SEPARATION PAIR DETECTION

Donald Fussell and Ramakrishna Thurimella

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-88-08 March 1988

Abstract

A separation pair of a biconnected graph is a pair of vertices whose removal disconnects the graph. The central part of any algorithm that finds triconnected components is an algorithm for separation pairs. Recently Miller and Ramachandran have given a parallel algorithm that runs in $O(\log^2 n)$ time using $O(m)$ processors. We present a new algorithm for finding all separation pairs of a biconnected graph that runs in $O(\log n)$ time using $O(m)$ processors. A direct consequence is a test for triconnectivity of a graph within the same resource bounds.

1 Introduction

Triconnected components are useful in testing a graph for planarity, finding how reliable a computer network is against node failures, etc. The central part of any triconnected component algorithm is an algorithm to find separation pairs. A separation pair of a biconnected graph is a pair of vertices whose removal disconnects the graph.

There are potentially $O(n^2)$ pairs that can qualify to be separation pairs. Consider a simple cycle of n vertices. It is biconnected and every nonconsecutive pair of nodes is a separation pair. Therefore any parallel algorithm that is required to list all the pairs must have a processor-time product of $\Omega(n^2)$. But such a list is not necessary when finding triconnected components. The output of our algorithm is a set of paths, where a pair of non-neighboring vertices of G is a separation pair if both vertices appear on a path.

The set of separation pairs can be found by n applications of parallel biconnectivity algorithm of Tarjan and Vishkin [TV 84] as follows. Notice that if w is an articulation point in the subgraph $V(G) - \{v\}$ then $\{v, w\}$ is a separation pair in G. Therefore the set of separation pairs can be found in $O(\log n)$ time using $O(mn)$ processors. The number of processors used by our algorithm is linear in the size of the edge set $O(m)$, and the algorithm runs in $O(\log n)$ time.

The paper is organized as follows. The next section contains definitions and some preliminary results. Section 3 contains two transformations that preserve separation pairs. The first transformation results in what we call a *reach* graph of ears and the second planarizes the reach graph. The last section establishes the complexity bounds of the algorithm.

A major part of the work reported here was developed prior to the announcement by Miller and Ramachandran [MR 87] of their parallel algorithm for triconnected components. The reach graphs described in section 3.1 are similar to the bridge graphs of ears. The graphs resulting from planarizing the reach graphs P_i and the star graphs of [MR 87] are the same.

The techniques we use differ significantly from the divide-and-conquer approach used by Miller and Ramachandran. The novelty in our approach comes from the extensive use of a method we refer to as "local replacement" to obtain an efficient parallel reduction of this problem to another which is solvable with known efficient techniques. The method of local replacement involves replacing each vertex v of the original graph by a sparse graph G_v. The sparse graph is such that $|V(G_v)|$ =degree(v). The technique of local replacement is of independent interest, and we have found it to be of use in other parallel graph algorithms as well [FT 87].

2 Preliminaries

Let $V(G)$ and $E(G)$ stand for the vertex set and the edge set of a given graph G respectively. Sometimes $V(G)$ and $E(G)$ are written as V_G and E_G respectively. The subgraph G' *induced* by a subset $V'(G)$ of the vertex set $V(G)$ is a graph whose vertex set is $V'(G)$ and the edge set $E(G')$ is a subset of $E(G)$ such that the edges in $E(G')$ are those with both end nodes in $V'(G)$. A vertex in a connected graph is an *articulation point* if removal of it results in disconnecting the graph. A connected graph G is *r-connected* if at least r vertices must be removed to disconnect the graph. A *biconnected graph (or a block)* is 2-connected. A *ear decomposition starting with a node* P_0 of an undirected graph G is $G = P_0 \cup P_1 \cup ... \cup P_k$ where P_{i+1} is a path whose end nodes belong to $P_0 \cup P_1 \cup ... \cup P_i$ but its internal nodes do not. Also the edge set of the graph $E_G = E_{P_0} \cup E_{P_1} \cup ... \cup E_{P_k}$ and the vertex set $V_G = V_{P_0} \cup V_{P_1} \cup ... \cup V_{P_k}$. Observe that k is the number of edge disjoint simple paths in a connected graph G, therefore $k = |E_G| - n + 1$. Each of the paths P_i is an *ear.* If the two end nodes of a path P_i do not coincide then P_i is called an *open ear*, otherwise the ear is *closed.* In *open ear decomposition* every ear P_i, $1 < i \leq k$, is open. Let an ear be a *short ear* if it consists of a single edge, otherwise call it a *long ear.* For any $i \leq k$, the subgraph induced by the ears $E_0, ..., E_i$ of an open ear decomposition is biconnected for a given biconnected graph [Wh 32]. A pair of vertices $\{x, y\}$ of a biconnected graph is a *separation pair* if the number of components of the subgraph induced by $V(G) - \{x, y\}$ is more than one. A vertex is an *internal* node of an ear if it is neither the first nor the last node of that ear. Notice that, except for the root, each node is an internal node of exactly one ear.

Remark: It is assumed that $r \notin \{x, y\}$ throughout this paper. 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 resource bounds claimed.

Lemma 1 *If $\{x, y\}$ is a separation pair of a graph G then there exists a long ear E_l in any open ear decomposition of G such that $\{x, y\}$ is a pair of non-consecutive nodes on E_l.*

Proof : Associate with each vertex $v \neq r$ the ear E_v, where v is an internal node of E_v. The fact that each vertex has a unique ear for which it is an internal node follows from the definition of an ear.

If $\{x, y\}$ is a separation pair, then the removal of these two vertices results in disconnecting the graph into more than one component. Let the connected components induced by $V(G)-\{x,y\}$ be $C_1, C_2, \ldots, C_k$, for some $k \geq 2$. Assume, without loss of generality, that the root $r \in V(C_1)$. See Fig 1.

Let E_l be the minimum of the ear numbers associated with the vertices of any C_i, for $i \neq 1$. E_l cannot start and end at x (or at y) since that would imply E_l is a closed ear attached to x (resp. y) and the only closed ear of a biconnected graph is the one that is attached to the root r. The end nodes of E_l should belong to an ear (or two ears) with smaller ear number (resp. numbers). Since x and y are the only two vertices through which an ear with a ear number smaller than E_l can be accessed from C_i, we conclude that $\{x, y\} \subset V(E_l)$.

x and y are not neighboring nodes on E_l since there is at least one vertex v where $ear(v) = E_l$ and $v \in V(C_i)$ that is between x and y on E_l. For the same reason E_l is a long ear. □

The following definition labels the vertices depending on the ear they belong to and the position of the vertex on that ear.

Definition 1 Starting with the end nodes, if x and y are the end nodes of E_l, pick one of the end nodes p arbitrarily, $p \in \{x, y\}$, and define $pos(p, E_l)$ to be zero. For rest of the nodes in $V(E_l)$, $pos(v, E_l)$ is the distance from p to v on E_l. The value of $pos(x, E_l)$, for $x \notin V(E_l)$ is undefined. If v is an internal node of E_l then it is denoted by $pos(v)$; since the second argument is unique it can be omitted.

