j, $1 \le j \le k$, there exists a vertex a_j in $V_{i_j}(x,y)$ and a vertex b_j in $V_{i_j}[x,y]$ belonging to a single star S in C_{i_j}, the bridge graph of P_{i_j}. Also, if P_{i_j} is an ear for which the above condition does not hold, then x,y is a separating pair that separates P_{i_j}.

Proof Let the pair x,y not be a separating pair, and consider any ear P_{i_j} that contains both x and y. Let a be any vertex in $V_{i_j}(x,y)$ and let b be any vertex in $V_{i_j}[x,y]$. Since x,y is not a separating pair, there exists a path $P=<a,w_1, \cdots ,w_l,b>$ in G that avoids both x and y. This implies that there is a subpath P' of P with $P'=<w_r, \cdots ,w_s>$ such that w_r is in $V_{i_j}(x,y)$, w_s is in $V_{i_j}[x,y]$, and none of the intermediate w_k lie on P_{i_j}. Hence there is a bridge B of P_{i_j} containing a and b, i.e., there is a star in C_{i_j} containing a and b.

If P_{i_j} is an ear for which every star in C_{i_j} has attachments only in $V_{i_j}(x,y)\cup\{x,y\}$ or only in $V_{i_j}[x,y]\cup\{x,y\}$, for each l, then clearly every path from a vertex in $V_{i_j}(x,y)$ to a vertex in $V_{i_j}[x,y]$ must pass through x or y, and hence x,y is a separating pair that separates P_{i_j}.[]

Observe that one consequence of Lemmas 3 and 4 is that the endpoints of an ear P_i do not form a pair separating P_i, if and only if there is a bridge B of P_i containing a vertex in an ear numbered lower than i, and B has an internal vertex of P_i as an attachment. Our algorithm will detect this situation as a special case, and hence we will call a pair of vertices x,y on an ear P_i a *candidate pair for* P_i if x,y is a pair separating P_i or (x,y) is an edge in P_i or x and y are endpoints of P_i $(i>0)$. Clearly, if we can determine the set of candidate pairs for P_i, we can extract from it the pairs separating P_i by deleting pairs that are endpoints of an edge in P_i, and checking if the endpoints of P_i form a pair separating P_i.

More generally, let $G(P)$ be a star graph. A pair of vertices x,y on P will be called a *pair separating* P if the vertices in the interior of $P(x,y)$ are separated from the vertices in $P[x,y]-\{x,y\}$ when the vertices x and y are deleted from G. A pair of vertices x,y on P will be called a *candidate pair* for P in G if x,y is a pair separating P, or x and y are endpoints of P, or (x,y) is an edge in P.

Lemmas 3 and 4 give us the following observation.

Observation 1: Let $G=(V,E)$ be a biconnected graph with an open ear decomposition $D=[P_0, \cdots ,P_{r-1}]$. Then x,y is a candidate pair for P_i in G if and only if it is a candidate pair for P_i in the bridge graph $C_i(P_i)$.

Let $G(P)$ be a star graph with stars $B,S_1, \cdots ,S_k$, where B is a star with the two endpoints of P as attachments. The *star path graph* H_j of $G(P)$ is the graph $H_j=P\cup B\cup S_j$. Analogously, given C_i, the bridge graph of P_i, together with its stars $S_{i,1}, \cdots ,S_{i,k}$, the star path graph $H_{i,j}$ is the graph $P_i\cup S_{i,j}\cup B$, where B is an auxiliary bridge with the two endpoints of P_i as attachments. As a consequence of Lemma 4 we have the following lemma about candidate pairs for an

ear. **Lemma 5** Let the stars on C_i be $S_{i,1}, \cdots, S_{i,k}$. Let $H_{i,j}^*$ be the planar embedding of the star path graph $H_{i,j}$ with P_i on the outer face. Then a pair of vertices on P_i is a candidate pair for P_i if and only if it lies on a common internal face in each $H_{i,j}^*$, $j=1, \cdots, k$.
Proof Follows from Lemma 4.[]

Corollary to Lemma 5 If C_i^* is a planar embedding of C_i with P_i on the outer face, then a pair of vertices x,y on P_i is a candidate pair for P_i if and only if x and y lie on a common inner face in C_i^*.

In general, this corollary may not apply, because C_i need not be planar. We now introduce the *star coalescing property:* namely, we will establish that if we enforce the planarity required in the corollary by *coalescing* the centers of any pair of stars that *interlace* (as defined below) to obtain a new graph C'_i, then the corollary applies to C'_i.

Two stars S_j and S_k in a star graph $G(P)$ *interlace* (see also [Ev, p. 149]) if one of following two hold:

1) There exist four distinct vertices a,b,c,d in increasing order in P such that a and c belong to $S_j(S_k)$ and b and d belong to $S_k(S_j)$; or

2) There are three distinct vertices on P that belong to both S_j and S_k.

Observation 2: (See, e.g., [Ev, p. 150].) A star graph $G(P)$ has a planar embedding with P on the outer face if and only if no pair of stars interlace.

The operation of *coalescing* two stars S_j and S_k is the process of forming a single new star S_l from S_j and S_k by combining the attachments of S_j and S_k, and deleting S_j and S_k. Given a star graph $G(P)$, the *coalesced graph* G' of G is the graph obtained from G by coalescing all pairs of stars that interlace. Note that G' is a star graph with respect to P, and G' has a planar embedding with P on the outer face, since no pair of stars interlace in G'.

