In SIAM J. Computing, 1993, pp. 587-616.

FINDING TRICONNECTED COMPONENTS BY LOCAL REPLACEMENT[1]

DONALD FUSSELL [3,5] , VIJAYA RAMACHANDRAN[2,3] AND RAMAKRISHNA THURIMELLA[4,5]

Abstract. We present a parallel algorithm for finding triconnected components on a CRCW PRAM. The time complexity of our algorithm is $O(\log n)$ and the processor-time product is $O((m+n)\log\log n)$ where n is the number of vertices, and m is the number of edges of the input graph. Our algorithm, like other parallel algorithms for this problem, is based on open ear decomposition but it employs a new technique, local replacement, to improve the complexity. Only the need to use the subroutines for connected components and integer sorting, for which no optimal parallel algorithm that runs in $O(\log n)$ time is known, prevents our algorithm from achieving optimality.

1. Introduction. A connected graph $G = (V, E)$ is k-vertex connected if it has at least $(k+1)$ vertices and removal of any $(k-1)$ vertices leaves the graph connected. Designing efficient algorithms for determining the connectivity of graphs has been a subject of great interest in the last two decades. Applications of graph connectivity to problems in computer science are numerous. Network reliability is one of them: algorithms for edge and vertex connectivity can be used to check the robustness of a network against link and node failures respectively. In spite of all the attention this subject has received, $O(m+n)$-time sequential algorithms for testing k-edge and k-vertex connectivity of an n node m vertex graph are known only for $k < 3$ [5],[12]. Recently, Gabow has devised a very nice algorithm for edge connectivity. His algorithm, unlike previous algorithms for connectivity, does not appeal to Menger's theorem. It runs in $O(km\log(n^2/m))$ time [9]. The algorithms for vertex connectivity for $3 < k \leq \sqrt{n}$ currently require $O(k^2n^2)$ time [14], [2], [20].

The subject of this paper is the parallel complexity of 3-vertex connectivity. The importance of 3-vertex connectivity stems from the fact that if a planar graph is 3-vertex connected (triconnected), then it has a unique embedding on a sphere. Hence an efficient algorithm that divides a graph into triconnected components is sometimes useful as a subroutine in problems like planarity testing and planar graph isomorphism.

We present in this paper an algorithm, based on open ear decomposition, for dividing a biconnected graph into triconnected components. The model of computation

[1] A preliminary version of this paper was presented at the 16th International Colloquium on Automata, Languages, and Programming.

[2] This work was supported in part by Joint Services Electronics Program under Contract N00014-85-C-0149 at the Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801, and by NSF Contract CCR 89-10707.

[3] Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712.

[4] This work was done while this author was with the University of Texas at Austin. Presently with Department of Mathematics and Computer Science, University of Denver, Denver, CO 80208.

[5] Supported in part by the ONR under Contracts N00014-86-K-0763 and N00014-86-K-0597.

used in this paper is a concurrent-read-concurrent-write PRAM where write conflicts are resolved arbitrarily (the ARBITRARY CRCW PRAM model). See [15] for a discussion on the PRAM model. Our algorithm runs in $O(\log n)$ time performing at most $O((m+n)\log\log n)$ work where m and n are the number of edges and the number of vertices of the input graph respectively.

The first optimal sequential triconnected component algorithm (based on depth-first search (DFS)) was given by Hopcroft and Tarjan in 1972 [11]. Several parallel algorithms (e.g. [13], [16]) have been developed since then for the triconnected component problem using techniques other than DFS, since the question of finding a DFS spanning tree efficiently in parallel remains one of the major open problems in the area of parallel algorithm design. The algorithms in [13], [16] use parallel matrix multiplication as a subroutine, hence their processor complexity is far from optimal. Significant progress has been made in recent years: first, Miller and Ramachandran [18] gave an $O(\log^2 n)$ parallel algorithm; later, Ramachandran and Vishkin [22] gave an algorithm with $O(\log n)$ parallel time for the more restricted problem of finding separating pairs. A drawback with both these algorithms is that the work performed by them in the worst-case is $O(\log^2 n)$ factor off the optimal. Independent of [18] and [22], Fussell and Thurimella [7] came up with a parallel algorithm for finding separating pairs whose time complexity is $O(\log n)$ while the work performed is only $O(\log n)$ factor off the optimal; detecting separating pairs forms the central part of any triconnected component algorithm.

The chief method employed by [18] and [22] can be broadly classified as divide-and-conquer. Additional complexity improvements are unlikely using this approach due to the "end-node sharing" problem: the difficulty arising from two or more ears sharing an end vertex. A novel technique known as *local replacement* was introduced in [7] as a method for obtaining efficient parallel reductions.

Using the local replacement technique and building on the algorithm in [7], we obtain an algorithm for triconnected components. A new linear-time sequential algorithm, an alternative to the algorithm of Hopcroft and Tarjan, for finding triconnected components can be easily extracted from our paper. We remark that a different presentation of the results of this paper is available in [21].

2. Preliminaries . Let $V(G)$ and $E(G)$ stand, respectively, for the vertex set and the edge set of a graph G. Assume $|V(G)| = n$ and $|E(G)| = m$. We denote an edge between x and y as (x, y) or simply xy. A connected graph G is *k-vertex connected* if $|V(G)| > k$ and at least k vertices must be removed to disconnect the graph. A *biconnected graph* (or a *block*) is a 2-vertex connected graph. A pair of vertices $\{x, y\}$ of a biconnected graph is a *separating pair* if the number of components of the subgraph induced by $V(G) - \{x, y\}$ is more than one. An *ear decomposition starting with a vertex* P_0 of an undirected graph G is a partition of $E(G)$ into an ordered collection of edge disjoint simple paths $P_0, P_1, ..., P_k$ such that P_1 is a simple cycle starting and ending at

P_0, and for P_i, $1 \leq i \leq k$, each end point is contained in some P_j for some $j < i$, and no internal vertex of P_i is contained in any $P_j, j < i$. Each of these paths P_i is an *ear*. P_0 is called the *root* of the decomposition and is referred to as r. If the two end vertices of a path P_i do not coincide, then P_i is an *open ear*. In an *open ear decomposition* every ear P_i, $1 < i \leq k$, is open.

THEOREM 2.1. ([Whitney]) [27] *A graph has an open ear decomposition iff it is biconnected.*

From the above theorem, we can conclude that the subgraph induced by the vertices of the ears of $P_1, P_2, ..., P_i$, for all i, $1 \leq i \leq k$, is biconnected.

An ear is a *nontrivial ear* if it consists of more than one edge; otherwise it is a *trivial ear*. For an ear P, and two vertices x and y of P, $P[x, y]$ (resp. $P(x, y)$) denotes the segment of P that is between x and y, inclusive (resp. exclusive) of x and y. The segments $P(x, y]$ and $P[x, y)$ of P are defined similarly. A vertex is *internal* to an ear if it is not one of the end vertices of that ear. For two vertices v and w on P, $P - P[v, w]$ refers to the segment(s) of P formed by $V(P) - V(P[v, w])$.

As an ear decomposition is a partition on the edge set of a graph, each edge (v, w) belongs to an unique ear (denoted by $ear(vw)$). Notice that, except for the root, each v is internal to exactly one ear; call it $ear(v)$. Refer to Fig. 2 for an example of a biconnected graph and an open ear decomposition for it. The following definition labels each vertex v depending on the *position* of v on $ear(v)$.

Starting with an arbitrary end vertex p of P, define the position of p on P, $pos(p, P)$, to be zero. For every vertex v of P, $v \neq r$, the position of v on P, $pos(v, P)$ is equal to the number of edges between p and v on P. When a vertex v is an internal vertex of P, we omit the second argument and write simply $pos(v)$. The value of $pos(x, P)$, for $x \notin V(P)$ is undefined. Some example pos values for the graph of Fig. 2 are $pos(g) = pos(g, 2) = 3$, $pos(d, 6) = 0$, $pos(r, 1) = 0$, $pos(4, e) = 4$ and $pos(1, j)$ is undefined. For a pair of vertices u, v of P, u is to the *left* (resp. *right*) of v if $pos(u, P) < pos(v, P)$ (resp. $pos(u, P) > pos(v, P)$). The vertex of P that has no vertices of P to its left (resp. right) is called the *left end point of P* (resp. *right end point of P*).

For the sake of completeness, we include the definition of bridges. Let $G = (V, E)$ be a biconnected graph, and let Q be a subgraph of G. We define the *bridges of Q in G* as follows (see, e.g., [5], p. 148): 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* $B = (V_B, P_B)$ 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 P_B containing the edges of G incident on a vertex in K. The vertices of Q that are connected by an edge to a vertex in K are called the *attachments* of B and these edges are called the *attachment edges* 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. The trivial and nontrivial bridges together constitute the bridges of Q.
Remark : Throughout the paper, we will only address how to detect pairs where $r \notin \{x, y\}$. The pairs in which one of the vertices is r can be detected as a special case by finding the articulation points of the graph induced by $V(G) - \{r\}$ within the claimed resource bounds.

LEMMA 2.2. *If $\{x, y\}$ is a separating pair of a graph G, then there exists a nontrivial ear P in any open ear decomposition of G such that $\{x, y\}$ is a pair of non-adjacent vertices on P.*

Proof. Let C be a connected component induced by $V(G) - \{x, y\}$ such that $r \notin V(C)$. Then, P is the minimum of $\{ear(v) \mid v \in V(C)\}$. ☐

DEFINITION 1. An ear P *is separated by* $\{x, y\}$, if x and y are non-adjacent vertices of P, and $P(x, y)$ is separated from the lower-numbered ears in $G - \{x, y\}$.

Notice that the number of ears separated by a pair of vertices $\{x, y\}$ is one less than the number of connected components of $G - \{x, y\}$. The location of separating pairs on an ear P can be stated precisely in terms of the attachments of the bridges of P.

THEOREM 2.3. *$\{x, y\}$ is a separating pair that separates ear P iff (i) there exists a nontrivial ear P containing x and y as non-adjacent vertices, (ii) the bridge B_r of P in G that contains r has no attachments on $P(x, y)$, and (iii) for all other bridges B of P in G, if B has an attachment on $P(x, y)$, then all attachments of B are on $P[x, y]$.*

Proof. ($\Longrightarrow$) Let P be an ear separated by $\{x, y\}$. Assume, for contradiction, that the forward implication is not true. From Definition 1, we know that x and y are not adjacent on P. For P, either B_r has an attachment $P(x, y)$ or there is a B with one attachment on $P(x, y)$ and one on $P - P[x, y]$. In either case, there is a path from one of the vertices of $P(x, y)$ to r in $G - \{x, y\}$ which contradicts the assumption that P is separated by $\{x, y\}$.
($\Longleftarrow$) There must be at least one vertex v of P between x and y as they are not adjacent on P. For all such v, there cannot be a path in $G - \{x, y\}$ from v to r as that would imply a bridge of P in G with at least two attachments—one on $P(x, y)$ and the other on $P - P[x, y]$. Therefore, the segment $P(x, y)$ is disconnected from the components containing lower-numbered ears when x and y are deleted from G. ☐

The theorem given above assures that each ear P together with the bridges of P are sufficient for extracting separating pairs and that each ear with its bridges can be considered in isolation. But if we were to consider each ear and all its bridges in their entirety, the number of edges involved could be far more than $O(m)$. However, notice that for an ear P, the information about its bridges that is of relevance in finding separating pairs is contained only in those edges of the bridges which are incident on P. We can succinctly encode the information about separating pairs by building a collection of simple graphs as shown in the following.

DEFINITION 2. The collection $\mathcal{H}$ consists of simple graphs one for each nontrivial ear of G. The graph $H_P \in \mathcal{H}$ for the ear P is as follows. Suppose that x and y are the end points of P and that $pos(x) < pos(y)$. Each H_P is such that $V(H_P) = V(P) \cup \{r_P\}$ and $E(H_P)$ is as follows. (i) For each vertex $v \in V(P)$, an edge (v, r_P) is added, if the bridge of P that contains the root r has an attachment at v. Otherwise, (ii) at most two edges are added to $E(H_P)$ by considering the bridges (possibly trivial) of P which have an attachment at v. Let a be the leftmost attachment of one such bridge where a is further to the left than the leftmost attachment for any other bridge that has an attachment at v. The edge (v, a) is added, if a belongs to $P[x, v)$. Similarly, an edge (v, b) is added where b the rightmost vertex that can be reached from v through a bridge of P. (See Fig. 2.)

