SIAM J. DISCRETE MATH.
Vol. 18, No. 1, pp. 41–70

AN EAR DECOMPOSITION APPROACH TO APPROXIMATING THE SMALLEST 3-EDGE CONNECTED SPANNING SUBGRAPH OF A MULTIGRAPH*

HAROLD N. GABOW†

Abstract. This paper gives a 3/2 approximation algorithm for the smallest 3-edge connected spanning subgraph of an undirected multigraph. The previous best algorithm of Khuller and Raghavachari [*J. Algorithms*, 21 (1996), pp. 434–450] has approximation ratio 5/3. The algorithm of Cheriyan and Thurimella [*SIAM J. Comput.*, 30 (2000), pp. 528–560] achieves ratio 3/2 for simple graphs. Our approach, based on the close relationship between an ear decomposition of a 2-edge connected graph and 3-edge connected components, enables us to achieve running time $O(m\alpha(m,n))$.

Key words. approximation algorithms, network design, multigraphs, graph connectivity, edge connectivity, ear decomposition, depth-first search

AMS subject classifications. 05C40, 05C85, 68R10, 68W25, 68W40, 90B18, 90C27

DOI. 10.1137/S0895480102405476

1. Introduction.

Finding the smallest k-edge connected spanning subgraph is a natural problem in network design. Since the problem is NP-complete even for $k = 2$, a large number of approximation algorithms have been developed. This paper provides an algorithm for $k = 3$ that has improved accuracy and runs in almost-linear time.

We begin by surveying the most relevant past work. Throughout this paper all graphs are undirected and parallel edges are allowed. n and m always denote the number of vertices and edges of the given graph, respectively.

Khuller and Raghavachari [8] give a 1.85 approximation algorithm for the smallest k-edge connected spanning subgraph for any k. A simpler version of that algorithm [7] achieves ratio $2 - 1/k$ and runs in linear time. For $k = 3$ this gives ratio 5/3, the best previous accuracy bound for our problem. Fernandes [4] shows the 5/3 bound is tight. She also improves the general bound to 1.75 (1.7 for large enough k) when the graph is simple.

Cheriyan and Thurimella [3] give more accurate algorithms for simple graphs. The performance bound is $1 + 2/(k + 1)$. For $k = 3$ this is 3/2. The time for the algorithm for $k = 3$ is $O(\sqrt{n}m + n^2)$. As pointed out in [3], the analysis relies on properties of simple graphs that need not hold for multigraphs. Indeed, [5] exhibits a family of multigraphs for every $k \geq 2$ where the approximation ratio of the algorithm is 2.

For $k = 2$ Vempala and Vetta approximate the smallest k-edge connected spanning subgraph to the ratio 4/3 [10]. As in [3], their approach is based on matching.

We use a simpler depth-first search approach. The approximation ratio is $\leq 3/2$ and the running time is $O(m\alpha(m,n))$ where α is the inverse Ackermann function. The starting point of our approach is the observation that, in an ear decomposition of

*Received by the editors January 15, 2002; accepted for publication (in revised form) August 21, 2003; published electronically July 2, 2004. A preliminary, abbreviated version of this paper appeared in *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, Philadelphia, ACM, New York, 2002, pp. 84–93.
http://www.siam.org/journals/sidma/18-1/40547.html
†Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309-0430 (hal@cs.colorado.edu).

a 2-edge connected graph, the ends of each ear are 3-edge connected. Cheriyan, Sebő, and Szigeti [2] use ear decomposition to approximate the smallest 2-edge connected subgraph.

Our lower bound is based on a generalization of the component lower bound introduced in [6]. We use a new operation,"breaking off," to strengthen that bound. We provide a simple proof that our algorithm has approximation ratio at most 14/9, and a more involved proof of the 3/2 accuracy bound. The latter requires combining three lower bounds (each one an instance of the component lower bound) and keeping track of the slack in those bounds. We give an example in which the algorithm has approximation ratio 17/12, i.e., 1/12 below our upper bound.

After the initial writing of this paper, we presented in [5] an improved analysis of the above-mentioned algorithm of Khuller and Raghavachari for the smallest k-edge connected spanning subgraph. The performance ratio of that algorithm is shown to be < 1.61 for any $k > 1$. To achieve this for odd values of k requires using the ear decomposition algorithm presented here as the base case. Furthermore, the analysis of [5] requires a stronger version of the performance ratio of the ear decomposition algorithm. The proof of the stronger version uses some parts of the argument for the 3/2 bound of this paper. For that reason the proof of the stronger version has been added as an appendix to this paper.

Section 2 gives basic facts that are used in the ear decomposition algorithm. Section 3 presents the algorithm. Section 4 gives a simple analysis showing that the approximation ratio is $\leq 14/9$. Section 5 refines the analysis to show the desired 3/2 bound. Section 6 shows the time bound. Appendix A gives the stronger version of the approximation ratio that is needed by [5] for general k. This section closes with our terminology and a background review. We use much of the notation of [2].

We often denote a singleton set $\{x\}$ by x. When we say a set is partitioned into subsets, each subset is required to be nonempty. For a family $\mathcal{S}$ of pairwise disjoint sets of vertices, the graph $G/\mathcal{S}$ is formed by contracting each set of $\mathcal{S}$ to a single vertex. We retain parallel edges but not loops.

We denote edges by juxtaposing the two vertices, e.g., vw. If vw is a tree edge or back edge of a depth-first search, the order of the vertices is significant: For a tree edge, v is the parent of w; for a back edge, v is a descendant of w. If the edge is not known to be a tree or back edge, the order is irrelevant.

In a graph $G = (V, E)$ with degree function d, if X and Y are disjoint sets of vertices, then $d(X, Y)$ is the total number of edges joining X and Y. $d(X)$ stands for $d(X, V - X)$. If H is a subgraph, then d_H denotes its degree function. (Sections 3 and 6 use the function d to denote depth in a tree, but this is clearly indicated.)

Two vertices are *k-edge connected* if they are joined by k edge-disjoint paths. Equivalently, the two vertices remain connected after deleting any $< k$ edges. This binary relation is an equivalence relation. A graph is k-edge connected if every two distinct vertices are k-edge connected. *k-ECSS* stands for k-edge connected spanning subgraph. For any k,

$$\varepsilon_k = \text{ the minimum number of edges in a } k\text{-ECSS}.$$

Throughout this paper we abbreviate ε_3 to ε.

We assume paths are simple, but they can be open or closed. A closed path has a distinguished vertex that plays the role of both endpoints. For a path P, $I(P)$ denotes the internal vertices of P, i.e., all the vertices except the endpoints. The symbol P denoting a path may reference the vertex set or the edge set of the path, as determined by context.

An *ear decomposition* of a graph is a partition of the edges into paths P_i, $i = 0, \ldots, q$, such that P_0 is a single vertex r and, for $i > 0$, each P_i has its ends and no other vertices in common with previous paths, i.e., $V(P_i) \cap (\bigcup_{j=0}^{i-1} V(P_j)) = V(P_i) - I(P_i)$ for $i = 1, \ldots, q$. A graph is 2-edge connected if and only if it has an ear decomposition [11]. For any vertex v the first ear containing v is denoted P_v. Any $v \neq r$ has $v \in I(P_v)$ (and for $v = r$, $I(P_r) = \emptyset$). An ear is *short* if its length is one, i.e., it has no internal vertices, otherwise the ear is *long*.

Given a spanning tree, a nontree edge *covers* every edge in its fundamental circuit. In a rooted tree we distinguish between *ancestor* and *proper ancestor*, the former relation being reflexive and the latter irreflexive. These distinctions hold similarly for *descendant* and *proper descendant*.

Khuller and Vishkin [9] define a *(dfs) tree carving* for any depth-first spanning tree T as follows. Do a bottom-up traversal of T, constructing a set of back edges B according to the following rule: When backing up from a vertex v to its parent p, if the tree edge pv is not covered by an edge of B, then add to B the back edge c that goes from a descendant of v to a vertex closest to the root. Edge c exists and covers pv, assuming G is 2-edge connected.

Each tree edge pv that forces a back edge to be added to B is a *carving edge*. Deleting the carving edges from T gives a forest called the dfs tree carving. Define

$$\gamma = \text{the number of carving edges.}$$

Khuller and Vishkin show that for any k,

$$\varepsilon_k \geq k\gamma.$$

This follows from the fact that no back edge covers two carving edges.

2. Basic facts. We increase the edge-connectivity by one using the following proposition, a slight strengthening of Lemma 3.1 of [8]. Let G be k-edge connected. Let K be a $(k-1)$-ECSS of G. Let $\mathcal{S}$ be a partition of V, each set of which is k-edge connected in K (i.e., any two vertices of the same set have $\geq k$ edge-disjoint paths between them). Let F be a maximal spanning forest of the graph $(G - K)/\mathcal{S}$.

PROPOSITION 2.1. *$K + F$ is k-edge connected.*

The next lemma gives our basic relation between an ear decomposition and 3-edge connectedness. (See Figure 1.) Consider an ear decomposition P_i, $i = 0, \ldots, q$, of a 2-edge connected graph. For any vertex v let the "closure" $Cl(v)$ be the smallest subset of $\{P_i\}$ that defines an ear decomposition of a (not necessarily spanning) subgraph containing v. (Closure can be defined inductively: For $P_0 = \{r\}$, $Cl(r) = P_0$. For $v \neq r$, if P_v goes from a to z, then $Cl(v) = \{P_v\} \cup Cl(a) \cup Cl(z)$.)

LEMMA 2.2. *Consider an ear decomposition of a 2-edge connected graph.*

(i) *The two endpoints of an ear are 3-edge connected.*

(ii) *If an ear has endpoints a and z, then each endpoint of an ear in $Cl(a) \oplus Cl(z)$ is 3-edge connected with a and z.*

Proof. Let P be an ear joining a and z.

(i) Let H be the subgraph $Cl(a) \cup Cl(z)$. a and z are 2-edge connected in H. Hence H contains two edge-disjoint az-paths. No edge of P belongs to H. This makes three edge-disjoint az-paths.

(ii) By symmetry it suffices to show that an arbitrary endpoint v of an ear in $Cl(a) - Cl(z)$ is 3-edge connected with z. Let H be the 2-edge connected graph $Cl(z) \cup Cl(v)$. The ears in $Cl(a) - Cl(v)$ contain a path Q from v to a. Combining

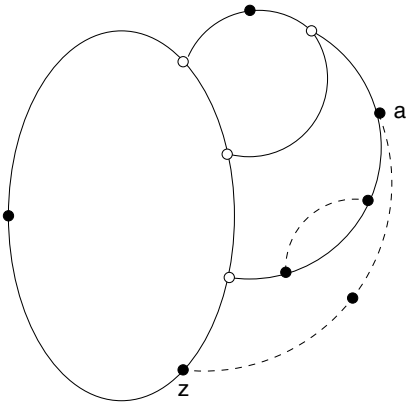

FIG. 1. *Illustration of Lemma* 2.2: $Cl(a)$ *includes the* 3 *solid ears but not the* 2 *dashed ears. The* 4 *hollow vertices are the endpoints of ears in* $Cl(a) \oplus Cl(z)$ *and so are* 3*-edge connected with* a *and* z.

Q with the given ear P gives a path from v to z that is edge-disjoint from H. As in part (i), v and z are 3-edge connected. □

Our algorithm forms an ear decomposition based on depth-first search. Let G be 2-edge connected and let T be a dfs tree of G. We shall construct ears consisting of a tree path followed by a back edge. We use the triple of vertices a, y, z to identify the ear consisting of the tree path from a to y followed by the back edge yz. Here z is an ancestor of a, which is an ancestor of y. (If $a = y$, the ear is short.)

To construct the ear decomposition for T let the first ear be r, the root of T. Suppose we have constructed a number of ears that collectively contain vertices $X \subset V$. Choose a tree edge ab with $a \in X$, $b \notin X$. Choose a back edge yz that covers ab. (yz exists since G is 2-edge connected.) The next ear is defined by the triple a, y, z. Adding this ear enlarges X by the vertices in the tree path from a to y. Repeat this step until $X = V$. Finally, make any back edge that is not yet in an ear into a short ear.

For the rest of this paper all ear decompositions are constructed in this manner. We use Figure 2 to illustrate various concepts involving ear decompositions.

We focus on the *first vertex* of an ear, i.e., vertex a in a, y, z. For each vertex x let $f(x)$ be the first vertex of ear P_x. f is represented by F, a tree on vertex set V: The root of F is r. The parent of a vertex $x \neq r$ is $f(x)$. In Figure 2 the first three proper ancestors of j in F are i, d, and a.

In this paper we use both the dfs tree T and the first vertex tree F. By default all tree terminology refers to T. For instance, we use "ancestor" (referring to T), "ancestor in T" (when there is danger of confusion), and "ancestor in F." r always denotes the root of T (r is also the root of F). It is easy to see that the proper descendants of x in F are the vertices that, in T, descend from a child x' of x with $x' \notin P_x$.

COROLLARY 2.3. *For an ear* a, y, z*, vertex* z *is* 3*-edge connected with every vertex that, in* F*, is an ancestor of* a *but not* $f(z)$.

Remark. It is possible that a is the only ancestor satisfying the conditions of the corollary.