Definition 2 Given an internal node v of E_l, define

$$reach(v) = \begin{cases} [-m, m] & \text{if } \exists \text{ path } P : v \rightarrow r \\ & \text{and } (V(P) - \{v\}) \cap \\ & V(E_l) = \emptyset \\ [left_v, right_v] & \text{otherwise} \end{cases}$$

where $left_v$ ($right_v$) is minimum (resp. maximum) over the set

$$\{pos(v', E_l) | \exists \text{ path } P : v \rightarrow v' \text{ and } (V(P) - \{v, v'\}) \cap V(E_l) = \emptyset\}$$

and m is any integer larger than the number of vertices in the graph. We will assume $m = 2n$ throughout.

We will refer to the first component of the interval as $reach(v).1$ and the second as $reach(v).2$.

Theorem 1 *$\{x, y\}$ is a separation pair iff $\exists$ a long ear E_l containing x and y such that x and y are not consecutive nodes on E_l and all nodes v between x and y on E_l satisfy $reach(v) \subseteq [pos(x, E_l), pos(y, E_l)]$ assuming $pos(x, E_l) < pos(y, E_l)$.*

Proof : **(only if)** Let the removal of the vertices x and y result in disconnecting the graph into connected components $C_1, C_2, \ldots, C_k$, for some $k \geq 2$. Assume without loss of generality that the root node r of the ear decomposition belongs to the vertex set of C_1. Consider any C_i, for $i \neq 1$. Let $E_l = \min \{ear(v) | v \in V(C_i)\}$. As proved in lemma 1, x and y are contained in $V(E_l)$.

Assume, for contradiction, that this part of the theorem is not true. That would imply, for all ears containing x and y, specifically E_l, there exists a vertex v such that

$$(pos(x, E_l) > reach(v).1) \text{ or } (pos(y, E_l) < reach(v).2)$$

In either case, we will exhibit a path from v to r that does not use x or y, thus implying $v \in V(C_1)$. As we assumed $v \in V(C_i)$, for some $C_i \neq C_1$, we would have a contradiction.

If $pos(x, E_l) > reach(v).1$ and $pos(x, E_l) = 0$ then $reach(v).1$ has to be $-2n$ as it is the only number less than 0 in our numbering. Therefore $reach(v) = [-2n, 2n]$ implying that there is a path from v to r that does not use any vertices of E_l, specifically x and y. Therefore assume $pos(x, E_l) \neq 0$ and $0 \leq reach(v).1 < pos(x, E_l)$. This implies there is a path P from v to v_1, where $pos(v_1, E_l) < pos(x, E_l)$. Let v_2 be the vertex with $pos(v_2, E_l) = 0$. From v_1 extend P first by taking edges of E_l to vertex v_2. To extend P further to r consider the block B induced by the ears $P_0, \ldots, ear(v_2)$. B cannot contain x but can contain y. But since there are two vertex-disjoint paths from v_2 to r in B it is always possible to extend P to r

without having to use y. Therefore there is a path from v to r that does not use either x or y.

The argument when $pos(y, E_l) < reach(v).2$ is similar.

(if) Since x and y are not neighbors on E_l, there exists at least one vertex between x and y on E_l. If the segment of the path E_l that is between x and y does not get disconnected from the component containing r, then there is a path from the root to a vertex v of the segment that avoids both x and y. That means

$$(pos(x, E_l) > reach(v).1) \text{ or } (pos(y, E_l) < reach(v).2),$$

a contradiction. □

3 An Algorithm for Separation pairs

The point of characterizing separation pairs in terms of the *reach* labeling is to be able to generate separation pairs efficiently. Let G_r be the graph obtained by adding to two edges $(v, reach(v).1)$ and $(v, reach(v).2)$ to $E(G)$ for each vertex $v \in V(G)$, for $reach(v) \neq [-2n, 2n]$, and by deleting all short ears from $E(G)$. In the case when $reach(v) = [-2n, 2n]$ add single edge (v, r). Define G_r to be the *reach graph of* G. It follows from Theorem 1 that the set of separation pairs of G_r and G is identical since the reach labeling of their respective vertices is the same.

The next subsection shows that an approximation of the *reach* labeling can be computed efficiently by making use of the parallel biconnectivity algorithm of Tarjan and Vishkin [TV 84]. The fact that the approximation suggested suffices when extracting separation pairs is shown in Theorem 3 in the next subsection.

Even though G_r is structurally simpler than G, a further simplification appears to be necessary to generate separation pairs rapidly in parallel.

Define R_i to be the *reach graph of* E_i as follows. The graph R_i is the same as E_i with the following additions. Create a new vertex r_i and add an edge (v, r_i) if $reach(v) = [-2n, 2n]$ or v is an end node of E_i; otherwise add two edges $(v, reach(v).1)$ and $(v, reach(v).2)$ to $E(R_i)$. Section 3.2 includes the definition of the planar versions P_i of R_i and a procedure that finds the P_i by a reduction to the connected component algorithm of Shiloach and Vishkin [SV 81]. See Fig 3. A local modification is suggested at the end of Section 3.2 that yields a set of paths where each nonconsecutive pair of vertices on each path is a separation pair.

3.1 Computing the *Reach* Labeling

This consists of two stages. In the first stage we define an approximation *reach′* and show how to compute it efficiently. In the second we show how to make local modifications of the original graph so as to offset the error introduced by the approximation. The graph G' that results after the local modifications has the following desirable property: corresponding to each separation pair $\{x, y\}$ and the ear E_l (as defined in Theorem 1), the *reach′* labeling of E_l in G' is the *reach* labeling of E_l of the original graph G.

Define the $reach'(v)$ labeling in the same way as $reach(v)$ but with the weaker restriction: P should not use the internal nodes of $ear(v)$ but is allowed to use the end nodes of $ear(v)$. Notice that if each vertex of G is an end node of at most one ear then *reach* and *reach′* are identical. Moreover, for each vertex v, $reach(v) \subseteq reach'(v)$. Moreover, if $reach(v) \neq reach'(v)$ then $reach'(v) = [-2n, 2n]$.

In the following we show how to label the vertices with the *reach′* labeling efficiently.

Consider the auxiliary multigraph G_e constructed from the given graph G as follows. Let E_v be a long ear $v_0, v_1, \ldots, v_k$ of length greater than two, i.e. $k \geq 3$. Now, G_e is obtained from G by contracting all such ears E_v by merging the internal vertices $v_1, v_2, ..., v_{k-1}$. For all v, $v \in V(G_e)$, label v with the ear number it represents. See Fig 2.

Let $B_1, B_2, ..., B_l$ be the biconnected components of G_e. Let the root node belong to B_1. For $1 \leq i \leq l$, let v_{a_i} be the node with minimum node label in $V(B_i)$. (Recall that the node labels in G_e correspond to the ear labels of long ears of G.) Define $B_2', ..., B_l'$ to be the subgraphs of $B_2, ..., B_l$ respectively, where for $2 \leq i \leq l$, B_i' is the subgraph of B_i induced by the vertex set $V(B_i) - \{v_{a_i}\}$. It is easily verified that the $V(B_i')$s define a partition over $V(G_e)$.

Consider a labeling α of the B_i's. This labeling will help us find the *reach′* values of a node v locally, that is by looking at the neighbors of v.

