Chapter 5

Separators in planar graphs

5.1 Triangulation

We say a planar embedded graph is *triangulated* if each face's boundary has at most three edges.

Problem 5.1. *Provide a linear-time algorithm that, given a planar embedded graph G, adds a set of artificial edgesx to obtain a triangulated planar embedded graph. Show that the number of artificial edges is at most twice the number of original edges.*

5.2 Weights and balance

Let G be a planar embedded graph, and let α be a number between 0 and 1. An assignment of nonnegative weights to the faces, vertices, and edges of G is an *α-proper* assignment if no element is assigned more than α times the total weight. A subpartition of these elements is *α-balanced* if, for each part, the sum of the weights of the elements of that part is at most α times the total weight.

5.3 Fundamental-cycle separators

Let G be a plane graph. A simple cycle C of G defines a subpartition consisting of two parts, the strict interior and the strict exterior of the cycle. If the subpartition is α-balanced, we say that C is an *α-balanced cycle separator*. The subgraph induced by the (nonstrict) interior, i.e. including C, is one *piece* with respect to C, and the subgraph induced by the (nonstrict) exterior is the other piece.

First we give a result on fundamental-cycle separators that are balanced with respect to an assignment of weights only to *faces*.

Lemma 5.3.1 (Fundamental-cycle separator of faces)**.** *For any plane graph G, $\frac{1}{4}$-proper assignment of weights to faces, and spanning tree T of G such that the*

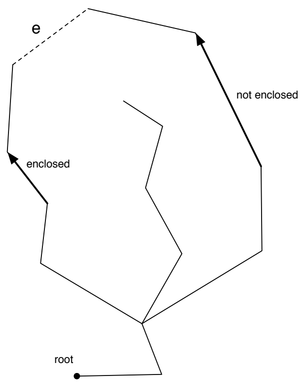


Figure 5.1: This figure illustrates the rule for assigning edges belonging to a fundamental cycle.

boundary of each face of G has at most three nontree edges, there is a nontree edge $\hat{e}$ such that the fundamental cycle of $\hat{e}$ with respect to T is a $\frac{3}{4}$-balanced cycle separator for G.

Proof. Let T^* be the interdigitating tree, the spanning tree of G^* consisting of edges not in T. The vertices of T^* are faces of G, and are therefore assigned weights. The property of T ensures that T^* has degree at most three. Using Lemma 1.3.2, let $\hat{e}$ be an edge separator in T^* of vertex weight. By the Fundamental-Cut Lemma (Lemma 3.2.2), the fundamental cut of $\hat{e}$ in G with respect to T^* is a simple cut in G^* (each side of which has weight at most three-fourths of the total) so it is, by the Simple-Cycle/Simple-Cut Theorem (Theorem 4.6.2), a simple cycle in G that encloses between one-fourth and three-fourths of the weight. □

Assignments of weight to faces, vertices, and edges can also be handled:

Lemma 5.3.2. *There is a linear-time algorithm that, given a triangulated plane graph G, a spanning tree T of G, and a $\frac{1}{4}$-proper assignment of weights to faces, edges, and vertices, returns a nontree edge $\hat{e}$ such that the fundamental cycle of $\hat{e}$ with respect to T is a $\frac{3}{4}$-balanced cycle separator for G.*

Problem 5.2. *Prove Lemma 5.3.2 using Lemma 5.3.1.*

We can give a better balance guarantee if we are separating just edge-weight.

Lemma 5.3.3 (Fundamental-cycle separator of edges)**.** *There is a linear-time algorithm that, given a triangulated plane graph G with a $\frac{1}{3}$-proper assignment of weights to edges, and a spanning tree T, returns a nontree edge $\hat{e}$ such that* *the fundamental cycle of* $\hat{e}$ *with respect to* T *is a* $\frac{2}{3}$*-balanced cycle separator for* G.

Problem 5.3. *Prove Lemma 5.3.3 by following the proof of Lemma 5.3.1 but using tree edge separators of vertex/edge weight (Lemma 1.3.3) instead of tree edge separators of vertex weight (Lemma 1.3.2).*

5.4 Breadth-first search

Let G be a connected, undirected graph, and let r be a vertex. For $i = 0, 1, 2, \ldots$, we say a vertex v of G has *level* i *with respect to* r and we define $\text{level}(v) = i$ if i is the minimum number of edges on an r-to-v path in G. (That is, the level of a vertex v is the distance of v from r where the edges are assigned unit length.) For $i = 0, 1, 2, \ldots$, let $L_i(G, r)$ denote the set of vertices having level i.

An edge uv is said to have level i (and we write $\text{level}(uv) = i$) if u has level i and v has level $i + 1$. (An edge whose endpoints have the same level is not assigned a level.) *Breadth-first search* from r is a linear-time algorithm that finds

- the levels of vertices and edges, and
- a spanning tree rooted at r such that, for each vertex v other than r, the parent of v has level one less than that of v (a *breadth-first-search tree*).

For brevity, we refer to the levels as *BFS levels* and we refer to the edges of the breadth-first-search tree as *BFS edges*.

5.5 $O(\sqrt{n})$-vertex separator

We use fundamental-cycle separators to prove a fundamental separator result for planar graphs.

Theorem 5.5.1 (Planar-Separator Theorem with Edge-Weights)**.** *There is a linear-time algorithm that, for a plane graph* G *and* $\frac{1}{3}$*-proper assignment of weights to edges, returns subgraphs* G_1, G_2 *such that*

- $E(G_1), E(G_2)$ *is a partition of* $E(G)$,
- *The partition* $E(G_1), E(G_2)$ *is* $\frac{2}{3}$*-balanced, and*
- $|V(G_1) \cap V(G_2)| \leq 4\sqrt{|V(G)|}$

The subgraphs G_1, G_2 are called the *pieces*. The set $V(G_1) \cap V(G_2)$ of vertices common to the two subgraphs is called the *vertex separator*.

The Planar-Separator Theorem can be used with an assignment of weight to vertices instead of edges. In this case, in evaluating the resulting balance, we do not count the weight of the vertices in the vertex separator.