An example of an ear P with its bridges and its corresponding H_P is illustrated in Fig. 2. Suppose that v and w are on P with $pos(v) < pos(w)$. From the definition of H_P, it follows that there is a bridge of P in G with one attachment on $P(v, w)$ and another on $P - P[v, w]$ iff there is a bridge of P in H_P with one attachment on $P(v, w)$ and another on $P - P[v, w]$. Hence, by Theorem 2.3,

COROLLARY 2.4. *$\{v, w\}$ is separating pair that separates P iff $\{v, w\}$ is a separating pair in H_P.*

3. An Algorithm for Separating Pairs. Lemma 2.2 tells us that, in our search for separating pairs, we do not have to consider those vertex pairs for which there is no single ear containing them. If we are further assured some how that *every* separating pair belongs to the *internal* vertices of *some* ear, then we can efficiently reduce the problem of finding separating pairs to that of finding biconnected components. The idea is to build a multigraph by collapsing the internal vertices of every nontrivial ear to a single vertex. Observe that if $\{x, y\}$ is a separating pair that is internal to P and $P(x, y)$ contains a vertex of degree greater than two, then the vertex in the multigraph that is obtained by shrinking the internal vertices of P is an articulation point. We elaborate more on this reduction in Section 3.2. But before we proceed to this reduction, we need to address a more serious difficulty: For a separating pair $\{x, y\}$, there *need not* be any ear in an open ear decomposition, for which x and y are internal. Section 3.1 shows how to circumvent this problem. In Section 3.2 we show how to find a collection of graphs similar to $\mathcal{H}$ that succinctly encode separating pairs. Finally, in Section 3.3 we address the extraction and the output representation of separating pairs.

3.1. Making Separating Pairs Internal to an Ear. In this subsection we show how to build a graph G', the *local replacement graph.* This graph is such that for every separating pair $\{x, y\}$ that separates P in G, there is a corresponding separating pair $\{x_P, y_P\}$ and an ear P' in G' such that x_P and y_P are internal vertices of P'. Furthermore, for every separating pair $\{x_P, y_Q\}$ of G', $\{x, y\}$ is a separating pair in G. Roughly, the strategy is to divide the graph G into a set of paths by splitting at the end vertices of the nontrivial ears of an open ear decomposition. This would result in making several copies of a vertex, i.e. one copy for each nontrivial ear that contains it. Next, we add new edges to connect up the different copies of a vertex. The main difficulty is in figuring out an efficient way to connect the split vertices without jeopardizing the primary goal of preserving the overall structure of separating pairs.

DEFINITION 3. (i) Define $\vec{G}$ to be a directed acyclic version of G which is obtained by the following construction. Suppose (a, b) is one of the end edges of P_1, and that (b, c) is next to (a, b) on P_1. Then, give a direction to (a, b). Orient the rest of P_1 in the opposite direction. Now, $\vec{G}$ is obtained by giving directions to the remaining ears of G so that the resulting digraph is acyclic.
(ii) Define $\vec{T}$ to be directed spanning tree rooted at a obtained by removing the last edge in each directed ear, P_i, $i > 1$, in $\vec{G}$, and deleting (b, c) in the case of P_1.
(iii) Define $\overleftarrow{T}$ to be the directed spanning tree rooted at b obtained as shown below. Let $\overleftarrow{G}$ be the graph resulting form reversing the directions of the edges of $\vec{G}$. Then, $\overleftarrow{T}$ is obtained by deleting the last edge of each ear P_i, $i > 1$, in $\overleftarrow{G}$, and deleting the end edge of P_1 other than (a, b), in the case of P_1.

Fig. 3.1 illustrates the above definitions. One efficient way to obtain $\vec{G}$, for a given G, is by st-numbering the vertices of G where (s, t) is an edge of P_1 with $s = r$, and directing each edge of an ear from the lower-numbered vertex to the higher-numbered. As each end point of every nontrivial ear belongs to a lower-numbered ear, we have

PROPOSITION 3.1. *The ear numbers of the edges of the tree path from any vertex v to the root r in $\vec{T}$ decrease monotonically.*

DEFINITION 4. P_{xy} refers to an ear P with $\{x, y\}$ as the end points. If P is directed from x to y in $\vec{G}$, then we denote it as $P_{\vec{xy}}$.
The following useful lemma relates the attachments of a bridge and $\vec{G}$.

LEMMA 3.2. *Let B be a bridge of $P_{\vec{xy}}$ that contains no edge with an ear number less than P. Then, if B has an attachment edge (z, y) (resp. (x, z)) at y (resp. x), then zy (resp. xz) is directed from z to y (resp. x to z) in $\vec{G}$, i.e. $xz \notin E(\vec{T})$.*

Proof. We will only prove the case when B has an attachment at y. A similar proof can be derived when B has an attachment at x. Fig. 3.1 illustrates the statement of the lemma. Let $ear(z) = Q$ where the edges of Q belong to $E(B)$. Suppose the lemma does not hold, i.e. Q is directed such that the edge zy is directed from y to z. Trace a path $\vec{G}$ as shown in the following. Start from y and traverse until the last vertex of Q, say w, is encountered. Suppose $ear(w) = S$, for some S that belongs to B. In an ear decomposition, as an end point of each ear belongs to a lower-numbered ear, we have $S < Q$. Now, trace the edges of the ear S along the direction given to S until the last vertex of S is reached. This process, when continued in this fashion, uses edges of B with monotonically decreasing ear labels. Therefore, we must eventually encounter a vertex of P. From that vertex of P, we can reach y by going along the direction given to P. In other words, if the lemma does not hold, we can trace a cycle starting and ending at y in $\vec{G}$ which contradicts the fact that $\vec{G}$ is acyclic. □

Using these directed graphs, we show how to build the local replacement graph in the following.

Algorithm *Build* G'

Input: A graph G, an open ear decomposition of G, and the directed graph $\vec{G}$ and its associated spanning trees $\vec{T}$, $\overleftarrow{T}$.

Output: A graph G' (local replacement graph) in which each separating pair of G is internal to some ear of G'.

1. *Construction of* $V(G')$
   $V(G') = \{v_P \mid v \in V(P)$ for some nontrivial ear $P\}$. Refer to v_P as a copy of v.
2. *Construction of* $E(G')$
   (a) Initialize $E(G')$ to $\{(u_P, v_P) \mid (u, v) \in E(P)\}$.
   (b) For every nontrivial ear $P_{\vec{vw}}$, add an edge as follows. If the least-common-ancestor of v and w (henceforth denoted as $lca(v, w)$) in $\vec{T}$ is v, then let Q be the ear number of the first tree edge in the path from v to w in $\vec{T}$. Then, add (v_P, v_Q) to $E(G')$. If $lca(u, w) \neq v$ in $\vec{T}$, then add (v_P, v_Q) to $E(G')$ where Q is such that v is internal to Q.
   (c) Repeat Step 2b with $\vec{T}$ and $\vec{G}$ replaced by $\overleftarrow{T}$ and $\overleftarrow{G}$, respectively.
   (d) For all trivial ears uv, if u and v are internal to ears P and Q, respectively, then add (u_P, v_Q) to $E(G')$.

Fig. 3.1 shows various stages of Step 2 on an example graph. The above algorithm does not quite suffice for our purposes, i.e. it need not be the case that there is a separating pair $\{x, y\}$ in G that separates an ear P iff $\{x_P, y_P\}$ is a separating pair in G'. This happens if the input graph G contains two or more ears with the same end points. Consider an example. Refer to the graph G of Fig. 3.1(i) and consider the subgraph $G - \{h\}$ obtained by deleting h and edges incident on h. In $G - \{h\}$, the separating pair $\{b, d\}$ separates the ear 2, but not 1. In the local replacement graph corresponding to $G - \{h\}$, i.e. in $G' - \{h_4, b_4, d_4\}$ of Fig. 3.1, the pair $\{b_2, d_2\}$ is not a separating pair. Before we give the final step of *Build* G', we need the following definitions.

DEFINITION 5. (i) An ear is a *parallel* ear if there is another ear with the same end points.

(ii) Consider the following partition on a set of parallel ears with $\{x, y\}$ as the end points. For each connected component C of $G - \{x, y\}$, denote the minimum of $\{ear(v) \mid v \in V(C)\}$ by P. Each such P is in a different partition. Additionally, if the end points of P are x and y, then the partition containing P contains exactly (no more or no less) those

ears parallel to P whose internal vertices belong to C. Remaining ears with $\{x, y\}$ as the end points, can be put in other (or additional) partitions arbitrarily. Each partition is called a *bundle* of parallel ears.
(iii) The ear with the smallest label of a bundle is called the *representative* for that bundle.
In the example graph of Fig. 3.1, ears 2, 3 are in one bundle and 4 is in a bundle by itself. Now, we describe a method to handle parallel ears.

Algorithmically, partitioning a collection of ears into sets of parallel ears is easy, as it is just grouping ears according to the end points. However, to further classify them into bundles is nontrivial. In the following, we give an alternate definition for a bundle of parallel ears that is equivalent to Definition 5(ii).

DEFINITION 6. Define, recursively, when an ear Q *depends* on an ear P as follows. (a) P *depends* on P. (b) An ear Q depends on P, if Q is not parallel to P and for each of the end points v of Q, there exists an ear R that depends on P such that $v \in V(R)$.

DEFINITION 7. Define, recursively, a *bundle* of parallel ears as follows. Two parallel ears P and Q are in the same bundle if one of the following holds. (a) There is a path from an internal vertex of P to an internal vertex of Q such that for every edge uv of that path, if $ear(uv) = R_{ab}$, then there exists ears that depend on either P or Q which contain $ear(a)$ and $ear(b)$. (b) There is another parallel ear R such that P, R and R, Q are in the same bundle.

Suppose that $\{x, y\}$ separates a parallel ear P and that the component C of $G - \{x, y\}$ containing $P(x, y)$ also contains the internal vertices of Q_{xy}. Then, notice that for all $v \in V(C)$, $ear(v)$ depends on P or an ear parallel to P (such as Q) whose internal vertices belongs to C. Therefore, it follows that the above definition is equivalent to Definition 5(ii).

In the following, the first step shows how to partition parallel ears into bundles efficiently. Informally, it is as follows. First, we build an auxiliary graph G_p based on the parallel ears of G. G_p is such that P and Q belong to the same bundle iff the corresponding pair of vertices p and q are in the same connected component in G_p. The graph G_p is built by making use of the local replacement graph that is available after Step 2b. Observe that after Step 2b, each ear is hooked only at one of its ends. Therefore, the partial G' after Step 2b is a tree. Denote it by $\vec{T_l}$. Now, we describe the method.

Algorithm *Build G'* (continued)

3. *Adjust $E(G')$ for parallel ears*
   (a) *Identify bundles*
   Let G_p be an auxiliary graph whose vertices p correspond to the parallel ears P of G. The edges of G_p are added by making use of the tree $\vec{T_l}$. Consider two parallel ears $P_{\vec{xy}}$ and $Q_{\vec{xy}}$. An edge $(p, q) \in E(G_p)$ iff there exists an ear $R_{\vec{ab}}$ (possibly, a trivial ear) that satisfies the following two

properties. (i) $\{a,b\} \cap \{x,y\} = \emptyset$. (ii) Let a and b are internal to the ears U and W, respectively. Then, the tree paths in $\vec{T_l}$ from the $lca(a_U, b_W)$ to a_U and b_W start with the edges added by Step 2b for P and Q.

(b) *Adjust $E(G')$ for parallel ears*
For each connected component C of G_p do the following. Find a spanning tree T and root it at the vertex with the smallest label (say p). Let P be the ear corresponding to p. For each ear $Q_{\vec{vw}}$, $Q \neq P$, such that q belongs to T: (i) delete the end edges of Q added to $E(G')$ in Step 2, and (ii) add (v_S, v_Q) and (w_S, w_Q) to $E(G')$ where s is the parent of q in T.

At the end of Step 3 of *Build* G', we still have a partition of the edge set into disjoint paths. However, because of the rearrangement of edges in Step 3b, the end points of an ear may not lie on a lower-numbered ear if we continue to use the old ear labels for G'. But this deficiency is inconsequential to the rest of our algorithm and hence we continue to use old ear labels. Notationally, if P is an ear of G, the path in G' consisting P together with two new end edges created by local modifications will be referred to as P'.

The following two propositions can be proved by induction and Definition 6 and Definition 7.