For all $v \in V(B_i')$, $\alpha(v) = \langle v_{a_i}, x, y \rangle$ where x (y) is minimum (resp. maximum) over the set

$\{pos(s) | (s,t) \in E(G), s$ and t correspond to v_{a_i} and a vertex in B_i' resp.$\}$

Extend the α labeling of the vertices of G_e to the vertices $V(G) - \{r\}$ as follows. Each v is an internal node of exactly one long ear E_x and the node corresponding to E_x belongs to exactly one B_i'.[1] For $v \in V(G)$, $\alpha(v)$ is the α label of the node in G_e that corresponds to the $ear(v)$. From the definition of an ear it follows that for all $v \in V(G)$, $\alpha(v).1 < ear(v)$.

Before we relate α and $reach'$ labelings, we need to prove

Lemma 2 *If $(v,w) \in E(G)$ then $\alpha(w).1 \leq ear(v)$.*

Proof : If $ear(w) \leq ear(v)$ then $\alpha(w).1 < ear(v)$ since $\alpha(w).1 < ear(w)$. Assume $ear(w) > ear(v)$. If there exists a path from w to r that avoids all the internal nodes of $ear(v)$ then both $ear(v)$ and $ear(w)$ belong to a biconnected component B_i in G_e, and $\alpha(v).1 = \alpha(w).1 = v_{a_i}$. So, $\alpha(w).1 < ear(v)$ since $\alpha(v).1 < ear(v)$. If all the paths from w to r have to use some internal nodes of $ear(v)$, then $ear(v)$ would be an articulation point in G_e , to which the biconnected component containing $ear(w)$ is attached. Therefore, $\alpha(w).1 = ear(v)$. □

Definition 3 For each vertex $v \in V(G)$, define a set of edges *escape-edges(v)* as

$\{(v,w) | (v,w) \in E(G) - E(ear(v))$ and $ear(v) \neq ear(w)\}$

See Fig 4.

Theorem 2 For a vertex v, $v \neq r$,

(i) $reach'(v) = [-2n, 2n]$ if and only if there exists $(v,w) \in$ escape-edges(v) and $\alpha(w).1 < ear(v)$.

(ii) if $reach'(v) \neq [-2n, 2n]$ then $reach'(v).1$ ($reach'(v).2$) is the minimum (resp. maximum) over the union of the following three sets

(a) $\{pos(v)\}$

(b) $\{pos(w) | \exists(v,w) \in E(G) - E(ear(v))$ and $ear(w) = ear(v)\}$

(c)$\{\alpha(w).2 (resp.\ \alpha(w).3) | \exists(v,w) \in escape\text{-}edges(v)\}$

Proof : (i) (only if) $reach'(v) = [-2n, 2n]$ implies that there is a path P from v to r in G that avoids all the internal nodes of $ear(v)$. Let the first two vertices of P be v and w. From the definition of $reach'(v)$, $ear(v) \neq ear(w)$. So, we have the following two cases to consider.

Case (a): $ear(w) < ear(v)$

For any vertex v_x, $\alpha(v_x).1 < ear(v_x)$ from Lemma 2. In particular for $v_x = w$. Therefore $\alpha(w).1 < ear(w) < ear(v)$.

Case (b): $ear(w) > ear(v)$

Corresponding to each long ear E_x of G there is a node in G_e (call it e_x). We will show that e_v and e_w belong to the same biconnected component B in G_e and B is attached to an articulation point e_y, where $ear(y) < ear(v) < ear(w)$. Since $\alpha(e_w).1$ would be e_y, it would follow that $\alpha(w).1 < ear(v)$.

Consider two paths P_1 and P_2 in G_e from e_v to e_0. Let P_1 be $e_v e_{v_1} e_{v_2} ... e_0$ where $e_v > e_{v_1} > e_{v_2} ... > e_0$. Such a path is guaranteed to exist given the definition of ear decomposition. Let P_2 be $e_v e_w ... e_0$, the path in G_e that corresponds to the path P that we assumed to exist in G. Let e_s be the first vertex after e_v common to P_1 and P_2, in the order they occur on the paths. Since node labels on P_1 decrease monotonically and as $e_v \neq e_s$, we conclude that $e_s < e_v < e_w$. Let B be the block that contains e_s, e_v and e_w. Then for all $e_l \in V(B)$, $\alpha(e_l).1 \leq e_s$. So $\alpha(e_w).1 \leq e_s < e_v$. Therefore $\alpha(w).1 < ear(v)$.

(if) Given that $\alpha(w).1 < ear(v)$ and that there exists an edge $(v,w) \in E(G) - E(ear(v))$, $ear(v) \neq ear(w)$, we need to prove that r can be reached from v without using any internal nodes of $ear(v)$. Notice that e_v and e_w belong to the same block in G_e because of the edge (v,w). And $\alpha(w).1$ is the articulation point in G_e to which B is attached. $\alpha(w).1 \neq ear(v)$ since $\alpha(w).1 < ear(v)$. Therefore there are two vertex-disjoint paths from e_w to $\alpha(w).1$. Assume e_w and e_v are in the same biconnected component in

[1]This statement would be incorrect if B_i' is replaced by B_i because if the node corresponding to E_x were an articulation point in G_e then it would be split and therefore present in more than one component.

G_e. Clearly there exists a path from e_w to $\alpha(w).1$ that does not contain e_v. From $\alpha(w).1$ take a path with monotonically decreasing node labels to e_0. So, we can always a find a path in G_e, from e_w to e_0 that avoids e_v. Hence, by taking a corresponding path in G, the root r can be reached without using any internal nodes of $ear(v)$.

(ii) From (i) we know that $reach'(v) \neq [-2n, 2n]$ implies that for all $(v, w) \in$ *escape-edges*(v), $\alpha(w).1 \geq ear(v)$. But from lemma 2 we know that $\alpha(w).1 \leq ear(v)$. Therefore, for all such w, $\alpha(w).1 = ear(v)$. This means that all the paths starting with an escape-edge lead back to an internal node of $ear(v)$.

The result directly follows from the definition of $reach'(v)$, $\alpha(v).2$ and $\alpha(v).3$. □

Since we do not how to find the *reach* labeling, we need a theorem similar to theorem 1 with *reach* replaced by *reach'*. To achieve this goal, we propose the following local modifications to G resulting in G'.

Definition 4 Let two ears be *parallel ears* if they have the same pair of end nodes. Let a *group of parallel ears* be parallel ears that are in the same connected component when the two end vertices are removed. See Fig 5.

Algorithm *Build G'*

1. Identify each ear as either a single ear or a parallel ear. Also mark the ear with the minimum ear label in a group of parallel ears.
2. Partition the edge set into a set of paths induced by ears by "splitting" at the end nodes.
3. Let ear E_x be $v, v_1, ..., v_k, w$. Rename each vertex s of E_x by s_{E_x}.
4. "Glue" these paths, to get G', as described below.
   (a) Let E_1 be the first long ear. Pick one of the two end edges of E_1 arbitrarily and give it a direction. Orient rest of E_1 in the opposite direction.
   (b) Give directions to the remaining ears of G so that the resulting digraph G_d is acyclic.
   (c) Remove the last edge in each directed ear in G_d to obtain a directed spanning tree T_d.
   (d) If E_x is a single ear or a parallel ear with minimum ear label in that group then do the following.
   Let (v_k, w) be the edge that was removed in step 3c for the ear E_x. If $lca(v_k, w) = v$ in T_d then let E_y be the ear number of the first edge in the tree path from v to w. Add an edge between v_{E_x} and v_{E_y}. If $lca(v_k, w) \neq v$ in T_d then add an edge between v_{E_x} and v_{E_y}, where v is an internal node of E_y in G.
   (e) If E_x is a parallel ear and E_x is not the minimum ear label in that group then do the following. Connect the end node v_{E_x} to v_{E_w} where E_w comes after E_x in the descending order of ear labels in that group.
   (f) Reverse the direction of edges of G_d and repeat steps 3c and 3d.