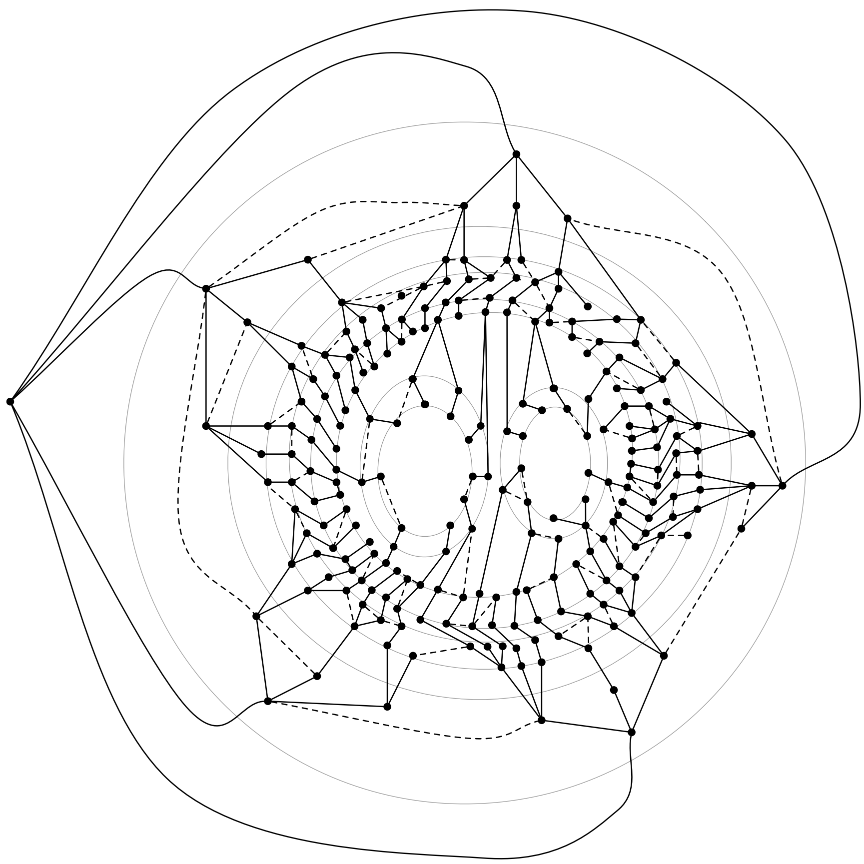


Figure 5.2: Shows the levels of breadth-first search.

Theorem 5.5.2 (Planar-Separator Theorem with Vertex-Weights)**.** *There is a linear-time algorithm that, for a plane graph G and $\frac{1}{3}$-proper assignment of weights to vertices, returns subgraphs G_1, G_2 such that*

- $E(G_1), E(G_2)$ *is a partition of* $E(G)$,
- *The subpartition* $V(G_1) - V(G_2), V(G_2) - V(G_1)$ *of* $V(G)$ *is* $\frac{2}{3}$*-balanced, and*
- $|V(G_1) \cap V(G_2)| \leq 4\sqrt{|V(G)|}$

Problem 5.4. *Show that the Vertex-Weights version (Theorem 5.5.2) follows easily from the Edge-Weights version (Theorem 5.5.1).*

Now we give the proof of the Edge-Weights version. Let $\boldsymbol{\omega}[\cdot]$ denote the edge-weight assignment. Assume for notational simplicity that the sum of edge-weights is 1. The basic idea is to find a balanced fundamental cycle separator, which might be too long, and shortcut it at small BFS levels.

The algorithm adds artificial zero-weight edges to G to triangulate it. Then, it performs a breadth-first search of G from r. Let T be the breadth-first-search tree. Let V_i denote the set of vertices at level i. Following Lemma 5.3.3, the algorithm finds a $\frac{2}{3}$-balanced fundamental-cycle separator C with respect to T. We partition the edges of G into two sets E_{in} and E_{ext}, so that E_{in} contains all edges strictly enclosed by C and E_{out} contains all edges not enclosed by C. The edges of C are assigned to E_{in} and E_{out} in a way that makes the partition $\frac{2}{3}$-balanced. Such a partitioned exists because the weight assignment $\boldsymbol{\omega}[\cdot]$ is $\frac{1}{3}$-proper, and can be computed in linear time by assigning the edges of C to E_{in} until the total weight of E_{in} is at least 1/3.

Let i_{min} and i_{max} denote the minimum and maximum level of a vertex of C, respectively. For an integer i, let $W_{\leq i}$ denote $\boldsymbol{\omega}(\{uv : u$ and v both have level $\leq i\})$, and let $W_{>i}$ denote $\boldsymbol{\omega}(\{uv :$ at least one of u or v have level $> i\})$. Let i_- be the greatest integer satisfying

- $i_{min} < i_- < i_{max}$
- $|V_{i_-}| \leq \sqrt{n}$
- $W_{\leq i_-} \leq 2/3$

If no such a level exists we set $i_- := i_{min}$. Let i_+ be the smallest integer satisfying

- $i_- < i_+ < i_{max}$
- $|V_{i_+}| \leq \sqrt{n}$
- $W_{>i_+} \leq 2/3$

If no such a level exists we set $i_+ := i_{max}$.

Since for any level i, the sets of edges corresponding to $W_{\leq i}$ and to $W_{>i}$ are disjoint, at least one of $W_{\leq i} \leq 2/3$, $W_{>i} \leq 2/3$ is true. It follows that each of the levels $i_- + 1, i_- + 2, \ldots, i_+ - 1$ has greater than $\sqrt{n}$ vertices. Hence, the total number of vertices among these levels is greater than $(i_+ - i_- - 1)\sqrt{n}$, so $i_+ - i_- - 1 < n/\sqrt{n} = \sqrt{n}$.

Let E_1 be the set of edges whose endpoints have level at most i_- if $i_- > i_{min}$, and the empty set if $i_- = i_{min}$. Let E_2 be the set of edges with at least one endpoint at level greater than i_+ if $i_+ < i_{max}$, and the empty set if $i_+ = i_{max}$. Let $E_3 = E_{in} - (E_1 \cup E_2)$, and let $E_4 = E_{out} - (E_1 \cup E_2)$.

It is immediate to verify that $\{E_j\}_{j=1}^4$ is a partition of the edges of G. By choice of i_+ and i_-, for $j = 1, 2$ we have $\boldsymbol{\omega}(E_j) \leq 2/3$. Since C is a balanced separator, for $j = 3, 4$ we have $\boldsymbol{\omega}(E_j) \leq 2/3$.

For $j = 1, 2, 3$, or 4, if $\boldsymbol{\omega}(E_j) \geq \frac{1}{3}$ then the algorithm sets $G_1 := E_j$ and $G_2 :=$ all other edges of G. Since $\frac{1}{3} \leq \boldsymbol{\omega}(E_j) \leq \frac{2}{3}$, it follows that $\boldsymbol{\omega}(E(G_1)) \leq \frac{2}{3}$ and $\boldsymbol{\omega}(E(G_2)) \leq \frac{2}{3}$.

Assume therefore that $\boldsymbol{\omega}(E_j) < \frac{1}{3}$ for $j = 1, 2, 3, 4$. Let j be the minimum integer such that $\boldsymbol{\omega}(E_1) + \cdots + \boldsymbol{\omega}(E_j) \geq \frac{1}{3}$. Then $\boldsymbol{\omega}(E_1) + \cdots + \boldsymbol{\omega}(E_{j-1}) < \frac{1}{3}$ and $\boldsymbol{\omega}(E_j) < \frac{1}{3}$ so $\boldsymbol{\omega}(E_1) + \cdots + \boldsymbol{\omega}(E_j) < \frac{2}{3}$. The algorithm sets $G_1 := E_1 \cup \cdots \cup E_j$ and $G_2 :=$ all other edges of G. Then $\boldsymbol{\omega}(E(G_1)) \leq \frac{2}{3}$ and $\boldsymbol{\omega}(G_2) \leq \frac{2}{3}$.

5.5.1 Size of the separator

We have completed the description of the algorithm, and have ensured that $E(G_1), E(G_2)$ is a $\frac{2}{3}$-balanced partition of $E(G)$. It remains to show that $|V(G_1) \cap V(G_2)| \leq 4\sqrt{|V(G)|}$.