Let $G(P)$ be a star graph in which no pair of stars interlace. We will call the planar embedding of G with P as the outer face the *star embedding* G^* *of* G.

Theorem 1 A pair of vertices x,y on the path P of a star graph $G(P)$ is a candidate pair for G if and only if it is a candidate pair for the coalesced graph $G'(P)$ of $G(P)$.
Proof By induction on the number of stars N in G.

Base: When $N=1$, then $G'=G$, and the theorem is trivially true.

Induction Step: Suppose the theorem is true for any $G(P)$ with up to k stars, and consider the case when $G(P)$ has $k+1$ stars $S_1, \cdots, S_{k+1}$. Let G_1 be G with S_{k+1} removed. By the induction hypothesis, a pair of vertices x,y on P is a candidate pair for G_1 if and only if it is a candidate pair for G'_1.

Now consider $G_2 = G'_1 \cup S_{k+1}$. First we note that from Lemma 5 and the induction hypothesis, a pair of vertices x,y on P is a candidate pair for G if and only if it is a candidate pair for G_2. We now prove that x,y is a candidate pair for G_2 if and only if it is a candidate pair for G'_2. Since $G'_2 = G'$, this will prove the theorem.

Suppose x,y is a candidate pair for G_2. Then both vertices lie on a common inner face F_1 in G'^*_1 and also on a common inner face F_2 in H^*_{k+1}. Hence all vertices on $P(x,y)$ lie in F_2. Since x and y lie on a single face F_2 in H^*_{k+1}, none of the stars of G'_1 that have all points of attachment between x and y (inclusive) in $P(x,y)$, interlace with S_{k+1}, and hence none of them are coalesced with S_{k+1} in G'_2. Hence x and y will lie on a common inner face in G'^*_2, and hence form a candidate pair for G'_2.

Suppose x,y is a candidate pair for G'_2. If (x,y) is an edge in P then x,y is a candidate pair for G_2. Suppose (x,y) is not an edge in P, and suppose that there is a path between a vertex a in $P(x,y) - \{x,y\}$ and a vertex b in $P[x,y] - \{x,y\}$ in G_2 that avoids both x and y. But every path in G_2 is present in G'_2 and hence there would be a path between a and b in G'_2 that avoids both x and y, contradicting the fact that x,y is a separating pair belonging to G'_2. Hence x,y must be a pair separating P in $G_2(P)$.

This establishes the induction step and the theorem is proved.[]

Corollary to Theorem 1 Let $G(P)$ be a star graph and let $G_1(P)$ be the graph derived from $G(P)$ by coalescing some of the interlacing stars on $G(P)$. Then x,y is a candidate pair for G if and only if it is a candidate pair for $G_1(P)$.

Let us refer to the set of vertices on P_i that lie on a common inner face in G'^*_i as a candidate set for P_i. A pair of vertices is a candidate pair for P_i if and only if it lies in a candidate set for P_i. A candidate set S for ear P_i is a *nontrivial* candidate set if it contains a pair separating P_i. Since every separating pair for G is a candidate pair for some ear P_i (Lemma 3), any algorithm that determines the candidate sets for all ears is an algorithm that finds all separating pairs for a graph.

By the results we have proved above, we can find all candidate sets in G by forming the bridge graph for each nontrivial ear, and then extracting the candidate sets from the coalesced graph of the bridge graph. In order to obtain an efficient implementation of this algorithm, we will not use the bridge graph of each ear, but instead a closely related graph called the *ear graph*, which we now define.

Let G be a biconnected graph with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$. Let $B_{r_1}, \cdots, B_{r_l}$ be the anchor bridges of P_i. The *ear graph of* P_i, denoted by $G_i(P_i)$ is the graph obtained from the bridge graph of P_i by a) coalescing all stars corresponding to anchor bridges; and b) removing any two-attachment bridges with the endpoints of the ear as attachments. We will call the star obtained by coalescing all anchor bridges, the *anchoring star* of $G_i(P_i)$.

Lemma 6 A pair of vertices x,y separates ear P_i in G if and only if it separates P_i in the ear graph $G_i(P_i)$.

Proof By Lemmas 3 and 4, x,y separates ear P_i in G if and only if it separates P_i in the bridge graph $C_i(P_i)$.

Now consider the ear graph $G_i(P_i)$. The ear graph $G_i(P_i)$ is obtained from the bridge graph $C_i(P_i)$ by coalescing all anchor bridges, and deleting multiple two-attachment bridges with the endpoints of the ear as attachments.

Deleting a star with attachments only to the endpoints of an ear can neither create nor destroy candidate pairs. We now observe that every pair of anchor bridges with an internal attachment on P_i must interlace: this is because every anchor bridge has the two endpoints of P_i as attachments. Hence $G_i(P_i)$ is the graph derived from $C_i(P_i)$ by coalescing some interlacing stars. The lemma now follows from the Corollary to Theorem 1.[]

Lemma 7 Let $G=(V,E)$ be a biconnected graph with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$, and let $|V|=n$ and $|E|=m$. Then the total size of the ear graphs of all ears in D is $O(m)$.

Proof Each ear graph consists of a nontrivial ear P_i together with a collection of stars on P_i. The size of all of the P_i is $O(m)$. So we only need to bound the size of all of the stars in all of the ear graphs.