Fig 7 illustrates the local modifications suggested by the above algorithm. Before we prove that *reach'* labeling of G' is sufficient to find separation pairs we need the following definitions.

Definition 5 If a vertex v is not an end node of any ear in G then it retains the same vertex label in G' and we say v of G and v of G' *correspond* to each other. Otherwise vertex v of G and v_{E_l} of G' *correspond* to each other if the latter is obtained from the former by splitting in step 1.

We also say a path (ear) of G *corresponds* to a path (ear) of G' if their vertices correspond to each other.

Definition 6 An ear E_x *dominates* another ear E_y if no vertex of E_y belongs to the component containing the root r in $V(G) - V(E_x)$. See Fig 6.

Lemma 3 *Let E_x dominate $E_{x_1}, E_{x_2}, ..., E_{x_k}$, for $x < x_1 < ... < x_k$. Let v be the end node of E_x, where the edge labeled E_x that is incident on v is in T_d. The subtree of T_d rooted at v contains all the vertices of the ears $E_{x_1}, E_{x_2}, ..., E_{x_k}$.*

Proof: Let the end nodes of E_x be v and w. The proof is by induction on k. For $k = 1$ notice that the end nodes of E_{x_1} are also in E_x. If both the end nodes of E_{x_1} are internal nodes of E_x then the lemma is clearly true. If one of the end nodes of E_{x_1} is v then lemma holds irrespective of whether E_{x_1} is attached in T_d by v or by the other end node. Assume one of the end nodes of E_{x_1} is w. If E_{x_1} is attached in T_d by the end node other than w then the lemma holds. So assume ear E_{x_1} is attached in T_d by w. But that means E_{x_1} together with the segment of E_x that is in between the end nodes of E_{x_1} forms a cycle contradicting that G_d is acyclic.

Let the lemma be true for $k = i - 1$. Consider the possible nodes of attachment of E_{x_i} in T_d. If the end nodes of E_{x_i} are either internal nodes of E_x or nodes in the vertex set of $E_{x_1}, ..., E_{x_{i-1}}$ then clearly the lemma holds. So assume one of the end nodes of E_{x_i} is an end node of E_x. The case when the vertex of attachment is v or one of the nodes in the vertex set of $E_{x_1}, ..., E_{x_{i-1}}$ is straightforward. Assume E_{x_i} is attached to w in T_d. The end node of E_{x_i} other than w is a vertex x in the subtree rooted at v by induction hypothesis. By taking the ear E_f to which x belongs to and the ear to which the end node of E_f belongs to and so on, the vertex w can be reached implying that G_d is not acyclic. $\square$

Lemma 4 *If E_x dominates E_y and if both have the same end-vertex v then both ears are either outgoing or incoming at v in G_d.*

Proof: Consider T_d where the edge with the ear label E_x that is incident on v is in the tree. If E_x and E_y have different directions at v then G_d has a cycle consisting of E_y and the tree path from v to the end node of E_y other than v. Such a tree path is guaranteed to exist by the previous lemma. $\square$

Assume for two ears E_x and E_y, $E_x < E_y$ and that if they are parallel then they are in the same group. Let v be an end node of E_x in G. Let T'_d be the spanning tree of G' corresponding to T_d in which the ear E_x is attached by a vertex corresponding to v.

Lemma 5 *If E_x dominates E_y then the subtree of T'_d rooted at v_{E_x} contains all the vertices of ear E_y.*

Proof: If E_x and E_y are parallel then clearly the end node of E_y is the descendant of v_{E_x} since the end nodes of parallel ears in a group appear in sorted order. So assume E_x and E_y are not parallel. If they do not share an end node then this lemma follows from Lemma 3. If they do then it is easy to complete the rest of the proof using Lemma 4 and induction. $\square$

Let the end nodes of E_y (E_x) be v and w_1 (resp. w_2). Assume $E_x < E_y$ and T_d is the spanning tree in which both E_x and E_y are attached to v.

Lemma 6 *If the fundamental cycle created by including the end edge of E_y that is incident at w_1 contains the ears E_x and E_y then for every vertex $w \neq v$ on the cycle we have $ear(w) \geq E_x$.*

Proof: Consider the following order on the fundamental cycle. Start at v and trace the cycle in the direction starting with edges of E_y. Notice that once a vertex x, where $ear(x) = E_x$ is reached then all vertices that come after x should also belong to E_x.Assume there is a vertex p for some $ear(p) < E_x$. We will show that this assumption leads to a contradiction. From the spanning tree construction, described in the algorithm, it is clear that if a vertex that is on $ear(p)$ can be reached then one of its end nodes q for which $ear(q) < ear(p)$ can be reached by going up the tree. This process can be repeated until a vertex that was used is encountered. Since the fundamental cycle is always a simple cycle the encountered vertex has to be v. But that implies that the cycle constructed does not contain any edges labeled E_x, a contradiction. $\square$

Remark: Notice that if $w_1 = w_2$, that is E_x and E_y are parallel then the lemma is vacuously true since the fundamental cycle never contains edges of two parallel ears.

Let $\{x, y\}$ be a separation pair. From Theorem 1 we know that there exists a long ear E_l containing two nonconsecutive nodes x and y with $pos(x, E_l) < pos(y, E_l)$.

Theorem 3 *$\{x, y\}$ is a separation pair in G iff for all nodes v that are between x_{E_l} and y_{E_l} on E_l in G' we have $reach'(v) \subseteq [pos(x_{E_l}, E_l), pos(y_{E_l}, E_l)]$.*

Proof: The local modifications suggested in the algorithm can be viewed as "local expansion" of the nodes. Clearly if there is no path between two nodes that avoids the internal nodes of an ear then there is no path that avoids the nodes of that ear in G. Therefore the reverse implication is trivial. The following argument establishes the forward implication.

Let E be the set of ears both of whose end nodes are in $C \cup \{x, y\}$ where C is the connected component containing the segment of E_l that is between x and y in $V(G) - \{x, y\}$.

Let C' be the connected component containing the segment of E_l that is between x_{E_l} and y_{E_l} in $V(G') - \{x_{E_l}, y_{E_l}\}$. Let E' be the set of ears in C'. We claim that the ears of E correspond in a one-to-one manner with the ears of E' and vice-versa. Let C'_E be the part of the connected component corresponding to the ears in E. We will show that $C' = C'_E$.

Assume $C' \neq C'_E$ and let $E_n \in C' - C'_E$. Notice that there cannot be an edge from a vertex that does not correspond to either x or y to a vertex of E_n as it would imply that the corresponding edge is present in $V(G) - \{x, y\}$ and hence E_n would be in E.

We will first establish that E_l is the minimum ear number in C'. Notice that it is sufficient if we prove for each split vertex x_{E_n} of x that if x_{E_n} is in C' then $E_n > E_l$. If there were a vertex x_{E_n} with $E_n < E_l$ then consider the ear label E_q of the edge (x_{E_n}, x_{E_q}) that connects x_{E_n} to one of the vertices of C'_E. Notice that this edge has to be either in the spanning tree T'_d or the tree obtained by reversing the directions. Assume without loss of generality that it is in T'_d. That means the subtree of T'_d rooted at x_{E_l} does not contain the ear E_q, contradicting Lemma 5.