Proof. If p is the parent of vertex x in T, then obviously $Cl(p) \subseteq Cl(x)$ (equality holds if $p \neq f(x)$). Iterating shows that any ancestor b of x has $Cl(b) \subseteq Cl(x)$. Thus

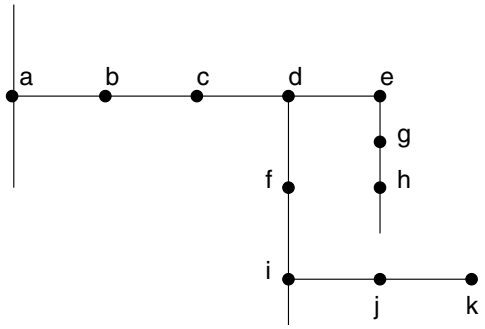

FIG. 2. *Schematic figure illustrating four long ears. Each horizontal or vertical line represents the tree path of one ear. Back edges of ears are not drawn. For example, one ear consists of the path from a to e followed by a back edge from e to an ancestor of a.*

in the statement of the corollary, $Cl(z) \subseteq Cl(a)$. In fact we get $Cl(a)$ from $Cl(z)$ by adding ears whose first vertices are the vertices that, in F, are proper ancestors of a but not proper ancestors of z. (Example: In Figure 2 we get $Cl(k)$ from $Cl(c)$ by adding two ears, with first vertex d and i, respectively.) Lemma 2.2 now implies the corollary. □

Garg, Santosh, and Singla [6] introduced the "component lower bound" for 2-ECSS. It was refined in [2]. We use the following generalization to k-ECSS. For any graph H let $c(H)$ be the number of connected components of H.

LEMMA 2.4 (component lower bound). *For any integer k let $G = (V, E)$ be k-edge connected. For any partition $\mathcal{S}$ of V, $\varepsilon_k \geq (k/2) \sum_{S \in \mathcal{S}} c(G - S)$.*

Remark. The degree lower bound $\varepsilon_k \geq kn/2$ amounts to the special case of the lemma where each $S \in \mathcal{S}$ consists of one vertex. The tree carving lower bound $\varepsilon_k \geq k\gamma$ is the special case where each S is a set of the tree carving.

Proof. Let H be a k-ECSS with ε_k edges. For any set S, each connected component of $G - S$ is joined to S by $\geq k$ edges of H. Hence

$$2\varepsilon_k = \sum_{v \in V} d_H(v) = \sum_{S \in \mathcal{S}} \sum_{v \in S} d_H(v) \geq \sum_{S \in \mathcal{S}} kc(G - S). \qquad \square$$

The rest of this section presents a method for strengthening the component lower bound. It is only used in section 5, so readers interested in just the 14/9 upper bound can skip this material.

We first give the idea and then a formalization. Suppose a set $S \subseteq V$ can be partitioned into sets S_0, S_1 so that ≤ 1 connected component of $G - S$ is adjacent to both S_0 and S_1. Then

$$c(G - S_0) + c(G - S_1) \geq c(G - S) + 1.$$

This follows since, for $i = 0$ or 1, a component of $G - S$ that is adjacent only to S_i contributes to $c(G - S_i)$. So ≤ 1 component of the right-hand side is not counted by the left-hand side. In addition, each $G - S_i$ has a component containing a vertex of S_{1-i}, which is also counted by the left-hand side.

We now define a configuration in which this principle can be applied repeatedly.

DEFINITION 2.5. *Consider a partition of a set $S \subseteq V$ into sets S_i, $i = 0, \ldots, h$, where each index i, $0 < i \leq h$ has a "parent" index $p(i)$, $0 \leq p(i) < i$. An edge joining S_i and $S_{p(i)}$ is a* bridging edge. *A component of $G - S$ that is adjacent to both S_i and*

$S_{p(i)}$ but no other set S_j is a bridging component. *The partition is an* enhancement *(of S) if the following two conditions hold:*

(i) *Every edge joining two distinct sets S_j is a bridging edge.*

(ii) *Every component of $G - S$ adjacent to two or more distinct sets S_j is a bridging component. Furthermore, at most one bridging component joins any two distinct S_j's.*

Note that the definition allows any number of components of $G-S$ to be adjacent to a single set S_i. We also remark that the following lemma remains valid for a slightly broader definition of enhancement, but Definition 2.5 suffices for our purposes.

LEMMA 2.6. *If $S \subseteq V$ has an enhancement S_i, $i = 0, \ldots, h$, then*

$$\sum_{i=0}^{h} c(G - S_i) \geq c(G - S) + h.$$

Proof. The proof is by induction on h. The above argument shows

$$c(G - (S - S_h)) + c(G - S_h) \geq c(G - S) + 1.$$

So if $h = 1$, we are done. For $h > 1$ we claim that the partition of $S - S_h$ into S_i, $i = 0, \ldots, h-1$, is an enhancement. This claim implies $\sum_{i=0}^{h-1} c(G - S_i) \geq c(G - (S - S_h)) + h - 1$ by induction. Together with the preceding inequality, this completes the inductive step.

To prove the claim, use the same parent function. Obviously condition (i) holds. The components of $G - (S - S_h)$ can be derived from the components of $G - S$ as follows: Let $\mathcal{C}$ be the family of components of $G - S$ that are adjacent to S_h and no other set S_i. Let B be the bridging component for S_h and $S_{p(h)}$, if it exists. The components of $G - (S - S_h)$ are those of $G - S$ with $\mathcal{C}$ and B replaced by one component, $S_h \cup B \cup \bigcup \mathcal{C}$. Since this new component is adjacent only to $S_{p(h)}$ (or to no set S_i, $i < h$), condition (ii) continues to hold. □

We form enhancements by starting with set S and repeatedly replacing it by $S - S_i$, where S_i is the next set of the enhancement. We call this operation *breaking off* the set S_i. So the enhancement of Definition 2.5 corresponds to breaking off sets $S_h, S_{h-1}, \ldots, S_1$ in that order. Details of the breaking off operation employed in this paper are given in section 5.2.

An *enhancement* of a partition $\mathcal{S}$ of V is formed by enhancing each set of $\mathcal{S}$. Suppose we do a total of β break off operations in various sets of $\mathcal{S}$. Lemmas 2.4 and 2.6 imply

$$\varepsilon_k \geq (k/2) \left(\sum_{S \in \mathcal{S}} c(G - S) + \beta \right). \tag{1}$$

The analysis of section 5 uses the degree lower bound, the carving lower bound, and (1). Furthermore, it analyzes the slack in these three lower bounds. We now present the version of (1) with slack terms.

Fix a k-ECSS H with ε_k edges. We will apply (1) to the graph H, *not* G. Start with a partition $\mathcal{S}$ of V. Form an enhancement, in H, by doing a total of β break off operations. An edge of H is nonbridging if it joins two vertices in the same set S_i of the enhanced partition; for any $S \in \mathcal{S}$, a component of $H - S$ is nonbridging if it is adjacent (in H) only to vertices in one set S_i of the enhanced partition. The reader should not forget that an enhancement may be valid in H but not in G. For instance,

a component may be nonbridging in H but may have neighbors in many different sets S_i in G.

Say that a component has s *surplus edges* if $k+s$ edges of H leave it. Define

θ_a = the total number of surplus edges for all nonbridging components of H;
θ_b = the number of nonbridging edges of H.

Note that θ_a is computed by summing, for every $S \in \mathcal{S}$, the number of surplus edges for all connected components of $H - S$ that are adjacent to only one set S_i of the enhancement. Introducing the terms θ_a and θ_b in the proof of Lemma 2.4 and using Lemma 2.6 gives

$$\varepsilon_k \geq (k/2)\left(\sum_{S\in\mathcal{S}} c(H-S) + \beta\right) + \theta_a/2 + \theta_b. \tag{2}$$

3. Approximation algorithm. Assume the given graph G is biconnected. If not, each block of G is 3-edge connected, and it suffices to run our approximation algorithm on each block. The algorithm consists of three phases that collectively construct a 3-ECSS A. The most involved part is Phase II. We begin by stating all three phases, and then we describe Phase II in detail.

Phase I does a depth-first search to construct a dfs tree carving. Let T denote the depth-first search tree and let r be its root. Phase I sets A to T. Let $C \subseteq T$ be the set of carving edges and write $\gamma = |C|$.

Phase II adds γ back edges to A, making A 2-edge connected. These back edges are chosen to also make a large number of pairs of vertices 3-edge connected. This is done by building A in the form of an ear decomposition. To create even more 3-edge connected pairs, additional back edges are added to A as short ears.

Phase III makes A 3-edge connected. This is done by adding a maximal forest of edges that span the 3-edge connected components of A, as described in Proposition 2.1.

The details of Phases I and III are straightforward. The rest of this section is devoted to Phase II, which is given by the pseudocode of Figure 3. The routines of Figure 3 use an auxiliary procedure `Multi-Merge` and an associated data structure to keep track of 3-edge connected pairs. We now describe both of these.

The data structure is a partition of V into sets of vertices that are known to be 3-edge connected in A. For $x \in V$, $t(x)$ denotes the set containing x. Thus x is 3-edge connected to every vertex of $t(x)$ in the subgraph A. The sets $t(x)$ are called *t-sets* and the corresponding partition of V is called the *t-partition*. The algorithm maintains the t-partition using the disjoint-set data structure, with operations `Union`(x, y) and `Find`(x) [1].

Phase II builds the first vertex tree F (defined in section 2) as it builds the ear decomposition of A. Say that an ear from a to z (with z an ancestor of a) *traverses* the set $t(z)$ and all the sets $t(x)$ where, in tree F, x is an ancestor of a but not an ancestor of $f(z)$. For instance, in Figure 2 an ear i, k, b traverses $t(i), t(d)$, and $t(b)$. In general, all the traversed sets can all be merged together according to Corollary 2.3. The purpose of `Multi-Merge`(a, z) is to execute this merge. Specifically, `Multi-Merge`(a, z) performs `Union`(z, x) for every distinct set $t(x) \neq t(z)$ traversed by the ear from a to z. Observe that an ear traversing s distinct t-sets causes exactly $s - 1$ union operations.

We turn to Figure 3. Let s be the child of r in the dfs tree T. (The root has a unique child since G is biconnected.) Here and throughout this section, $d(v)$ denotes the depth of vertex v in T.

```
Phase II
1.  Long_Ear(s);
2.  Short_Ear(s, 3);
3.  Short_Ear(s, 2);

Long_Ear(b)
1.  let a be the parent of b in T;
2.  choose an edge c ∈ C that descends from edge ab and is covered by a back
    edge yz with y descending from c and z an ancestor of a;
3.  choose the above yz so
    (i) z ∉ t(a) if possible /* merging ear */
    (ii) subject to (i), the depth d(y) is maximal;
/* the new ear P_b consists of the tree path from a to y followed by edge yz */
4.  add the back edge yz to A;
5.  Multi-Merge(a, z);
6.  for each tree edge xx' with x ∈ I(P_b), x' ∉ P_b do Long_Ear(x');

Short_Ear(b, i)
1.  for each tree edge xx' with x ∈ I(P_b), x' ∉ P_b do Short_Ear(x', i);
2.  for each x ∈ I(P_b) do
3.      if some back edge xz traverses > i distinct t-sets then { /* short ear */
4.          let xz be such an edge with minimum depth d(z);
5.          add xz to A;
6.          Multi-Merge(x, z) }
```

FIG. 3. *Algorithms for Phase* II.

Phase II has three main steps (see the top of Figure 3). It starts with every vertex being a singleton t-set. `Long_Ear` enlarges A from the dfs tree T to a 2-edge connected graph. This is done in a top-down traversal of T, as described in section 2, determining the back edges of A that form long ears. Next, the first execution of `Short_Ear` enlarges A with back edges that form short ears, each one causing ≥ 3 unions by `Multi-Merge`. This is done in a bottom-up traversal of T. Then the second execution of `Short_Ear` adds short ears that cause 2 unions by `Multi-Merge`.

The recursive procedure `Long_Ear`(b) starts by constructing a new ear whose first internal vertex is b. To do this, in lines 2–3 we choose the back edge yz of the new ear a, y, z. The back edge yz covers a carving edge c. It is easy to see that if $ab \notin C$, then the carving edge c descends from b; on the other hand, if $ab \in C$, then $c = ab$. The existence of carving edge c and back edge yz is guaranteed by the definition of tree carving.

A long ear is classified as *merging* if $z \notin t(a)$ in line 3(i); otherwise it is *nonmerging*. It is clear that the call to `Multi-Merge` (line 5) performs one or more unions if the ear is merging. The remark after Lemma 4.1 shows that no union is performed for a nonmerging ear.

Line 3(ii) ensures that the depth $d(y)$ is maximal; i.e., if a, y, z is nonmerging, then no ear a, y', z' with y' properly descending from y is possible, and if a, y, z is merging, then no merging ear a, y', z' with y' properly descending from y is possible.

Line 4 updates the tree F when it adds yz to A. Specifically, each vertex of $I(P_b)$ is made a child of a. Finally, line 6 of `Long_Ear` grows the rest of the ear decomposition in recursive calls.