Consider an edge (u,v) in G. This edge appears as an internal attachment edge in at most two ear graphs: once for the ear $P_{earnumber(u)}$ and once for ear $P_{earnumber(v)}$. Thus the number of internal attachment edges in all of the stars is no more than $2m$.

We now bound the number of attachment edges to endpoints of ears. Since we delete all stars with only the endpoints of an ear as attachments, every star in an ear graph $G_i(P_i)$ with an attachment to an endpoint of P_i also has an internal attachment in P_i. A star can contain at most two attachments to endpoints of an ear. Hence for each star that contains attachments to endpoints of its ear, we charge these attachments to an internal attachment. Since the number of internal attachment edges is no more than $2m$, the number of attachment edges to endpoints of ears is no more than $4m$. Hence the total size of all of the ear graphs is $O(m)$.[]

Lemma 5, Theorem 1 and Lemma 6 establish the validity of the following algorithm to find the nontrivial candidate sets in a biconnected graph.

Algorithm 1 Finding the Nontrivial Candidate Sets

Input A biconnected graph $G=(V,E)$.

1. Find an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$ for G.

2. *For* each nontrivial ear P_j *do*

A) Construct the ear graph G_j.

B) Coalesce all interlacing stars on $G_j(P_j)$ to form the coalesced graph G'_j. Find a planar embedding of G'_j, with P_j on the external face, and identify each set of vertices on P_j on a common inner face in this embedding as a candidate set.

C) *If* $j>0$ let the endpoints of P_j be u and v. *If* the anchor bridge of P_j has an internal attachment on P_j, then delete the candidate set $\{u,v\}$.

D) Delete any doubleton candidate set that contains the endpoints of an edge in P_j.

Let $|V|=n$ and $|E|=m$. Step 1 has an $O(\log m)$ time parallel algorithm with $O(m)$ processors on a CRCW PRAM [MaScVi, MiRa]. In the next section, we give an $O(\log^2 m)$ time parallel algorithm on a CRCW PRAM with a linear number of processors for steps 2A and 2B. Clearly steps 2C and 2D are trivial to implement. Finally in section 6, we show how to obtain the triconnected components of a biconnected graph, given the nontrivial candidate sets. We find the triconnected components using *Tutte splits* [Tu, HoTa2] in contrast to the earlier algorithm based on depth first search [HoTa].

We also note that Algorithm 1 has an efficient sequential implementation. Step 1 has a linear-time algorithm [MaScVi, MiRa]. The ear graphs can be constructed sequentially for the non-trivial ears, starting from the highest-numbered non-trivial ear, by collapsing together all vertices in higher numbered ears that belong to the same bridge with respect to lower numbered ears. This algorithm runs in time $O((m+n)\cdot\alpha(m,m))$ where $\alpha(m,m)$ is the inverse Ackermann function. Step 2B can has a fairly straightforward linear-time algorithm [MiRa2]. The remaining steps are easy to implement in linear time.

5. Finding Candidate Sets

In this section we give algorithms to implement steps 2A and 2B in Algorithm 1.

5.1. Forming the Ear Graphs

Let $G=(V,E)$ be a biconnected graph with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$. Let $|V|=n$ and $|E|=m$.

We first provide some preliminary definitions. Given $G=(V,E)$ with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$, we define $G_{i,j}$, the (i,j) *ear graph of* G, *for* $i<j$, as follows: Let $P_{i,j}=\bigcup_{k=i}^{j} P_k$, and let $U_{i,j}$ be the set of vertices in $P_{i,j}$ that are contained in ears numbered lower than i (these are some of the endpoints of $P_i, P_{i+1}, \cdots, P_j$). Let $R_1, \cdots, R_l$ be the bridges of $P_{i,j}$ that contain a vertex on an ear numbered lower than i (the *anchor bridges* of $P_{i,j}$), and let $S_1, \cdots, S_k$ be the bridges of $P_{i,j}$ whose attachments are all in $U_{i,j}$. Let $C_{i,j}$ be the bridge graph of $P_{i,j}$. Then $G_{i,j}$ is the graph $C_{i,j}$ with the bridges $R_1, \cdots, R_l$ coalesced (the anchoring star of $C_{i,j}$), and with the bridges $S_1, \cdots, S_l$ deleted. Note that $G_{i,i}$ is simply the ear graph G_i, and $G_{0,r-1}$ is the input graph G.

Let $G'_{i,j}$ be a graph consisting of $P_{i,j}$, together with a collection of stars (i.e., connected graphs in which exactly one vertex has degree greater than 1) with attachments on $P_{i,j}$, of which a subset $\{R_k, k=1,2, \cdots, l\}$ are 'marked'. Then $G'_{i,j}$ is a *partial ear graph of* $P_{i,j}$ if the graph obtained by coalescing the $R_k, k=1,2, \cdots, l$ is the ear graph $G_{i,j}$, excluding the attachments of the anchoring star to vertices in $U_{i,j}$.

The following algorithm constructs the ear graph of each ear in parallel. The construction proceeds in stages. In each stage the algorithm constructs a partial ear graph $G'_{i,j}$ for a collection of $P_{i,j}$.'s. Finally, after having obtained $G'_{i,i}$ for each i, the algorithm coalesces all of the marked stars in $G'_{i,i}$ and provides attachments for this coalesced star to the two endpoints of P_i, to form the ear graph G_i.