Assume E_n greater than the maximum ear label in C'_E. Also assume, without loss of generality, that E_n is connected directly to the component C'_E.But if the only edges of attachment of E_n are two edges connected to x_{E_l} and y_{E_l} then E_n would get disconnected when these two vertices are removed. Therefore assume without loss of generality that x_{E_m} is a vertex of attachment of E_n to C'_E for some $E_l < E_m < E_n$. If E_m and E_n are parallel then they are in the same group and therefore there is a path in G from an internal node of E_m to E_n that does not use either x or y. This means $E_n \in E$. If E_m and E_n are not parallel, then from the previous lemma, there is a path in G from an internal node of E_m which uses vertices whose ear label is greater than or equal to E_m. But as $E_n > E_m$ neither x nor y can occur on this path hence $E_n \in E$. Therefore such an E_n cannot exist. □

3.2 Planar Versions of Reach Graphs of Ears

The reach graphs R_i for the ears E_i are defined at the beginning of Section 3. Refer to the edges in $E(R_i) - E(E_i)$ as *arcs*. Define a relation on $V(E_i)$ as follows. Two vertices x and y are related if there is an arc between them or there exists a pair of arcs (x, u), (y, v) with $pos(x) < pos(y) < pos(u) < pos(v)$. Extend the relation to include r_i as follows. Define x to be related r_i if there is edge $(x, r_i) \in E(R_i)$ or there is an arc (x, y) and (u, r_i) such that $pos(x) < pos(u) < pos(y)$. Now consider the partition $p_1, p_2, ..., p_k$ induced by this relation. The planar version P_i of R_i is obtained as shown below. The vertex set $V(P_i)$ is the union of $V(E_i)$, $\{r_i\}$ and $\{v_{p_i} \mid p_i$ is a partition that does not include $r_i\}$. The edge set $E(P_i)$ is the union of $E(E_i)$,$\{(x, v_{p_i}) \mid x$ belongs to the partition $p_i\}$ and $\{(x, r_i) \mid x$ and r_i are in the same partition $\}$.

The following algorithm builds P_i using the connected component algorithm [SV 81]. See Fig 3.

Algorithm *Build P_i*

1. For each arc (x, y), identify two arcs (a, b) and (c, d) where $(a, b) = \max \{ pos(f) \mid (e, f)$ is an arc with $pos(x) < pos(e) < pos(y)\}$. Define (c, d) similarly with max replaced by min.
2. Replace each arc (x, y) with two edges $(x, v_{<x,y>})$, $(v_{<x,y>}, y)$. Refer to $v_{<x,y>}$ as an *arc vertex*.
3. If (a, b) is an edge identified with the arc (x, y) in the first step then add an edge $(v_{<x,y>}, v_{<a,b>})$.
4. Assume p_i is a connected component in the subgraph induced by the arc vertices after the Step 3. For all i, merge all the arc vertices of p_i into a single vertex v_{p_i} using the connected component algorithm.

Refer to the vertices resulting from the Step 4 of the previous algorithm also as *arc vertices*. For each arc vertex v_{p_i}, define $left(v_{p_i})$ ($right(v_{p_i})$) to be the *pos* value of the vertex on E_i that is connected to v_{p_i} and that has a minimum (resp. maximum) *pos* value. Call the edges connecting v_{p_i} to its *left* and *right* vertices in P_i, *extreme edges*. It is easy to see that each vertex of E_i can have at most one non-extreme edge incident on it in P_i.

Theorem 4 *The equivalence classes defined in the beginning of this subsection and the equivalence classes produced by the above algorithm are identical.*

Proof: It is required to show that it suffices to identify the arcs (a, b) and (c, d) for each (x, y), where these edges are as defined in Step 1 of the algorithm. Define the *right-max*(x,y) to be the arc (a,b). Similarly the *left-max*(x,y) to be the arc (c,d). Consider the arcs (x,y) and (u,v) where $(u,v) \neq (a,b)$, and $pos(a) < pos(u) < pos(b) < pos(v)$. Let left-max$(u,v) = (e,f)$ and right-max$(x,y) = (g,h)$. Assume $pos(e) < pos(f)$ and $pos(g) < pos(h)$. If e,f,g and h are in the same partition then a,b,u and v also belong to the same partition. Otherwise we have two arcs $(e,f),(g,h)$ where $(pos(h)-pos(e)) > (pos(v)-pos(x))$. Repeat the argument if e,f,g and h are not in the same partition. Every application of the argument results in a new pair of arcs for which the distance between the vertices at the extreme is strictly greater than the corresponding distance for the old pair of arcs. As the length of E_i is finite we conclude that a,b,u and v are in the same partition. □

Theorem 5 *The graphs P_i resulting from the above construction are planar.*

Proof: We claim that if P_i is not planar then there exists a pair of arc vertices x,y such that subgraph of P_i induced by $V(E_i) \cup \{x,y\}$ is not planar. Assume the claim is not true. Let $x_1, x_2, ..., x_k$ be arc vertices. Now assume that there is an arc vertex z such that the subgraphs $V(E_i) \cup \{x_1, x_2, ..., x_k\}$ and $V(E_i) \cup \{z, x_h\}, 1 \leq h \leq k$ are planar but the subgraph $V(E_i) \cup \{z, x_1, x_2, ..., x_k\}$ is not. Consider the following embeddings of $V(E_i) \cup \{z, x_h\}$. Assume the graph E_i is represented as a horizontal line and the vertices z and x_h and the edges incident on z and x_h appear above it. We say x_h is *inside* z if x_h is inside the closed curve consisting of z and part of E_i that is between $left(z)$ and $right(z)$. Otherwise we say x_h is *outside* z. Applying an inductive argument we can conclude the following. The node z can be "inserted" such that the nodes that are inside z would remain inside and the nodes outside z would remain outside without destroying the planarity of $V(E_i) \cup \{z, x_1, x_2, ..., x_k\}$. The claim follows since we assumed otherwise.

Notice that for a given arc vertex x and a vertex $w \in V(E_i)$, $left(x) < pos(w) < right(x)$, there exists an arc $(u,v) \in E(R_i)$ such that $pos(w) < pos(v) < pos(v)$. This follows from the relation defined at the beginning of the section. That is if there is no arc that connects the different sides of w then $left(x)$ and $right(x)$ cannot be in the same partition.

Assume the theorem is not true. Let x,y be two arc vertices such that the subgraph of P_i induced by $V(E_i) \cup \{x,y\}$ is not planar. We claim that there are two arcs (a,b) and (c,d) , $pos(a) < pos(c) < pos(b) < pos(d)$, where a,b (c,d) are in the same partition as x (resp. y). To prove the claim assume (a,b) is the arc obtained by applying the argument from the previous paragraph where the arc vertex is x and w is the vertex whose pos value is $left(y)$. The second arc is obtained by again applying the same argument but now with the arc vertex y and $w = b$. But, from the way the relation is defined, a,b,c and d are in the same partition. Therefore $x = y$ and P_i is planar as there is only one arc vertex. A contradiction. □

Theorem 6 *Assuming $\{x,y\} \in V(G)$ the following statements are equivalent:*

(i) $\{x,y\}$ is a separation pair in G.