`Short_Ear`(b, i) starts by recursively processing the ears descending from P_b. Then it adds short ears that have their deeper vertex in $I(P_b)$ and cause i or more unions. (In line 3, it makes sense to speak of the t-sets traversed by back edge xz since xz is an ear.) Note that in line 4, xz is not already in A since a back edge of A traverses at most two distinct t-sets. Also note that xz need not be the back edge with minimum $d(z)$. For instance, in Figure 2 if $t(a) = \{a, b, d\}$, then the back edge jc traverses more t-sets than ja or jb.

We note that the call to `Short_Ear`$(s, 3)$ is irrelevant to section 4: All results in that section hold if the call is omitted. We also note that section 5.1 adds some natural rules for choices made in the algorithm of Figure 3.

Figure 4 illustrates the algorithm on a family of graphs in which the approximation ratio approaches 17/12. (The illustration obeys the rules given in section 5.1 too.) First we describe the graph. The number of vertices is divisible by 4, so we write $n = 2h$ with h even. The vertices are identified by the integers $0, \ldots, n-1$. All arithmetic on vertex numbers is done modulo n. It is convenient to describe the edges as a union of five sets. An important property is that no two even-numbered vertices are adjacent. Figure 4(a) illustrates the first edge set,

$$H = \{(2i, 2i-1), (2i, 2i+1), (2i, 2i+3) : 0 \le i < h\}.$$

It is easy to see that the edges of H induce a 3-edge connected graph. Since each vertex has degree 3, H is a smallest 3-ECSS. The second edge set constitutes the dfs tree shown in Figure 4(b)–(c),

$$T = \{(2i-1, 2i+1) : 1 \le i < h\} \cup \{(2h-1, 2i) : 0 \le i < h\}.$$

The third edge set, shown in Figure 4(b), contains the back edges that complete merging ears,

$$M = \{(2i, 2h-2i-3) : h/2 - 1 \le i \le h-2\}.$$

The fourth edge set consists of the back edges that form short ears, shown in Figure 4(c). Writing $a = 3h/2$, the set is

$$S = \{(2i, 3h-2i-3) : a/2 - 1 \le i \le h-2\}.$$

The last edge set forms the spanning forest added to A in Phase III,

$$F = \{(1, 2i) : 0 \le i \le a/2 - 2 \text{ or } i = h-1\} \cup \{(1, 2i+1) : a/2 \le i \le h-2\}.$$

Since no two even-numbered vertices are adjacent, T is a valid dfs tree of the entire graph. Phase I constructs T as the dfs tree. The carving edges are the h edges incident to the leaves of T. In Phase II, `Long_Ear` works as follows (Figure 4(b)): The first ear is $1, 3, 5, \ldots, 2h-1, 2h-2, 1$. (The back edge comes from H.) The remaining long ears each have first vertex $2h-1$. First, $h/2$ merging ears are formed using the back edges of M. These ears build up a t-set of $h/2+1$ vertices, drawn black in Figure 4(b), $t(1) = \{2i-1,\ 2h-1 : 1 \le i \le h/2\}$. All other t-sets are singletons. Next, $h/2-1$ nonmerging ears are formed using back edges contained in H, $(h-4, h-3)$, $\ldots$, $(2, 3)$, $(0, 1)$. (The choice of the second vertex of these edges

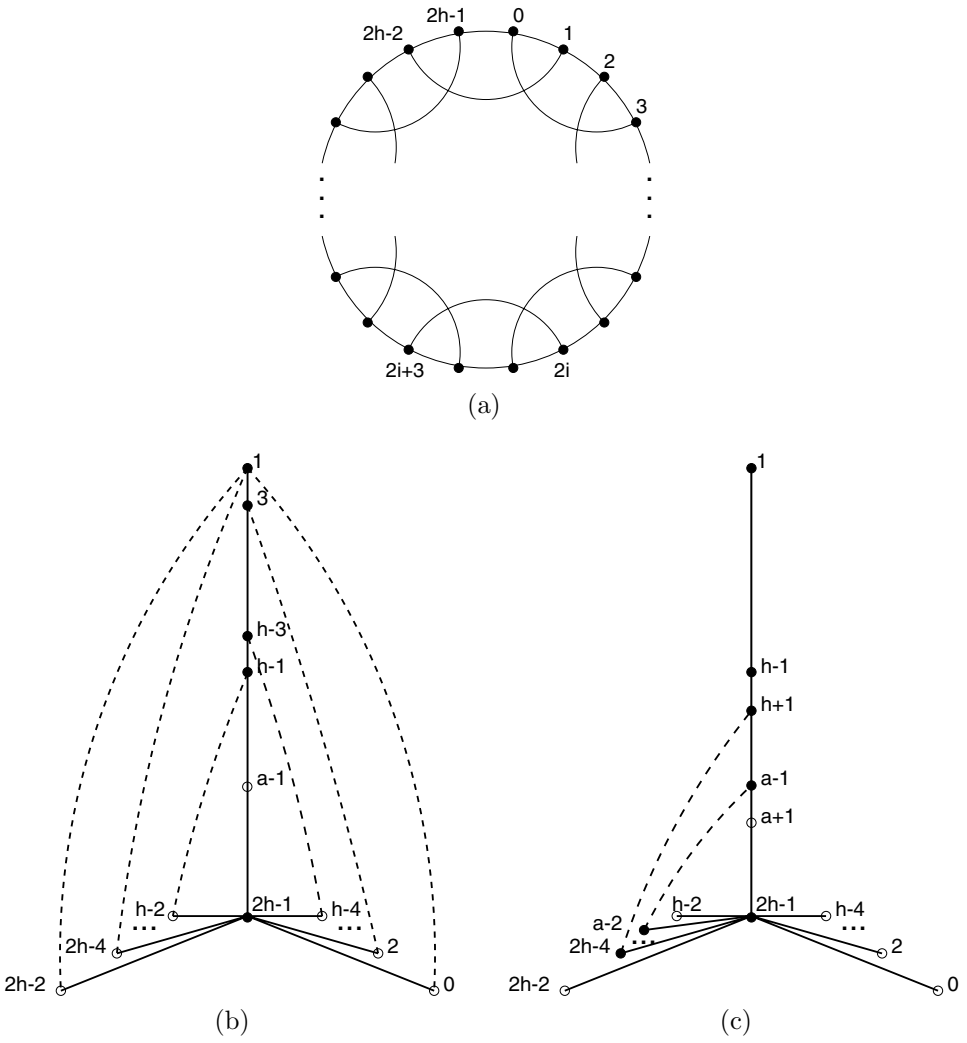

FIG. 4. *Execution of Phase* II *on an example graph.* (a) *Minimum* 3*-ECSS.* (b) *Subgraph* A *after* `Long_Ear`*. Dfs tree edges are solid. Back edges of long ears are dashed.* (c) *Dfs tree with the edges added in* `Short_Ear`*. The parameters of section* 4 *for this example are* $\gamma = h,\ \mu = \overline{\mu} = h/2,\ \sigma = h/4,\ \nu = h,\ \chi = 0,\ \kappa = 3h/4$.

is not crucial.) Observe that no merging ear is possible for any of the corresponding carving edges.

Figure 4(c) illustrates the rest of Phase II, i.e., the two executions of `Short_Ear`. `Short_Ear`$(s, 3)$ does nothing. `Short_Ear`$(s, 2)$ adds the edges of S as $h/4$ short ears. These add $h/2$ vertices to $t(1)$, drawn black. No short ears are added from vertices $a-4, \ldots, 2, 0$, since each back edge from these vertices traverses only two t-sets.

Phase III adds the forest F to A, joining each hollow vertex of Figure 4(c) to vertex 1. Phases I, II, and III add $2h-1$, $h + h/4 = 5h/4$, and $h-1$ edges to A, respectively. The approximation ratio is $(17h/4 - 2)/3h$, which approaches $17/12$ as $h \to \infty$.

Returning to our general discussion, it is clear that Phase II works correctly,

and in fact the whole algorithm correctly produces a 3-ECSS. We note one further property of Phase II.

LEMMA 3.1. *If a nonmerging ear has exactly one internal vertex, that vertex is a leaf.*

Proof. Consider a nonmerging ear P whose first edge is ab. Assume ab is a carving edge; otherwise, obviously P has > 1 internal vertex. P is created in `Long_Ear`(b). Assume b is not a leaf of T. Hence b has a child v. Since b is not an articulation point, some back edge e joins a descendant of v to an ancestor of a. Since P is nonmerging, e can be chosen in line 3(ii), which is a contradiction. Hence P has > 1 internal vertex. □

4. Basic analysis. This section proves some basic properties of the algorithm, leading to the conclusion that the approximation ratio is $\leq 14/9$. We begin by bounding the size of the algorithm's subgraph A. Define the following quantities that satisfy (3):

$$\begin{aligned} \mu &= \text{the number of merging ears,} \\ \overline{\mu} &= \text{the number of nonmerging ears.} \\ \gamma &= \mu + \overline{\mu}. \end{aligned} \tag{3}$$

By definition, a merging ear does at least one nontrivial union operation in `Multi-Merge`. A short ear does at least two nontrivial union operations. All other unions executed by `Multi-Merge` are called *surplus unions*. So an ear causes s surplus unions if it is a merging ear causing $1 + s$ unions, or a short ear (in `Short_Ear`$(s, 3)$) causing $2 + s$ unions. Define the following quantities that satisfy (4):

$$\begin{aligned} \sigma &= \text{the number of short ears,} \\ \nu &= \text{the total number of unions,} \\ \chi &= \text{the number of surplus unions.} \\ \nu &= \mu + 2\sigma + \chi. \end{aligned} \tag{4}$$

These quantities are illustrated in Figure 4.

Phase I adds $n - 1$ tree edges. Phase II adds γ back edges for long ears and σ edges that are short ears. Phase III adds $\leq (n - 1) - \nu$ edges to make A 3-edge connected. Thus the number of edges added by the algorithm is

$$|A| \leq (n-1) + \gamma + \sigma + (n-1) - \nu = 2(n-1) + \overline{\mu} - \sigma - \chi.$$

We turn to lower-bounding ε. We first prove a structural property of the t-partition: At any point in time, any set $t(x)$ contains $f(x)$ if it contains an ancestor of $f(x)$. In fact, we prove the following more general property.

LEMMA 4.1. *At any point in time, any set $t(x)$ contains any ancestor of x in F that has an ancestor in T belonging to $t(x)$.*

Example. In Figure 2, if $t(j)$ contains a proper ancestor of a, then it contains i, d, and a. The lemma is not true if we change F to T; e.g., we may have $b \in t(j)$, but $c, f \notin t(j)$.

Proof. We claim the following.

CLAIM. *At any point in time for any vertex x, $t(x)$ contains $f(x)$ if it contains a vertex not descending (in T) from $I(P_x)$.*

This implies the lemma as follows: Suppose $a \in t(x)$ is an ancestor in T of $f(x)$. The claim implies $f(x) \in t(x)$. Iterating this argument gives the lemma.

We will use this simple consequence of the claim: Call a vertex x *ancestral* if every vertex of $t(x)$ descends (in T) from $I(P_x)$. The claim implies that if $f(x) \notin t(x)$, then x is ancestral.

We now prove the claim by contradiction. Consider the first time the claim fails, say, as a result of the operation `Multi-Merge`(y, z). The new t-set τ formed by this operation must violate the claim. τ is the union of all t-sets traversed by the ear from y to z. Specifically, if W is the set of vertices that, in F, are ancestors of y but not $f(z)$, $\tau = t(z) \cup \bigcup_{x \in W} t(x)$. (Throughout this argument the notation $t(u)$ refers to the t-set of u immediately before `Multi-Merge`(y, z).) Let w be the vertex of W that is shallowest in F. Since some back edge goes from a descendant of y to z, w is a descendant of z; also $f(w) = f(z)$.

We now show that τ consists of $t(z)$, $t(w)$, and some descendants of w in T. A vertex $u \in \tau - t(w) - t(z)$ comes from a set $t(x)$, where $x \in W - t(w)$ and, without loss of generality, $f(x) \notin t(x)$. The latter implies that x is ancestral. This implies u descends from $f(x)$ in T. Since $f(x)$ descends from w in F it descends from w in T. Hence u descends from w in T, as desired.

To show the claim actually is not violated, consider a vertex $v \in \tau$ with $f(v) \notin \tau$. Clearly $f(v) \notin t(v)$, so v is ancestral. We first show that $v \in t(w) \cup t(z)$. Suppose not. Let $x \in W \cap t(v)$ with $f(x) \notin t(v)$. (x exists since $t(v) \neq t(w)$.) $f(x)$ gets added to τ since $f(x) \in W$. Furthermore, $f(x) = f(v)$ since both v and x are ancestral. But this contradicts $f(v) \notin \tau$.

We have shown either $v \in t(z)$ or $v \in t(w)$. Also note that v is ancestral, so every vertex in $t(v)$ descends from $I(P_v)$. We consider four cases depending on how $f(z)$ relates to $t(w)$ and $t(z)$.

Suppose $f(z) \in t(w) - t(z)$. (This is possible even though w descends from z: recall the above example.) Thus every vertex of $t(z)$ descends from $I(P_z)$, $v \in t(w) - t(z)$, and every vertex of $t(w) \cup P_z$ descends from $I(P_v)$. The latter implies that every vertex of τ descends from $I(P_v)$. Thus the claim holds.