Algorithm 2A Forming the Ear Graphs $G_i=(V_i,E_i), i=0, \cdots, r-1$.

Input: Undirected biconnected graph $G=(V,E)$ with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$.

```
1. For $i=1, \cdots, \lceil \log r \rceil$ do
for $j=0$ (step 2) to $2^i$ pardo
```

let $a=\left\lceil \frac{jr}{2^i} \right\rceil, b=\left\lceil \frac{(j+1)r}{2^i} \right\rceil, c=\left\lceil \frac{(j+2)r}{2^i} \right\rceil$.

a) Form $G'_{a,b-1}$ from $G'_{a,c-1}$ as follows:

i) Delete the subgraph induced by $P_{a,b-1}$ from $G'_{a,c-1}$. Call the resulting graph $H_{b,c-1}$.

ii) Find connected components in $H_{b,c-1}$.

iii) Mark any connected component that contains a vertex that was previously marked.

iv) Mark any connected component containing a vertex on an ear numbered lower than a (these will be some of the endpoints of ears in $P_{b,c-1}$).

v) Collapse each connected component into a single vertex.

vi) Restore $P_{a,b-1}$ together with all edges incident on it. Remove multiple copies of any edge in this graph, and delete all vertices not connected to $P_{a,b-1}$.

vii) Remove the bridges of $P_{a,b-1}$ with attachments only to vertices in $U_{a,b-1}$, i.e, only to some of the endpoints of $P_{a,b-1}$ which lie on ears numbered lower than a.

viii) Remove any edge connecting a marked vertex to a vertex in $U_{a,b-1}$.

b) Form $G'_{b,c-1}$ from $G'_{a,c-1}$ in a similar manner.

2. For each i, coalesce all marked bridges in $G'_{i,i}$ to form star S_i, and introduce attachments from S_i to the two endpoints of P_i; if there is no marked bridge in $G'_{i,i}$, then introduce a new marked star S_i with attachments only to the two endpoints of P_i. Label this the anchor bridge of P_i.

Lemma 8 In Algorithm 2A, for each i, the size of all of the $G'_{j,k}$'s present in this step is $O(m)$.

Proof Note that the size of all of the $G'_{j,k}$'s excluding attachments to the corresponding $U_{j,k}$'s is $O(n+m)$, since any edge of G appears at most twice in this collection. We now bound the attachment edges to $U_{j,k}$ in $G'_{j,k}$ over all j,k's for fixed i.

Consider $G'_{j,k}$, where $j=y\cdot 2^i$ and $k=(y+1)\cdot 2^i$, for some j. First observe that for any marked vertex in $G'_{j,k}$ we remove all attachments to $U_{j,k}$ in step viii. So we only need to consider bridges containing only vertices belonging to higher-numbered ears ($P_x, x>k$). For each such bridge B in $G'_{j,k}$ the algorithm retains an edge for each distinct attachment to a vertex v in $U_{j,k}$. We can charge this attachment to the lowest numbered ear P_l that is contained in B and has an attachment to v. Note that $l>k$, since B is a bridge containing vertices only from ears numbered larger than k. Any edge is charged at most twice by this scheme (once for each endpoint of the edge) among bridges of all $P_{j,k}$ with i fixed. Hence the sizes of all of the $G'_{j,k}$'s present in this step is $O(m)$.[]

Lemma 9 Algorithm 2A correctly finds the ear graph of each ear.

Proof It is clear that Algorithm 2A without parts vii) and viii) in steps 1a and 1b, constructs the bridge graph of each ear. We need to show that the algorithm as specified constructs the ear graph of each ear.

We establish this by showing by induction that in the *i*th iteration of the main step, each $G'_{j,k}$ formed is a partial ear graph of the corresponding $P_{j,k}$.

Base i=1: Let $b=\lceil r/2 \rceil$. When i=1 the algorithm constructs two graphs $G'_{0,b-1}$ and $G'_{b,r-1}$. Since the bridge graph $C_{0,b-1}$ of $P_{0,b-1}$ contains no anchor bridges, step 1a of Algorithm 2A constructs $C_{0,b-1}$, which is clearly a partial ear graph of $P_{0,b-1}$.

The bridge graph $C_{b,r-1}$ of $P_{b,r-1}$ contains only anchor bridges and all of these bridges have attachments only to $U_{b,r-1}$. Hence any partial ear graph of $P_{b,r-1}$ contains no bridges. Algorithm 2A removes all bridges of $P_{b,r-1}$ in step 1b vii) and hence the graph constructed by the algorithm is a partial bridge graph of $P_{b,r-1}$.

Induction step Suppose the result is true until the $(i-1)$st iteration, and consider iteration i. Let $G'_{a,b-1}$ be constructed in step 1a from $G'_{a,c-1}$ in the *i*th stage.

First we show that every attachment of the anchoring bridge of $G_{a,b-1}$ is contained in one of the marked stars of $G'_{a,b-1}$. Let v be an attachment vertex of an anchoring bridge of $P_{a,b-1}$ in G. If v is also an attachment of an anchoring bridge of $P_{a,c-1}$ in G then by the induction hypothesis there is a marked star in $G'_{a,c-1}$ that has an attachment to v. The connected component containing this marked star is marked in step 1a iii) of Algorithm 2A and hence v is an attachment of a marked star in $G'_{a,b-1}$. If v is not an attachment of an anchoring bridge of $P_{a,c-1}$ in G then in order for v to be an attachment edge of an anchoring bridge in $C_{a,b-1}$, either it must be connected to an anchoring bridge of $P_{a,c-1}$ through internal vertices in $P_{b,c-1}$ or it must be connected through internal vertices in $P_{b,c-1}$ to a vertex in $U_{b,c-1}$ that belongs to an ear numbered lower than a. In the former case the corresponding connected component is marked in step 1a iii) of Algorithm 2A and in the latter case it is marked in step 1a iv) of Algorithm 2A.