PROPOSITION 3.3. *If an ear Q depends on $P_{\vec{xy}}$, then both end points of Q' belong to the subtree of $\vec{T_l}$ rooted at x_P.*

PROPOSITION 3.4. *If an ear Q depends on P, then there is a path from any vertex of Q to an internal vertex of P such that if uv belongs to this path, then $ear(uv)$ depends on P.*

THEOREM 3.5. *Two parallel ears $P_{\vec{xy}}$ and Q are in the same bundle iff p and q are in the same connected component in G_p.*

Proof. In what follows, we use the notation Step 3. Since, P and Q are parallel, Q is directed from x to y in $\vec{G}$. Furthermore, from the construction in Step 2 it follows that they have the same parent in $\vec{T_l}$; Call it x_S. Denote the subtrees of $\vec{T_l}$ rooted at x_P and at x_Q by T_p and T_q, respectively. Assume w.l.o.g. that a_U is in T_p.
($\Longleftarrow$) Assume $(p,q) \in E(G_p)$. If an ear R satisfies the condition (ii) of Step 3a, then by Proposition 3.3, R depends neither on P nor on Q. Conversely, if R is the ear with smallest label that depends on neither P nor Q, then, by Definition 6, R cannot have both its ends points in one of T_p, T_q. Hence, R satisfies the conditions of Step 3a. Assume w.l.o.g. that R is the ear with the smallest label that satisfies the conditions of Step 3a. As U and W contain the end vertices of R, $U < R$ and $V < R$. Since R is the ear with smallest label that satisfies the conditions of Step 3a, U depends on P and W depends on Q. Then, by condition (i) of Step 3a and Proposition 3.4, there is a path from an internal vertex of P to an internal vertex of Q which satisfies the condition (a) of Definition 7.

($\Longrightarrow$) Assume there are no ears parallel to P except Q in the bundle containing P. As the condition (b) of Definition 7 is transitive, it suffices to prove just this case. Suppose that $P < Q$. We are required to show that there exists an ear R that satisfies the conditions of Step 3. Consider the subgraph D of G formed by P, Q, and all those ears of G that depend on either P or Q. It can be proved by induction that P and Q are in different bridges of $\{x, y\}$ in D, i.e. there is no path from an internal vertex of P to that of Q. Denote the bridge of $\{x, y\}$ in D containing P and Q by D_p and D_q, respectively. If P and Q are in the same bundle, then, by Definition 6 and the fact that there are no other parallel ears in D, there must exist an ear R that connects a non-attachment vertex of D_p, say a, to a non-attachment vertex of D_q, say b. As a and b are non-attachment vertices of D_p and D_q, neither of them is from $\{x, y\}$. Therefore, R satisfies condition (i) of Step 3a. Since a and b are internal to ears that depend on P and Q, respectively, the end vertices of R' belong to T_p and T_q by Proposition 3.3. That is, R satisfies condition (ii) of Step 3a. □

The reverse direction of the above theorem can be stated equivalently as follows.

COROLLARY 3.6. *If $(p, q) \in G_p$, then there is a path from an internal vertex of P to an internal vertex of Q such that this path does not use any vertices of P or of ears parallel to P.*

Next, we prove the correctness of *Build G'*. The following simple facts are useful in the proofs. Observe that the number of copies of a vertex v in G' is one more than the number of nontrivial ears for which v is an end point, i.e. it is $1 + (\text{degree}(v) - 2)$. The algorithm *Build G'* adds an edge only between two copies of the same vertex and it adds one edge for every end point of every nontrivial ear. See Fig. **??**. Therefore,

PROPOSITION 3.7. *(i) The edges that connect different copies of a single vertex v form a tree. Therefore, there is a path between any two copies of v in G' that uses only other copies v. As a consequence, for a vertex $v \notin V(P)$, all copies of v belong to a single bridge of P' in G'. (ii) The graph G can be obtained from G' by collapsing, for each v, all copies of v into one.*

DEFINITION 8. For an ear P, we say a bridge of P is a *relevant* bridge if for each ear Q that belongs to it (a) $Q > P$, and (b) if Q parallel to P, then Q is in the same bundle as P. If a bridge is not relevant, then it is said to be *irrelevant*.

Observe that for a set of parallel ears with x and y as the end vertices, if one of the ears in $\vec{G}$ is directed from x to y, then all of them are directed from x to y. After Step 2 (but before Step 3), if P and Q are parallel in G, then P' and Q' are parallel in G' by the construction of Step 2. As Step 3 does not change the end edges created in Step 2 for the representatives, we have

PROPOSITION 3.8. *If P and Q are parallel ears such that each is a representative of its bundle, then P' and Q' have the same end points in G'.*

For a subgraph D of G, let $\mathcal{E}(D) = \{ear(uv) \mid uv \in E(D)\}$. The following can be derived from Definition 8 and the definition of open ear decomposition.

Proposition 3.9. *Suppose that B is a relevant bridge of P. If B contains an edge of an ear Q, then B contains all edges of Q,i.e. if $Q \in \mathcal{E}(B)$, then all edges of Q belong to B.*
Another useful fact can be derived from the definition of open ear decomposition by using the minimum of $\mathcal{E}(B)$.

Proposition 3.10. *If B is a bridge of an ear P that contains an edge whose ear label is less than P, then B has attachments at the end vertices of P.*

Lemma 3.11. *Consider a relevant bridge B of $P_{\vec{xy}}$ (resp. $P_{\overleftarrow{xy}}$) in G. Suppose it contains an ear Q with one end point at x (resp. y) and the other at $v \notin V(P)$. Then, the $lca(x,v)$ (resp. $lca(y,v)$) in $\vec{T}$ (resp. $\vec{T}$) is x (resp. y).*

Proof. We will prove the lemma when Q has an attachment at y. A similar proof works when Q has an attachment at y. We will show that x is an ancestor of v in $\vec{T}$. Observe that all paths from v to r in G must go through a vertex of P because v belongs to a bridge of P and this bridge does not contain r. Specifically, the tree path from v to r must have a vertex, say w, of P. This vertex w of P cannot be y, because, by Lemma 3.2, y is not reachable in $\vec{T}$ from v. Hence, the tree path from v to r must encounter a vertex w from $P[x,y)$. Now, the last vertex of P on this tree path is x because of the direction given to P. Therefore, the tree path from v to r in $\vec{T}$ contains x. In other words, x is an ancestor of v in $\vec{T}$. □

Lemma 3.12. *Consider a relevant bridge B of $P_{\vec{xy}}$ in G. Suppose it contains an ear Q with one end point at $u \in V(P)$ and the other at $v \notin V(P)$. If $lca(u,v) = u$, then the edges on the tree path from v to u in $\vec{T}$ belong to either P or to those ears of B whose label is less than Q.*

Proof. Assume $ear(v)$ is U for some $U \in \mathcal{E}(B)$. By the definition of a relevant bridge, $U > P$. We traverse the tree path from v to u. As argued in the proof for the previous lemma, we must encounter a vertex z of P from $P[x,y)$ in this traversal. Now, u is an ancestor of v because $lca(u,v) = u$. Therefore, z is a descendant of u. Finish the traversal of the tree path from z to u by taking the edges of P from z to u. In this traversal, initially U belongs to a relevant bridge, namely B. Also, whenever the ear labels change from, say, P_1 to P_2, for $P_1, P_2 \neq P$, the switch occurs at the end vertex of P_1. This end vertex is a non-attachment vertex of B, because otherwise we would have moved from P_1 to P instead of to P_2. Hence, P_1 and P_2 are in the same bridge of P. Since U belongs to P, P_1 and, hence, P_2 belongs to B. In summary, we showed that the traversal of the tree path from v to u uses the edges of the ears of B and possibly some of P. □

Theorem 3.13. *Let B_q be the bridge of $P_{\vec{xy}}$ in G containing the ear Q. Let BL_q be the bridge of P' in G' containing the ear Q'. Then, (i) for every ear S, if $S \in \mathcal{E}(B_q)$, then $S' \in \mathcal{E}(BL_q)$. Additionally, (ii) if B_q is relevant bridge, then for every S', if $S' \in \mathcal{E}(BL_q)$, then $S \in \mathcal{E}(B_q)$.*

Proof. If two ears Q and S are in the same bridge B of P in G, then there must be a path between a vertex of Q and a vertex of S consisting of only the non-attachment vertices of B. Then, by Proposition 3.7, it follows that there is a single bridge of P' in G' containing Q' and S'. That proves part (i).

Next, we will prove using induction that if two ears belong to two different bridges B_1 and B_2 in G, then they belong to two different bridges of P' in G' provided at least one of B_1, B_2 is a relevant bridge. For the base of the induction, we start with the subgraph D_1 of G' formed by P' plus $\{Q' \mid$ either $Q < P$, or Q is the minimum labeled ear of a bridge of P, or Q is parallel to P and Q is the representative of its bundle$\}$. The ith induction step consists of building D_i from D_{i-1} by adding the smallest ear from $\mathcal{E}(G) - \mathcal{E}(D_{i-1})$. This ear is added in a manner that conforms with the construction rules of Step 2 and Step 3, and thus we maintain the invariant that D_i is a subgraph of G' at all times.

To show that D_1 satisfies the base case, we claim that each ear of a relevant bridge of P in G is in a bridge (of P') by itself in D_1. First, observe these ears have their end points on P in G. In *Build* G', the end edges for the minimum labeled ears of a relevant bridge of P are decided in Step 2 and these are not changed later in Step 3. To prove the claim, it suffices to show that each of these ears is attached in Step 2 to some vertices of P'. All parallel ears are attached to the end vertices of P' by Proposition 3.8. For any other ear $Q_{\vec{vw}}$, $v, w \in V(P)$, if $w = y$, then $v \neq x$ because we assumed that Q is not parallel to P. In that case, $lca(v, w) \neq v$ in $\vec{T}$ (because the end edge of P incident on y is not present in $\vec{T}$) and v_Q is attached to v_P. The other end point w is attached to w_P, because $lca(w, v) = w = y$ in $\overleftarrow{T}$ and the tree path from w to v starts with an edge of P. The cases arising from the other positions for v and w on P can be analyzed similarly to conclude that v_Q and w_Q are attached to v_P and w_P respectively.

Now, we prove the induction step. Assume, inductively, the theorem holds when the smallest $(i-1)$ ears from $\mathcal{E}(G) - \mathcal{E}(D_1)$ are added. Consider the ith smallest ear $Q_{\vec{uv}}$.

Suppose there is an ear (possibly P) parallel to $Q_{\vec{uv}}$. Then, from the construction of Step 3b we notice that Q is attached to the copies of u and v say u_S and v_S where s is (in the notation of Step 3b) the parent of q in T. The case when $S = P$ results in creating a new bridge when Q' is added to D_{i-1} and the induction step is trivially true. But if $s \neq p$ and $(s, q) \in E(G_p)$, then by Corollary 3.6 there is a path between an internal vertex of Q to an internal vertex of S that does not use any vertices of ears parallel to Q. Specifically, this path does not use any vertices of P. Hence S and Q are in the same bridge of P in G.

Next, assume $Q_{\vec{uv}}$ has no parallel ears. Let $ear(u)$ and $ear(v)$ be S and U, respectively. We will do a case analysis depending on the position of u and v.

Assume neither u nor v belong P. Then, by Proposition 3.7, Q' is connected to a non-attachment vertex of the bridge of P' (in D_{i-1}) containing S' (call it BL_s) to a non-attachment vertex of the bridge containing U' (call it BL_u). Clearly S and U (and hence the ears of G corresponding to the ears of BL_s and BL_u) are in one bridge of P in G because of the path (namely Q) between them that uses no vertices of P.

Now, consider the case when both u and v belong to P, i.e. $P = S = U$. Then, it cannot be that $u = x$ and $v = y$ as that would make Q have an ear (namely P) parallel to it. Hence one of the end vertices of Q must be internal to P. In this case, by a proof similar to the one used for the base case, it can be argued that Q' is attached to two of the vertices of P'. In other words, if both u and v belong to $V(P)$, then a new bridge gets created and the induction step holds.