Any vertex in $V(G_1) \cap V(G_2)$ is an endpoint of edges in distinct sets E_i and E_j. Such vertices fall into one of three categories:

1. level-i_- vertices (if $i_- \neq i_{min}$), or the (at most 2) vertices of C at level $i_m in$ (if $i_- = i_{min}$).
2. level-i_+ vertices (if $i_+ \neq i_{max}$), or the (at most 2) vertices of C at level i_{max} (if $i_+ = i_{max}$).
3. vertices of the cycle C in the level range (i_-, i_+).

There are at most $\sqrt{n}$ in each of the first two categories (assuming $\sqrt{n} \geq 2$). We claim that the number in the third category is at most $2(i_+ - i_- - 1)$, which in turn is at most $2\sqrt{n}$. This is because the monotonicity property of levels of vertices on leafward paths in the BFS tree implies that each of the two leafward tree paths comprising C has at most one vertex at each BFS level. This completes the proof of Theorem 5.5.1.

Problem 5.5. *Show how to modify the algorithm so that the size of the separator is at most $c\sqrt{n}$ for some constant $c < 4$.* Hint: *The size criterion used to decide whether a level i qualifies to be designated level i_- can depend on $|i_{min} - i_-|$.*

Problem 5.6. *Let G be a directed planar graph with real arc weights. A negative cycle in G is a cycle whose arc weights sum to a negative number. Let k be the minimum number of arcs on a negative cycle in G. Give an $O(n^{3/2}k)$-time algorithm for finding k.*

Problem 5.7. *Let G be a biconnected planar graph with m edges and face weights summing to 1 such that no face weighs more than $\frac{3}{4}$. Give an $O(m)$ algorithm that finds a balanced simple-cycle separator C in G. (Note there is no size requirement on C.)*

5.6 Noncrossing families of subsets

Two nonempty sets A and B *cross* if they are neither disjoint nor nested, i.e. if $A \cap B \neq \emptyset$ and $A \not\subseteq B$ and $B \not\subseteq A$.

A family $\mathcal{C}$ of nonempty sets is *noncrossing* (also called *laminar*) if no two sets in $\mathcal{C}$ cross.

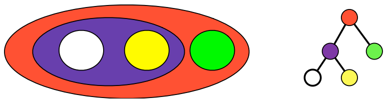


Figure 5.3: The diagram on the left shows a Venn diagram of some noncrossing sets, and the diagram on the right shows the corresponding rooted forest (a tree in this case).

As illustarted in Figure 5.3, under the subset relation, a noncrossing family $\mathcal{C}$ of sets forms a rooted forest $F_{\mathcal{C}}$. That is, each set $X \in \mathcal{C}$ is a node, and its ancestors are the sets in $\mathcal{C}$ that include X as a subset. To see that this is a forest, let X_1 and X_2 be two supersets of X in $\mathcal{C}$. Since X is nonempty, X_1 and X_2 intersect, so one must include the other. This shows that the supersets of X are totally ordered by inclusion. Hence if X has any proper supersets, it has a unique minimal proper superset, which we take to be the parent of X.

5.7 The connected-components tree

5.7.1 Vertex and face labels

Let G be a connected triangulated undirected graph. Let r be a vertex of G, and let f_∞ be a face incident to r. Recall the definition of the level of a vertex v with respect to r given in Section 5.4. For a face f of G, define its level to be the minimum level of a vertex incident to f. Thus, level$(r) = 0$, and all faces to which r is incident is also 0.

The following lemma follows immediately from G being a connected triangulated graph.

Lemma 5.7.1. *A vertex v of G with level$(v) = i \geq 1$ is incident to at least one level-$(i-1)$ face, zero or more level-i faces, and no faces at other levels.*

Corollary 5.7.2. *For every face f of G other than f_∞, there is a face g of level less than that of f and an f-to-g path P in G^* all of whose vertices (which are faces of G) except g have the same level as f.*

Proof. Let $i > 0$ be the level of f. Consider the vertex v incident to f whose level is i. By Lemma 5.7.1, v is incident to at least one level $i-1$ face and only to faces at levels $i-1$ or i. Let d be the reverse of a dart of f such that $\text{head}_G(d) = v$. Let P be a path in G^* consisting of darts belonging to v, in order of the permutation cycle of v, starting at d and ending at the first dart whose head is a level-$(i-1)$ face of G. By construction, P starts at f, and all vertices of P except the last one have level i. □

5.7.2 The connected-components tree

For an integer $i \geq 0$, let F_i^+ denote the set of faces of G with level at least i. Let $\mathcal{K}_i^+$ denote the set of connected components of the subgraph of G^* induced by F_i^+. We refer to a connected component $K \in \mathcal{K}_i^+$ as a *level-i component*, and we define level$(K) = i$. With a slight abuse of notation we identify a connected compoenet K of G^*, which is a subgraph of G^*, with the set of vertices of G^* (faces of G) that belong to K. Thus, when it is convenient, we regard K as a subset of F(G).

Lemma 5.7.3. *For any triangulated connected graph G and face f_∞, the components $\bigcup_i \mathcal{K}_i^+$ form a noncrossing family of subsets of $F(G)$.*

Proof. Let K_1 and K_2 be two components in $\bigcup_i \mathcal{K}_i^+$. Assume without loss of generality that level$(K_1) \leq$ level(K_2). If any face of K_2 belongs to K_1 then, by definition, *every* face of K_2 belongs to K_1. □

As explained in Section 5.6, there is a rooted forest $\mathcal{T}$ corresponding to the noncrossing family $\bigcup_i \mathcal{K}_i^+$.

Lemma 5.7.4. *The parent in $\mathcal{T}$ of a level-i component is a level-$(i-1)$ component if $i > 0$.*

Proof. Let K be a level-i component where $i > 0$. Since the level-i components are disjoint, K is not contained in any other level-i component. Since K must contain some level-i face f of G, K is not contained in any level-j component where $j > i$. Let g be a level $i-1$ face that shares a vertex with f. Then the level-$i-1$ component K' containing g must contain K. Furthermore, for $j \leq i-1$, any level-j component that contains K must also contain K'. This shows that K' is the unique minimal proper superset of K among the components. □

Lemma 5.7.4 shows that a root of the forest of components must have level 0. Since we assume G is connected, there is only one level-0 component, namely the whole graph G^*, so it is the only root, and the forest is in fact a tree. We call $\mathcal{T}$ the *component tree.*

Lemma 5.7.5 (BFS-Component-Tree Construction)**.** *There is a linear-time algorithm to construct the component tree* $\mathcal{T}$*.*

Problem 5.8. *Prove the Component-Tree Construction Lemma.*

Lemma 5.7.6. *For any level* i*, the cuts* $\{\delta_{G^*}(K) \;:\; K \in \mathcal{K}_i^+, i > 0\}$ *are edge-disjoint.*

Proof. Suppose K is a level-i component. Let fg be an edge of $\delta_{G^*}(K)$ where $g \in K$. Then the level of f is less than i (else f would belong to K). Since faces f and g share a vertex, their level differ by at most one, so level$(f) = i-1$. Therefore, for any component K' such that $fg \in \delta_{G^*}(K')$, we must have level$(K') = i$ and K' must contain g, hence $K' = K$. □

Lemma 5.7.7. *For any level* i *and any component* $K \in \mathcal{K}_i^+$*,* $\delta_{G^*}(K)$ *is a simple cut.*

Proof. Clearly K is connected in G^*. We need to show that $V(G^*) - K$ is also connected in G^*. Consider a face $f \in G^* - K$ with level$(f) = j > 0$. Let P be the path in Corollay 5.7.2. Since $f \notin K$, and since all faces along the path P except the last have level j, P is a path in $G^* - K$. An induction by level then shows that every vertex in $V(G^*) - K$ is connected in $G^* - K$ to a level-0 face. Since all level zero face are incident to the same vertex r, they are connected in $G^* - K$, so $G^* - K$ is connected. □

The simple-cut/simple-cycle theorem (Theorem 4.6.2) implies that for any component K, the edges of $\delta_{G^*}(K)$ form a simple cycle in G whose enclosed set of faces is exactly K. We call such a cycle the *level cycle* associated with K and denote it by $X(K)$. If K is a level-i component then we say that $X(K)$ is a level-i cycle.