A similar argument holds for attachments of the anchoring bridge of $G_{b,c-1}$.

If an attachment does not belong to an anchoring star of $G_{a,b-1}$ or $G_{b,c-1}$ then we claim that it cannot be an attachment of a marked star in $G'_{a,b-1}$ or $G'_{b,c-1}$. This is so since the algorithm only deletes edges from the graph, and hence never induces a path between two vertices in a subgraph of $G'_{x,y}$ if a path did not exist in the subgraph of G induced by $V(G'_{x,y})$.

Finally we note that every non-anchor bridge of $P_{a,b-1}$ in G appears as a non-anchor bridge in $G'_{a,b-1}$ as constructed by Algorithm 2A. This is because steps 1a vii) and viii) and steps 1b vii) and viii) delete only edges incident on vertices belonging to ears numbered lower than a and b respectively, and hence none of the edges in non-anchor bridges are removed.

This establishes the induction step and the lemma is proved.[]

The major computation in each parallel step of the algorithm involves finding connected components in subgraphs of the $G'_{i,j}$'s. Since by Lemma 8, the total size of the graphs present at any given step is $O(m)$, each parallel step can be implemented in $O(\log m)$ time on a CRCW PRAM with $O(m)$ processors [ShVi]. Finally since the algorithm has $\log m$ parallel stages, it runs on a CRCW PRAM in $O(\log^2 m)$ time with $O(m)$ processors.

5.2. Forming the Coalesced Graph

We now turn to step 2B, which finds the coalesced graph G' of a star graph $G(P)$, and determines from it, the candidate sets of $G(P)$.

Our parallel algorithm to coalesce all interlacing stars in a star graph $G(P)$ runs in $O(\log^2 q)$ time using $O(q)$ processors, where q is the number of edges in the stars and in the path. Recall that the star embedding of a star graph with no interlacing stars consists of embedding the path P as a horizontal line, and all the stars and their edges above the line in a well nested fashion with no two edges crossing. Each face in this embedding, ignoring the exterior face, lies directly below exactly one star and every star with k edges sits directly above $k-1$ faces. Thus, associated with the embedded stars is an ordered rooted tree: one vertex for each star and one for each face, which we call *star-vertex* and *face-vertex* respectively. The root of this tree is the exterior face. The children of a star-vertex are its faces in order and the children of a face-vertex are the stars that lie directly below it in order. We call this tree the *face-star tree*. The face-star tree is the main data structure that we use for the parallel star coalescing algorithm.

We distinguish between distinct and nondistinct attachments of the stars. In figure 3 we show a star embedding of a star graph $G(P)$ together with its face-star tree for the case of distinct attachments, i.e., each vertex on P is common to at most one star. As one can see from the example, the face-star tree is any rooted and ordered tree such that 1) the root is labeled a face, and each level of vertices is labeled alternately as faces and stars, and 2) each leaf is labeled as a face. The last condition just requires that all leaves are of even depth.

The face-star tree gives us most of the information about the star embedding. We show how to extract the information that is in the embedded star graph from the face-star tree. We start with a definition. The *Euler tour* of an ordered rooted tree T is the cycle starting at the root that traces the tree in a counter-clockwise order. In our example from figure 3 the Euler tour is $<A,a,B,a,C,c,D,c,C,d,E,d,C,a,A,b,F,b,G,e,H,e,G,f,\ I,f,J,f,G,b,A>$, where face vertices are labeled by upper case letters and star vertices by lower case letters. A *corner* of an Euler tour $<v_1, \cdots, v_k>$ is any triple (v_{i-1}, v_i, v_{i+1}) of successive vertices on the tour. The triple is a corner of star (face) if v_i is a star (face). Note that each attachment is defined by a corner of the star it belongs to, with the two face vertices in the corner corresponding to the faces bounding the attachment edge.

Star embedding

Face-star tree

figure 3
A star embedding with distinct attachments
and its face-star tree

The general case of nondistinct attachments is slightly more complicated. Since $G(P)$ has no multiple two-attachment stars with the same pair of attachments (by definition), any star graph with a star embedding that is derived from $G(P)$ by coalescing stars has a unique embedding. The attachment vertices correspond to a consecutive set of star corners in the Euler tour of the face-star tree of such an embedding. In figure 4 we exhibit a star embedding in which the attachments are not all distinct. Here the attachment x corresponds to the corners (F,c,B), (B,a,A), (A,d,G) and (G,e,I). Thus, since our algorithm will maintain the face-star tree and not the star graph, we need to maintain a list of corners that determine the beginning and end of each attachment vertex.

figure 4

A general star embedding with its face-star tree

Our parallel star coalescing algorithm is a divide-and-conquer algorithm that divides the set of stars in half and recursively and in parallel finds the face-star tree for the coalesced graph in each half, and then combines the two trees using *Merge-Tree*, the heart of the algorithm. We give an $O(\log p)$ time p processor algorithm to merge the face-star trees of the two halves, where p is the total number of attachments in the two two coalesced graphs. This gives an $O(\log^2 q)$ time q processor algorithm for star coalescing.