Next, assume that one of u, v belongs to P and the other does not. Assume w.l.o.g. u belongs P. Clearly, U and Q are in a single bridge of P in G because they are connected at v. Therefore, if Q' attaches itself to the bridge of P' in D_{i-1} that contains U' (denote this bridge by BL_u), then the induction step holds because U (and all the ears corresponding to the ears of BL_u) and Q are in a single bridge of P in G. We will show that this is indeed the case, i.e. Q' attaches itself to BL_u. Notice that, by Proposition 3.7, as $v \notin V(P)$, there exists a bridge of P' in D_{i-1} that contains all copies of v. Therefore, v_Q is connected to a copy of v that belongs to BL_u. It remains to be shown that v_Q is not connected to a non-attachment vertex of a bridge other than BL_u. Consider the $lca(u, v)$ in $\vec{T}$. If $lca(u, v) \neq u$, then u_Q is connected to u_P by Step 2. Otherwise, by Lemma 3.12, the tree path in $\vec{T}$ from v to u consists of edges of either P or of the ears from $\mathcal{E}(BL_u)$. Therefore, u_Q is connected to u_P or u_W where $W \in \mathcal{E}(BL_u)$. □

3.2. A Reduction to Biconnected Components. We briefly alluded to reducing triconnectivity to biconnectivity at the beginning of Section 3. We elaborate more on this reduction here. In this subsection, we show how to find a collection similar to $\mathcal{H}$ that encodes separating pairs. There are two reasons for finding a collection that is only similar to $\mathcal{H}$ and not $\mathcal{H}$ itself as defined before. The first is that the new collection suffices for our purposes, and the second reason is that it can be computed by an easy reduction to any biconnected component algorithm.

Recall the definition of $\mathcal{H}$ from Definition 2. It consists of a graph H_P for each nontrivial ear $P_{\vec{xy}}$. We slightly change the definition of H_P as it is difficult to efficiently check, for $v \in V(P)$, if the bridge of P containing the root r is adjacent to v.

Definition 9. The graph H_P is defined as in Definition 2 with the following modification to part (i) of that definition. We say that (v, r_P) is added to $E(H_P)$ if v is adjacent to w and w belongs to an irrelevant bridge of P.

Even though the resulting collection is slightly different, we will continue to denote it by $\mathcal{H}$. Let us examine and see whether this new definition of graph H_P also encodes the set of separating pairs of G. If the bridge containing w also contains the root r, then the new rule results in the same H_P. Otherwise, the bridge containing w must necessarily have attachments at x and y by Proposition 3.10. Therefore, we would have added (v, x) and (v, y) to $E(H_P)$ instead of (v, r_P). A pair of vertices of P that is not a separating pair before would not be a separating pair now. The converse also holds except for the pair $\{x, y\}$. In this case, notice that we would detect $\{x, y\}$ as a separating pair on H_Q where $\{x, y\}$ separates Q. This is because, by Definition 1, all vertices of $Q(x, y)$ are adjacent to vertices of bridges of Q relevant to Q. Therefore, the edges added to $E(H_Q)$ for the vertices of $Q(x, y)$ are identical irrespective of whether the old rule is used or the new one.

Next, we show how to build $\mathcal{H}$ by a reduction to biconnected components. The biconnected component algorithm is run on a multigraph G_e (the subscript e to indicate that G_e is built from the ears of G') constructed from G' and its ears.

DEFINITION 10. Let P' be a nontrivial ear $\langle v_0, v_1, \ldots, v_k \rangle$ of length greater than two, i.e. $k \geq 3$. The graph G_e is obtained from G' by contracting all such ears P' by merging the internal vertices $v_1, v_2, ..., v_{k-1}$ into a single vertex p.

Fig. 3.2 shows the multigraph G_e for the local replacement graph and its ear decomposition shown Fig. **??**. Recall that the representative (say P) of a bundle of parallel ears is the ear of that bundle with the smallest label. Observe that the end points P belong to a lower-numbered ear that is not parallel to P. This observation together with the definition of open ear decomposition imply the following simple fact.

PROPOSITION 3.14. *If D is a biconnected component of G_e, then the vertex of D with the minimum label is either r or an articulation point of G_e.*

There is a correspondence between the relevant bridges of P_{xy} in G, the bridges of $P'[x_P, y_P]$ in G', and the connected components resulting from deleting p from G_e. The relation between the latter two is simple: From the definition of G_e, it follows that the ears Q and R are in the same bridge of $P'[x_P, y_P]$ in G' iff q and r are in the same component in $G_e - \{p\}$. The relation between the attachments of the bridges of P in G and that of the bridges of $P'[x_P, y_P]$ in G' is stated below.

THEOREM 3.15. *Suppose that B_q (resp. BL_q) is a bridge of $P_{\vec{xy}}$ (resp. P') in G (resp. G') containing Q (resp. Q').*
(i) B_q relevant to P iff BL_q has no attachments at the end vertices of P' in G'.
(ii) Assume that B_q is relevant to P. Then, B_q has an attachment at v iff BL_q has an attachment at v_P.

Proof. **(i)** Assume B_q is a relevant bridge for P. It can be proved that the attachments of BL_q are from $P'[x_P, y_P]$ by an inductive proof similar to the one used to show part (ii) of Theorem 3.13. The invariant that should be maintained at all times is that if Q belongs to a relevant bridge, then the bridge of P' containing Q' in D_i has all its attachments on $P'[x_P, y_P]$.

Assume B_q is not a relevant bridge. There are two kinds of irrelevant bridges. The ones that have an ear less than P and the ones that have an ear (say Q) parallel to P from a different bundle. If B_q is of the first kind, then BL_q contains an ear less than

P' by Theorem 3.13(i). If in addition BL_q contains a parallel ear to P from the bundle of P, then from the construction of Step 3b it follows that BL_q has attachments at the end vertices of P'. Otherwise, consider the subgraph of G' consisting of P', the ears of BL_q that have no parallel ears, plus the representatives of the bundles of ears that belong to BL_q. Clearly, this subgraph is biconnected and the labels on the ears define an open ear decomposition. Therefore, by Proposition 3.10, BL_q has attachments at the end vertices of P'. Next, assume that B_q is an irrelevant bridge because it contains an ear S parallel to P such that P and S are in different bundles. Assume w.l.o.g. that S is the representative of its bundle. By Proposition 3.8, the end points of S' and the end points of the representative of the bundle of P' are the same. Therefore, BL_q has attachments at the end vertices of P'.
(ii) Assume BL_q has an attachment at v_P and that this attachment belongs to an ear S' of BL_q. Since B_q is a relevant bridge, by part (ii) of Theorem 3.13, $S \in \mathcal{E}(B_q)$. By Proposition 3.7, $v \in V(S)$. Therefore, B_q has an attachment at v.

Consider the only if direction of the theorem. Let $S_{\vec{vw}}$ denote the ear from the bridge B_q with the least label that has an attachment at v. Then, if $lca(v,w) \neq v$ in $\vec{T}$, then, by Lemma 3.11, v is an internal vertex of P. Therefore, by Step 2c, v_S is attached to v_P. Otherwise, i.e. $lca(v,w) = v$, then, by Lemma 3.12, the tree path from u to v consists of the edges from either P or the ears of B_q with a label less than S. As there are no ears of B_q less than S incident v, the tree path must end with the edges of P. Hence, by Step 2, v_S is attached to the vertex v_P. □

The (ii)nd part of the above theorem implies the following.

COROLLARY 3.16. *$\{x, y\}$ is a separating pair that separates P in G iff $\{x_P, y_P\}$ separates P' in G'.*

Proof. Consider the bridges of $P[x,y]$ in G and that of $P'[x_P, y_P]$ in G'. □

We exploit this correspondence in building $\mathcal{H}$. The nontrivial part in building H_P is in identifying which edges to add, if any, for a vertex v of $P[x,y]$. This involves knowing the extreme attachments of the relevant bridges of P adjacent to v. If there are any bridges of P that are relevant, then p would be an articulation point in G_e; and each block D of G_e attached to p that does not contain r' corresponds to a relevant bridge of P. Therefore, we give a common label, called α label, to all vertices (except p) of each such block D of G_e. This α label is a 3-tuple $\langle P, a, b \rangle$: a and b are the *pos* labels of the extreme attachments of the bridge of P that corresponds to D and the first tuple reflects the fact that these attachments are on P. We will denote the first, second and the third tuple of $\alpha(q)$ by $\alpha(q).1$, $\alpha(q).2$ and $\alpha(q).3$, respectively.

It turns out that the α labeling can be computed efficiently by a slight modification of any biconnected component algorithm as shown below. Given α labeling, the edges that need to be added to build H_P can be figured out quite easily.

Algorithm *Build $\mathcal{H}$*
Input: A graph G, an open ear decomposition of G, *pos* labeling for each ear, and the

local replacement graph G' for that decomposition.
Output: A collection of graphs $\mathcal{H}$ (built on the top of each nontrivial ear of G) that encode separating pairs succinctly.

1. Build a multigraph G_e from G' by merging the internal vertices of each nontrivial ear P' into a single vertex p. Discard the self loops from G_e.
2. Find the biconnected components $D_1, D_2, ..., D_k$ of G_e.
3. Label the vertices of G_e with 3-tuples:
   For each D_i do the following. Let p be the vertex of D_i with the smallest label. For each $q \in V(D_i) - \{p\}$, set the 1st component of $\alpha(q)$ to P. Of all edges ps of D_i incident on p, consider those that have an image, say cd, in G. Assume $c \in V(P)$. (See Fig. 3.2.) Let a and b the minimum and the maximum, respectively, of the *pos* labels of all such c. Then, a and b are the second and third components of $\alpha(q)$.
4. Build H_P for each nontrivial ear P using the α labels:
   (a) Assign $V(H_P) = V(P) \cup \{r_P\}$.
   (b) Initialize $E(H_P) = E(P)$. For every vertex $v \in V(P)$, add the following edges. For an edge $vw \in E(G)$, $w \notin V(P)$, denote $ear(w)$ by Q.
      (i) If $\alpha(q).1$ is less than P, then add (v, r_P) to $E(H_P)$. Otherwise, add two edges as shown in the next step.
      (ii) Let a be such that $pos(a) = \min\{\{\alpha(q).2 \mid ear(w) = q \text{ and } vw \in E(G),\ w \notin V(P)\} \cup \{pos(w) \mid vw \in E(G) \text{ and } v, w \in V(P)\}\}$. Add (v, a) to $E(H_P)$. Also, add (v, b) where b is obtained similarly by using max and $\alpha(q).3$.

LEMMA 3.17. *Consider an ear P, and an edge $vw \in E(G)$, $w \notin V(P)$, where v is an internal vertex of P. Let $ear(w)$ be Q. Then, if the bridge B_q of P containing Q contains no ear less than P, then $\alpha(q).1 = P$.*

Proof. If B_q contains ears parallel to P, then they would have to be in the same bundle with P because of the the edge vw. Therefore, B_q is a relevant bridge of P. Now, it follows from part (i) of Theorem 3.15, that the bridge containing Q' would have all its attachments on the internal vertices of P'. Then, p would be an articulation point that is on every path between q and the root vertex r in G_e. Therefore, $\alpha(q).1 \geq P$. But p and q are in the same block because the edge vw of G causes an edge between p and q in G_e. Hence, $\alpha(q).1 = P$. ☐

THEOREM 3.18. *Algorithm Build $\mathcal{H}$ computes $\mathcal{H}$ as defined in Definition 9.*

Proof. Consider an ear $P_{\vec{xy}}$ and an internal vertex v of P. We need to argue that the construction carried out in Step 4 is correct. Suppose there is an edge $vw \in E(G)$, $w \notin V(P)$. Denote $ear(w)$ by Q. Denote the bridge of P containing Q by B_q and the $\min\{\mathcal{E}(B_q)\}$ by N. Note that whether vw is a trivial ear or one of the end edges of Q, it causes an edge to be added between p and q in G_e. Therefore, there is a block that contains both p and q.

According to Definition 9, the edge (v, r_P) is added to $E(H_P)$ iff N is less than P. Consider the construction in Step 4. We claim that $N < P$ iff $\alpha(q).1$ is less than P. To prove the forward direction of the claim, we need to prove that the block D of G_e containing p and q contains an articulation point m such that $M < P$. By Proposition 3.14, it is sufficient to show that p is not the articulation point in D with the smallest label. Because $N < P$, the bridge B_q is irrelevant to P. Now, if p is an articulation point that appears on every path from q to the root r', then BL_q of P' containing Q' in G' has all its attachments on $P[x_P, y_P]$. This contradicts part (i) of Theorem 3.15. Hence, p cannot be the articulation point with the smallest label in D. Consider the reverse direction. Assume for contradiction that $\alpha(q).1$ less than P but $N \geq P$. Clearly, $N \neq P$ because N belongs to a bridge of P. But if $N > P$, then, from Lemma 3.17, $\alpha(q).1$ is exactly P and not less than P.