The definition of face-levels with respect to the face-vertex incidence graph implies that level-cycles at different levels are vertex disjoint.

Lemma 5.7.8. *Let* K, K' *be components at distinct levels.* $X(K)$ *and* $X(K')$ *are vertex disjoint.*

Proof. Suppose K is a level-i component, and let v be a vertex of $X(K)$. Thus v is incident to a level-i face and to a level-$(i-1)$ face. By Lemma 5.7.1, any face incident to v has levels i or $i-1$. Hence, if v is a vertex of $X(K')$, then K' must be a level-i component as well. □

PK: Need a figure to give the reader a sense of the structure.

5.8 Cycle separators

Theorem 5.8.1 (Planar-Cycle-Separator Theorem). *There is a linear-time algorithm that, for any simple undirected triangulated plane graph and any $\frac{3}{4}$-proper assignment of weights to faces, edges, and vertices, returns a $\frac{3}{4}$-balanced cycle separator C of size at most $4\sqrt{n}$.*

The algorithm is very similar to the vertex separator of Section 5.5. The main difference is that it shortcuts that fundamental cycle at small level-cycles, rather than at small BFS levels.

Assume for notational simplicity that the total weight is 1 and that weight is assigned only to faces. The algorithm first computes a BFS tree T rooted at an arbitrary vertex r. It designates some face incident to r as f_∞ and computes the component tree $\mathcal{T}$.

5.8.1 Shortcutting a fundamental-cycle separator

Following Lemma 5.3.1, the algorithm finds a $\frac{3}{4}$-balanced fundamental-cycle separator C with respect to T. Let i_{min} and i_{max} denote the minimum and maximum level of a vertex of C, respectively.

We say that an edge uv *penetrates* a component K at u if $u \in X(K)$ and v is strictly enclosed by $X(K)$. A set of edges penetrates K if some edge in the set penetrates K.

Lemma 5.8.2. *An edge uv with level(u) $= i$, level(v) $= i+1$ penetrates exactly one component K. The level of K is i.*

Proof. Since level(u) $= i$, and level(v) $= i+1$, the two faces f_1 and f_2 to which uv belongs must have level i. Let K be the unique level-i component to which f_1 and f_2 belong. Since level(u) $= i$, $u \in X(K)$. Since level(v) $= i+1$, v is strictly enclosed by $X(K)$.

We next show that K is the unique such component. Since the level-i components are disjoint, v is in no other level-i component. Since the level of u is i, u is in no level-j component for $j > i$. For any level-j component K' with $j < i$, The vertices of $X(K')$ have level j, so $u \notin X(K')$. □

Lemma 5.8.3. *For every integer $i \in (i_{min}, i_{max})$, C penetrates exactly one level-i component K_i, and it does so at exactly two distinct vertices of $X(K_i)$.*

Proof. Let uw be the non-tree edge of C. Let v be the least common ancestor of u and w in T. The cycle C consists of the edge uw, of a v-to-u leafward paths P_1, and a v-to-w leafward path P_2. The level of v is level(v) $= i_{min}$. The level of u is level(u) $= i_{max}$, and the level of w is either i_{max} or $i_{max} - 1$. Since T is a BFS tree, and by Lemma 5.8.2, each of these paths penetrates exactly one component at every level in (i_{min}, i_{max}). Let K_{max} be the level i_{max}-component penetrated by P_1. That is, u is strictly enclosed by $X(K_{max})$. Since the edge uw exists, w is also enclosed by $X(K_{max})$. It follows that P_1 and P_2 penetrate the same components. Since P_1 and P_2 are vertex disjoint (except at their start vertex v), C penetrates each of these components at exactly two distinct vertices. □

Let K_i denote the unique level-i component penetrated by C. For convenience, let X_i denote $X(K_i)$. Let $W(X_i)$ denote the weight of faces enclosed by X_i (that is, the weight of the component K_i), and let $\bar{W}(X_i)$ denote the weight of faces not enclosed by X_i. Let i_- be the greatest integer satisfying

- $i_{min} < i_- < i_{max}$
- $|V(X_i)| \leq \sqrt{n}$
- $\bar{W}(X_i) \leq 3/4$

If no such a level exists we set $i_- := i_{min}$. Let i_+ be the smallest integer satisfying

- $i_- < i_+ < i_{max}$
- $|V(X_i)| \leq \sqrt{n}$
- $W(X_i) \leq 3/4$

If no such a level exists we set $i_+ := i_{max}$.

Since for any level i, the sets of faces enclosed and not enclosed by X_i are disjoint, at least one of $W(X_i) \leq 3/4$,$\bar{W}(X_i) \leq 3/4$ is true. It follows that each of the levels $i_- + 1, i_- + 2, \ldots, i_+ - 1$ has greater than $\sqrt{n}$ vertices. By Lemma 5.7.8 all these level-cycles are vertex disjoint, so the total number of vertices among these levels cycle is greater than $(i_+ - i_- - 1)\sqrt{n}$, so $i_+ - i_- - 1 < n/\sqrt{n} = \sqrt{n}$.

Let F_1 be the set of faces enclosed by X_{i_+} if $i_+ < i_{max}$, and the empty set if $i_+ = i_{max}$. Let F_4 be the set of faces of G not enclosed by X_{i_-} if $i_- > i_{min}$, and the empty set if $i_- = i_{min}$. Let F_2 be the set of faces enclosed by C, and that belong to neither F_1 nor to F_4, and let F_3 be the set of faces not enclosed by C, and that belong to neither F_1 nor to F_4.

By choice of i_+ and i_-, for $j = 1, 4$ we have $\boldsymbol{\omega}(F_j) \leq 3/4$. Since C is a balanced separator, for $j = 2, 3$ we have $\boldsymbol{\omega}(F_j) \leq 3/4$. Let j be such that $\boldsymbol{\omega}(F_j) \geq \frac{1}{4}$. Let C^* be the boundary of F_j. That is, C^* is the set of edges that belong to exactly one face in F_j. The algorithm returns C^*.

