Efficient Parallel Ear Decomposition with Applications

*Gary L. Miller**

MSRI, Berkeley, and University of Southern California, Los Angeles

*Vijaya Ramachandran***

MSRI, Berkeley, and University of Illinois, Urbana

1. Introduction

A *ear decomposition* of an undirected graph is a partition of its edge set into an ordered collection of paths having certain properties. This is useful in determining several connectivity properties of the graph. In this paper we present an efficient parallel algorithm for ear decomposition and a triconnectivity test based on it. Our algorithm runs in $O(\log m)$ parallel time using $O(n+m)$ processors, where n is the number of vertices in the graph and m is the number of edges.

2. Definitions

An *undirected graph* $G=(V,E)$ consists of a *vertex set* V and an *edge set* E containing unordered pairs of distinct elements from V. A *path* P in G is a sequence of vertices $<v_0,\cdots,v_k>$ such that $(v_{i-1},v_i)\in E, i=1,\cdots,k$. The path P *contains* the vertices $v_0,\cdots,v_k$ and the edges $(v_0,v_1),\cdots,(v_{k-1},v_k)$ and has *endpoints* v_0 and v_k. P is a *simple path* if $v_0,\cdots,v_{k-1}$ are distinct and $v_1,\cdots,v_k$ are distinct. P is a *simple cycle* if it is a simple path and $v_0=v_k$.

A *ear decomposition* D of an undirected graph $G=(V,E)$ is a partition of E into an

** Supported by NSF under ECS 8404866 and by an IBM Faculty Development Award.

ordered collection of edge disjoint simple paths $P_0, \cdots P_{r-1}$ such that P_0 is a simple cycle and each endpoint of $P_i, i=1, \cdots, r-1$ is contained in some $P_j, j<i$. The P_i's are called the ears of D. D is an *open ear decomposition* if none of the $P_i, i=1, \cdots, r-1$ *is a simple cycle.*

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

An undirected graph $G=(V,E)$ is connected if there exists a path between every pair of vertices in V. A *connected component* of G is an induced subgraph of G which is maximally connected.

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

Let $G=(V,E)$ be a biconnected undirected graph. A pair of vertices $v_1, v_2 \in V$ is a *separation pair* for G if the induced subgraph on $V-\{v_1,v_2\}$ contains two connected components. G is *triconnected* if it contains no separation pair.

An undirected graph $G=(V,E)$ is k-vertex (k-edge) connected if there exist k vertex-disjoint (k edge-disjoint) paths between every pair of vertices in V. Note that G is connected iff it is 1-vertex or 1-edge connected, G is biconnected iff it is 2-vertex connected, and G is triconnected iff it is 3-vertex connected.

An undirected graph $G=(V,E)$ is 2-edge connected iff every edge lies on a simple cycle. A *2-edge connected component of* G is a maximal set of edges $E' \subseteq E$ such that each $e \in E'$ lies on a simple cycle in G with edges from E'. Let $E_c = E - \{e \mid e$ is not on a simple cycle in $G\}$. Then the set of 2-edge connected components of G forms a partition of E_c. An edge $e \in E-E_c$ is called a *bridge* of G. A 2-edge connected component consisting of a single edge is a *bridge* of G.

3. Some Properties of Ear Decomposition

Lemma 0 Let $G=(V,E)$ be an undirected graph with a ear decomposition containing r ears. Then $r=|E|-|V|+1$.

Proof By induction on r.

Basis $r=1$. Then by definition G is a simple cycle. Hence $|E|=|V|$ and $|E|-|V|+1=1=r$.

Induction Step Let lemma be true for all $r \leq i$ and let $r=i$.