The procedure Merge-Tree is substantially simplified if we first sort the original stars as follows. Let $left(S)$ and $right(S)$ be the first and last attachments of S on P. We say a vertex u on P is less than another vertex v on P if u appears before v on P. We say that star S is before star S' in the *Euler order* if (1) $left(S)<left(S')$ or (2) $left(S)=left(S')$ and $right(S)<right(S')$. Note that in the star embedding, the Euler order gives the sequence in which the stars first appear on the Euler tour.

We first sort the stars in $G(P)$ with respect to the Euler order in $O(\log q)$ time using q processors [Co] (as a preprocessing step we coalesce all stars with the same span). Note that since each recursive call of the parallel star coalescing algorithm returns the face-star tree, by the above remark we may think of the stars being returned as sorted. On the other hand, by the following observation it is relatively easy to maintain the stars in order even if they are being coalesced: If interlacing stars are coalesced, then the Euler order on the new stars is obtained from the old Euler order by taking each interlacing class of stars and associating it with the minimum numbered star in its class. Thus we can maintain the Euler order throughout the computation without sorting.

The main procedure Merge-Tree has as input the face-star trees of two star embeddings $G^*(P)$ and $G'^*(P)$. By presorting, as defined above, we may also assume that there is a point δ such that every star in $G(P)$ has a point of attachment at δ or to its left, and every star in $G'(P)$

has all of its attachments at δ or to its right. The procedure returns with the face-star tree of the star embedding of the coalesced graph of $G(P) \cup G'(P)$. The procedure has four steps which may be combined in a slightly more efficient algorithm, but for clarity, are best viewed separately:

1) Determine which stars in T (T') interlace with the same star in T' (T).

2) Coalesce all stars in T (T') that interlace with the same star in T' (T).

3) For each pair of interlacing stars S, S', with $S \in T$ and $S' \in T'$, add the attachments of S to the right of δ to those of S'.

4) Splice the tree T' into the tree T.

To implement step 1 we start by determining those stars in T' that interlace with the same star in T. We use the fact that every star S in T has an attachment at δ or to its left. Note that if S' is a star in T' that interlaces with S, then S interlaces with the parent star of S', if it exists. Thus, if $S'_1, \cdots, S'_k$ are the stars that are the children of the root of T', and S' is a descendant of S'_i, then S must interlace with every star on the path from S' to S'_i in T'. Thus it suffices to find, for each attachment of each star S in T, the lowest level star in T' with which it interlaces. We can find this star by determining in which corner of T' the attachment lies. We do this by merging the attachment of $G(P)$ with those of $G'(P)$, either by sorting the points or by using a relatively simple pointer jumping scheme. The latter idea can be performed from the original sorting as part of the preprocessing. Thus the merging takes only a constant amount of time.

To find the set of stars of T that interlace with a given star in T', we note that the stars of T that interlace with some star in T' all lie on the rightmost path, say τ, in T, i.e., the path that starts from the root and ends at the rightmost leaf. Each star S' in T' interlace with exactly those stars in T that have an attachment in the span of S'; this corresponds to stars on a single segment of τ. Thus each star in T' can determine this segment of τ in constant time given our preprocessing. In fact it is sufficient for only the stars $S'_1, \cdots, S'_k$, the children of the root in T' to determine their segments on τ.

We now have the set of stars in T that must be coalesced and the set of stars in T' that must be coalesced. We do this coalescing in step 2. Clearly we can coalesce two neighboring stars in constant time, since one star must be the parent of the other, or the two stars must be siblings. Thus this step can be done in $O(\log p)$ time with $O(p)$ processors using either parallel tree contraction [MiRe] or the Euler tour technique on trees [TaVi].

At this point at most one star in T' can interlace with a given star in T and at most one star in T can interlace with a star in T'. Let us call a star that currently interlaces with a star in the other half, an *interlacing star*. The only interlacing stars in T lie on the path τ, and the only interlacing stars in T' are children of the root. In step 3 we add the attachments to the right of δ of each interlacing star in T to its mate in T' so that the interlacing stars in T' have their own attachments as well as the attachments (to the right of δ) of their mate in T.

Finally in step 4 we replace all subtrees to the right of δ for each interlacing star S in T with all the subtrees of its mate S'_j in T'. This can be performed in constant time. On the other hand, if S'_i is a star in T' that does not interlace with any star in T then it must lie in one of the faces defined by T. In this case we attach the subtree at S'_i to this face.

This concludes the description of the algorithm to coalesce all interlacing stars in a star graph with q edges in $O(\log^2 q)$ time using $O(q)$ processors. Since the total size of the ear graphs of all nontrivial ears is $O(m)$ this gives an algorithm to find the coalesced graphs of all ear graphs of G in $O(\log^2 n)$ time using $O(m)$ processors.

The face-star tree data structure now allows us to extract the candidate sets efficiently. For each face vertex f, the information available at its parent in the tree gives the leftmost vertex l and rightmost vertex r on P that belong to the face f. Similarly, the information at each child vertex c gives a segment s_c on P between l and r that does not belong to face f. The candidate set defined by f is the set of vertices in the interval $[l,r]$ excluding the vertices in the segments s_c, where c ranges over the children of f in the face-star tree. Thus the vertices on each face in the star embedding of the coalesced graph can be obtained as a circular linked list in constant time by having a processor at each vertex in the face-star tree.