The three other possibilities are $f(z) \notin t(w) \cup t(z)$, $f(z) \in t(z) - t(w)$, and $f(z) \in t(w) \cap t(z)$ (i.e., $t(w) = t(z)$). The argument for each is similar to the one just given. □

The lemma justifies the term "nonmerging ear": It is easy to see that `Multi-Merge` does not perform any unions for a nonmerging ear.

For the rest of this paper, all sets $t(a)$ refer to their value at the end of Phase II unless explicitly stated otherwise. The *(component) cluster* of an ear P with first vertex a consists of all descendants in F of vertices in $I(P) - t(a)$. For example, in Figure 2 if $P_b \cap t(a) = \{a, c, e\}$, then the cluster of P_b is $\{b, d, f, i, j, k\}$. A cluster can be empty; i.e., we can have $I(P) \subseteq t(a)$. For instance, any short ear has an empty cluster. We will be interested only in nonempty clusters, which occur only for long ears. The next lemma gives basic properties of clusters; the term "cluster" is motivated by property (ii).

LEMMA 4.2. *Let K be the cluster of an ear with first vertex a.*

(i) $K \cap t(a) = \emptyset$.

(ii) *K is a union of connected components of $G - t(a)$.*

Proof. Let K be the cluster of ear P.

(i) Suppose $y \in K \cap t(a)$. In F, y has an ancestor $x \in I(P) - t(a)$. Now $t(y)$ contains a, an ancestor of x, but not x itself. This contradicts Lemma 4.1.

(ii) By part (i), it suffices to show that every edge leaving K goes to $t(a)$. A tree edge leaving K must be an edge of P. Since $P \subseteq K \cup t(a)$, the edge goes to $t(a)$. Now

suppose a back edge yb (with b an ancestor of y) leaves K but does not go to $t(a)$. There are two possibilities.

Case 1. $y \in K$ and $b \notin K \cup t(a)$.

$b \notin P$ since $P \subseteq K \cup t(a)$. Hence b is a proper ancestor of a. Edge yb traverses $t(y)$, $t(a)$, and $t(b)$. These sets are distinct at the end of Phase II. ($b \notin t(a)$ by Case 1, $y \notin t(a)$ by part (i), and $b \notin t(y)$ by Lemma 4.1 with $a \notin t(y)$.) Hence the sets are distinct when line 2 of `Short_Ear` is executed with $x = y$ and $i = 2$. Thus lines 4–5 add a back edge yz with z an ancestor of b. The subsequent execution of `Multi-Merge`(y, z) merges $t(y)$ and $t(a)$, which is a contradiction.

Case 2. $y \notin K \cup t(a)$ and $b \in K$.

The definition of K implies that $b \in P$. Let x be the deepest ancestor of y in P. Since $y \notin K$, $x \in t(a)$. Thus b is a proper ancestor of x. Edge yb traverses $t(y)$, $t(x)$, and $t(b)$. These sets are distinct at the end of Phase II. ($y, b \notin t(x) = t(a)$ by Case 2 and $b \notin t(y)$ by Lemma 4.1 with $x \notin t(y)$.) Now the argument follows Case 1: Lines 4–5 of `Short_Ear` add a back edge yz with z an ancestor of b, and `Multi-Merge`(y, z) merges $t(y)$ and $t(x) = t(a)$, which is a contradiction. □

Define

$$\kappa = \text{the total number of nonempty clusters.}$$

(See Figure 4.) Call an ear *depleted* if all its vertices belong to the same t-set. Equivalently, ear P defined by a, y, z is depleted if $I(P) \subseteq t(a)$ (since we always have $z \in t(a)$). Thus κ equals the number of nondepleted ears.

LEMMA 4.3. $\varepsilon \geq (3/2)(n - 1 + \kappa - \nu)$.

Proof. Use the t-partition in the component lower bound. Consider a set $t(a)$. Let κ_a nondepleted ears have their first vertex in $t(a)$. Each of these ears gives a nonempty cluster. All of these clusters for $t(a)$ are pairwise disjoint (Lemma 4.2(i)). Hence $c(G - t(a)) \geq \kappa_a$. This gives a total of $\geq \kappa$ components in the component lower bound.

If $r \notin t(a)$, then $t(a)$ has at least one more component. To prove this, it suffices to show that r does not belong to any cluster of $t(a)$. This follows since r, as the root of F, does not descend (in F) from any internal vertex of any ear.

The number of distinct sets $t(a)$ is n decreased by the number of union operations, $n - \nu$. So the previous observation gives $n - \nu - 1$ more components in the component lower bound. We conclude that the total number of components in the component lower bound is $\geq \kappa + n - \nu - 1$. This gives the lemma. □

The following inequality has some slack in it, but see the remark after Lemma 4.5.

LEMMA 4.4. $\kappa \geq \overline{\mu} - \nu/2$.

Proof. Consider a nonmerging depleted ear P with first vertex a. Since each internal vertex gets merged into $t(a)$, we can associate $|I(P)|$ unions with P. We will prove $|I(P)| \geq 2$. Since a nonmerging ear is either depleted or nondepleted, we get $2\overline{\mu} \leq \nu + 2\kappa$ as desired.

We need only show that a nonmerging ear P with one internal vertex is not depleted. Let P be a, y, z. Lemma 3.1 shows that y is a leaf of T. So it suffices to prove the following claim. The claim drops the assumption that a, y, z has one internal vertex, since we need this more general fact in section 5.

CLAIM. *A nonmerging ear a, y, z with y a leaf has $t(y)$ a singleton.*

To prove this, we need only show that Phase II does not add a back edge yw as a short ear. When ear a, y, z is created, any back edge yw has w either descending from a or belonging to $t(a)$. So Lemma 4.1 (with $x = a$) shows that yw only traverses

two distinct sets, $t(y)$ and $t(w)$. Thus line 5 of `Short_Ear` does not add a back edge from y, even when $i = 2$. □

We can now bound the approximation ratio.

LEMMA 4.5. *The algorithm has approximation ratio* $\leq 14/9$.

Proof. Define the quantity δ to satisfy

$$\overline{\mu} - \sigma - \chi = (2/3)\gamma + \delta. \tag{5}$$

Thus $|A| \leq 2(n-1) + (2/3)\gamma + \delta$. We will show that

$$(4/3)\varepsilon \geq 2(n-1) + \delta.$$

The tree carving lower bound $\varepsilon \geq 3\gamma$ implies $(2/9)\varepsilon \geq (2/3)\gamma$. Thus $|A| \leq (4/3 + 2/9)\varepsilon = (14/9)\varepsilon$ as desired.

If $\delta \leq 0$, then the degree lower bound $\varepsilon \geq (3/2)n$ implies $(4/3)\varepsilon \geq 2n \geq 2(n-1) + \delta$ as desired. Hence we assume $\delta > 0$. Lemmas 4.3 and 4.4 combined give $\varepsilon \geq (3/2)(n - 1 + \overline{\mu} - 3\nu/2)$.

Some algebra shows $\overline{\mu} - 3\nu/2 \geq 3\delta$ as follows: Substitute (3) into (5) to get $\overline{\mu}/3 - \sigma - \chi = 2\mu/3 + \delta$ or, equivalently,

$$\overline{\mu} = 2\mu + 3(\sigma + \chi + \delta).$$

Combining with (4),

$$\overline{\mu} - 3\nu/2 = \mu/2 + 3\chi/2 + 3\delta \geq 3\delta.$$

We have shown $\varepsilon \geq (3/2)(n + 3\delta - 1)$. Hence $(4/3)\varepsilon \geq 2(n + 3\delta - 1) \geq 2(n-1) + \delta$. □

As already mentioned, Lemma 4.4 has some slack. However, the interested reader can check that even if we replace the term $\overline{\mu}$ with γ in that lemma, an argument similar to the above does not yield a lower approximation ratio.

5. Sharper analysis. This section proves that a natural implementation of the algorithm has approximation ratio $\leq 3/2$. Section 5.1 states three rules we require in the implementation; it also introduces some basic concepts for the analysis. Section 5.2 discusses how we use the breaking off operation. Section 5.3 proves the 3/2 bound, assuming a key inequality. Finally, section 5.4 proves the key inequality.

5.1. Algorithm rules and basic notions. We begin with some additional terminology. We often designate an ear a, y, z as a, z. The last internal vertex of a long ear (y) is its *tip*. Each vertex v is uniquely classified as a tip or nontip since P_v is unique. We often apply terminology for an ear to its tip; e.g., a merging tip is a tip whose ear is merging. The main issue of section 5 is bounding the number of nonmerging depleted ears (recall the proof of Lemma 4.4). Towards this end, let $\mathcal{Y}$ denote the set of all nonmerging depleted tips. A *child ear* of vertex x is a long ear whose first edge goes to a child of x.

To prove the upper bound, we incorporate several rules specifying choices made by the algorithm. For each rule, we specify the line number to which it applies.

Rule 1 (line 3(ii) of `Long_Ear`). Once y is chosen, z is chosen so a merging ear from y merges as many t-sets as possible.

Rule 2 (line 6 of `Long_Ear`). Vertex x progresses through the internal vertices of P_b in order; i.e., x moves from b to y.

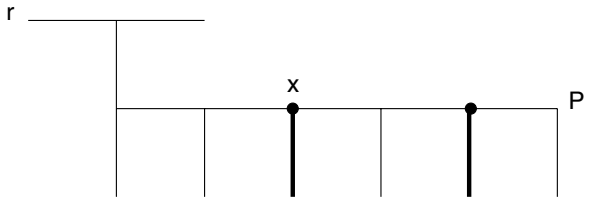

FIG. 5. *$t(x)$ contains the two solid vertices but not the first vertex of P. The root cluster of $t(x)$ consists of all vertices except the ones on the two heavy paths.*

Rule 3 (line 6 of `Long_Ear`). The first child ear of vertex x is merging if possible.

Rule 3 refers to the first child ear created at vertex x. We also call this the *first merging ear at x* (Lemma 5.10).

The following simple consequence of line 3(ii) is useful.

PROPOSITION 5.1. *Consider a back edge xz, where x properly descends from a tip y.*

(i) *If y is nonmerging, then z is a proper descendant of $f(y)$.*

(ii) *If y is merging, then z descends from $f(y)$ or belongs to $t(f(y))$.* □

The analysis of section 4 uses the notion of a cluster of an ear, and the proof of Lemma 4.3 shows there are additional components containing the root. We will call these "root clusters," defined formally as follows. Consider a vertex $x \neq r$ that is the shallowest vertex in $t(x)$. Every vertex in $t(x)$ has an ancestor in $t(x) \cap I(P_x)$. (This follows from the claim of Lemma 4.1.) The *root cluster* of $t(x)$ consists of all vertices that do not descend in F from $I(P_x) \cap t(x)$. (Contrast this with the definition of ear cluster in section 4; see Figure 5.) Equivalently, the root cluster consists of all vertices that do not descend in F from a vertex of $t(x)$.

In addition we use the term *clusters of $t(x)$* to refer to all clusters associated with the set $t(x)$; specifically the term refers to the root cluster of $t(x)$ plus the cluster of each ear a, y, z that has $a \in t(x)$.

We will bound the number of nonmerging depleted ears by associating various sets with them. The following notions are used to accomplish this.

DEFINITION 5.2. *For a vertex $y \in \mathcal{Y} \cup r$, T_y denotes the set of all vertices that descend from y but no deeper vertex of $\mathcal{Y} \cup r$. The* representative vertex *of a set is*

- *for an edge, its deeper vertex;*
- *for an ear, its first vertex;*
- *for the cluster C of an ear, the shallowest vertex of C;*
- *for the root cluster of $t(a)$, the shallowest vertex of $t(a)$;*
- *for a component C of a t-set, the representative of the cluster containing C.*

A set with representative vertex $v \in T_y$ is launched by *any ancestor of v in T_y. The set is* properly launched *if $v \in T_y - y$.*

Note that the representative of a short ear e is the same when we consider e to be an ear or an edge. In the last bulleted item, a component C of t-set $t(a)$ refers to a connected component of $G - t(a)$ (recall the proof of Lemma 4.3).

Any set S with a representative vertex is launched by exactly one vertex of $\mathcal{Y} \cup r$. So S is launched by at most one tip $y \in \mathcal{Y}$. This is our mechanism for establishing "ownership" of sets. Note that, in such an ownership relation, we cannot rely on any particular relationship between $t(y)$ and any t-set associated with S. For instance, if S is a cluster of $t(a)$ that is launched by tip $y \in \mathcal{Y}$, we might hope that $t(y) = t(a)$, but this is not true in general.

Throughout this entire section we fix a smallest 3-ECSS H of G for the analysis.

5.2. Breaking off operation. Our analysis starts with the t-partition of V and forms an enhancement, called the *t*-partition*, by doing a number of break off operations. This section describes these operations. It also proves a lemma, allowing us to enhance the t-partition to the t*-partition yet carry out the subsequent analysis by referring only to t-sets.