Next, consider the case when there is an edge $vw \in E(G)$, $w \notin V(P)$, but for all such edges $\alpha(q).1$ is not less than P. By the claim of the previous paragraph, if $\alpha(q).1$ is not less than P, then $N > P$. Therefore, by Lemma 3.17, $\alpha(q).1 = P$. That is, every bridge of P' with an attachment at v has all its attachments on $P'[x_P, y_P]$. Because of the edge vw, each such bridge B_q is a relevant bridge of P. Therefore, by part (ii) of Theorem 3.15, the attachments of BL_q on P' are identical to the attachments of B_q on P. From the definition of G_e, the attachments of B_q become the edges of G_e incident on p where these edges belong to the block containing q. The attachments of B_q that are closest to x and y are reflected in the second and third tuples of $\alpha(q)$, as computed in Step 3. Therefore, Step 4b(ii) computes the $E(H_P)$ as defined in part (ii) of Definition 2. $\square$

3.3. The Extraction and Output Representation of Separating Pairs. Observe that in $O(m + n)$ time we cannot possibly list all separating pairs of a graph as there could be $O(n^2)$ of them. For example, in a simple cycle every pair of non-adjacent vertices is a separating pair. That is, a cycle of n vertices has $n(n-3)/2$ separating pairs. Our output representation is a set of paths referred to as candidate lists such that a pair of vertices $\{u, v\}$ separates P iff u and v are non-adjacent on P and there is a candidate list generated from H_P that contains u and v. Such a representation of separating pairs is sufficient to divide a graph into triconnected components.

Let us reexamine the graph H_P of $\mathcal{H}$ from the previous subsection. Assume H_P happens to be planar and that it is drawn in the plane as shown in Fig. 3.3, i.e. P appears as a horizontal line and all its bridges in H_P are drawn on the top. (This embedding will be referred, henceforth, as a *canonical* embedding.) Then, an important, but easy to see, observation that follows from Theorem 2.3 is that a pair of vertices x and y separates P iff a bounded region in the canonical embedding of H_P contains x and y. This fact can be used quite effectively in producing candidate lists as demonstrated below.

Algorithm *Find Candidate Lists*

Input: A collection of planar graphs encoding separating pairs.
Output: A set of paths, called candidate lists, encoding separating pairs. The output representation is a linked list.

1. For every bridge B of P in H_P that does not contain r_P, let u and v, respectively, be the attachments of B with the minimum and maximum *pos* value.
   (i) output the edge (u, v),
   (ii) if v is not adjacent to r_P in H_P and if the furthest vertex to the left that is reachable from v through a bridge is u, then perform this step. Let w be the furthest vertex to the right that is reachable through a bridge adjacent to v. Add a pointer from the output location of (u, v) to that of (v, x) where $x = w$ if $w \neq v$; otherwise, x is the successor of v on P. (See Fig. 3.3.)
2. For every edge (a, b) of P, $pos(a) < pos(b)$, that is not part of a triangular region,
   (i) output the edge (a, b),
   (ii) if the furthest vertex to the left that is reachable from b through a bridge is b, then perform this step. Let c be the furthest vertex to the right that is reachable through a bridge adjacent to b. Add a pointer from the output location of (a, b) to that of (b, d) where $d = c$ if $c \neq b$; otherwise, d is the successor of b on P. (See Fig. 3.3.)

In the rest of this subsection, we show how to take a non-planar H_P and produce a planar graph I_P on the same set of vertices such that the set of separating pairs of H_P and that of I_P are identical. We will denote the resulting planar collection by $\mathcal{I}$. The process of planarizing H_P involves coalescing interlacing bridges. Call a pair of bridges B_1 and B_2 of P as interlacing bridges if two of the attachments of B_1 and B_2 are at x, y and u, v, respectively, such that that $pos(x) < pos(u) < pos(y) < pos(v)$. The operation of coalescing is to discard the bridges B_1 and B_2 and to put in a new, single bridge whose attachments are a union of the attachments of B_1 and B_2. A nice property of the operation coalescing that follows from Theorem 2.3 is that it preserves the set of separating pairs. Now, if bridges of H_P are coalesced until no more bridges are interlacing, then the resulting graph is planar; this fact can easily be proven by constructing a canonical embedding of the resulting graph. Denote the resulting planar graph by I_P.

In the following, we give a fast parallel algorithm for finding $\mathcal{I}$. At a high level, the algorithm reduces the problem of planarizing H_P to that of finding connected components of a graph G_b. The vertices of G_b correspond to bridges of P. Two vertices b_1 and b_2 of G_b are in the same connected components iff the bridges corresponding to the two vertices B_1 and B_2 are interlacing. In constructing G_b, if we add edges between b_1 and all vertices whose corresponding bridges in H_P interlace with B_1, then there could be far too many edges in G_b (as many as $O(n^2)$). The trick is to add at most two edges per vertex. Add (b_1, b_2) (resp. (b_1, b_3)) where B_2 (resp. B_3) interlaces B_1 and furthermore it is the bridge with an attachment that is furthest to the left (resp. right) with respect to the leftmost (resp. rightmost) attachment of B_1. It turns out that employing this trick does not alter the connected components of G_b. (See Appendix A for a proof of this fact.) We summarize the ideas below.

Algorithm *Planarize* $\mathcal{H}$

Input: The collection $\mathcal{H}$ of graphs encoding separating pairs of G.

Output: A collection $\mathcal{I}$ of planar graphs encoding separating pairs of G succinctly.

1. For each H_P, build a graph G_b based on the bridges of P_{xy} in H_P that do not contain r_P. These bridges are all single edges (u, v).
   (i) For each such (u, v), create a vertex in G_b denoted by the 2-tuple $\langle u, v\rangle$.
   (ii) Find the bridge (a, b) (resp. (c, d)) with one attachment on $P(u, v)$ and the other furthest to the left (resp. right) from u (resp. v) on $P[x, u)$ (resp. $P(v, y]$).
   (iii) Add an edge between $\langle u, v\rangle$ and $\langle a, b\rangle$, and between $\langle u, v\rangle$ and $\langle c, d\rangle$ in G_b.
2. Find the connected components $C_1, C_2, ..., C_k$ of G_b.
3. Build I_P using the connected components of G_b.
   (i) $V(I_P) = V(H_P) \cup \{c_i \mid C_i \text{ is a component of } G_b\}$
   (ii) $E(I_P) = E(P)$. Add the edges incident on r_P to $E(I_P)$. In addition, include the following edges. Add (c_i, u) if there a vertex in C_i with u as one of the 2-tuples in its label.

3.4. Complexity on a CRCW Pram. The results of the previous subsections show that the following algorithm generates all separating pairs (as candidate lists) of a given biconnected graph G.

Algorithm *Separating Pairs*

1. Find an open ear decomposition of G.
2. Construct the local replacement graph G' by executing *Build* G'.
3. Succinctly encode separating pairs using *Build* $\mathcal{H}$.
4. Use *Planarize* $\mathcal{H}$ and obtain the collection $\mathcal{I}$.
5. Generate separating pairs as candidate lists by invoking *Find Candidate Lists* with $\mathcal{I}$ as its input.

The most expensive step of the above algorithm from the deterministic complexity point of view is providing the required input representations to some of the subroutines we use, and building the adjacency lists of the auxiliary graphs. We show in Appendix B how to construct the needed representation of graphs assuming that the input is a list of edges. This construction runs in $O(\log n)$ time with $((m+n)\log\log n)$ work. In the remainder of this subsection we will argue that the complexity of the rest of the algorithm is identical to that of the best known parallel connected component algorithm.

Let G have n vertices and m edges. We say an algorithm has an 'almost-optimal' processor-time bound, if it runs in $O(\log n)$ parallel time with $O((m+n)\alpha(m,n)/\log n)$

processors on a CRCW PRAM, where α is the inverse Ackermann function. We first note the following results on optimal and almost-optimal parallel algorithms.

A List ranking on n elements can be performed optimally in $O(\log n)$ on an EREW PRAM [3].

B Connected components and spanning tree of an n-node, m-edge graph can be found in time $O(\log n)$ with $O((m+n)\alpha(m,n)/\log n)$ on an ARBITRARY CRCW PRAM [3] provided the input is presented as an adjacency list.

C Least common ancestors of k pairs of vertices in an n-node tree can be found in $O(1)$ time with k processors after $O(\log n)$ time preprocessing using $O(n/\log n)$ processors on an EREW PRAM using the algorithm in [24].

D The Euler-tour technique on trees of [25] can be implemented optimally in $O(\log n)$ time with $O(n/\log n)$ processors on an EREW PRAM using **A**.

E Using the above mentioned results as subroutines, we obtain an almost-optimal parallel algorithm for finding an open ear decomposition from the algorithm in [19], [17], for finding biconnected components from the algorithm of [25], and for finding an *st*-numbering in a biconnected graph from the algorithm of [19].

We will refer to the above five results while describing the processor-time complexity of Algorithm *Separating Pairs*.

Step 1 can be done almost optimally by **E**.

Consider Step 2. The digraphs $\vec{G}$, $\vec{T}$, $\overleftarrow{G}$, and $\overleftarrow{T}$ can be constructed almost optimally using the *st*-numbering algorithm of **E**. Splitting and renaming can be achieved by making the vertex labels a 2-tuple: the first component representing the vertex label, and the second representing the ear label of the edge that is incident on that vertex. We can implement this in constant time per split node. The processors assigned to the end edges of each ear can be made responsible for adding the edge to attach that ear. The *lca* values of non-tree edges can be computed optimally by **C**. Step 3a of *Build G'* identifies the bundles of parallel ears. Let us analyze its complexity. For building G_p, the processor assigned to the last edge (a,b) of each ear examines the ear labels of the two edges incident on the $lca(a,b)$ in the fundamental cycle created by including (a,b) in $\vec{T_l}$. Let the ear labels be P and Q. Next, it checks to see if P and Q have identical pair of end vertices. If so, an edge is added in G_p between p and q. Now using the almost-optimal connected component algorithm of **B** we can implement Step 3b of *Build G'*. This leads to an almost-optimal algorithm to implement *Build G'*.

Let us examine the complexity of *Build $\mathcal{H}$*. The *pos* labeling can be computed optimally in $O(\log n)$ time using the Euler-tour technique by **D**. The auxiliary multigraph G_e can be constructed as follows. The vertex set is easy to create. The processor assigned to the first edge of an ear R does the following. (i) It finds the end vertices u and v of that ear. (ii) It also finds the ears P and Q where $P = ear(u)$ and $Q = ear(v)$, and the values $pos(u,P)$ and $pos(v,Q)$. Assume that the vertices corresponding to P, Q and R in G_e are p, q and r respectively. Finally, (iii) it creates edges (p,r) and (r,q)

and labels these edge with a 2-tuple $\langle pos(u,P), pos(v,Q)\rangle$.

The α labeling of the vertices of G_e consists of the following steps. Find the blocks of G_e almost optimally as in **E**. Treat each block D separately and construct a spanning tree T_b in each block almost optimally as in **B**. The articulation point q with the smallest label in a block D can be found optimally by using the Euler-tour technique. That gives us the first component of α, i.e. Q. The values a and b can be found by examining the 2-tuple labels of the edges of D incident on q. These values are broadcast to all the vertices in the block D using, again, the Euler-tour technique. Finally, building H_P involves computing the minimum and maximum of the labels of edges incident on a vertex. This can be done optimally in $O(\log n)$ time using **A**.

In *Planarize* $\mathcal{H}$, the only nontrivial step is implementing the algorithm of Appendix A, i.e. the construction of the G_p. It involves the identification of the arcs (a,b) and (c,d) for each arc (x,y). An optimal algorithm for this problem is given in [1] (this problem is also known as the range-minima problem). The building of I_P can be done almost-optimally by an easy reduction to the connected components.

Finally, *Find Candidate Lists* requires computing the minimum and the maximum of the labels of edges incident on a vertex. This can be done optimally in $O(\log n)$ time using **A**.

4. An Algorithm for Finding Triconnected Components. We start with some definitions.

Let $G=(V,E)$ be a biconnected graph, and let Q be a subgraph of G. We define the *bridge graph of* Q, $S=(V_S,P_S)$ as follows (this is a little modified from the usual definition as in [5], [18], [22]). Let the bridges of Q in G be $B_i, i=1,...,k$. Then $V_S=V(Q)\cup\{B_1,...,B_k\}$ and $P_S=E(Q)\cup\{\text{edge } (v,B_i) \text{ for each edge } (v,w)\in B_i \text{ with } w\in V(Q), 1\le i\le k\}$. Note that S is a *multigraph*, i.e., a graph in which there can be several edges between the same pair of vertices. Each $B_i\in V_S$ together with the edges incident on B_i is a bridge of Q in S.