5.8.2 Balanced, short, and simple cycle

Lemma 5.8.4. *C^* is a simple cycle.*

Proof. If $j = 1$ then C^* is the simple cycle X_{i_+}. If $j = 4$ then C^* is the simple cycle X_{i_-}. By lemma 5.8.3, C intersect each of $X(i_-)$ and $X(i_+)$ exactly twice. Thus, each of X_{i_-} and X_{i_+} can be decomposed into two paths, one enclosed by C and the other not enclosed by C. If $j = 2$ then C^* is the simple cycle formed by the edges of C enclosed by X_{i_-} but not by X_{i_+} and by the edges of X_{i_-} and

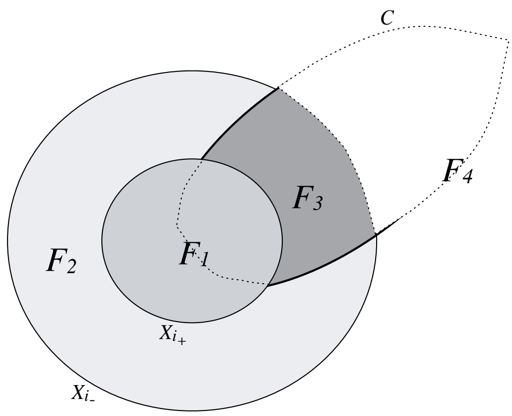


Figure 5.4: This figure shows the cycle separator C penetrating the vertex disjoint level-cycles X_{i_-} and X_{i_+}, each at two distinct nodes. The sets of faces F_j are also indicated. Their boundaries are simple cycles.

X_{i_+} enclosed by C. If $j = 3$ then C^* is the simple cycle formed by the edges of C enclosed by X_{i_-} but not by X_{i_+} and by the edges of X_{i_-} and X_{i_+} not enclosed by C. See Figure 5.4 for an illustration. □

By choice of j, C^* is a $\frac{3}{4}$-balanced cycle separator. It remains to show that C^* consists of fewer than $4\sqrt{n}$ edges. By choice of i_- and i_+, $|V(X_{i_-})| \leq \sqrt{n}$. The monotonicity property of levels of vertices on leafward paths in T implies that each of the two leafward tree paths comprising C has at most one vertex at each level. Combining this with the fact that $i_+ - i_- \leq \sqrt{n}$, we get that the number of edges of C enclosed by X_{i_-} and not enclosed by X_{i_+} is $2\sqrt{n}$. Thus C^* has fewer than $4\sqrt{n}$ edges. This completes the proof of Theorem 5.8.1.

5.9 Division into regions

In some cases it is useful to divide a graph into more than two regions. We next discuss such divisions.

Definition 5.9.1. *A* region *R of G is an edge-induced subgraph of G.*

Definition 5.9.2. *A* division *of G is a collection of regions such that each edge is in at least one region. A vertex is a* boundary vertex *of the division if it is in more than one region. A division is an r–division if there are $O(n/r)$ regions, and each region has at most r vertices and $O(\sqrt{r})$ boundary vertices.*

Definition 5.9.3. *A* natural *face of a region R is a face of R that is also a face of G. A* hole *of R is a face of R that is not natural.*

Definition 5.9.4. *An r–*division with few holes *is an r–division in which*

- *any edge of two regions is on a hole of each of them, and*
- *every region has $O(1)$ holes.*

Definition 5.9.5. *A* decomposition tree *for G is a rooted tree in which each leaf is assigned a region of G such that each edge of G is represented in some region. For each node x of the decomposition tree, the* region R_x corresponding to x *is the subgraph of G that is the union of the regions assigned to descendants of x.*

To avoid clutter we shall sometimes refer to $x \in \mathcal{T}$ simultaneously as a node of $\mathcal{T}$ and as the corresponding region R_x of G. Thus, for example, we may refer to the number of boundary vertices of x instead of writing the number of boundary vertices of R_x.

Definition 5.9.6. *A decomposition tree $\mathcal{T}$* admits *an r–division with few holes if there is a set S of nodes of $\mathcal{T}$ whose corresponding regions form an r–division of G with few holes.*

5.10 Computing a Decomposition Tree

In what follows we prove the following theorem, which states that one can efficiently compute a decomposition tree that simultaneously admits essentially all $\boldsymbol{r}$–division.

Theorem 5.10.1. *For a constant s, there is an $O(n \log n)$-time algorithm that, for any triangulated planar embedded graph G, outputs a binary decomposition tree $\mathcal{T}$ for G that simultanously admits an r–division of G with few holes for every $r \geq s$.*

In the remainder of this section we prove the theorem. Since the input graph G is assumed to be triangulated, it follows that, for every region R of G, every natural face is a triangle. The algorithm is invoked by calling the procedure RecursiveDivide$(G, 0)$, given in Algorithm 5.1.

Given a connected region R with more than s edges, and given a recursion-depth parameter ℓ, RecursiveDivide first triangulates each hole of R by adding an artificial vertex and attaching it via artificial edges to each occurrence of a vertex on the boundary of the hole. Let R' be the resulting graph. See Figures 5.5(a), 5.5(b) for an illustration. The vertices and edges that are not artificial are *natural*.

Next, the procedure uses the simple cycle separator algorithm (Theorem 5.8.1) to find a simple-cycle separator C consisting of at most $c\sqrt{n}$ natural vertices, where c is a constant and $n := |V(R)|$ is the number of vertices of R (which is equivalent to the number of natural vertices of R'). Depending on the current recursion depth ℓ, the cycle separates R in a balanced way with respect to either vertices, boundary vertices, or holes.

Algorithm 5.1 RecursiveDivide(R, ℓ)

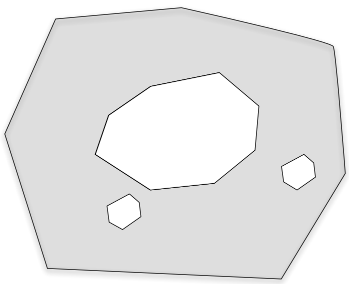
let $n = |V(R)|$
if $n \leq s$ **then**
  return a decomposition tree consisting of a leaf assigned R
if $\ell \bmod 3 = 0$ **then**
  separator chosen below to balance number of vertices
else if $\ell \bmod 3 = 1$ **then**
  separator chosen to balance number of boundary vertices
else if $\ell \bmod 3 = 2$ **then**
  separator chosen to balance number of holes
let R' be the graph obtained from R as follows: triangulate each hole by placing an artificial vertex in the hole and connecting it via artificial edges to all occurrences of vertices on the boundary of the hole
find a balanced simple-cycle separator C in R' with at most $c\sqrt{n}$ natural vertices
let F_0, F_1 be the sets of natural faces of R enclosed and not enclosed by C, respectively
for $i \in \{0, 1\}$ **do**
  let R_i be the region consisting of the edges of faces in F_i and the edges of C that are in R
  $\mathcal{T}_i \leftarrow$RecursiveDivide($R_i, \ell + 1$)
return the decomposition tree $\mathcal{T}$ consisting of a root with left subtree $\mathcal{T}_0$ and right subtree $\mathcal{T}_1$

Note that the cycle separator algorithm is invoked on R', which has more than n vertices since it also has some artificial vertices. However, we show in Lemma 5.10.4 that there are at most twelve artificial vertices, the bound of $c\sqrt{n}$ still holds for some choice of c (since $n \geq s$). The number of artificial vertices on the separator does not matter in the analysis of RecursiveDivide.