Section 2 defined the operation of breaking off in general. Now we specify this operation for our analysis. Initially the t*-partition is identical to the t-partition. Let b be a vertex with parent p. To *break off vertex* b means to replace $t^*(p)$ with $t^*(p)-B$ and B, for B the set of descendants of b belonging to $t^*(p)$. Vertex b is *breakable* if

(i) $t(p)$ contains at least one descendant of b;

(ii) at most one component of $H - t(p)$ is adjacent (in H) to both descendants and nondescendants of b;

(iii) b or p is a tip;

(iv) $b \notin \mathcal{Y}$.

Conditions (iii)–(iv) can be weakened, but they suffice for our purposes.

The t*-partition is formed by breaking off zero or more breakable vertices of each t-set. Let us describe how a t-set, say $t(a)$, gets partitioned into t*-sets. Choose a as the shallowest vertex in $t(a)$. Suppose we break off h vertices b_i whose parent p_i belongs to $t(a)$, $i = 1, \ldots, h$. The set $t(a)$ gets partitioned into $h+1$ t*-sets, specifically the vertices of $t(a)$ that descend from b_i but no deeper b_j, for $i = 0, \ldots, h$. Here we take a to be b_0. It is easy to see that condition (i) implies each of these t*-sets is nonempty. In Definition 2.5 the parent of the t*-set of b_i is $t^*(p_i)$ $(i \geq 1)$.

LEMMA 5.3. *Suppose the t*-partition is defined by breaking off* ≤ 1 *breakable vertex in each set* T_y.

(i) *The t*-partition is an enhancement of the t-partition (in graph* H*).*

(ii) *If no vertex of* T_y *is broken off, then an edge properly launched by* y *with both ends in the same t-set is a nonbridging edge (of the t*-partition).*

(iii) *Suppose* y *launches a bridging component* C *of* $H - t(a)$ *for some vertex* a*. Then a vertex* $b \in T_y$ *whose parent belongs to* $t(a)$ *was broken off and* C *is adjacent to both descendants and nondescendants of* b*.*

Proof. For any $y \in \mathcal{Y} \cup r$, let $\overline{T}_y$ be the set T_y enlarged at its leaves; i.e., add to T_y each $w \in \mathcal{Y}$ having $f(w) \in T_y$. Let $X_y = I(P_y) \cup \overline{T}_y$. (For $y = r$ recall $I(P_r) = \emptyset$.)

CLAIM 1. *For any vertex* a*,* $t(a) \cap X_y$ *is contained either in one t*-set or in some t*-set and its parent set.*

Proof. The only possible breakable vertices of X_y are the breakable vertices b of T_y and the first vertex b' of $I(P_y)$ (recall (iii)–(iv) of the definition). If we break off b', $t(a) \cap X_y$ is still contained in one t*-set. If we break off a vertex $b \in T_y$ with parent p, the t*-set changes only if $p \in t(a)$. In that case, $t(a) \cap X_y$ is contained in a t*-set descending from b and its parent set $t^*(p)$. □

CLAIM 2. *An edge* vz *with deeper vertex* $v \in t(z)$ *either has both ends in the same t*-set or joins a t*-set and its parent set.*

Proof. Choose y so $v \in \overline{T}_y - y$. Proposition 5.1(i) shows $z \in X_y$. Together with Claim 1 this implies Claim 2. □

Claim 2 gives condition (i) of Definition 2.5 for the t*-partition. It also gives (ii) of the current lemma.

CLAIM 3. *Consider a cluster* C *of* $t(a)$ *with representative vertex* c*. Suppose* $c \in T_y - y$ *for* $y \in \mathcal{Y} \cup r$*. Then any edge with exactly one end in* C *has the other end in* X_y*.*

Proof. We remind the reader that C is either a root cluster or an ear cluster. Our assertions must be checked in both cases! For root clusters it is convenient to refer to Figure 5.

We begin by noting that vertex y of the claim always exists; this is equivalent to $c \notin \mathcal{Y} \cup r$. P_c is not depleted (for both cluster types). This implies $c \notin \mathcal{Y}$. $c \neq r$ follows from the fact that r is not the representative of any cluster (of either type).

Now let vz be an edge, with deeper vertex v, having exactly one end in C. (That end can be v or z.) Suppose v is a proper ancestor of c. If C is an ear cluster, this implies $v, z \notin C$. If C is a root cluster, it implies $v, z \in C$. In both cases, vz violates its definition. So assume that v descends from c.

Since $c \neq y$, we have $v \neq y$. If $v \in \overline{T}_y - y$, then $z \in X_y$ (Proposition 5.1(i)) and the claim holds (regardless of which vertex is in C). So assume that v is a proper descendant of some $w \in \overline{T}_y \cap \mathcal{Y} - y$.

Let b be the first vertex of $I(P_w)$. We show that (a) b is an ancestor of z, (b) $b \notin P_c$, and (c) b descends from c. (a) follows from the definition of b and Proposition 5.1(i). (b) follows since P_c is not depleted but P_w is. For (c), recall our assumption that v descends from c. The tree path from v to c enters T_y along P_w and then goes to P_c. (c) follows.

(b) and (c) imply either that all descendants of b belong to C or that none do (for either ear type). With (a) this makes vz violate its definition. □

Now consider a component C of $H - t(a)$. Since C is contained in a cluster, Claims 3 and 1 show that all the neighbors of C belong either to the same t*-set or to some t*-set and its parent. Together with condition (ii) of the definition of breakable, this gives condition (ii) of Definition 2.5. It also gives (iii) of the current lemma. □

5.3. The approach. In the following definitions we fix a t*-partition formed according to Lemma 5.3. If a cluster of $t(x)$ contains $1 + s$ connected components of $H - t(x)$, it has s *surplus components*. If $3 + s$ edges of H cover an edge $e \in C$, then e is *redundantly covered* s times. Define

κ = the number of nondepleted ears, plus the number of surplus components in clusters;
β = the number of break offs that form the t*-partition;
θ_a = the total number of surplus edges for all nonbridging components of H;
θ_b = the number of nonbridging edges of H;
θ_c = the number of edges of H not covering an edge of C or redundantly covering an edge of C;
θ_d = the number of vertices that have degree > 3 in H.

(Subscript b stands for "both ends," c stands for "carving," and d stands for "degree.") Our new definition of κ generalizes the definition in section 4. β, θ_a, and θ_b are defined as at the end of section 2.

The next section bounds the number of nonmerging ears, showing

$$\overline{\mu} \leq \kappa + \chi + \beta + \theta_d + (\theta_a + 2\theta_b + \theta_c)/3. \tag{6}$$

We now demonstrate that this implies the approximation ratio is $\leq 3/2$.

We use the following three lower bounds:

$$\varepsilon \geq 3\gamma + \theta_c; \tag{7}$$

$$\varepsilon \geq (3/2)n + \theta_d/2; \tag{8}$$

$$\varepsilon \geq (3/2)(n - 1 + \kappa + \beta - \nu) + \theta_b + \theta_a/2. \tag{9}$$

It is clear that (7) and (8) are true. To prove (9), recall that Lemma 4.3 is proved using the component lower bound. Instead use (2), which is our extension of the component lower bound. Inequality (9) follows easily.

Define the quantity δ to satisfy

$$\overline{\mu} - \sigma - \chi = \gamma/2 + \delta. \tag{10}$$

(δ may be positive, negative, or 0.) Thus

$$|A| \le 2(n-1) + \gamma/2 + \delta.$$

Combining 1/6 times (7) with 4/3 times (8) gives $(3/2)\varepsilon \ge 2n + \gamma/2 + \theta_c/6 + (2/3)\theta_d$. Hence we can assume

$$\delta > \theta_c/6 + (2/3)\theta_d$$

since otherwise we are done. To handle this case we will show that (7) and (9) imply

$$(3/2)\varepsilon \ge 2(n-1) + \gamma/2 + 4\delta - 2\theta_d - \theta_c/2. \tag{11}$$

Inequality (11), together with our assumption $3\delta > \theta_c/2 + 2\theta_d$, implies the desired result.

We begin by reexpressing $\overline{\mu}$ as follows: Substitute (3) into (10) to get $\overline{\mu}/2 - \sigma - \chi = \mu/2 + \delta$ or, equivalently, $\overline{\mu} = \mu + 2(\sigma + \chi + \delta)$. Combining with (4), we have

$$\overline{\mu} - \nu - \chi = 2\delta.$$

Combining this with (6) gives

$$\kappa + \beta - \nu \ge \overline{\mu} - \chi - \nu - \theta_d - (\theta_a + 2\theta_b + \theta_c)/3 = 2\delta - \theta_d - (\theta_a + 2\theta_b + \theta_c)/3.$$

Thus (9) gives

$$\varepsilon \ge (3/2)(n - 1 + 2\delta - \theta_d - (\theta_a + 2\theta_b + \theta_c)/3) + \theta_b + \theta_a/2 = (3/2)(n - 1 + 2\delta - \theta_d - \theta_c/3).$$

Combining 4/3 times this inequality with 1/6 times (7) gives the desired inequality (11).

5.4. Bounding $|\mathcal{Y}|$. We begin by noting that this section is concerned with both the given graph G and the fixed 3-ECSS H. Proposition 5.4 and Lemmas 5.5–5.9 are properties of G alone. The last three results, Lemmas 5.10–5.12, depend on H.

PROPOSITION 5.4. *Let a 3-edge connected graph $G = (V, E)$ have degree function d. Suppose sets X, X', and x form a partition of V, and $d(x) = 3$. Then $d(X, X') \ge 2$.*

Proof. Without loss of generality, assume $d(x, X) \le 1$. Hence $3 \le d(X) = d(X, X') + d(X, x) \le d(X, X') + 1$ as desired. □

A long ear is *tight* if its last tree edge is a carving edge. Otherwise the ear is *loose*. Every pendant edge of T is obviously tight. The following lemma generalizes this fact.

LEMMA 5.5. *The tip of a loose ear has a merging child ear.*

Proof. Let the loose ear contain the carving edge vw, and let its last edge be xy. Thus y is the ear's tip; possibly $w = x$. Consider the depth-first search of Phase I that finds carving edges. Edge xy does not force a back edge to be added. So the search added a back edge uz from a descendant u of y to an ancestor z of x. Furthermore,

the tree path from y to u contains a carving edge. When `Long_Ear` constructs P_y, y, u, z is a possible merging ear at y. So Rule 3 shows y has a merging child. □

We note a related property for use in Lemma 5.9. A tight ear has only nonmerging children. This follows since a back edge covers at most one carving edge.

LEMMA 5.6. *Let y be a nonmerging tip. An ear of A with its first vertex descending from y has its last vertex properly descending from $f(y)$.*

Proof. Let the ear be a, u, z and, for the sake of contradiction, assume z is an ancestor of $f(y)$. Since u is a descendant of y, Proposition 5.1(i) implies $u = y$. Since u descends from a, we get $a = y$; i.e., the ear is short. But the algorithm never adds a short ear originating at a nonmerging tip (this is proved in the claim of Lemma 4.4). □

The next lemma consists of two similar parts. After stating the lemma we give an example showing that a plausible generalization is false.

LEMMA 5.7. *Let an ear P have first vertex a and tip u.*

(i) *The first ear that adds a vertex of $I(P)$ to $t(a)$ is launched by a.*

(ii) *The first ear that adds a proper ancestor of u to $t(u)$ is launched by u.*

Example 1. For an arbitrary vertex x, the first ear that adds a proper ancestor of x to $t(x)$ need not be launched by x. In Figure 2 take x to be d. Let $e \in \mathcal{Y}$. After the short ear h, c is added the short ear g, d adds c to $t(d)$. But g, d is launched by e and is not launched by d. A similar example uses merging ears: The first merging ear goes from e to c and the second from e to d.

Proof. (i) Consider the first execution `Multi-Merge`(b, z) that adds a vertex of $I(P)$ to $t(a)$. b descends from $I(P)$ and z is an ancestor of a. We claim that ear b, z was launched by the first vertex of $I(P)$. (This is slightly stronger than the lemma.) We prove this by showing that the tree path from a to b does not contain a vertex $w \in \mathcal{Y} - a$.

Suppose w exists. $f(w)$ is a descendant of a (possibly equal to it). So Lemma 5.6 shows z is a proper descendant of a, which is a contradiction.

(ii) Consider the first execution `Multi-Merge`(b, z) that adds a proper ancestor of u to $t(u)$. b descends from u and z is a proper ancestor of u. Suppose the tree path from u to b contains a vertex $w \in \mathcal{Y} - u$. Since u is a tip, $f(w)$ is a descendant of u (possibly equal to it). So Lemma 5.6 shows that z is a proper descendant of u, which is a contradiction. □

Remark. The stronger version of the lemma that we proved leads to a stronger version of Lemma 5.8, but we do not require it.

A *surplus ear* is a merging or short ear that causes a surplus union (i.e., a union counted in χ).

LEMMA 5.8. *Suppose $y \in \mathcal{Y}$ does not launch a surplus ear and some back edge xz has $x \in T_y - y$.*

(i) *If z is an ancestor of $f^3(x)$, then $f^3(x)$ launches a merging ear.*

(ii) *If z is a proper ancestor of $f^2(x)$ and $f^2(x)$ is a tip, then $f^2(x)$ launches a merging ear.*

Remark. A plausible common generalization of (i)–(ii) fails: Assuming the lemma's hypothesis, z can be a proper ancestor of $f^2(x)$ without $f^2(x)$ launching a merging ear. For instance, let xz be the back edge jc in Figure 2, and assume the merging ear scenario of Example 1.