6. Finding Triconnected Components

We first review some material from [Tu, HoTa2]. A *multigraph* $G=(V,E)$ is an undirected graph in which there can be several edges between the same pair of vertices. An edge e in a multigraph is denoted by (a,b,i) to indicate that it is the ith edge between a and b; the third entry in the triplet may be omitted for one of the edges between a and b.

A pair of vertices a,b in a multigraph $G=(V,E)$ is a separating pair if and only if there are two nontrivial bridges, or at least three bridges, one of which is nontrivial, of $\{a,b\}$ in G. If G has no separating pairs then G is triconnected. The pair a,b is a *nontrivial* separating pair if there are two nontrivial bridges of a,b in G.

Let a,b be a separating pair for a biconnected multigraph $G=(V,E)$, and let B be a bridge of a,b in G which is biconnected and has $|E(B)|\geq 2$. We can apply a *Tutte split* [Tu, HoTa2] $s(a,b,i)$ to G by forming G_1 and G_2 from G, where G_1 is $B\cup\{(a,b,i)\}$ and G_2 is the induced subgraph on $(V-V(B))\cup\{a,b\}$, together with the edge (a,b,i). The graphs G_1 and G_2 are called *split graphs of G with respect to a,b*. The *Tutte components* of G are obtained by successively applying a Tutte split to split graphs until no Tutte split is possible. Every Tutte component is one of three types: i) a triconnected simple graph; ii) a simple cycle (a *polygon);* or iii) a pair of vertices with at least three edges between them (a *bond);* the Tutte components of a biconnected multigraph G are the unique *triconnected components* of G.

Let $G=(V,E)$ be a biconnected graph with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$. Let a,b be a pair separating P_i. Let $B_1, \cdots, B_k$ be the bridges of P_i with an attachment in the interior of $P_i(a,b)$, and let $T_i(a,b)=(\bigcup_{j=1}^{k} B_i) \cup P_i(a,b)$. It is easy to see that $T_i(a,b)$ is a bridge of a,b. Then the *ear split* $e(a,b,i)$ consists of forming the *upper split graph* $G_1=T_i(a,b) \cup \{(a,b,i)\}$ and the *lower split graph* G_2, which is the induced subgraph on $(V-V(T_i(a,b))) \cup \{a,b\}$ together with the edge (a,b,i). Note that the ear split $e(a,b,i)$ is a Tutte split if one of $G_1-\{(a,b,i)\}$ or $G_2-\{(a,b,i)\}$ is biconnected.

Let S be a nontrivial candidate set for ear P_i. Two vertices u,v in S are an *adjacent separating pair for P_i* if u,v is a pair separating P_i and S contains no vertex in $V_i(u,v)$. Two vertices a,b in S are an *extremal separating pair for P_i* if S contains no vertex in $V_i[a,b]$.

We now prove the following theorem.

Theorem 2 Let $G=(V,E)$ be a biconnected graph with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$. Let a,b be an adjacent (extremal) separating pair for P_i in G, and let G_1 and G_2 be, respectively, the upper and lower split graphs obtained by the ear split $e(a,b,i)$. Then,

a) $G_1-\{(a,b,i)\}$ $(G_2-\{(a,b,i)\})$ is biconnected.

b) The ear decomposition D_1 induced by D on G_1 by replacing P_i by the simple cycle formed by $P_i(a,b)$ followed by the newly added edge (b,a,i) is a valid open ear decomposition for G_1; likewise, the ear decomposition D_2 induced by D on G_2 by replacing $P_i(a,b)$ by the newly added edge (a,b,i) is a valid open ear decomposition for G_2.

c) Let c,d be a pair separating some $P_j, 0 \le j \le r-1$ in G. If $\{c,d\} \ne \{a,b\}$ or $i \ne j$ then c and d lie in one of G_1 or G_2, and c,d is a separating pair for P_j in the split graph in which P_j, c, and d lie.

d) Every separating pair in G_1 or in G_2 is a separating pair in G.

Proof

a) If $G_1-\{(a,b,i)\}$ is not biconnected then let c be an articulation point in the graph. c cannot lie on $P_i(a,b)$ since this would imply that it is part of the candidate set for which a,b is an adjacent separating pair. But c cannot lie on a bridge of $P_i(a,b)$ since this would imply that G is not biconnected.

Similarly $G_2 - \{(a,b,i)\}$ is biconnected if a,b is an extremal separating pair.

b) If there were an ear in D with one endpoint in G_1 (G_2) and the other endpoint not in G_1 (G_2) then a,b would not be a separating pair.

c and d) If $i \neq j$ let P_j lie in G_k (where k=1 or 2). We note that the ear graph of P_j in G_k is the same as the ear graph of P_j in G. Hence c,d is a pair separating P_j in G if and only if it is a pair separating P_j in G_k.

If $i=j$ we note that in G_1 the bridges of P_i are precisely those bridges of P_i in G that have attachments on an internal vertex of $P_i(a,b)$. Hence if c and d lie on $P_i(a,b)$ then c,d separates P_i in G if and only if it separates $P_i(a,b)$ in G_1. An analogous argument holds for G_2 in the case when c and d lie on $P_i[a,b]$.[]

We now present the algorithm for finding triconnected components.