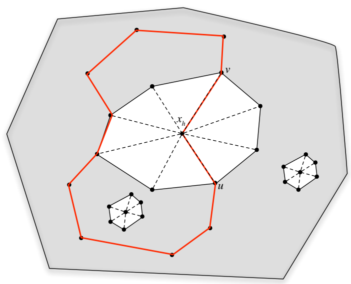
The cycle C determines a bipartition of the faces of the triangulated graph R', which in turn induces a bipartition (F_0, F_1) of the natural faces of R. For $i \in \{0, 1\}$, let R_i be the region consisting of the edges bounding the faces in F_i, together with the edges of C that are in R (i.e. omitting the artificial edges added to triangulate the artificial faces). See Figures 5.5(c), 5.5(d) for an illustration.

Lemma 5.10.2. *If R is connected then R_0 is connected and R_1 is connected.*

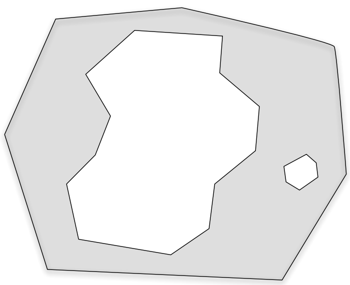
The procedure calls itself recursively on R_0 and R_1, obtaining decomposition trees $\mathcal{T}_0$ and $\mathcal{T}_1$, respectively. The procedure creates a new decomposition tree $\mathcal{T}$ by creating a new root corresponding to the region R and assigning as its children the roots of $\mathcal{T}_0$ and $\mathcal{T}_1$.



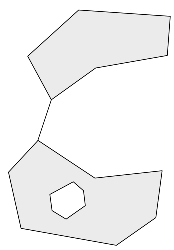
(a) A schematic diagram of a region R with four holes (white faces, one of them being the unbounded face).



(b) The graph R' and a cycle separator C (solid red). Artificial triangulation edges are dashed (triangulation edges are not shown for the unbounded hole to avoid clutter).



(c) The region R_0 consisting of the edges bounding the faces not enclosed by C together with the edges of C that belong to R. Equivalently, R_0 is the subgraph of R' not strictly enclosed by C without any artificial edges and vertices. R_0 has three holes.



(d) The region R_1 consisting of the edges bounding the faces enclosed by C together with the edges of C that belong to R. Equivalently, R_1 is the subgraph of R' enclosed by C without any artificial edges and vertices. R_1 has two holes. Note that a hole is not necessarily a simple face.

Figure 5.5: Illustration of triangulating a hole and separating along a cycle.

5.10.1 Number of Holes

The triangulation step (Line 10) divides each hole h into a collection of triangle faces. We say a hole h is *fully enclosed by* C if all these triangle faces are enclosed by C in R'.

Lemma 5.10.3. *Suppose that there are k holes that are fully enclosed by C. Then R_0 has $k+1$ holes.*

Proof. We give an algorithmic proof. See Figure 5.5 for an illustration. Initialize R'_0 to be the graph obtained from R' by deleting all edges not enclosed by C. Then C is the boundary of the infinite face of R'_0. Consider in turn each hole h of R such that a nonempty proper subset of h's triangle faces are enclosed by C. For each such face h, C includes the artificial vertex x_h placed in h, along with two incident edges ux_h and x_hv where u and v are distinct vertices on the boundary of h. Deleting all the remaining artificial edges of h modifies the boundary of the infinite face by replacing ux_h x_hv with a subsequence of the edges forming the boundary of h. In particular, deleting these artificial edges does not create any new faces.

Finally, for each hole h that is fully enclosed by C, delete the artificial edges of h, turning h into a face of R'_0. The resulting graph is R_0, whose holes are the holes of R that were fully enclosed by C, together with the infinite face of R_0. □

If the recursion depth $\mod 3$ is 2, Line 11 of RecursiveDivide must select a simple cycle in R' that is balanced with respect to the number of holes. To achieve this, for each hole h of R, the algorithm assigns weight 1 to one of the triangles resulting from triangulating h in Line 10, and weight 0 to all other triangles of h. Then the algorithm finds a cycle C that is balanced with respect to these face-weights.

Lemma 5.10.4. *For any region created by* RecursiveDivide*, the number of holes is at most twelve.*

Proof. By induction on the recursion depth ℓ,

(A) if $\ell \bmod 3 = 0$ then R has at most ten holes;

(B) if $\ell \bmod 3 = 1$ then R has at most eleven holes;

(C) if $\ell \bmod 3 = 2$ then R has at most twelve holes.

For $\ell = 0$, there are no holes. Assume (C) holds for ℓ. Since the cycle separator C encloses at most 3/4 of the weight, it fully encloses at most nine holes. By Lemma 5.10.3, R_0 has at most ten holes. The symmetric argument applies to R_1. Thus (A) holds for $\ell + 1$.

Similarly, by Lemma 5.10.3, if (A) holds for ℓ then (B) holds for $\ell + 1$, and if (B) holds for ℓ then (C) holds for $\ell + 1$. □

5.10.2 Number of Vertices and Boundary Vertices

If the recursion depth mod 3 is 0, Line 11 of RecursiveDivide selects a simple cycle in R' that is balanced with respect to the number of natural vertices. To achieve this, for each natural vertex v, the algorithm selects an adjacent face in R', dedicated to carry v's weight. The weight of each face is defined to be the number of vertices for which that face was selected. Since each face in R' is a triangle, every weight is an integer between 0 and 3. A cycle C is then chosen that is balanced with respect to these face-weights.

If the recursion depth mod 3 is 1, the cycle must be balanced with respect to the number of boundary vertices. For each boundary vertex, the algorithm selects an incident face; the algorithm then proceeds as above.

In either case, the total weight enclosed by the cycle C is an upper bound on the number of vertices (natural or boundary) strictly enclosed by C. Thus at most 3/4 of the vertices (natural or boundary) of R' are strictly enclosed by C in R'. Similarly, at most 3/4 of the vertices are not enclosed by C in R'.

The vertices of R_0 are the natural vertices of R' enclosed by C (including the natural vertices on C, which number at most $c\sqrt{|V(R)|}$), and the vertices of R_1 are the natural vertices of R' not strictly enclosed by C. Let $n := |V(R)|$ and, for $i \in \{0, 1\}$, let $n_i := |V(R_i)|$. We obtain

$$n_0 + n_1 \leq n + c\sqrt{n}. \tag{5.1}$$

Moreover, if the recursion depth mod 3 is 0, then

$$\max\{n_0, n_1\} \leq \frac{3}{4}n + c\sqrt{n}. \tag{5.2}$$

Similarly, let b be the number of boundary vertices of R, and, for $i \in \{0, 1\}$, let b_i be the number of boundary vertices of R_i. We obtain

$$b_0 + b_1 \leq b + c\sqrt{n}. \tag{5.3}$$

Moreover, if the recursion depth mod 3 is 1, then

$$\max\{b_0, b_1\} \leq \frac{3}{4}b + c\sqrt{n}. \tag{5.4}$$

5.10.3 Admitting an r–division

Let N be the number of vertices in the original input graph G. Consider the decomposition tree $\mathcal{T}$ of G produced by RecursiveDivide. Each node x corresponds to a region R_x. We define $n(x) := |V(R_x)|$, and $b(x)$ to be the number of boundary vertices of R_x. We will show that for any given $r \geq s$, $\mathcal{T}$ admits an r–division of G. The analysis consists of two phases. In the first phase (Lemma 5.10.8), we identify a set of $O(N/r)$ regions for which the number of vertices is at most r, and the *average* number of boundary vertices is $O(\sqrt{r})$. However, some of the individual regions in this set might have too many boundary vertices (since the number of vertices and boundary vertices do not necessarily decrease at the same rate). We show that each such region can be replaced with smaller regions in $\mathcal{T}$ so that every region has $O(\sqrt{r})$ boundary vertices, and the total number of regions remains $O(N/r)$ (Lemma 5.10.9).