(ii) $\{x,y\}$ is a separation pair in P_i.

(iii) $\{x,y\}$ lie on a bounded face in P_i in the embedding given in the proof of the previous theorem.

Proof: Note that G and its reach graph G_r have an identical set of separation pairs as mentioned at the beginning of Section 3. If $\{x,y\}$ is a separation pair G then there is arc that connects the segment S that is between x and y to the rest of the ear E_i in G_r. Therefore all the arc vertices that have an edge that connects to a vertex of S should also have their extreme edges with in the segment between x and y. Therefore $\{x,y\}$ is a separation pair in P_i. Conversely if $\{x,y\}$ is not a separation pair then there is an arc that connects the segment between x and y to the rest of the ear E_i . Therefore there is an arc vertex with at least two edges: one connecting a vertex from the segment, and the other connecting a vertex outside the segment. Therefore the first two statements are equivalent.

The equivalence between the last two statements follows in a straightforward manner from the construction given before. □

The following algorithm shows how to obtain separation pairs from the P_i. Basically, starting with E_i each node is expanded linearly by considering the edges incident on the corresponding vertex in P_i. This expanded path is then broken off at places where the vertices from the opposite sides cannot form a separation pair.

Algorithm *Generate-separation-pairs*

For a vertex $v \in V(E_i)$, let $x_1, x_2, ..., x_k$ be v's neighbors in P_i. Assume $x_1, x_k \in V(E_i)$. Assume $left(x_i)$, for $1 < i < k$, is in sorted order. In addition, for all j, if $left(x_j) = v$ then assume the relative ordering of x_j's is with respect to their $right(x_j)$ in decreasing order. Assume $Pred(v)$ and $Succ(v)$ refer to the vertex before and the vertex after v on the line segment under consideration.

1. If there is a non-extreme edge (x, x_j) incident on x in P_i then replace x in E_i by the following two chains

$$v_{x_2} - v_{x_3} - \ldots - v_{x_j},$$
$$v'_{x_j} - v_{x_{j+1}} - \ldots - v_{x_{k-1}}.$$

   If all the edges incident on x are extreme in P_i then replace x in E_i by the following chain

$$v_{x_2} - v_{x_3} - \ldots - v_{x_j} - v_{x_c} - v_{x_{j+1}} \cdots\ v_{x_{k-1}}$$

   where, for all $2 \leq i \leq j$, $left(x_i) < pos(v)$ and, for all $j < i < k$, $left(x_i) = pos(v)$.

2. For an arc vertex x, let p and q be vertices such that $left(x) = pos(p)$ and $right(x) = pos(q)$. For each arc vertex $x \neq r_i$, add an edge $(Pred(p_x), Succ(q_x))$ and destroy the edges $(Pred(p_x), p_x)$ and $(q_x, Succ(q_x))$.

3. For the arc vertex r_i, destroy the edges as in the previous step but add an edge only if there are no non-extreme edges incident on r_i.

4. Discard all single-vertex-paths resulting from the previous step.

Fig 8 illustrates the local modifications suggested by the above algorithm .

Theorem 7 *Two vertices x, y are on a path p produced by the above algorithm if and only if x, y is a separation pair in P_i.*

Proof: If x and y are consecutive on E_i then the theorem is trivially true since the only edges destroyed by the algorithm are the edges introduced by the chain(s) by the first step. Therefore assume they are not consecutive but $pos(x) < pos(y)$.

Consider the forward direction.

We will first establish that after the algorithm terminates there can be no edge between any two split vertices v_1 and v_2 of a vertex v. From the construction it is clear that v_1 and v_2 have to be consecutive. From the order we assumed, as explained at the beginning of the algorithm, we can conclude that only edges between split vertices are destroyed. An inspection of the algorithm reveals that there are $k-1$ split vertices and $k-2$ edges between them for each vertex of degree k. Notice that there are exactly $k-2$ edges from $E(P_i) - E(E_i)$ incident on a vertex of degree k. Since each such an edge destroys one edge in Step 2 of the above algorithm we conclude that no two split vertices of the same vertex can be connected by an edge after the algorithm terminates.

Now assume that the nodes x and y are consecutive nodes on p. Since edges are created only by arc vertices assume z is the arc vertex that created the edge between x and y in p. If this part of the theorem is not true then x and y are not on a common face. That implies there is a vertex s such that $pos(s) \in [pos(x), pos(y)]$ and $reach(s) \notin [pos(x), pos(y)]$. Since x, y and z are in the same equivalence class the edges (x, z) and (y, z) cannot both be the extreme edges of z. Therefore we have a contradiction. Now consider the case when x, z and y are consecutive on p where x and z, and z and y are on a common face but not x and y. Notice that z is connected to at least one of its split vertices. Since no two vertices among x, y and z are split vertices of the same vertex this situation cannot arise.

Conversely if x and y are not consecutive on E_i but occur consecutively on a face in P_i then there exists an arc vertex z for which (x, z) and (y, z) are left and right extreme edges respectively. Notice that the predecessor of x_z (also the successor of y_z) has to be a split vertex of x (resp. y). As Step 2 adds an edge between the predecessor of x_z and successor of y_z the nodes x and y are on a path generated by the algorithm .

4 Complexity on a CRCW Pram

Maon, Vishkin and Schieber [MSV 86] describe how to find open ear decomposition in at most $O(e \log n)$ operations. Let T be the spanning tree that was used to find open ear decomposition in the above algorithm. Then the position of vertices on an ear–*pos* labeling–can be computed using the Euler-tour technique of Tarjan and Vishkin [TV 84] on T by doubling in $O(\log n)$ time with $O(e)$ processors.

Consider the complexity of $reach'$ labeling.

The auxiliary multigraph G_e can be constructed as follows. Assume that the processor assigned to the first edge of each ear is responsible for creating the edge between the appropriate vertices of G_e. Recall that in the case of short ears there is only one edge which is also the first edge. Each processor P_i assigned to the first edge of an ear E_i finds the end nodes u and v of E_i. The processor P_i also finds the ear numbers E_j, E_k for which the pair u, v are interval nodes and the values $pos(u, E_j)$ and $pos(v, E_k)$. Assume the corresponding nodes in G_e are e_j, e_k. The processor P_i creates the edge (e_j, e_k) and labels the edge with a 2-tuple $< pos(u, E_j), pos(v, E_k)$.

The α labeling of the vertices of G_e consists of the following steps. Find the blocks of G_e by the algorithm given by Tarjan and Vishkin [TV 84]. Treat each block B_i separately and construct a spanning tree T_{b_i} in each block using the modification of the connected component algorithm of Shiloach and Vishkin [SV 81]. The articulation point v_{a_i} that represents the minimum ear label in B_i can be found by using the Euler-tour technique. The values x and y can be found by examining the 2-tuple labels of the edges incident on v_{a_i}. These values are broadcast to all the vertices in the block B_i using the Euler-tour technique again.

Extending the α labeling to the vertices of G is straightforward. Let P_{e_i} be the processor assigned to the vertex $e_i \in V(G_e)$. As mentioned in the construction of G_e, P_{e_i} can be the same processor that was assigned to the first edge of E_i in G. Now P_{e_i} can broadcast the value of $\alpha(e_i)$ to all the internal vertices of E_i using the Euler-tour technique on the spanning tree T of G.