A *star* is a connected graph with a vertex v such that every edge in the graph is incident on v. A *star graph* $G(P)$ is a graph G consisting of a simple path P, each of whose bridges is a star. Thus if Q is a simple path in G, then S, the bridge graph of Q, is a star graph. Let B be a star in a star graph $G(P)$, where for convenience let $P=\langle 0,...,k-1\rangle$. Let the attachments of B on P be $v_0,...,v_j$, with $v_0<v_1<...<v_j$. Then the vertices v_0 and v_j are the *end attachments* of B and the remaining attachments are its *internal attachments*. We will also refer to v_0 as the *left attachment* of B and v_j as its *right attachment*. The closed interval $[v_0,v_j]$ is the *span* of B and it contains all of the vertices on P between v_0 and v_j (both vertices inclusive).

We now review some material from [26], [12], [18] relating to triconnected components. This material deals with multigraphs. An edge e in a multigraph is denoted by (a,b,i) to indicate that it is an edge between a and b; here i is the label that distinguishes e from the other edges 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 pair, 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 $G = (V, E)$. For any bridge X of $\{a, b\}$, let $\bar{X}$ be the induced subgraph on $(V - V(X)) \cup \{a, b\}$. Let B be a bridge of G such that $|E(B)| \geq 2, |E(\bar{B})| \geq 2$ and either B or $\bar{B}$ is biconnected. We can apply a *Tutte split* ([26], [12]) $s(a, b, i)$ to G by forming G_1 and G_2 from G, where G_1 is $B \cup \{(a, b, i)\}$ and G_2 is $\bar{B} \cup \{(a, b, i)\}$. Note that we consider G_1 and G_2 to be two separate graphs. Thus it should cause no confusion that there are two edges (a, b, i) since one of these edges is in G_1 and the other is in G_2. The graphs G_1 and G_2 are called the *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. In this section we give an almost-optimal $O(\log n)$ time parallel algorithm to find the triconnected components of G corresponding to triconnected simple graphs and polygons. The bonds can be inferred, if necessary, by counting the number of triconnected components with respect to each separating pair.

Let $G = (V, E)$ be a biconnected graph with an open ear decomposition $D = [P_0, ..., P_{r-1}]$. When referring to vertices on a specified ear P_i or on a path P, we will assume for convenience that they are numbered in sequence from one end point of the path (its *left end point*) to the other (its *right end point*). Let $\{a, b\}$ be a pair separating P_i. Let $B_1, ..., B_k$ be the bridges of P_i with no attachments outside the interval $[a, b]$ on P_i, and let $T_i(a, b) = (\cup_{j=1}^{k} B_j) \cup P_i(a, b)$, where $P_i(a, b)$ is the segment of P_i between and including vertices a and 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* $G_2 = \bar{T}_i(a, b) \cup \{(a, b, i)\}$. An ear split $e(a, b, i)$ is a Tutte split if either $G_1 - \{(a, b, i)\}$ or $G_2 - \{(a, b, i)\}$ is biconnected.

Let S be a nontrivial candidate list for ear P_i. Two vertices u, v in S are an *adjacent separating pair for* P_i if u and v are not adjacent to each other on P_i and S contains no vertex in the interval (u, v) on P_i. Two vertices a, b in S are an *extremal separating pair for* P_i if $|S| \geq 3$ and S contains no vertex in the interval outside $[a, b]$. An ear split on an adjacent or extremal separating pair is a Tutte split, and the Tutte components of G are obtained by performing an ear split on each adjacent and extremal separating pair [18].

With each ear split $e(a, b, i)$ corresponding to an adjacent or extremal pair separating P_i, we can associate a unique Tutte component of G as follows. Let $e(a, b, i)$ be such a split. Then by definition $T_i(a, b) \cup \{(a, b, i)\}$ is the upper split graph associated with the ear split $e(a,b,i)$. The *triconnected component of the ear split* $e(a,b,i)$ denoted by $TC(a,b,i)$ is $T_i(a,b) \cup \{(a,b,i)\}$ with the following modifications: Call a pair $\{c,d\}$ separating an ear P_j in $T_i(a,b)$ a *maximal pair for* $T_i(a,b)$ if there is no e,f in $T_i(a,b)$ such that $\{e,f\}$ separates some ear P_k in $T_i(a,b)$ and c and d are in $T_k(e,f)$. In $T_i(a,b) \cup \{(a,b,i)\}$ replace $T_j(c,d)$ together with all two-attachment bridges with attachments at c and d, for each maximal pair $\{c,d\}$ of $T_i(a,b)$, by the edge (c,d,j), to obtain $TC(a,b,i)$. We denote by $TC(0,0,0)$, the unique triconnected component that contains a specified edge on P_0.

We note that $TC(a,b,i)$ as defined above is a triconnected component of G since each split of $T_i(a,b)$ in the above definition is a valid Tutte split, and the final resulting graph contains no unprocessed separating pair. Further, we also note that every triconnected component of G appears as $TC(a,b,i)$ for some adjacent or extremal separating pair. This is seen as follows. Let T be a triconnected component of G. By the results in [18] we know that T can be obtained by a sequence of ear splits at adjacent and extremal pairs separating ears in the open ear decomposition D of G. Since the order of processing these ear splits is arbitrary, let us consider a sequence in which these splits are performed in nonincreasing order of ear number. In this case, every upper split graph formed at the end of processing ear P_i must be a triconnected component since it will contain no unprocessed separating pairs. Let P_i be the lowest numbered ear that contains a separating pair whose copies are present in T and let $e(a,b,i)$ be the last ear split performed while generating T. Then clearly, $T = TC(a,b,i)$.

In our parallel algorithm, we will make the collection of splits S_1 corresponding to adjacent separating pairs simultaneously, followed by the collection of splits S_2 for extremal separating pairs. We will call each component present after completion of splits in S_1 an *adjacent triconnected component*, and denote it by $TC_A(a,b,i)$. Since the virtual edges corresponding to the splits will be inserted by concurrent writes, we will have only one copy of each such edge between a given pair of vertices. Hence we will not generate the triconnected components corresponding to bonds. These can be inferred, if necessary, by counting the number of triconnected components of the other two types that are present at each separating pair.

The rest of this section is devoted to describing an almost-optimal algorithm for performing these operations. We first review some further material from [18] and [22].

Let G be a biconnected graph with an open ear decomposition $D = [P_0, ..., P_{r-1}]$. Let $B_1, ..., B_l$ be the bridges of P_i that contain a non-attachment vertex on an ear numbered lower than i; we call these 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) Replacing all of the anchor bridges by a new star whose attachments edges are the union of the internal attachments edges of all anchor bridges, deleting the attachments of anchor bridges to the end points of P_i, and replacing them by one new edge to each end point. We will call this new star the *anchoring star* of $G_i(P_i)$.

b) Removing any multiple two-attachment bridges with the same two attachments, and also removing any two-attachment bridge with the end points of P_i as attachments.

Note that $G_i(P_i)$ is a multigraph. (This definition of ear graph is slightly modified from that in [18], [22] to reflect the change made in the definition of bridge graph.) $G_i(P_i)$ is also a star graph.

Two stars S_j and S_k in a star graph $G(P)$ *interlace* (see also [5], page 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.

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* $G_c(P)$ of $G(P)$ is the graph obtained from G by coalescing all pairs of stars that interlace. Note that $G_c(P)$ is a star graph with respect to P, and $G_c(P)$ has a planar embedding with P on the outer face, since no pair of stars interlace on P.

Let $G(P)$ be a star graph in which no pair of stars interlace. If $G(P)$ contains no star that has attachments to the end points x and y of P, then add a virtual star X to $G(P)$ with attachments to x and y. The *star embedding* $G^*(P)$ of $G(P)$ is the planar embedding of (the possibly augmented) $G(P)$ with P on the outer face. A star B is the *parent-star* of star B' and B' a *child-star* of B if there is a face in the star embedding $G^*(P)$ that contains the left and right attachments x and y of B' as well as an attachment edge of B in each of the intervals $[l, x]$ and $[y, r]$, where l and r are the left and right end points of P.

The following lemma is shown in [18].

LEMMA 4.1. *A pair* $\{a, b\}$ *separates* P_i *in the coalesced graph* $G_{i_c}(P_i)$ *if and only if* $\{a, b\}$ *separates* P_i *in* G.

We will use the following corollary to the lemma given above.

COROLLARY 4.2. *An edge* (x, y) *incident on* P_i *is in* $TC(a, b, i)$ *if and only if* (x, y) *is in the triconnected component associated with pair* $\{a, b\}$ *separating* P_i *in* $G_{i_c}(P_i)$.

Proof. Let $C_i(P_i)$ be the bridge graph of P_i and let $C_{i_c}(P_i)$ be its coalesced graph. A straightforward extension of the proof of Lemma 4.1 given in [18] (Theorem 1 of that paper) shows that an edge incident on P_i is in $TC(a, b, i)$ if and only if it is in the triconnected component associated with the pair $\{a, b\}$ separating P_i in $C_{i_c}(P_i)$. We then observe that the edges of $C_i(P_i)$ that are deleted in the ear graph $G_i(P_i)$ cannot appear in $TC(x, y, i)$ for any pair x, y separating P_i. □

For convenience of notation, we will denote $G_{i_c}(P_i)$ by $G_c(P_i)$. [22] give an almost-optimal algorithm to form the coalesced graph of a star graph $G(P)$ that runs in logarithmic time on a CRCW PRAM. This algorithm has the same processor-time complexity as that of finding connected components.

LEMMA 4.3. *In the coalesced graph $G_c(P_i)$, for each adjacent pair $\{a, b\}$ separating P_i, there is at most one bridge of P_i with attachments on a, b and a vertex in (a, b), the portion of P_i between a and b.*

Proof. Suppose not, and let B_1 and B_2 be two bridges of P_i in $G_c(P_i)$ that have attachments on a, b and a vertex in (a, b). Then B_1 and B_2 must interlace, which contradicts the fact that $G_c(P_i)$ is the coalesced graph of the ear graph $G_i(P_i)$. □

LEMMA 4.4. *Let B be a two-attachment bridge of P_i in $G_c(P_i)$ with attachments a and b. Then*

- *a) If the span $[a, b]$ is degenerate (i.e., (a, b) is an edge in P_i) or if there is a bridge B' of P_i with attachments on a and b and at least one other vertex, then $G_c(P_i) - \{B\}$ defines the same set of polygons and simple triconnected components $TC(x, y, i)$, for i fixed, as $G_c(P_i)$.*
- *b) If part a does not hold, then $\{a, b\}$ is an extremal pair separating P_i as well as an adjacent pair separating P_i.*

Proof. Let P_j be the lowest-numbered ear in B. Then, $j > i$ and a and b are the end points of P_j. Hence the ear split $e(a, b, j)$ separates B from P_i, and thus B is not part of $TC(x, y, i)$ for any pair $\{x, y\}$ separating P_i. So a 2-attachment bridge on P_i is never a part of a triconnected component associated with a pair separating P_i, though it may define some adjacent and extremal separating pairs as in case b) of the lemma.

We now prove parts a) and b) of the lemma.

- a) Suppose span $[a, b]$ is degenerate. Then the triconnected component associated with split $e(a, b, i)$ is the single edge (a, b), which is a bond. Otherwise, if there is a bridge B' with attachments on a, b and at least one other vertex v, then the triconnected component associated with split $e(a, b, i)$ contains a portion of P_i between a and b, together with B' if v is in the interval (a, b) and is a polygon if v is not in $[a, b]$. Both of these situations can be inferred without the presence of B. Note that it is not possible for B' to have an attachment v in the interval (a, b) and another attachment w that is not in $[a, b]$, since the bridge B would interlace with B' in such a case.
- b) Let the span $[a, b]$ be non-degenerate and let the portion of P_i between a and b be $\langle a = a_1, a_2, ..., a_k = b\rangle$. Since there is no k-attachment bridge, $k > 2$, with span $[a, b]$, there must exist an $a_i, 1 < i < k$, such that a, a_i and b are in the same candidate list C, and no vertex outside $[a, b]$ is in C. Hence $\{a, b\}$ is an extremal separating pair. Also, since there is no bridge with attachments on a, b and some other vertex c outside $[a, b]$, there must be some vertex c on P_i such that either $c < a < b$ or $a < b < c$, and a, b and c are in the same candidate list C'. Further, no vertex in the interval (a, b) can belong to C'. Hence $\{a, b\}$ is an adjacent pair in the candidate list C'.