Deonte $r_k = N\left(\frac{3}{4}\right)^k$ for any integer $0 \le k \le \lceil \log_{\frac{4}{3}}(N) \rceil$. Define the rank of a node x of $\mathcal{T}$ to be the integer k such that $r_{k+1} < n(x) \le r_k$.

Proposition 5.10.5. *For any $x \in \mathcal{T}$, the rank of x is not smaller than the rank of any descendant of x in $\mathcal{T}$.*

Lemma 5.10.6. *There is a constant s' such that for any $k < s'$, any rank-k node in T is the ancestor of at most 3 rank-k nodes in $\mathcal{T}$.*

Proof. Let k be the rank of $x \in \mathcal{T}$. Thus, $n(x) \le r_k$. If one child of x has rank k, then it has at least $\frac{3}{4}n(x)$ vertices. Thus, by Equation 5.1, the other child of x has at most $\frac{1}{4}n(x)+c\sqrt{n}(x)$ natural vertices. This can be guaranteed to be at most $\frac{3}{4}n(x)$, which by an appropriate choice of s'. Therefore, at most once child of x has rank k. The lemma follows since the separator is chosen to balance the number of natural vertices at levels congruent to 0 modulo 3 of $\mathcal{T}$. □

Whenever a region x of $\mathcal{T}$ is separated into two descendant regions using a cycle separator C, the vertices of C may become boundary vertices, which appear now in both subregions. To bound the total number of boundary vertices we shall bound in the following lemma the number of boundary vertices created when separating regions of nodes of the same rank.

Lemma 5.10.7. *There exists constants s', γ' depending on c such that, for any rank $k < \log N - s'$, the number of new boundary vertices (counting multiplicities) created by separating rank-k nodes is at most $\gamma' \frac{N}{\sqrt{r_k}}$.*

Proof. By induction on k. The claim clearly holds for $k = 0$ since the root of $\mathcal{T}$, namely all of G, is the only node with rank 0, and separating a graph with N vertices introduces $c\sqrt{N} = cN/\sqrt{r_0}$ boundary vertices.

Assume the lemma is correct for all $k' < k$. The sum over the number of vertices in all the rootmost nodes of rank k (i.e., rank-k nodes that are not descendants of any other rank-k nodes) is at most N plus the number of boundary vertices contributed by all nodes with rank strictly less than k. By the inductive hypothesis, this is at most

$$N + \sum_{k'<k} \gamma' \frac{N}{\sqrt{r_{k'}}} = N(1 + \frac{\gamma'}{\sqrt{N}} \sum_{k'} (4/3)^{\frac{k'}{2}}).$$

Since the number of vertices in each rank-k node is at least r_{k+1}, there are at most $\frac{N}{r_{k+1}}(1+\frac{\gamma'}{\sqrt{N}}\sum_{k'}(4/3)^{\frac{k'}{2}})$ rootmost rank-k nodes. Therefore, by Lemma 5.10.6, there are at most $3\frac{N}{r_{k+1}}(1+\frac{\gamma'}{\sqrt{N}}\sum_{k'}(4/3)^{\frac{k'}{2}})$ rank-k nodes altogether. Each such node creates at most $c\sqrt{r}_k$ new boundary vertices, so the number of new boundary nodes contributed by rank-k nodes is bounded by

$$\frac{3cN}{r_{k+1}/\sqrt{r_k}}(1 + \frac{\gamma'}{\sqrt{N}} \sum_{k'} (4/3)^{\frac{k'}{2}}) = \frac{N}{\sqrt{r_k}}\sqrt{12}c(1 + \frac{\gamma'}{\sqrt{N}} \sum_{k'<k} (4/3)^{\frac{k'}{2}}).$$

This can be made at most $\gamma' \frac{N}{\sqrt{r_k}}$ by choosing s' to be a constant satisfying $\sum_{k' < \log_{\frac{4}{3}}(N) - s'} \left(\frac{4}{3}\right)^{k'/2} < \frac{\sqrt{N}}{2}$, and $\gamma' > \sqrt{48}c$. □

Fix r and let S_r be the set of nodes y of $\mathcal{T}$ such that y's region has no more than r vertices but the region of y's parent has more than r vertices. Note that no node in S_r is an ancestor of any other. Let $\hat{x}$ be the root of $\mathcal{T}$. Recall that, for any $x \in \mathcal{T}$, $b(x)$ denotes the number of boundary vertices of R_x.

Lemma 5.10.8 (Total Number of Boundary Vertices)**.** *There are constants s and γ, depending on c, such that, $\sum_{x \in S_r} b(x) \le \frac{\gamma N}{\sqrt{r}}$, provided that $r > s$.*

Proof. Let k be the integer such that $r_{k+1} < r \le r_k$. The parent of each node in S_r has rank at most k. The sum $\sum_{x \in S_r} b(x)$ equals the sum over the new boundary vertices created for each of the strict ancestors of nodes in S_r. All of these ancestors have rank at most k, so by Lemma 5.10.7, there are constants γ', s' such that, if $k < \log_{\frac{4}{3}} N - s'$,

$$\sum_{k' \le k} \gamma' \frac{N}{\sqrt{r_{k'}}} = \gamma' N \frac{1}{\sqrt{N}} \sum_{k' \le k} \left(\frac{4}{3}\right)^{k/2}. \tag{5.5}$$

This geometric sum is dominated by its last term, so, for some constant α, it is bounded by

$$\gamma' \alpha N \frac{1}{\sqrt{N}} \left(\frac{4}{3}\right)^{k/2} = \gamma' \alpha \frac{N}{\sqrt{r_k}} \le \sqrt{\frac{4}{3}} \gamma' \alpha \frac{N}{\sqrt{r}}.$$

Choosing $s > \left(\frac{4}{3}\right)^{s'}$ guarantees that $k < \log_{\frac{4}{3}} N - s'$. Then, setting $\gamma = \sqrt{\frac{4}{3}} \gamma' \alpha$ yields the lemma. □

Lemma 5.10.8 implies that $\sum_{x \in S_r} n(x)$ is $O(N)$, since the regions in S_r are disjoint *except* for boundary vertices, of which there are at most $O(N/\sqrt{r})$. For each parent y of a node $x \in S_r$, the corresponding region R_y has more than r vertices, so the number of such parents is $O(N/r)$, so $|S_r|$ is $O(N/r)$. Let c' be a constant to be determined. For a node x, let $S'_r(x)$ denote the set of rootmost descendants y of x (where x is a descendant of itself) such that R_y has at most $c'\sqrt{r}$ boundary vertices. Let $S'_r = \bigcup_{x \in S_r} \{S'_r(x)\}$.

Lemma 5.10.9. *The regions $\{R_y \,:\, y \in S'_r\}$ form an r–division with a constant number of holes per region.*

Proof. It follows from the definition of S'_r that each region in this set has at most r vertices and at most $c'\sqrt{r}$ boundary vertices. It follows by induction from Lemma 5.10.2 that each of these regions is connected. It follows from Lemma 5.10.4 that each region has at most twelve holes. It remains to show that $|S'_r|$ is $O(N/r)$.