Let G have a ear decomposition $P_0, \cdots, P_{i-1}$. Let $G'=(V',E')$ be obtained from G by removing P_{i-1} (except for its endpoints). Then G' has a ear decomposition with $i-1$ ears and hence by the induction hypothesis $i-1=|E'|-|V'|+1$. Let the number of edges in P_{i-1} be e, and let the number of vertices in P_{i-1}, excluding the endpoints, be v. Then $e=v+1$ since P_{i-1} is a simple path. But $|E|=|E'|+e$ and $|V|=|V'|+v$, hence $|E|-|V|+1=|E'|-|V'|+1=i-1+1=i$.[]

Lemma 1 An undirected graph $G=(V,E)$ has a ear decomposition iff G is 2-edge connected.

Proof See [Wh].

Lemma 2 An undirected graph has an open ear decomposition iff it is biconnected.

Proof See [Wh].

Lemma 3 Let $G=(V,E)$ be a biconnected undirected graph for which vertices v_1 and v_2 form a separation pair. Let **D** be an open ear decomposition for G. Then there exists a ear P_i in **D** that contains both v_1 and v_2.

Proof Since v_1 and v_2 form a separation pair, the subgraph induced by $V-\{v_1,v_2\}$ contains at least two connected components. Let C_1 and C_2 be two such connected components (see figure 1).

Case 1 The first ear P_0 lies wholly outside of C_2:

Consider the lowest-numbered ear, P_i, that passes through a vertex in C_2. Since its endpoints are distinct and must be contained in earlier ears, P_i must contain v_1 and v_2.

figure 1
Illustrating the proof of Lemma 3

Case 2 P_0 contains a vertex in C_2:

If P_0 lies wholly outside of C_1, then case 1 applies to C_1. Otherwise P_0 contains at least one vertex from C_1, and one vertex from C_2. But then, since P_0 is a simple cycle, it must contain v_1 and v_2.[]

4. A Fast Parallel Ear Decomposition Algorithm

In this section we present an efficient parallel algorithm for finding ear decomposition for 2-edge connected components of a connected graph, such that each biconnected component of the graph has an open ear decomposition.

Algorithm A: Efficient Parallel Ear Decomposition

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

Output A numbering on the edges of 2-edge connected components, specifying their ear number, such that each biconnected component has an open ear decomposition.

begin

1) *Preprocess* Find a rooted spanning tree T for G (with root r), and number the vertices of T in preorder. Let $low(v)$= minimum numbered vertex reachable from a descendent of v followed by exactly one nontree edge, $high(v)$= maximum numbered vertex reachable from

a descendent of v followed by exactly one nontree edge, $nd(v)$= number of descendents of v, and let $p(v)$ be the parent of v in T for each $v \in V-\{r\}$. *Mark* each vertex v for which $low(v)<p(v)$ or $high(v)>p(v)+nd(v)$. All of these preprocessing steps can be done in $O(\log n)$ time with $O(m)$ processors [TaVi].

2) *Assign ear numbers to nontree edges in* T Find the least common ancestor (lca) in T for each nontree edge e in G, and label the nontree edges from 0 to $|E|-|V|$ in nondecreasing order of their lca in a proper way to form the required ear decomposition. This step is executed using rake and compress [MiRe]. Each nontree edge is given an ordered pair as a label, the first element of which is the preorder number of its lca, and the second element gives the position of the edge among those with the same lca. The first element of each label is computed during a tree contraction of T and the second elements are computed at the end, as follow:

A rake step is executed at a leaf only when all but one of its siblings have been evaluated. Let $v_1, \cdots, v_k$ be the children of vertex v, and let $v_1, \cdots, v_{k-1}$ be leaves at this stage of the computation. Let $b_1, \cdots, b_r$ be the nontree edges from the v_i's, $i=1, \cdots, k-1$ to the subtree, T_k, rooted at v_k. Detach each b_i from its endpoint in T_k and coalesce the free endpoints into a new vertex v'_k. If v_k is a marked vertex, then mark v'_k. Form G_v as the subgraph induced on $v_1, \cdots, v_{k-1}, v'_k$, and save it for later processing.