Proof. Proposition 5.1(i) shows that z is a proper descendant of $f(y)$. Hence any vertex $f^i(x)$ that descends from z, properly or not, belongs to T_y, e.g., $i \leq 3$ in (i) and $i \leq 2$ in (ii).

(i) Suppose $f^3(x)$ does not launch a merging ear. We prove the lemma by showing y launches a surplus ear. Lemma 5.7(i) shows that $x, f(x), f^2(x)$, and $f^3(x)$ are in distinct t-sets at the end of `Long_Ear`. Hence Lemma 5.7(i) implies $f^3(x)$ launches a surplus ear during `Short_Ear`$(s, 3)$—either before x is scanned or when a back edge at x (e.g., xz) is added as a short ear.

(ii) The argument is similar to (i). Suppose $f^2(x)$ does not launch a merging ear. We show y launches a surplus ear. Lemma 5.7(i)–(ii) shows that $x, f(x), f^2(x)$, and z are in distinct t-sets at the end of `Long_Ear`. Hence Lemma 5.7(i)–(ii) implies that $f^2(x)$ launches a surplus ear during `Short_Ear`$(s, 3)$—either before x is scanned or when a back edge at x (e.g., xz) is added as a short ear. □

An ear P with first vertex a is *penetrated* if $I(P) \cap t(a) \neq \emptyset$.

LEMMA 5.9. *A nonleaf tight tip either launches a surplus ear or has a penetrated child ear.*

Proof. Let u be a nonleaf tight tip. In this proof, say that $t(u)$ *is enlarged by ear* b, z if the execution `Multi-Merge`(b, z) changes $t(u)$ from a singleton set to a nonsingleton. Clearly, in this case z is an ancestor of u, which is an ancestor of b. If $b \neq u$, then some vertex x with $f(x) = u$ gets added to $t(u)$. This makes P_x a penetrated child ear of u.

The rest of the argument is in three cases. First suppose $t(u)$ is enlarged by a merging ear b, z. No child ear of u is merging (as remarked after Lemma 5.5). So b properly descends from u. The opening remark gives the lemma.

Suppose $t(u)$ is enlarged during `Short_Ear`$(s, 3)$ by a short ear b, z. Either $b \neq u$ or u launches a surplus ear. In both cases the lemma holds.

Finally, suppose $t(u)$ is a singleton at the end of `Short_Ear`$(s, 3)$. Let P be the first child ear of u. Some back edge xz joins a descendant x of $I(P)$ to a proper ancestor z of u, since G has no articulation point. Since `Short_Ear`$(s, 2)$ works bottom-up, some short ear launched by a descendant of $I(P)$ enlarges $t(u)$. (This either occurs before x is scanned or when a back edge at x, e.g., xz, is added as a short ear.) Again the lemma holds. □

LEMMA 5.10. *Suppose $y \in \mathcal{Y}$ does not launch a surplus component or a surplus ear. If the first merging ear at a given vertex of T_y is depleted and its tip u is a leaf, then u is breakable.*

Proof. Let $a = f(u)$. Thus $a \in T_y \cap t(u)$. We claim u is adjacent to at most one cluster C of $t(u)$; furthermore

$$C \cap P_a \neq \emptyset$$

if C exists. This claim implies the lemma. To prove this, we need only verify condition (ii) of the definition of breakable. (For condition (i) note that the parent of u belongs to $t(u)$.) Assume C exists; otherwise, condition (ii) is vacuous. The above set inequality makes P_a nondepleted (since it contains vertices of two t-sets). Hence $f(a) \in T_y$, and C (root cluster or ear cluster) is launched by y. Now the lemma's hypothesis shows that C consists of exactly one component of $H - t(u)$. Hence (ii) holds and u is breakable.

To prove the claim, first suppose $f(a) \notin t(u)$. So any ancestor of u belongs to $t(u)$ or to the root cluster C of $t(u)$. Since $f(a) \in C$, we have $C \cap P_a \neq \emptyset$, and the claim holds.

Next suppose $f(a) \in t(u)$. Consider an edge uz with $z \notin t(u)$, and suppose z does not descend from $f(a)$. When ear P_u is created, $t(a)$ is a singleton by Rules 2–3. Hence $a, f(a)$, and z are in distinct t-sets at that time. (Note that at the end of Phase II, $t(u) = t(a) = t(f(a))$, so $z \notin t(f(a))$.) Rule 1 shows P_u is a surplus ear.

But the lemma assumes this is not the case (recall $a \in T_y$). Hence z descends from $f(a)$. We conclude that cluster C of P_a is the only cluster of $t(u)$ that is adjacent to u. Obviously $C \cap P_a \neq \emptyset$ if $C \neq \emptyset$. □

For any $y \in \mathcal{Y}$ define

$\kappa(y)$ = the number of merging nondepleted ears launched by y plus the number of surplus components launched by y;
$\chi(y)$ = the number of surplus ears launched by y;
$\beta(y)$ = the number of breakable vertices in $T_y - y$;
$\theta_a(y)$ = the number of surplus edges of H leaving components launched by y;
$\theta_b(y)$ = the number of edges of H properly launched by y having both ends in the same t-set;
$\theta_c(y)$ = the number of edges of H launched by y not covering a carving edge or redundantly covering a carving edge launched by y;
$\theta_d(y)$ = the number of vertices of T_y having degree > 3 in H.

$\chi(y)$ is the only one of the above quantities that is a function of G, not H. None of the quantities involve the t*-partition. The definitions of $\kappa(y), \theta_a(y)$, and $\theta_b(y)$ differ slightly from their counterparts in section 5.3. The differences are reconciled in the last argument of this section.

LEMMA 5.11. *For $y \in \mathcal{Y}$ let $u \in T_y$ be the tip of a depleted ear. Suppose u does not launch a merging ear and u has a penetrated child ear. Then either*

(i) *u has a breakable child belonging to T_y, or*

(ii) *$\theta_a(y) + 2\theta_b(y) + \theta_c(y) \geq 3$, or*

(iii) *$\kappa(y) + \chi(y) + \theta_d(y) \geq 1$.*

Proof. The argument is illustrated in Figure 6. Lemma 5.5 shows u is tight. Let U be the set of proper ancestors of u. Consider the components of $H - t(u)$. For an arbitrary ear X, C_X denotes the cluster of X (which may be empty). For $S \subseteq V$, $\overline{S}$ denotes $V - S$. We assume the lemma is false and derive a contradiction.

CLAIM 1. *If S is the set of all descendants of a child of u, then $d_H(S, U) \geq 2$. Furthermore each edge from S to U is launched by y.*

Proof. For the first part we have $d_H(u) = 3$, since otherwise $\theta_d(y) \geq 1$ and (iii) holds. Now Proposition 5.4 shows $d_H(S, U) = d_H(S, \overline{S} - u) \geq 2$.

For the second part, an edge e from S to U is launched by y since u is tight and e cannot cover two carving edges. □

Remark. If u has ≥ 3 children, the claim shows $\theta_c(y) \geq 3$ and we are done. However, we do not use this principle.

The rest of the argument focuses on P, a penetrated child ear of u. Let p be the tip of P. Let D be the set of all descendants of the child of u that belongs to P.

CLAIM 2. *Let xz be an edge of G with $x \in D$, $z \in \overline{D}$.*

(a) *$z \in t(u)$.*

(b) *Suppose x is a proper descendant in F of a vertex $w \in P$. Then $w \neq p$, the tree path from w to x contains no carving edge, and $x \in T_y$.*

(c) *x belongs to P or a child ear of $I(P)$. In the latter case $z = u$.*

Proof. (a) Since u is depleted, this part follows from Proposition 5.1(i)–(ii).

(b) Since P is nonmerging, $w \neq p$ by Proposition 5.1(i). Hence $w \in T_y$. The lemma's hypothesis shows w is not the first vertex of a merging ear. So Rule 3 implies there is no carving edge on the tree path from w to x. This makes $x \in T_y$.

(c) First suppose x is a proper descendant of a child ear of $I(P)$. (b) implies $x \in T_y$. Now Lemma 5.8(i) and the hypothesis of our lemma show y launches a surplus ear. Thus $\chi(y) \geq 1$, and (iii) holds.

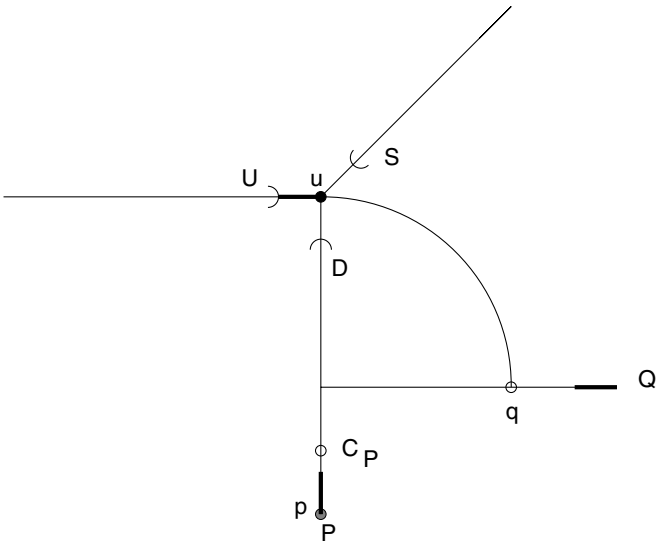

FIG. 6. *Proof of Lemma* 5.11. *Hollow vertices are not in* $t(u)$. *Heavy lines denote carving edges.*

Next suppose x belongs to a child ear of $I(P)$ and $z \in U$. As before, $x \in T_y$. Since u is a tip, Lemma 5.8(ii) and the hypothesis of our lemma show y launches a surplus ear. This gives (iii). □

CLAIM 3. *P is nondepleted and tight. Hence cluster C_P is nonempty.*

Proof. If P is depleted, then by Claims 2(a) and 2(c) the two edges of Claim 1 have both ends in $t(u)$. The deeper end of each edge belongs to $T_y - y$ since u is tight and no edge covers two carving edges. This makes $\theta_b(y) \geq 2$, so (ii) holds. We conclude P is nondepleted.

This implies $p \in T_y$. Hence p does not have a merging child. Lemma 5.5 shows P is tight. □

CLAIM 4. *H has an edge with deeper end in $D \cap T_y$ that does not cover a carving edge. So $\theta_c(y) \geq 1$.*

Proof. The child of u on P is not breakable (otherwise (i) holds). Since P is penetrated, this means ≥ 2 components of $H - t(u)$ are adjacent (in H) to both D and $\overline{D}$. Since C_P is launched by y, it does not contain a surplus component (otherwise $\kappa(y) \geq 1$ and (iii) holds). So another cluster of $t(u)$, root or ear, is adjacent to both D and $\overline{D}$. Claim 2(a) shows the cluster is C_Q for an ear Q descending from $I(P)$. Claim 2(c) shows Q is a child ear of $I(P)$ and a vertex $q \in C_Q \cap Q$ is adjacent to u in H. Since P is tight, Claim 2(b) shows qu does not cover a carving edge. It also shows $q \in T_y$. Hence $\theta_c(y) \geq 1$. Claim 4 follows. □

CLAIM 5. *Every edge from D to $\overline{D}$ in H goes from a cluster of $t(u)$.*

Proof. Let xz be an edge of H with $x \in D$ and $z \in \overline{D}$. Assume $x \in t(u)$; otherwise we are done. Claim 2(a) shows $z \in t(u)$. Now it suffices to show $x \in T_y$. For that makes $\theta_b(y) \geq 1$ which, with $\theta_c(y) \geq 1$ (Claim 4), gives (ii).

If $x \in P$, then Claim 3 shows $x \in T_y$. If $x \notin P$, then Claims 2(b) and 2(c) show $x \in T_y$. □

CLAIM 6. *$d_H(C_P, U) \geq 2$. $d_H(C_P, D - C_P) \geq 2$. Hence $\theta_a(y) \geq 1$.*

Proof. Claim 1 shows that at least two edges e of H join D and U. Claim 5 shows each e has exactly one end in a cluster C of $t(u)$ with $C \subseteq D$. Now Claim 2(c) shows

$C = C_P$. So the two edges e give the first assertion of Claim 6.

For the second assertion, first note that $D - C_P \neq \emptyset$ since P is penetrated. We have already assumed $d_H(u) = 3$. By the preceeding paragraph, an edge of H leaving $D-C_P$ does not go to U. Since V is partitioned into sets $D-C_P$, $X = V-(D-C_P)-u$, and u, Proposition 5.4 shows $d_H(D - C_P, C_P) = d_H(D - C_P, X) \geq 2$.

We turn to the third assertion of Claim 6. C_P is launched by y, and we have already noted that it does not contain a surplus component (Claim 4). Hence $\theta_a(y) \geq 1$. □

CLAIM 7. *u has only one child. $d_H(U, u) \leq 1$.*

Proof. Suppose either part of the claim is false. We first show at least four edges of H launched by y cover the carving edge of P_u. If u has two children, this follows from Claim 1. If $d_H(U, u) \geq 2$, it follows from these two edges plus Claim 1 applied to D.