Theorem 2 gives a relation between $reach'(v)$ and $\alpha(v)$. Therefore $reach'(v)$ can be found by finding the minimum (or maximum as appropriate) of the components of α values of the edges incident on v. This can be performed using the doubling technique by the processors assigned to the edges incident on v.

From the explanation given above it is easy to see that the $reach'$ labeling can be done in $O(\log n)$ time using $O(m)$ processors. The following argument shows that G' can be constructed within the same resource bounds.

To identify parallel ears do the following. Let $(u, v_1), (u, v_2), ..., (u, v_k)$ be the edges incident on a vertex v and let $E_{x_1}, E_{x_2}, ..., E_{x_k}$ be their ear labels respectively. Let the end nodes of $E_{x_1}, E_{x_2}, ..., E_{x_k}$ be $(u, w_1), (u, w_2), ..., (u, w_k)$ respectively. Now assume the edge list at v is in the increasing order of the pos values of $w_1, w_2, ..., w_k$. This can be achieved by any of the parallel sorting algorithms [AKS 83]. Consider an auxiliary graph G_p where the vertices correspond to the long ears of G. Now the processor P_i assigned to the first edge of each ear E_i examines the neighboring edge with ear label E_j to see if E_i and E_j have the same end nodes. If so, P_i adds an edge between the corresponding vertices in G_p. Now using the connected component algorithm of Shiloach and Vishkin on G_p, we can identify all groups of parallel ears.

Splitting and renaming can be achieved, for example, by making the node labels a 2-tuple: the first component representing the vertex label, and the second representing the property by which the graph is being split. In our case the second component is the ear label of the edge that is incident on that vertex. The edge list for the split version of the graph can be constructed by using a parallel sorting algorithm as explained in [TV 84].

In general the above approach can be adopted for the construction of any locally modified graph of a given graph.

The digraph G_d can be constructed efficiently as shown in [MSV 86]. The same paper also contains a method to compute lca values of non-tree edges in $O(\log n)$ parallel time.

The only nontrivial step in the construction of the P_i is the identification of the arcs (a, b) and (c, d) for each arc (x, y). It is easily seen that (a, b) and (c, d) can be found by doubling for each (x, y) in $O(\log n)$ time using at most $O(m)$ processors.

Finally the implementation of the algorithm that generates separation pairs involves building a locally modified graph. This can be done in within the claimed resource bounds as explained before.

References

[AKS 83] M. Ajtai, J. Komlos and E. Szemeredi, "An $O(n \log(n))$ sorting network," *Combinatorica 3:1*, 1983, pp. 1-19.

[FT 87] D. Fussell and R. Thurimella, " Finding a sparse graph that preserves biconnectivity," *submitted.*

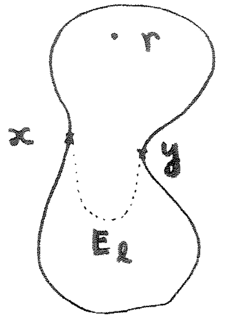
[MR 87] G.L. Miller and V. Ramachandran, "A new triconnectivity algorithm and its applications," *Proc. 19th annual STOC* , NY, May 1987, pp. 335-344.

[MSV 86] Y. Maon, B. Schieber and U. Vishkin, "Parallel ear decomposition search (EDS) and ST-numbering in graphs," *TR 46/86*, School of Mathematical sciences, Tel Aviv University, Feb. 1986.

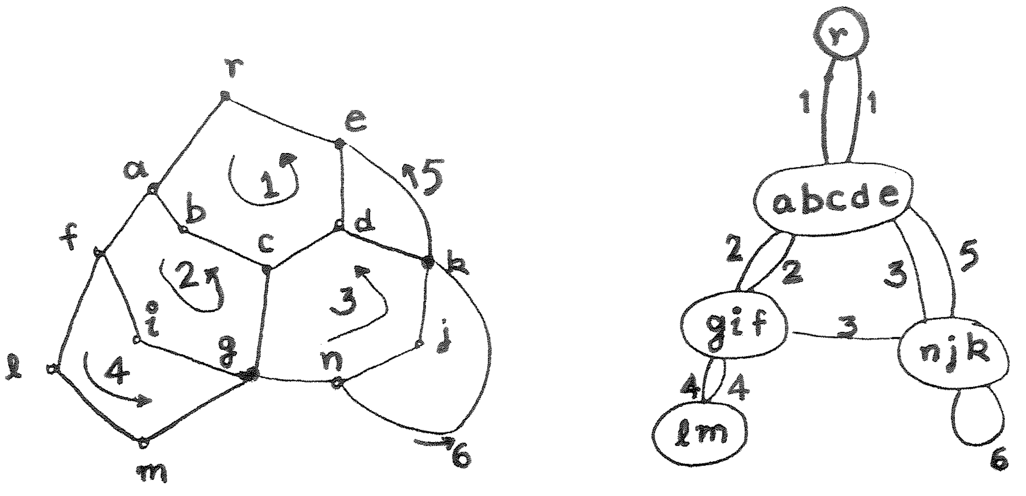
[SV 81] Y. Shiloach and U. Vishkin, "An $O(\log n)$ parallel connectivity algorithm," *J. Algorithms 2*, (1981), pp. 57–63.

[TV 84] R. E. Tarjan and U. Vishkin, "An efficient parallel biconnectivity algorithm," *SIAM J. Computing*, 14, (1984), pp. 862-874.

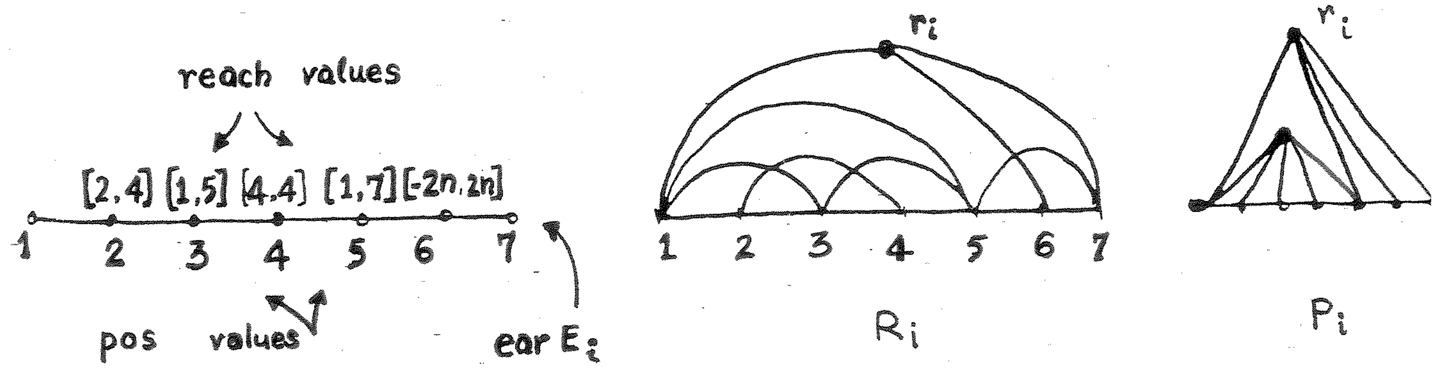
[Wh 32] H. Whitney, "Non-separable and planar graphs," *Trans. Amer. Math. Soc.,* 34 (1932), pp 339-362.