Restore the detached endpoint of each b_i to its original position in T_k and move all nontree edges at the raked leaves $v_1, \cdots, v_{k-1}$ to their parent v.

A compress step moves all nontree edges at a vertex being compressed to the vertex it is compressed into.

A contract step is a rake and compress step performed simultaneously.

Before the first contract step and after each contract step, check for each nontree edge, if one endpoint is an ancestor of the other in the current contracted tree. If so, the ancestor is the lca of the nontree edge. Remove this nontree edge after assigning the preorder number of the ancestor endpoint to the first element of its label pair. To check for ancestor-

descendent relationship, note that u is an ancestor of v in T iff $u \leq v \leq u + nd(u)$.

After T has been contracted to the root, find a spanning forest for each G_v in parallel. Order the trees in the forest arbitrarily, and root each tree at a marked vertex, if one exists. Number the vertices of the forest in preorder. For each edge in the spanning forest assign the number of its child in the spanning forest as the second element in its label. For nontree edges not in the spanning forest and with lca v, assign $l+1$ as the second element of the label, where l is the number of vertices in the forest.

Sort the nontree edges in nondecreasing order of their labels in $O(\log n)$ time with $O(m)$ processors [AjKoSz] and renumber using prefix sums by initializing all labels to 1, again in $O(\log n)$ time with $O(m)$ processors [TaVi].

3) *Extend the numbering assigned in step 2 to the tree edges* Number each tree edge t by the label of the smallest numbered nontree edge e for which t is on a path from one of the endpoints of e to the lca of e. This gives the required ear decomposition. Label the tree edges using rake and compress as follows:

a) Each vertex gets a label which is the number of the minimum numbered nontree edge incident on it. This can be determined in $O(\log n)$ time with $O(m/\log m)$ processors. Delete all nontree edges.

b) Implementing rake: Let v be a vertex for which children $v_1, \cdots, v_k$ are leaves that need to be raked. Let the label of v be b and the label of each v_i be b_i. Edge e_i connecting v to v_i gets label $b_i, i=1, \cdots, k$. The label of v gets labeled by $\min(b, b_1, \cdots, b_k)$. Although we are doing a minimum computation over k elements, we can break this down into binary computation by transforming our spanning tree T into a binary tree by introducing new vertices. The number of new vertices introduced is $O(n)$. Hence the tree size remains $O(n)$.

c) Implementing compress: Let v_2 be a vertex with label b_2 that needs to be compressed into vertex v_1 with label b_1. Let the edge connecting the two vertices be e. Then e gets label b_2 and v_1 gets label $\min(b_1, b_2)$.

Step 3 can be implemented in $O(\log n)$ time with $O(m)$ processors.

end.

Claim 1 Assume we number nontree edges with the same lca arbitrarily in Algorithm A (i.e., we do not form G_v to obtain the second element of the label of nontree edges). Then Algorithm A obtains a valid ear decomposition of a 2-edge connected graph.

Proof We prove the result in four parts:

a) The first ear is a simple cycle.

b) Ears formed by nontree edges with lca 1 are simple paths that lie on earlier ears (by induction on ear number). These could be *closed* ears.

c) The first ear formed by a nontree edge with lca i is a simple path that lies on earlier ears (by induction on lca number). This could be a simple cycle.

d) Ears formed by nontree edges with lca i are simple paths that lie on earlier ears (by induction on ear number). These could be closed ears.

The details of each part are straightforward.[]

Claim 2 Let G be a biconnected graph. Then Algorithm A obtains an open ear decomposition for G.

Proof First note that if G is biconnected then each forest in G_v, for each v, will have a marked vertex (if not, then v is an articulation point). If we can ensure that each edge from v to a child is assigned a different ear number then we are done. Edges from v to marked children belong to ears with lca higher than v and hence satisfy this condition. Edges from v to unmarked vertices belong to ears that 'hook on' to smaller numbered ears with same lca and hence again form open ears.[]