Now Claim 4 shows $\theta_c(y) \geq 2$. With $\theta_a(y) \geq 1$ (Claim 6) we get (ii). □

CLAIM 8. *$d_H(D, u) \leq 1$.*

Proof. Claim 5 shows $d_H(D, u) = \sum\{d_H(C, u) : \text{cluster } C \subseteq D\}$. First consider $C = C_P$. If $d_H(C_P, u) \geq 1$, then five edges leave C_P (Claim 6) so $\theta_a(y) \geq 2$. With $\theta_c(y) \geq 1$ (Claim 4), we get (ii). We conclude $d_H(C_P, u) = 0$.

Next consider $C \neq C_P$. Claim 2(b) shows that an edge from C to u does not cover a carving edge and, further, C is launched by y. If H contains two such edges, then $\theta_c(y) \geq 2$, and $\theta_a(y) \geq 1$ (Claim 6) gives (ii). We conclude there is only one such edge, as desired. □

Claims 7 and 8 show $d_H(u) \leq 2$, the desired contradiction. □

LEMMA 5.12. *Any tip $y \in \mathcal{Y}$ has*

$$(\kappa + \chi + \beta + \theta_d)(y) + (\theta_a + 2\theta_b + \theta_c)(y)/3 \geq 1.$$

Proof. Assume $\kappa(y) = 0$; otherwise we are done. Hence any merging ear launched by y is depleted. Choose vertex $u \in T_y$ to be the tip of a depleted ear as follows. If y does not launch a merging ear, then $u = y$. In the opposite case, choose u as the tip of a merging ear a, u, z, where $a \in T_y$ has the greatest depth possible. If two merging ears have the same first vertex a, choose u on the first merging ear at a.

Vertex u belongs to T_y. Hence it does not launch a merging ear. Lemma 5.5 shows u is tight. Now consider two cases.

First suppose u is a leaf. P_u is merging since a nonmerging depleted tip is not a leaf (by the claim of Lemma 4.4). Assume $\chi(y) = 0$; otherwise we are done. Since $\kappa(y) = 0$, Lemma 5.10 shows that u is breakable. Hence $\beta(y) \geq 1$.

Now suppose u is not a leaf. Since we are assuming $\chi(y) = 0$, Lemma 5.9 shows that u has a penetrated child ear. Now the desired conclusion follows from Lemma 5.11. □

We can now achieve the goal of proving inequality (6). Recall the definitions of all the right-hand quantities from section 5.3. Apply Lemma 5.12 to each $y \in \mathcal{Y}$. Define the t*-partition by breaking off a breakable vertex in $T_y - y$ whenever $\beta(y) \geq 1$. Lemma 5.3(i) shows that this gives a valid enhancement. This defines the quantity β of (6).

Clearly

$$\sum\{\chi(y) : y \in \mathcal{Y}\} \leq \chi.$$

(Inequality may hold since we ignore surplus ears launched by r and, further, we do not count the total number of surplus unions.) Analogous inequalities hold for θ_c and

θ_d. Lemma 5.3(ii) shows

$$\sum\{\theta_b(y) : y \in \mathcal{Y},\ \beta(y) = 0\} \le \theta_b.$$

Lemma 5.3(iii) shows the analogous inequality for θ_a. Let $\overline{\mu}_n$ be the number of nonmerging nondepleted ears. Then

$$\sum\{\kappa(y) : y \in \mathcal{Y}\} \le \kappa - \overline{\mu}_n.$$

Since $|\mathcal{Y}| = \overline{\mu} - \overline{\mu}_n$, Lemma 5.12 implies

$$\overline{\mu} - \overline{\mu}_n \le \kappa - \overline{\mu}_n + \chi + \beta + \theta_d + (\theta_a + 2\theta_b + \theta_c)/3.$$

This amounts to (6).

6. Efficient implementation. This section presents an implementation of the algorithm that runs in time $O(m\alpha(m,n))$. It is straightforward to find the biconnected components and implement Phases I and III in linear time. So we limit our attention to Phase II. The implementation must incorporate Rules 1 and 3 of section 5 (Rule 2 is trivial). This section first describes how the t-partition is maintained and manipulated. Then it describes how long ears are constructed. The remaining details of Figure 3 are obvious.

6.1. The t-partition. We maintain the t-partition using the following properties of F. For any vertex x, let $\ell(x)$ be the ancestor of x in F that belongs to $t(x)$ and has minimum depth. Lemma 4.1 shows that $t(x)$ contains every vertex on the path in F from x to $\ell(x)$. Write $fl(x)$ for $f(l(x))$. Note that every $y \in t(x)$ has $f\ell(y) = f\ell(x)$. (Recall Figure 5.) In the following lemma assume $f(r) = r$.

LEMMA 6.1. *For any $i \ge 0$, a back edge xz traverses $> i+1$ distinct t-sets if and only if z is an ancestor of $[f\ell]^i(x)$ in T and $z \notin t([f\ell]^i(x))$.*

Proof. If $r \notin t(x)$, then the deepest ancestor of x in F not belonging to $t(x)$ is $fl(x)$. Hence for $i \ge 1$, if $r \notin t([fl]^{i-1}x)$, then the path in F from x to $[fl]^i(x)$ contains vertices in exactly $i+1$ distinct t-sets. If z is an ancestor (in T) of $[fl]^i(x)$ but not $[fl]^{i+1}(x)$ and $z \notin t([fl]^i(x))$, then an edge xz traverses exactly $i+2$ t-sets. □

The t-partition is maintained by a disjoint-set data structure. In addition, each set $t(x)$ is labelled by the node $f\ell(x)$. As already noted, this label is well defined. The labels allow `Multi-Merge` to be implemented using a number of finds proportional to the number of unions. Line 3 of `Short_Ear` is implemented using Lemma 6.1. It performs $O(1)$ finds per back edge, since i equals 2 or 3. Line 3(i) of `Long_Ear` uses one find per back edge.

To implement Rule 1 in line 3(ii) of `Long_Ear`, tentatively choose the back edge yz from y that has $z \notin t(a)$ and minimum $d(z)$. (Throughout this section $d(v)$ denotes the depth of vertex v in tree T, as in section 3.) Find the maximum index i with vertex $v = [fl]^i(a)$ descending from z. A merging ear with tip y merges at most $i+2$ t-sets. Any back edge yz' with z' an ancestor of v and $z' \notin t(v)$ gives such an ear. If no such z' exists, the back edge yz gives an ear merging $i+1$ t-sets, the maximum possible in this case. In either case, `Multi-Merge` performs at least i unions for this ear.

We conclude that the disjoint-set data structure performs a total of $O(m)$ find operations in a universe of n elements. Thus the total time for manipulating the t-partition is $O(m\alpha(m,n))$ [1].

6.2. Constructing long ears. This section describes how lines 2–3 of `Long_Ear` find edge $c \in C$ and back edge yz to construct the new ear. We accomplish this in linear time, implying the desired time bound for Phase II.

At any point during the execution of `Long_Ear`, let R be the subtree of edges of T that already belong to long ears. The idea of the implementation is to maintain an "active" subtree S, $R \subseteq S \subseteq T$. When xx' is chosen in `Long_Ear` (line 6), the subtree of S rooted at x' will lead to the edges of C that can be covered by new ears having xx' as their first edge. We now provide the details.

Phase I labels each back edge e with the carving edge $c[e]$ covered by e, if it exists; if not, then $c[e] = \Lambda$. It is easy to do this labelling in linear time after identifying the connected components of $T - C$.

In Phase II, say that vertex x gets *visited* the first time it is reached in line 6 of `Long_Ear`. Also, as we have implicitly done, say that a back edge is *directed to* its shallower vertex. `Long_Ear` maintains a list $L[c]$ for each $c \in C$, defined by

$$L[c] = \{e : e \text{ is a back edge covering } c \text{ and directed to an already visited vertex}\}.$$

Subtree S is maintained according to the following invariant.

S-Invariant. The pendant edges of $S - R$ are precisely the edges of $c \in C - R$ with $L[c] \neq \emptyset$.

`Long_Ear` constructs L lists and grows S as follows. When line 6 visits x, it first does some processing, described below, that implements Rule 3. Then it scans each back edge e directed to x. If $c[e] \neq \Lambda$, do the following: Add e to $L[c[e]]$. If $c[e]$ is not in S, join it into S by the tree path to its first ancestor already in S.

This procedure maintains the definition of L. It also preserves the S-Invariant, since the definition of tree carving shows the edges added to S do not descend from a pendant edge of $S - R$.

In order to implement Rule 3, we use another variable, edge c^*. If c^* is defined, then it is used as edge c of line 2 to form the merging ear required by Rule 3. We define c^* when visiting x. The complete procedure for visiting x is as follows.

When x is first reached in line 6, if x already has descendants in S not in P_x, follow a path in S from x to a pendant edge. Take that pendant edge to be c^*. Then scan the back edges directed to x, as described above. Finally, if c^* is defined, choose x' (for the first child ear at x) as the child of x that is an ancestor of c^*. Otherwise choose x' arbitrarily.

Note that when this procedure begins, all proper ancestors of x have been visited (by Rule 2) but no descendant of x has been visited. Hence for every edge $c \in C - R$ descending from x, $L[c]$ contains precisely the edges that can be used to form a merging child ear at x (by the definition of L). If such a c exists, the procedure's edge c^* qualifies, since the S-Invariant guarantees that $c^* \in C$ and $L[c^*] \neq \emptyset$.

Now we describe the implementation of lines 2–3 of `Long_Ear`(b). The purpose of line 2 is to define c. If c^* is defined, then $c = c^*$. This gives a merging ear, satisfying Rule 3.

If c^* is not defined, then follow a path in S from b to a pendant edge of S. Take that edge as c. The S-Invariant guarantees that $c \in C$ and $L[c] \neq \emptyset$. Furthermore, any edge of $L[c]$ can be used to form an ear P_b. This follows from the definition of $L[c]$, since a and all its ancestors have been visited, but no descendant of b has been visited.

Next we describe line 3. If $L[c]$ contains an edge yz with $z \notin t(a)$, choose one with maximum $d(y)$ to define a merging ear. Otherwise choose any edge $yz \in L[c]$ with

maximum $d(y)$ to define a nonmerging ear. In both cases, 3(ii) is satisfied. Note that for merging ears the procedure described in section 6.1 determines the final merging ear. Finally, line 4 adds the tree path from b to y to both R and S.

Appendix A. Analysis for *k*-edge connectivity. This appendix extends the analysis of section 5 to k-edge connectivity. Specifically, let B denote the set of nontree edges in the approximation algorithm's solution graph. (B consists of all edges added after Phase I.) We show that for any integer $k \geq 3$,

$$|B| \leq (5/2k)\ \epsilon_k. \tag{12}$$

(It is easy to see that this implies $|A| \leq (9/2k)\epsilon_k$. So for $k = 3$ we again have a 3/2 performance ratio.) This result is used in [5] to get an approximation algorithm for k-ECSS. We still assume Rules 1–3 of section 5.1 are used in the algorithm, but there are no other additional rules.

The core of the derivation is a new version of Lemma 5.11. (Taking $k = 3$, the new version can replace the one in section 5, but the new argument is slightly longer.) There are a number of additional changes, but all changes are in sections 5.3 and 5.4. The new versions of these sections are Appendices A.1 and A.2, respectively.

A.1. The approach for general *k*. We make some small changes in the definitions of our fundamental quantities as follows: If $k + s$ edges of H cover an edge $e \in C$, then e is *redundantly covered* s times. A vertex with degree $k + s$ in H has *surplus degree* s. Quantities $\kappa, \beta, \theta_a, \theta_b$, and θ_c are defined exactly as before. (For θ_c use the new definition of redundant covering.) The new definition of θ_d is

$$\theta_d = \text{the total surplus degree of all vertices in } H.$$

We modify the key inequality (6) in two ways, changing a factor 3 to k and changing the term involving θ_d as follows:

$$\overline{\mu} \leq \kappa + \chi + \beta + (\theta_a + 2\theta_b + \theta_c + \theta_d)/k. \tag{13}$$

The proof that this inequality implies (12) is entirely analogous to the proof given in section 5.3. For completeness, the rest of this section gives all the details.

Our three lower bounds are

$$\varepsilon_k \geq k\gamma + \theta_c; \tag{14}$$

$$\varepsilon_k \geq (k/2)n + \theta_d/2; \tag{15}$$

$$\varepsilon_k \geq (k/2)(n - 1 + \kappa + \beta - \nu) + \theta_a/2 + \theta_b. \tag{16}$$

It is obvious that (15) holds for the new definition of θ_d. The two other inequalities are the same as in section 5.3 with the factor 3 changed to k.

For convenience we restate here the previous equation defining δ as follows:

$$\overline{\mu} - \sigma - \chi = \gamma/2 + \delta. \tag{17}$$

The definition of B gives

$$|B| \leq \gamma + \sigma + (n - 1 - \nu) = n - 1 + \overline{\mu} - \sigma - \chi = n - 1 + \gamma/2 + \delta.$$

Combining $1/(2k)$ times inequality (14) with $2/k$ times inequality (15) gives $(5/2k)\,\varepsilon_k \geq n + \gamma/2 + \theta_c/(2k) + \theta_d/k$. Hence we can assume

$$\delta > \theta_c/(2k) + \theta_d/k$$

since otherwise we are done. To handle this case, we will show that (14) and (16) imply

$$(5/2k)\,\varepsilon_k \geq n - 1 + \gamma/2 + 2\delta - \theta_c/(2k) - \theta_d/k. \tag{18}$$

Inequality (18), together with our assumption on δ, implies the desired result.