□

Let $\{a, b\}$ be an adjacent separating pair for ear P_i. The pair a, b is a *non-vacuous adjacent separating pair for* P_i if there is a bridge of P_i in $G_c(P_i)$ with attachments on a, b and one other vertex in the interval (a, b) on P_i; otherwise the pair $\{a, b\}$ is a *vacuous adjacent separating pair.* We leave it as an exercise to verify that if $\{a, b\}$ is a non-vacuous adjacent separating pair then $TC(a, b, i)$ is a simple triconnected graph and if $\{a, b\}$ is a vacuous adjacent separating pair, then $TC(a, b, i)$ is a bond; if $\{a, b\}$ is an extremal separating pair then $TC(a, b, i)$ is a polygon.

Lemmas 4.3 and 4.4, in conjunction with Corollary 4.2 tell us that we can compute the triconnected components of G by the following method. Make the splits corresponding to the adjacent separating pairs by performing, for each star B in $G_c(P_i)$, an ear split $e(a, b, i)$, where $[a, b]$ is the span of B. Then, break off chains of degree-2 vertices on the paths in the resulting star graphs to perform the splits corresponding to the extremal separating pairs.

There are two problems with using the above approach in an efficient logarithmic time algorithm for forming the triconnected components of a graph. One is that we are working with the ear graphs of the ears and the total size of these graphs need not be linear in the size of G. The second is that this approach will not work if a vertex a appears in an ear split for two different ears. In particular, two-attachment bridges corresponding to adjacent separating pairs will be separated on two different ears and this would cause processor conflicts.

We now turn to G', the local replacement graph of G which we defined in Section 3.1, in order to develop an efficient method of identifying the associated triconnected components.

Let G' be the local replacement graph of G and let $D' = [P'_0, ..., P'_{r-1}]$ be the corresponding open ear decomposition. By Corollary 3.16, a pair $\{a, b\}$ separates P_i in G if and only if the pair $\{a_{P_i}, b_{P_i}\}$ separates P'_i in G'. Further, neither a_{P_i} nor b_{P_i} is an end point of P'_i. The following lemma shows that in G' we can efficiently identify any bridge B of an ear P'_i which has no attachment to an end point of P'_i.

LEMMA 4.5. *Let G_e be the graph obtained from G' by collapsing all internal vertices of each ear into a single vertex. Let vertex v_i represent ear P'_i in G_e. Then the edges incident on v_i in each block of G_e whose lowest-numbered vertex is v_i correspond to the attachment edges of a bridge of P'_i in G' and conversely, each bridge of P'_i in G' that has no attachments to the end points of P'_i corresponds to a block of G_e.*

Proof. Let e_1 and e_2 be any pair of edges incident on v_i that lie in the same block B of G_e whose lowest-numbered vertex is v_i. Then there is a path between e_1 and e_2 in B that avoids v_i and hence in G' there is a path between e_1 and e_2 that avoids internal vertices of P'_i. But since the lowest-numbered vertex in B is v_i, the path between e_1 and e_2 in B does not contain any vertex on an ear numbered lower than i, and hence e_1 and e_2 must lie in a connected component in $G' - \{P'_i\}$.

Conversely, let B be a bridge of P_i' in G' that has no attachments to the end points of P_i'. Then, when the internal vertices of P_i' are collapsed into v_i, all of the attachments of B on P_i' become incident on v_i. Thus B becomes a block in G_e with articulation point v_i. Further since B has no attachments to the end points of P_i', B is not an anchor bridge of P_i' and hence v_i is the minimum-numbered vertex in B in G_e. □

Recall that (Proposition 3.7) the copies of a vertex v in G' are connected in the form of a tree. For the following lemma, assume this local tree that replaces v is rooted at v_S where $ear(v) = S$.

LEMMA 4.6. *Let $\{x, y\}$ be a separating pair that separates P in G. Let C be a connected component in $G - \{x, y\}$ that contains $P(x, y)$. If Q is an ear label of one of the edges of C and if x is one the end points of Q, then x_P is an ancestor of x_Q in the local tree that replaces x.*

Proof. If Q is parallel to P, then as $\{x, y\}$ separates P it has the smallest ear label among the labels of edges of C. Hence, P would have to be the representative of the bundle containing Q and the lemma is clearly true. Otherwise, notice that $V(C) - V(P)$ consists of non-attachment vertices of a relevant bridge of P. If the lca of the end points of Q is not x, then x_Q is a child of x_P by Step 2 of *Build G'*. Otherwise, by Lemma 3.12, x_Q is a descendant of x_P. □

LEMMA 4.7. *Any bridge of P_i' in G' with an attachment to an end point of P_i' must be either part of the anchoring star of $G_i'(P_i')$ or a bridge of P_i' with attachments only to the end points of P_i'.*

Proof. Let B be a bridge of P_i' in G' with an attachment to one of its end points x_{P_j}.

We first show that the internal vertices on P_j' are part of the anchoring star of P_i'. If P_j is not parallel to P_i, then $j < i$ and the result follows directly. If P_j is parallel to P_i, then let C be the connected component constructed in Step 3 of Algorithm Build G' that contains P_i and P_j and let P_l be the root of the spanning tree of C constructed in that step. Hence $l \leq j$ and x_{P_l} is an ancestor of x_{P_j} in LT_x where LT_x the local tree that replaces the vertex x in G'. Further, by the construction in Step 3 of Algorithm Build G' there is a path in G' between an internal vertex of P_j' and an internal vertex of P_l' that avoids all vertices on P_i'. Hence the vertices on P_j' belong to an anchor bridge of P_i'.

Let $e = (y, x_{P_j})$ be an attachment edge of bridge B of P_i'. We will show that B is an anchor bridge of P_i'. Let e belong to ear P_k'.

If $y \neq x_{P_k}$, then e is an edge on P_j'. Hence e, and thus B, is part of the anchoring star of P_i'. If $y = x_{P_k}$ then consider the fundamental cycle completed by the non-tree edge (u, v) in P_k' in the tree $\vec{T}$ in which (x_{P_j}, x_{P_k}) is a tree edge. If P_k is not parallel to P_i, then the presence of edge (x_{P_k}, x_{P_j}) in G' implies that either this fundamental cycle contains an edge on P_j' and no vertex on P_i', or there is a path from v to the root s of G' that avoids all vertices in P_i'. In either case, e is part of a bridge of P_i' that contains a non-attachment vertex on an ear numbered lower than i.

If P_k and P_i are parallel to each other, then if P_j is not parallel to P_i, each of P_i and P_k correspond to the root of the spanning tree of a connected component constructed in Step 3 of Algorithm Build G'. Hence by Lemma 4.6, B is a bridge of P_i' with no internal attachment on P_i'. Finally, if P_i, P_j and P_k are all parallel to each other, then since x_{P_j} is the parent of x_{P_k} in LT_x, there is a path in G' between an internal vertex of P_k' and an internal vertex of P_j' that avoids all vertices in P_i'. Hence e is part of the bridge of P_i' that contains the internal vertices of P_j'. This bridge was shown to be an anchor bridge of P_i'. □

Lemmas 4.5 and 4.7 tell us that the following algorithm generates the ear graph of each ear in G'.

Algorithm *Ear Graphs of* G'
Input A local replacement graph G' with its associated open ear decomposition $D' = [P_0', ..., P_{r-1}']$.

1. Form G_e.
2. For each block B in G' do
   a) Let the minimum-numbered vertex in B be v_i. Make the image e in G' of each edge e' in B incident on v_i as an attachment edge of non-anchor bridge B in the ear graph of P_i'.
   b) for each vertex $v_j \neq v_i$ in B make the image e in G' of each edge e' of B incident on v_j as an attachment edge of the anchoring star of the ear graph of P_j'.
3. For each ear P_i', add attachment edges to the end points of P_i' for the anchoring star created in Step 2b.

Step 1 is the same as Step 1 of Algorithm *Build* $\mathcal{H}$ applied to G' (Section 3.2). Steps 2 and 3 can be implemented in constant time per edge using the α-values of each vertex computed in Step 3 of Algorithm *Build* $\mathcal{H}$. The total size of all of the ear graphs is $O(m)$, where m is the number of edges in G', since each edge in G' appears in at most two ear graphs (corresponding to the ears containing the two end points of the edge).

Having obtained the ear graph $G_i'(P_i')$ of each ear in G', we can obtain the coalesced graph $G_c'(P_i')$ of each of the ear graphs using the algorithm of [22]. By Corollary 3.16 and Lemma 4.1, a pair $\{x_{P_i}, y_{P_i}\}$ is an adjacent (extremal) pair separating P_i' in $G_c'(P_i')$ if and only if $\{x, y\}$ is an adjacent (extremal) pair separating P_i in $G_c(P_i)$. It turns out that the relation between G and G' extends beyond separating pairs to triconnected components. The following two lemmas allow us to relate the bridges of ears in G' with the bridges of ears in G, and hence develop an efficient algorithm to find the triconnected components of G.

LEMMA 4.8. *Let x be a vertex in P_i in G (possibly its end point) and let $e_1 = (u_1, x) \in P_j$ and $e_2 = (u_2, x) \in P_k$ be two edges incident on x that belong to different bridges (B_1 and B_2 respectively) of P_i, each of which has an internal attachment on P_i.*

Then the least common ancestor (lca) of x_{P_j} and x_{P_k} in LT_x is x_{P_p}, where x_{P_p} is an ancestor of x_{P_i} in LT_x.

Proof. Suppose not and let x_{P_l} be $lca(x_{P_j}, x_{P_k})$, where x_{P_l} is a proper descendant of x_{P_i}.

Case 1: There are no parallel ears incident on x in G.

Let the vertices on the path from x_{P_l} to x_{P_j} in LT_x be $x_{P_{j_1}} = x_{P_j}, x_{P_{j_2}}, ..., x_{P_{j_r}} = P_l$. Then by construction the fundamental cycle of P_{j_h} contains an edge in $P_{j_{h+1}}, h = 1, ..., r-1$, and no edge in P_i in G. A similar situation holds for x_{P_k}. But then all of these ears would lie in the same bridge of P_i.

Case 2: There are some parallel ears incident on x in G.

Again, by construction, a pair of parallel ears have an ancestor-descendant relationship in LT_v only if they are connected to each other by a path that avoids P_i and their two end points. Hence again, by a combination of this observation and the argument in case 1 we deduce that P_j and P_k must be in the same bridge of P_i. □

LEMMA 4.9. *Let $\{a, b\}$ be an adjacent or extremal pair separating P_i in G and let $\mathbf{B} = \{B_1, ..., B_r\}$ be the bridges of P_i with an internal attachment in (a, b). Similarly, let $\mathbf{B}' = \{B'_1, ..., B'_{r'}\}$ be the bridges of P'_i in G' with an internal attachment in (a_{P_i}, b_{P_i}). Then, $r = r'$, and there is a one-to-one correspondence between the bridges in $\mathbf{B}$ and the bridges in $\mathbf{B}'$ (without loss of generality we assume that the correspondence is between B_i and B'_i for $i = 1, ..., r$) such that an edge e is in B_i if and only if the corresponding edge e' is in B'_i.*

Proof. We only need to verify that the connectivity at LT_v in $G' - \{P'_i\}$, for $v \in P_i$, since the connectedness in the rest of the graph remains unaltered when a vertex u in G is replaced by the tree LT_u in G'. But we note from Lemma 4.8 that if two edges $e_1 = (u, v) \in P_j$ and $e_2 = (u_2, v) \in P_k$, $v \in P_i$ are in different bridges of P_i, then e_1 and e_2 are separated from each other in $G' - \{v_{P_i}\}$. The lemma follows. □

From Lemma 4.9, we see that given an adjacent pair $\{a, b\}$ separating P_i, the bridges of P'_i with no attachments outside the interval $[a_{P_i}, b_{P_i}]$ on P_i, together with the path from a_{P_i} to b_{P_i} on P'_i will correspond to the upper split graph of the ear split $e(a, b, i)$ in G. Now, we can further apply the Corollary 4.2 to G', and work with $G'_c(P'_i)$ to directly identify the triconnected components of G. This is done in the following algorithm.

Algorithm *Triconnected Components*

Input A biconnected graph G with an open ear decomposition $D = [P_0, ..., P_{r-1}]$, its local replacement graph G', together with its associated open ear decomposition $D' = [P'_0, ..., P'_{r-1}]$, and the coalesced graph $G_c(P'_i)$ of the ear graph of each ear in D'.