Algorithm 3 Finding Triconnected Components

Input A biconnected graph G=(V,E) with an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$, and the nontrivial candidate sets for each ear.
Output The triconnected components of G.

```
1. For each nontrivial ear $P_i$ do
      for each nontrivial candidate set $S$ for $P_i$ do
      a) For each adjacent separating pair $u,v$ in $S$ replace $G$ by its upper split graph $G_1$ and its
         lower split graph $G_2$ for the ear split $e(u,v,i)$. Replace $D$ by the ear decomposition $D_1$ for
         $G_1$ and the ear decomposition $D_2$ for $G_2$ as in part b) of Theorem 2.
      b) If $|S|>2$, then replace $G$ by its upper split graph $G_1$ and its lower split graph $G_2$ for the
         extremal separating pair $u,v$ in $S$. Replace the ear decomposition $D$ by the corresponding ear
         decompositions $D_1$ and $D_2$. (If $i$=0 and $u$ and $v$ are endpoints of $P'_0$ then perform this ear
         split only if there are at least two edges between $u$ and $v$.)
2. Split off multiple edges in the remaining split graphs to form the bonds.
```

Lemma 10 Algorithm 3 generates the Tutte components of G.
Proof Follows from Theorem 2.[]

For a parallel implementation of Algorithm 3 we implement step 1 in $\log r$ stages. In the first stage we locate the median *active* ear, call it P_i, where an *active ear* is an ear having a non-trivial candidate set. Then we find the bridges of $G_{0,i-1}$, and in each bridge we locate the smallest-numbered active ear. Let these ears be $P_i, P_{i_1}, \cdots, P_{i_t}$. In parallel we perform step 1 on each of these ears; this is essentially a connectivity algorithm. All of the preceding can be done in $O(\log m)$ time with a linear number of processors on a CRCW PRAM. At this point we have a collection of split graphs, and by Theorem 2, the number of active ears in any of them is at most half the number of active ears in the original graph. We now recursively apply the above step in each of the split graphs, and in $\log r$ stages we will be done. This gives an $O(\log^2 m)$ algorithm with $O(m)$ processors to find the triconnected components.

We also note that Algorithm 3 can be easily modified to obtain the tree of 3-connected components (or "auxiliary graph" [HoTa2]) with the same time and processor bounds.

REFERENCES

[Ev] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, MD, 1979.

[GaTa] H. N. Gabow, R. E. Tarjan, "A linear-time algorithm for a special case of disjoint set union," *J. Comput. Syst. Sci.*, vol. 30, 1985, pp. 209-221.

[HoTa] J. E. Hopcroft, R. E. Tarjan, "Dividing a graph into triconnected components," *SIAM J. Comput.*, 1973, pp.135-158.

[HoTa2] J. E. Hopcroft, R. E. Tarjan, "Finding the triconnected components of a graph," TR 72-140, Computer Science Department, Cornell University, Ithaca, NY, 1972.

[JaKo] J. Ja'Ja, S. R. Kosaraju, private communication.

[JaSi] J. Ja'Ja, J. Simon, "Parallel algorithms in graph theory: planarity testing," *SIAM J. Comput.*, 11, 1982, pp. 314-328.

[KanRa] A. Kanevsky, V. Ramachandran, "Improved algorithms for graph four-connectivity," *Proc. 28th IEEE Symp. on Foundations of Comp. Sci*, IEEE Press, 1987.

[KarRa] R. M . Karp, V. Ramachandran, "Parallel algorithms for shared memory machines," *Handbook of Theoretical Computer Science*, J. Van Leeuwen, ed., North Holland, 1988, to appear.

[Lo] L. Lovasz,"Computing ears and branchings in parallel," *Proc. 26th IEEE Ann. Symp. on Foundations of Comp. Sci.*, 1985, pp. 464-467.

[MaScVi] Y. Maon, B. Schieber, U. Vishkin, "Parallel ear decomposition search (EDS) and st-numbering in graphs," *VLSI Algorithms and Architectures*, Lecture Notes in Computer Science vol. 227, 1986, pp. 34-45.

[MiRa] G. L. Miller, V. Ramachandran, "Efficient parallel ear decomposition with applications," unpublished manuscript, MSRI, Berkeley, CA, January 1986.

[MiRa2] G. L. Miller, V. Ramachandran, "A new graph triconnectivity algorithm and its parallelization," *Proc. 19th Annual ACM Symp. on Theory of Computing*, 1987, pp. 254-263.

[MiRe] G. L. Miller, J. H. Reif, "Parallel tree contraction and its applications," *Proc. 26th FOCS*, 1985.

[RaVi] V. Ramachandran, U. Vishkin, "Efficient parallel triconnectivity in logarithmic time," *Proc. Aegean Workshop on Computing*, Corfu, Greece, June-July 1988, Springer Verlag Lecture Notes in Computer Science, 1988.

[Ta] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," *JACM*, vol. 22, 1975, pp. 215-225.

[TaVi] R. E. Tarjan, U. Vishkin, "Finding biconnected components and computing tree functions in logarithmic parallel time," *SIAM J. Comput.*, vol. 14, 1985, pp. 862-874.

[Tu] W. T. Tutte, *Connectivity in Graphs*, University of Toronto Press, 1966.

[Wh] H. Whitney, "Non-separable and planar graphs," *Trans. Amer. Math. Soc.* 34, 1932, pp.339-362.