1. E_1 contains x,y

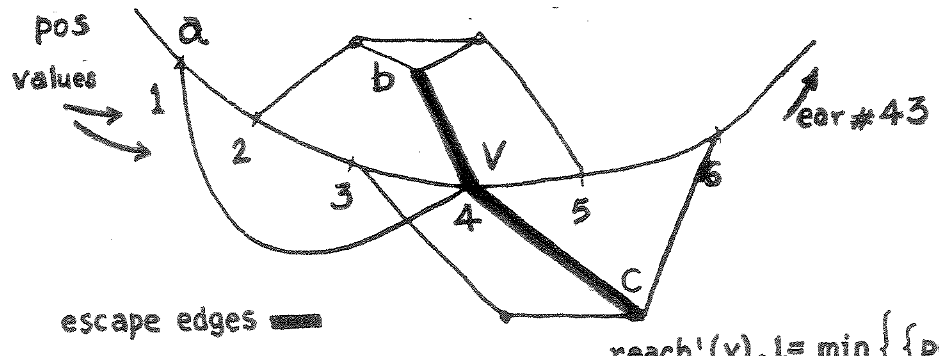


2. A Graph G and its auxiliary graph G_e



3. The reach graph R_i of ear E_i and its P_i

4. Computing reach'(v)



$$\text{reach}'(v).1 = \min\left\{\{pos(v)\} \cup \{pos(a)\} \cup \{\alpha(b).2, \alpha(c).2\}\right\}$$

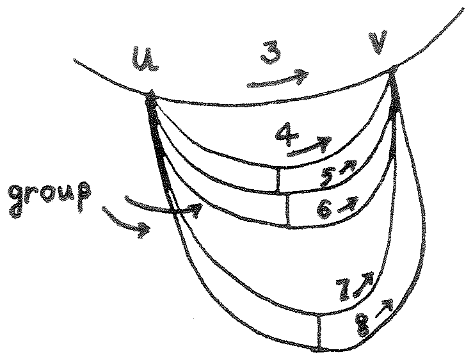
$$=1$$

$$\alpha(b) = \langle 43, 2, 5 \rangle$$

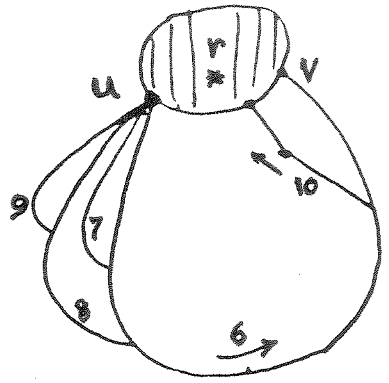
$$\alpha(c) = \langle 43, 3, 6 \rangle$$

$$\text{reach}'(v).2 = \max\left\{\{pos(v)\} \cup \{\alpha(b).3, \alpha(c).3\}\right\} = 6$$

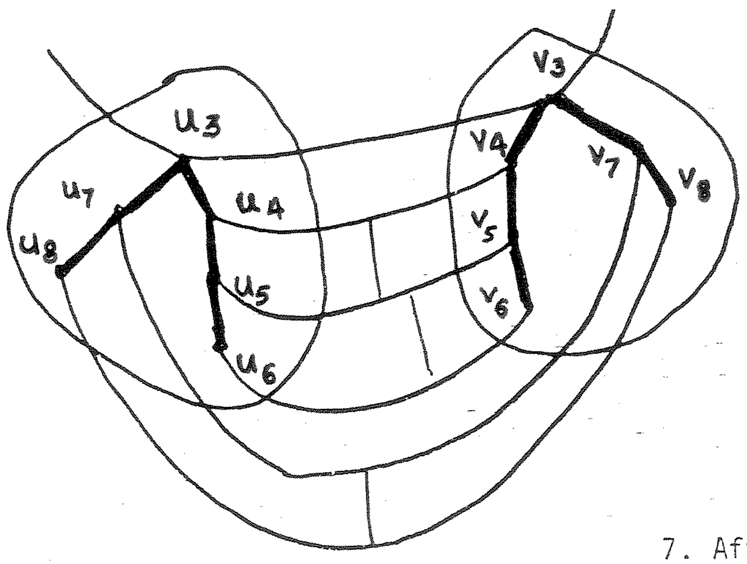
$$\text{reach}'(v) = [1,6]$$

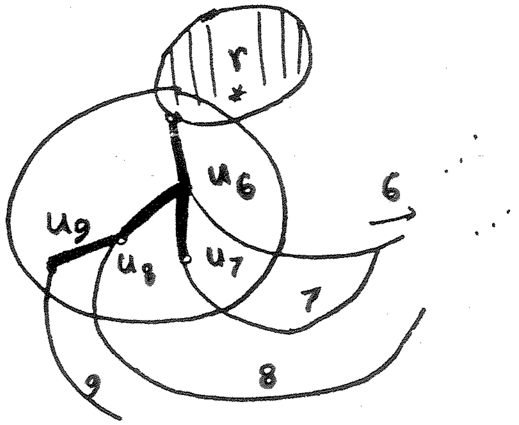


5. Parallel Ears and a Group of Parallel Ears

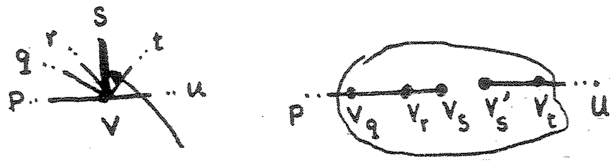


6. Ear 6 dominates 7,8,9

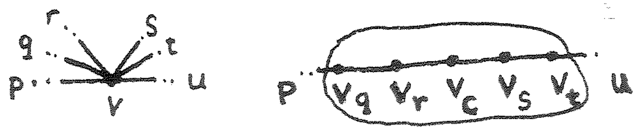




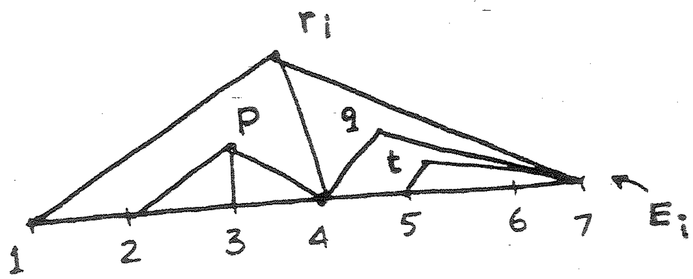
7. After local modifications

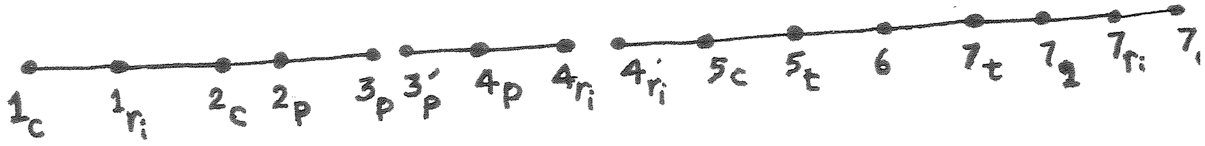


non-extreme edge



At the node level





8. After Step 1