Lemma 5.10.8 implies that $\sum_{x \in S_r} b(x) = O\left(\frac{N}{\sqrt{r}}\right)$. We claim that, for every node $x \in S_r$,

$$|S'_r(x)| \leq \max\{1, \frac{b(x)}{c\sqrt{r}} - 12\}. \tag{5.6}$$

Summing over $x \in S_r$ and using $\sum_{x \in S_r} b(x) = O\left(\frac{N}{\sqrt{r}}\right)$ proves that $|S'_r|$ is $O(N/r)$.

We set $c' = 40c$. The proof of Equation (5.6) is by induction. If $b(x) \leq c'\sqrt{r}$ then $S'_r(x) = \{x\}$, so the claim holds. Assume therefore that $b(x) > c'\sqrt{r}$. Let $y_1, \ldots, y_k$ be the rootmost descendants y of x such that $b(y) \leq c'\sqrt{r}$ or $\ell(y) - \ell(x) = 3$, ordered such that $|S'_r(y_1)| \geq |S'_r(y_2)| \geq \cdots \geq |S'_r(y_8)|$. Let q be the cardinality of $\{i \ : |S'_r(y_i)| > 1\}$. Observe that $q \leq 8$.

Case 0: $q = 0$. In this case, $|S'_r(x)| \leq 8$, and $\frac{b(x)}{c\sqrt{r}} - 12 > 40 - 12 \geq 8$.

Case 1: $q = 1$. For some ancestor y of y_1 that is a descendant of x, the separator chosen for R_y is balanced in terms of boundary vertices. It follows that $b(y_1) \leq \frac{3}{4}b(x) + 3c\sqrt{r}$. By the inductive hypothesis, $|S'_r(y_1)| \leq \frac{b(y_1)}{c\sqrt{r}} - 12$, so

$$\begin{aligned}
|S'_r(x)| &= |S'_r(y_1)| + k - 1 \\
&\leq \frac{b(y_1)}{c\sqrt{r}} - 12 + 7 \\
&\leq \frac{\frac{3}{4}b(x) + 3c\sqrt{r}}{c\sqrt{r}} - 12 + 7 \\
&\leq \frac{b(x)}{c\sqrt{r}} - \frac{b(x)}{4c\sqrt{r}} + 3 - 12 + 7 \\
&\leq \frac{b(x)}{c\sqrt{r}} - 12,
\end{aligned}$$

because $b(x) > c'\sqrt{r} = 40c\sqrt{r}$.

Case 2: $q = 2$. In this case, $b(y_1) + b(y_2) \leq b(x) + 6c\sqrt{r}$. Using the inductive hypothesis on y_1 and y_2, we have

$$\begin{aligned}
|S'_r(x)| &= |S'_r(y_1)| + |S'_r(y_2)| + k - 2 \\
&\leq \frac{b(y_1)}{c\sqrt{r}} - 12 + \frac{b(y_2)}{c\sqrt{r}} - 12 - 6 \\
&\leq \frac{b(x) + 6c\sqrt{r}}{c\sqrt{r}} - 12 - 12 - 6 \\
&\leq \frac{b(x)}{c\sqrt{r}} - 12.
\end{aligned}$$

Case 3: $q > 2$. This case is similar to Case 2. □

5.10.4 Running time

Now we consider the running time of 5.1. Clearly, the running time is dominated by the invocations of the simple cycle separator algorithm (Theorem 5.8.1 in

Line 11, which take time linear in the size of R', which is $O(|R|)$. The running time is therefore bounded by $\sum_{x\in\mathcal{T}} |R_x|$. We shall count the contribution of nodes of $\mathcal{T}$ to this sum by the rank of nodes of $\mathcal{T}$.

The sum of $b(x)$ over all rank-k nodes x in $\mathcal{T}$ is bounded by the sum over the new boundary vertices created for all nodes of $\mathcal{T}$ with rank at most k. This is precisely the quantity we had shown to be $O(N/\sqrt{r_k})$ in the proof of Lemma 5.10.8. Therefore, by Lemma 5.10.6 and since regions of nodes that are not an ancestor of one another are disjoint except for boundary vertices,

$$\sum_{x \text{ has rank } k} |R_x| = O(N).$$

Since the number of different ranks is $O(\log N)$, the total running time for computing the decomposition tree $\mathcal{T}$ is $O(N \log N)$. It is easy to see that finding the set S'_r of nodes of $\mathcal{T}$ that correspond to an r–division can be done in additional $O(N)$ time.

5.11 Recursive divisions

Definition 5.11.1. *For an increasing sequence $\boldsymbol{r} = (r_0, r_1, \ldots)$ of numbers, a* recursive $\boldsymbol{r}$–division of G (with few holes *is a decomposition tree for G in which, for $i = 1, 2, \ldots$, the nodes at height i correspond to regions that form an r_i–division of G (with few holes).*

A region consisting of a single arc uv is denoted $R(uv)$. Such a region is said to be *atomic*. It follows from Theorem 5.10.1 that

Corollary 5.11.2. *There is an $O(n \log n)$ that, for any triangulated plane graph G and for any increasing sequence $\boldsymbol{r}$, outputs a recursive $\boldsymbol{r}$–division of G with few holes.*

5.12 Chapter Notes

Ungar [Ungar, 1951] gave the first separator theorem for planar graph. His separator had $O(\sqrt{n}\log^{3/2} n)$ vertices. Lipton and Tarjan [Lipton and Tarjan, 1979] proved the first $O(\sqrt{n})$ separator theorem for planar graphs. Constant-factor improvements were found by Djidjev [Djidjev, 1981] and Gazit [Gazit, 1986].

Miller [Miller, 1986] proved the first cycle-separator theorem for planar graphs. Miller's result was more general and had better constants than the cycle separator algorithm in Section 5.8. A constant-factor improvement was found by Djidjev and Venkatesan [Djidjev and Venkatesan, 1997].

The $O(\sqrt{n})$-vertex separator in Section 5.5, and the $O(\sqrt{n})$ cycle separator in Section 5.8 are not quite the algorithms of Lipton and Tarjans [Lipton and Tarjan, 1979] and of Miller [Miller, 1986]. They were designed so that the cycle separator algorithm will follow more naturally from the vertex separator algorithm. The cycle separator algorithm includes ideas from [Klein et al., 2013, Fox-Epstein et al., 2013], and from [Har-Peled and Nayyeri, 2017].

The notion of an r-division is due to Frederickson [Frederickson, 1987], who gave an algorithm for finding one in $O(n \log n)$ time. Goodrich [Goodrich, 1995] gave a $O(n)$-time algorithm for computing the complete decomposition tree resulting from recursively separating an input graph using small vertex separators, until each region contains a single edge. His method can be adapted to produce a recursive $\boldsymbol{r}$–division, using the overall approach of Section 5.10. As for r–divisions with few holes, these were first used in algorithms of Klein and Subramanian [Klein and Subramanian, 1993, Subramanian, 1995], and subsequently in many other works. Klein, Mozes and Sommer [Klein et al., 2013] gave a linear time algorithm for computing a recursive $\boldsymbol{r}$–division with few holes.