Reexpress $\overline{\mu}$ exactly as in section 5.3 as follows: Substituting (3) into (17) and using (4) gives

$$\overline{\mu} - \nu - \chi = 2\delta.$$

Combining this with (13) gives

$$\kappa + \beta - \nu \geq \overline{\mu} - \chi - \nu - (\theta_a + 2\theta_b + \theta_c + \theta_d)/k = 2\delta - (\theta_a + 2\theta_b + \theta_c + \theta_d)/k.$$

Thus (16) gives

$$\varepsilon_k \geq (k/2)(n-1+2\delta-(\theta_a+2\theta_b+\theta_c+\theta_d)/k)+\theta_a/2+\theta_b = (k/2)(n-1+2\delta-(\theta_c+\theta_d)/k).$$

Combining $2/k$ times this inequality with $1/(2k)$ times (14) gives the desired inequality (18).

A.2. Bounding $|\mathcal{Y}|$ for general k. As in section 5.4, we still deal with the given graph G and the fixed 3-ECSS H. We do not use Proposition 5.4. Lemmas 5.5–5.9 are properties of G alone and do not involve the connectivity of G, so they are still valid. Lemma 5.10 involves H but not its connectivity, so it is still valid. Our analogue of Lemma 5.11 uses the same quantities $\kappa(y), \chi(y), \beta(y), \theta_a(y)$, and $\theta_b(y)$. The two remaining quantities are defined slightly differently as follows:

$\theta_c(y)$ = the number of edges of H launched by y not covering a carving edge or redundantly covering the carving edge of an ear launched by y;

$\theta_d(y)$ = the total surplus degree of vertices of T_y in H.

The new definition of $\theta_d(y)$ makes it consistent with θ_d. The definition of $\theta_c(y)$ differs from section 5.4 in a substantive way: It associates a carving edge vw that has $v \in T_y$ and $w \in \mathcal{Y}$ with y rather than with w as in section 5.4.

LEMMA A.1. *For $y \in \mathcal{Y}$ let $u \in T_y$ be the tip of a depleted ear. Suppose u does not launch a merging ear and u has a penetrated child ear. Then either*

(i) *u has a breakable child belonging to T_y, or*

(i) *$\theta_a(y) + 2\theta_b(y) + \theta_c(y) + \theta_d(y) \geq k$, or*

(iii) *$\kappa(y) + \chi(y) \geq 1$.*

Proof. As in Lemma 5.11, we argue by contradiction. As in that lemma, u is tight. We use the same notation U, C_X, $\overline{S}$, P, p, D with one change noted at the beginning of Case 2 below. We use Claims 2 and 4 but no others. Their proofs are unchanged. The proof of Claim 4 shows that there is an ear Q with first vertex in P, nonempty cluster C_Q, and vertex $q \in C_Q \cap Q \cap T_y$ adjacent to u in H. Furthermore, edge qu does not cover a carving edge. Throughout this proof we write d for the degree function in H, d_H.

Case 1. P is depleted.

We show that (ii) holds in this case, more specifically, $2\theta_b(y) + \theta_d(y) \geq k$. Claim 2(c) and the fact that H is k-edge connected implies these two inequalities:

$$\begin{array}{lcll} d(D) & = & d(P,U) + d(D,u) & \geq \quad k, \\ d(D+u) & = & d(P,U) + d(u,\overline{D}) & \geq \quad k. \end{array}$$

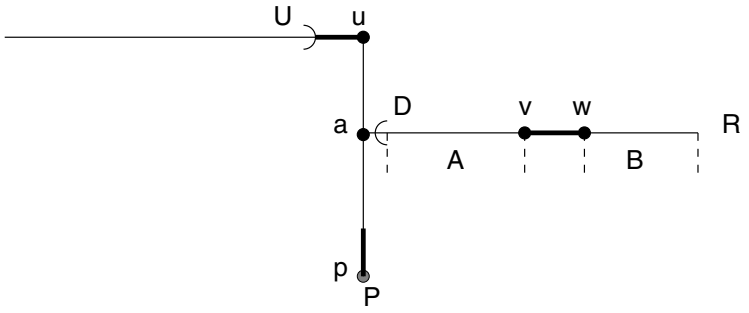

FIG. 7. *Proper child ear R.*

Adding them gives $2d(P,U)+d(u) \geq 2k$. With $\theta_d(u) = d(u)-k$ this gives $2d(P,U)+\theta_d(y) \geq k$. Now it suffices to show that every edge $xz \in E(H)$ from P to U is counted in $\theta_b(y)$. This requires that (a) x and z belong to the same t-set, and (b) xz is properly launched by y, i.e., $x \in T_y - y$. For (a) observe that Claim 2(a) shows $z \in t(u)$, and P depleted shows $x \in t(u)$. For (b) observe that $x \neq p$ since otherwise edge xz covers two carving edges (recall Figure 6).

Case 2. P is nondepleted.

As in Claim 3, the assumption implies P is tight. Consider (see Figure 7) a child ear R of P whose first vertex is a and whose carving edge is vw (with v the parent of w as usual). Say R is a *proper* child ear of P if $a \neq p, u, v$. For instance, the ear Q from Claim 4 is proper, by Claim 2(b).

Now consider a proper child ear R of P. Redefine D to be the set of all descendants of $I(R)$. (We will manipulate this new D in a manner analogous to the original D in Case 1.) Let $\theta_c(D)$ denote the contribution to $\theta_c(y)$ of all edges that have at least one end in D.

OBSERVATION 1. $\theta_c(D) \geq k/2$.

Proof. Partition D into two sets: A consists of all vertices that descend in F from a vertex of $I(R)$ that precedes (or equals) v; B consists of all vertices that descend in F from a vertex of $I(R)$ that follows (or equals) w. Both sets are nonempty ($v \in A, w \in B$). Claim 2(c) and H k-edge connected imply

$$\begin{aligned} d(D) &= d(A,P)+d(B,P) \geq k, \\ d(A) &= d(A,P)+d(A,B) \geq k. \end{aligned}$$

Adding the inequalities gives $2d(A,P)+d(B) \geq 2k$. Equivalently, $d(A,P)+(d(B)-k)/2 \geq k/2$. So it suffices to show $\theta_c(D) \geq d(A,P)+(d(B)-k)$.

$d(B)$ equals the number of edges covering the carving edge vw. Hence $d(B)-k$ is the number of times vw is redundantly covered. This is included in $\theta_c(D)$ because R is launched by y. (Note that this is not true if we use the original definition of $\theta_c(y)$.)

It remains to show that every edge $xz \in E(H)$ from A to P is counted in $\theta_c(y)$. For this it suffices to prove (a) xz does not cover a carving edge, and (b) xz is launched by y, i.e., $x \in T_y$.

We prove (a) by contradiction. Suppose xz covers $e \in C$. Let s be the deepest vertex of R that is an ancestor of x. We will show that e descends from s in F. $x \in A$ shows that the tree path from s to $f(s)$ does not contain the carving edge of R. R proper and P tight implies that the tree path from $f(s)$ to z does not contain the carving edge of P. Hence the carving edge e covered by xz must descend from s in F.

Now Rule 3 shows that R has a child ear, launched by u, that is merging. This follows since, after `Long_Ear` has constructed ear R, the possible ear s, x, z is merging (as $s \neq z$). But the existence of such a merging ear contradicts the hypothesis of Lemma A.1.

The contradiction proves (a). It is easy to see that (a) implies (b). □

Consider again the child ear Q of Claim 4. Edge qu is not a short ear (since $q \notin t(u)$). Hence $f(q) \in t(u)$ when `Short_Ear`$(\cdot, 2)$ scans q (i.e., when line 2 of `Short_Ear` has $x = q$ and $i = 2$). Consider the first ear that adds a vertex of $I(P)$ to $t(u)$. Since `Short_Ear` works bottom-up, Claim 2(c) implies this ear is a short ear xu for some vertex x internal to a child ear R of P. Claim 2(b) shows that x is an ancestor of the carving edge of R, so R is a proper child.

Case 2.1. $R \neq Q$.

Observation 1 applies to both R and Q, since both are proper. Write $D(R)$ for the set of all descendants of $I(R)$ and, similarly, write $D(Q)$. No edge contributes to both $\theta_c(D(R))$ and $\theta_c(D(Q))$. (More generally, let Q' range over all child ears of P except R. An edge leaving $D(R)$ does not leave $D(Q')$ or cover any edge of Q'. Similarly, an edge covering an edge of R does not leave $D(Q')$ or cover any edge of Q'.) Now Observation 1 shows $\theta_c(y) \geq k/2 + k/2 = k$. Hence Lemma A.1(ii) holds.

Case 2.2. $R = Q$.

For consistency we refer to the ear as R. Let vw be the carving edge of R. Recall vertices $x \in I(R) \cap t(u)$ and $q \in I(Q) - t(u)$, both adjacent to u. This implies $x \neq q$, and both vertices are proper ancestors of w (Claim 2(b)).

Let D denote the set of all descendants of $I(R)$. Partition D into three sets: A (C) consists of all vertices that descend in F from the first (last) vertex of $I(R)$, respectively; B consists of all vertices that descend in F from a vertex of $I(R)$ other than the first or last. (For the remainder of the proof we are discarding the use of "C" as the set of carving edges.) All three sets are nonempty: Since $R = Q$ is tight, $w \in C$. Since x and q are proper ancestors of w in $I(R)$, $A, B \neq \emptyset$.

Claim 2(c) and H k-edge connected imply

$$\begin{array}{rcl} d(A) & = & d(A,P) + d(A,B) + d(A,C) \quad \geq \quad k, \\ d(B) & = & d(B,P) + d(B,A) + d(B,C) \quad \geq \quad k, \\ d(D) & = & d(A,P) + d(B,P) + d(C,P) \quad \geq \quad k. \end{array}$$

Adding the inequalities gives $2d(A \cup B, P) + 2d(A, B) + d(C) \geq 3k$. Equivalently, $d(A \cup B, P) + d(A, B) + (d(C) - k)/2 \geq k$. We will show $\theta_c(y) \geq d(A \cup B, P) + d(A, B) + (d(C) - k)$, giving Lemma A.1(ii) as desired.

As in the proof of Observation 1, $d(C) - k$ is the number of times the carving edge vw is redundantly covered, and it is included in $\theta_c(y)$ since R is launched by y. Similarly, the proof of Observation 1 shows that every edge $xz \in E(H)$ from $A \cup B$ to P is counted in $\theta_c(y)$. Finally, we must show the same for edges from A to B. This follows by the same argument as in Observation 1. □

The analogue of Lemma 5.12 is that any tip $y \in \mathcal{Y}$ has

$$(\kappa + \chi + \beta)(y) + (\theta_a + 2\theta_b + \theta_c + \theta_d)(y)/k \geq 1.$$

This is proved by exactly the same argument as before, using Lemma A.1 in place of Lemma 5.11. The desired inequality (13) is then proved by the argument for (6) in section 5.4.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill, New York, 1990.

[2] J. Cheriyan, A. Sebő, and Z. Szigeti, *Improving on the* 1.5*-approximation of a smallest* 2*-edge connected spanning subgraph*, SIAM J. Discrete Math., 14 (2001), pp. 170–180.

[3] J. Cheriyan and R. Thurimella, *Approximating minimum-size k-connected spanning subgraphs via matching*, SIAM J. Comput., 30 (2000), pp. 528–560.

[4] C. G. Fernandes, *A better approximation ratio for the minimum size k-edge-connected spanning subgraph problem*, J. Algorithms, 28 (1998), pp. 105–124.

[5] H. N. Gabow, *Better performance bounds for finding the smallest k-edge connected spanning subgraph of a multigraph*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, ACM, New York, 2003, pp. 460–469.

[6] N. Garg, V. S. Santosh, and A. Singla, *Improved approximation algorithms for biconnected subgraphs via better lower bounding techniques*, in Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, ACM, New York, 1993, pp. 103–111.

[7] S. Khuller, *Approximation algorithms for finding highly connected subgraphs*, in Appoximation Algorithms for NP-hard Problems, D. S. Hochbaum, ed., PWS Publishing, Boston, MA, 1997 pp. 236–265.

[8] S. Khuller and B. Raghavachari, *Improved approximation algorithms for uniform connectivity problems*, J. Algorithms, 21 (1996), pp. 434–450.

[9] S. Khuller and U. Vishkin, *Biconnectivity approximations and graph carvings*, J. ACM, 41 (1994), pp. 214–235.

[10] S. Vempala and A. Vetta, *Factor* 4/3 *approximations for minimum* 2*-connected subgraphs*, in Approximation Algorithms for Combinatorial Optimization, K. Jansen and S. Khuller, eds., Lecture Notes in Comput. Sci. 1931, Springer-Verlag, Berlin, 2000, pp. 262–273.

[11] D. B. West, *Introduction to Graph Theory*, 2nd ed., Prentice-Hall, Upper Saddle River, NJ, 2001.