1. For each ear P'_i do

   for each vertex v on P'_i, make a copy, v_B, of v for each star B in $G_c(P'_i)$ that has an attachment on v. If there is no star with an internal attachment on v, then make an additional copy v_P of v to represent the lower split graph formed when all adjacent pairs containing v have been processed.

2. Assign vertices to edges on P_i'
   a) For $j = 0, 1, ..., k-1$ do
      If there is no bridge with its leftmost attachment on j, then replace edge $(j, j+1)$ on P_i' by an edge incident on j_C, where C is B if there is a bridge B with an internal attachment on j and is P otherwise.
   b) For $j = 1, ..., k$ do
      If there is no bridge with its rightmost attachment on j, then replace edge $(j-1, j)$ on P_i' by an edge incident on j_D, where D is B' if there is a bridge B' with an internal attachment on j and is P otherwise.
3. Make the splits corresponding to adjacent separating pairs:
   For each star B in $G_c(P_i')$ do
   Let the end attachments of B on P_i' be v and w, $v < w$.
   a) Replace all edges in B incident on v by edges incident on v_B. Similarly replace all edges in B incident on w by edges incident on w_B.
   b) If B has no child-star with an attachment at v, then replace edge $(v, v+1)$ on P by an edge incident on v_B. Similarly, if B has no child star with an attachment at w, then replace edge $(w-1, w)$ by an edge incident on w_B.
   c) Place a virtual edge between v_B and w_B, and another virtual edge between v_C and w_D, where C (resp. D) is the parent-star of B if the parent star of B has an attachment at v (resp. w) and is P otherwise.
   d) Replace each internal attachment edge of B on a vertex u in P_i' by an edge incident on u_P.
4. Process extremal pairs:
   For each star B in $G_c(P_i')$ do
   Let the attachments of B on P_i' be $v_0 < v_1 < ... < v_l$.
   For $j = 0, ..., l-1$ do
      if (v_{j_B}, v_{j+1_B}) is not an edge in the current component containing B, then
      For convenience of notation let x denote v_j and let y denote $v_j + 1$.
      a) Make a copy x_{B_r} of x and a copy y_{B_l} of y.
      b) Replace the edge on P_i' connecting x_B to the next larger vertex in the current graph by an edge incident on x_{B_r}.
      c) Replace the edge on P_i' connecting y_B to the next smaller vertex in the current graph by an edge incident on y_{B_l}.
      d) Place a virtual edge between x_B and y_B and another virtual edge between x_{B_r} and y_{B_l}.
5. Convert the vertices in G' into vertices in G.
   In each of the components formed, collapse all vertices that correspond to a given vertex v in G into a single copy of v to construct the triconnected

components of G.

THEOREM 4.10. *Algorithm Triconnected Components correctly finds the simple triconnected components and the polygons of G.*

Proof. Consider a bridge B' in $G_c(P_i')$ with span $[x_{P_i}, y_{P_i}]$. By Corollary 3.16 and Lemma 4.9 we can map each edge e' in B' (that is not in any LT_v) to an edge e in a bridge B of $G_c(P_i)$ with span $[x, y]$. A similar argument holds for the bridges of P_i' in $G_c(P_i')$ corresponding to the maximal pairs in $T_i(x, y)$. Finally, any two-attachment bridge B'' with attachments c_{P_i} and d_{P_i} on P_i' is split off in P_j' at c_{P_j} and d_{P_j}, where P_j' is the minimum-numbered ear in B''. Hence when we make the split corresponding to B' in Step 3 of Algorithm Triconnected Components, the edges in the component formed must correspond to the edges in the adjacent triconnected component $TC_A(x, y, i)$. Finally, the polygons generated in Step 4 are clearly the polygons of the triconnected components of G since all vertices on a polygon are local to a given ear.

Thus, when we implement Step 5 of the algorithm in a component to get back original vertices of G, we get back a triconnected component of G. □

For the processor-time complexity of Algorithm Triconnected Components, we note that steps 1, 2 and 4 can be performed optimally in logarithmic time. So can all of the steps in Step 3 except Step 3c, which requires identifying the parent-star of a star in a star embedding. This step can be performed using the bucket-sort algorithm of Hagerup [10]. It can also be performed optimally in logarithmic time using list ranking and making use of the fact that $G_c(P_i')$ is planar. The details of this implementation are given in [8]. They are omitted here, since the overall complexity of the algorithm is dominated by the need to perform bucket sort in order to obtain the adjacency lists of the various graphs. Step 5 can be performed with the same bounds as that of finding the connected components of a graph.

Hence Algorithm Triconnected Components runs in $O(\log n)$ time deterministically on a CRCW PRAM while performing $O((m + n) \log \log n)$ work.

5. Conclusion. We have presented an efficient parallel algorithm for dividing a graph into triconnected components. We conclude the paper by mentioning the following remarks.

1. Our algorithm can be adapted to test 3-edge connectivity within the same bounds. For this we use an ear decomposition instead of an open ear decomposition and look for separating pairs of edges. It turns out that in this case it is not necessary to construct the local replacement graph since each edge of the graph is contained in exactly one ear. Hence the resulting algorithm is simpler than the one we have presented for testing (vertex) triconnectivity.
2. Our parallel algorithm is slightly sub-optimal in the work it performs due to the sub-optimality of the currently known parallel algorithms for finding connected components and performing bucket sort. It will be interesting to find improvements in these parallel algorithms, which in turn will lead to improvements in

the bounds for our algorithm.

REFERENCES

[1] N. Alon and B. Schieber, "Optimal preprocessing for answering on-line product queries," tech. report 71/87, Tel Aviv University, Israel, 1987.

[2] J. Cheriyan and R. Thurimella, "Algorithms for parallel k-vertex connectivity and sparse certificates," *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, 1991, pp. 391-401.

[3] R. Cole and U. Vishkin, "Approximate and exact parallel scheduling with applications to list, tree, and graph problems," *Proc. 27th Symp. Found. Comp. Sci.*, 1986, pp. 478-491.

[4] S. Even, "An algorithm for determining whether the connectivity of a graph is at least k," *SIAM J. Computing*, vol. 4, 1975, pp. 393-396.

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

[6] S. Even, and R. E. Tarjan, "Network flow and testing graph connectivity," *SIAM J. Computing*, vol. 4, 1975, pp. 507-518.

[7] D. Fussell and R. Thurimella, "Separation pair detection," *VLSI Algorithms and Architectures*, Springer-Verlag LNCS, vol. 319, 1988, pp. 149-159.

[8] D. Fussell, V. Ramachandran and R. Thurimella, "Finding triconnected components by local replacements," *Proc. of ICALP 89*, Springer-Verlag LNCS, vol. 372, 1989, pp. 379-393.

[9] H.N. Gabow, "A matroid approach to finding edge connectivity and packing arborescences," *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, 1991, pp. 112-122.

[10] T. Hagerup, "Towards optimal parallel bucket sorting," *Information and Computation*, vol. 75, pp. 39-51.

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

[12] J. E. Hopcroft, and R. E. Tarjan, "Dividing a graph into triconnected components," *SIAM J. Computing*, vol. 2, 1973, pp. 135-158.

[13] J. JáJá, and J. Simon, "Parallel algorithms in Graph theory: Planarity Testing," *SIAM J. Computing*, vol. 11, 1982, pp. 314-328.

[14] A. Kanevsky, V. Ramachandran, "Improved algorithms for graph four-connectivity," Journal of Computer and System Sciences, 42 (1991), pp. 288–306.

[15] R. M. Karp, V. Ramachandran, "Parallel algorithms for shared-memory machines," *Handbook of Theoretical Computer Science*, North-Holland, 1990, pp. 869-941.

[16] G.L. Miller, and J. Reif, "Parallel tree contraction and its applications," *Proc. 26th Symp. Found. Comp. Sci.*, October 1985, pp. 478-489.

[17] G.L. Miller, an V. Ramachandran, "Efficient parallel ear decomposition with applications," Manuscript, MSRI, Berkeley, CA, January 1986.

[18] G.L. Miller, and V. Ramachandran, "A new triconnectivity algorithm and its applications," *Proc. 19th Ann. ACM Symp. on Theory of Computing*, May 1987, pp. 335-344; also, *Combinatorica*, vol. 12, 1992, to appear.

[19] Y. Maon, B. Schieber, and U. Vishkin, "Parallel ear decomposition search (EDS) and ST-numbering in graphs," *Theoretical Computer Science*, vol. 47, 1986, pp. 277-298. vol. 227,

1986, pp. 34-45.

[20] H. Nagamochi and T. Ibaraki, "Linear time algorithms for finding a sparse k-connected spanning subgraph of a k-connected graph," *Algorithmica,* to appear.

[21] V. Ramachandran, "Parallel open ear decomposition and its application to graph biconnectivity and triconnectivity," to appear as a chapter in *Synthesis of Parallel Algorithms*, J.H. Reif, ed., Morgan-Kaufmann.

[22] V. Ramachandran, and U. Vishkin, "Efficient parallel triconnectivity in logarithmic time," *VLSI Algorithms and Architectures*, Springer-Verlag LNCS, vol. 319, 1988, pp. 33-42.

[23] S. Rajasekaran and J. H. Reif, "Optimal and sublogarithmic time randomized parallel sorting algorithms," *SIAM J. Computing*, vol. 18, 1989, pp. 594-607.

[24] B. Schieber, U. Vishkin, "On finding lowest common ancestors: simplification and parallelization," *VLSI Algorithms and Architectures*, Springer-Verlag LNCS 319, 1988, pp. 111-123.

[25] R. E. Tarjan, and U. Vishkin, "An efficient parallel biconnectivity algorithm," *SIAM J. Computing*, 14 (1984), pp. 862-874.

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

[27] H. Whitney, "Non-separable and Planar Graphs," *Trans. Amer. Math. Soc.*, 34 (1932), pp. 339-362.

Appendix

A. Keeping G_b Sparse. Recall, from Section 3.3, the trick to keep G_b sparse (Step 1 of *Planarize* $\mathcal{H}$). We add at most two edges for each vertex b_1. The edge (b_1, b_2) (resp. (b_1, b_3)) is such that B_2 (resp. B_3) interlaces B_1 and furthermore it is the bridge with an attachment that is furthest to the left (resp. right) with respect to the leftmost (resp. rightmost) attachment of B_1.

We show that the connected components of the graph obtained by adding an edge between b_1 and b_2 for *every* interlacing pair of bridges B_1 and B_2, and that of G_b are identical. Consider a proof by contradiction. Let $B_1 = (s, t)$ be the rightmost bridge (i.e. highest $pos(t)$ value) with an interlacing bridge $B_2 = (u, v)$, $pos(u) < pos(s) < pos(v) < pos(t)$, such that $(b_1, b_2) \notin E(G_b)$. Then, by our construction, there must exist of (b_2, r_1) and (b_1, l_1) in G_b where the bridges $R = (a, b)$ and $L = (c, d)$ interlace with B_2 and B_1, respectively. Further, R and L are such that $pos(d) < pos(u)$ and $pos(b) > pos(t)$. Now, since R interlaces with B_2, $a \in P(u, v)$. If $a \in P(s, v)$, then R and B_1 interlace. As we assumed that B_1 is the rightmost bridge that belongs to a "bad" interlacement pair, we can conclude that $(r, b_1) \in E(G_b)$. On the other hand, if $a \in P(u, s)$, R and L interlace and $(r, l) \in E(G_b)$. In either case, b_1 and b_2 belong to the same connected component in G_b. A contradiction.

B. Building an Adjacency list using Bucket Sort. The complexity of the procedure we are above to describe is $O(\log n)$ time using $O((m+n) \log\log n)$ processors. The needed representation is a special kind of adjacency list. In this representation, every edge (u, v) appears as two directed edges $\langle u, v\rangle$ and $\langle v, u\rangle$, i.e., the vertices u and v appear in each others adjacency lists. Given a list of edges, we can accomplish this by (1) sorting the edges to make sure that no edge appears twice initially, (2) creating $\langle v, u\rangle$ for every $\langle u, v\rangle$, (3) evaluating the degree of each vertex v by using the difference in the addresses in the sorted array of the first occurrence and last occurrence of v, (4) allocating in memory one array of size degree(v) for each v, and finally (5) making the processor allocated to (u, v) responsible for creating the entry in the list of u. We use the parallel bucket-sort algorithm of Hagerup [10] which runs in $O(\log n)$ time with $(n \log\log n)$ operations to sort n numbers to achieve the desired complexity.