

# Fully-dynamic Planarity Testing in Polylogarithmic Time

Jacob Holm<sup>\*1</sup> and Eva Rotenberg<sup>†2</sup>

<sup>1</sup>University of Copenhagen    jaho@di.ku.dk

<sup>2</sup>Technical University of Denmark    erot@dtu.dk

## Abstract

Given a dynamic graph subject to insertions and deletions of edges, a natural question is whether the graph presently admits a planar embedding. We give a deterministic fully-dynamic algorithm for general graphs, running in amortized  $\mathcal{O}(\log^3 n)$  time per edge insertion or deletion, that maintains a bit indicating whether or not the graph is presently planar. This is an exponential improvement over the previous best algorithm [Eppstein, Galil, Italiano, Spencer, 1996] which spends amortized  $\mathcal{O}(\sqrt{n})$  time per update.

---

<sup>\*</sup>Partially supported by the VILLUM Foundation grant 16582, Basic Algorithms Research Copenhagen (BARC).

<sup>†</sup>Partially supported by Independent Research Fund Denmark grant “AlgoGraph” 2018-2021 (8021-00249B).

# 1 Introduction

A linear time algorithm for determining whether a graph is planar was found by Hopcroft and Tarjan [14]. For the partially dynamic case, where one only allows insertion of edges, the problem was solved by La Poutré [17], who improved on work by Di Battista, Tamassia, and Westbrook [5, 19, 20], to obtain an amortized running time of  $\mathcal{O}(\alpha(q, n))$  where  $q$  is the number of operations, and where  $\alpha$  is the inverse-Ackermann function. Galil, Italiano, and Sarnak [8] made a data structure for fully dynamic planarity testing with  $\mathcal{O}(n^{\frac{2}{3}})$  amortized time per update, which was improved to  $\mathcal{O}(\sqrt{n})$  by Eppstein et al. [7].

While there has for a long time been no improvements upon [7], there have been different approaches in works that address the task of maintaining an embedded graph. In [15], Italiano, La Poutré, and Rauch present a data structure for maintaining a planar embedded graph while allowing insertions that do not violate the embedding, but allowing arbitrary deletions; its update time is  $\mathcal{O}(\log^2 n)$ . Eppstein [6] presents a data structure for maintaining a dynamic embedded graph, which handles updates in  $\mathcal{O}(\log n)$  time if the embedding remains plane—this data structure maintains the genus of the embedding, but does not answer whether another embedding of the same graph with a lower genus exists.

Pătraşcu and Demaine [16] give a lower bound of  $\Omega(\log n)$  for fully-dynamic planarity testing. For other natural questions about fully-dynamic graphs, such as fully dynamic shortest paths, even on planar graphs, there are conditional lower bounds based on popular conjectures that indicate that subpolynomial update bounds are unlikely [2, 1].

In this paper, we show that planarity testing does indeed admit a subpolynomial update time algorithm, thus exponentially improving the state of the art for fully-dynamic planarity testing. We give a deterministic fully-dynamic algorithm for general graphs, running in amortized  $\mathcal{O}(\log^3 n)$  time per edge insertion or deletion, that explicitly maintains a single bit indicating whether the graph is presently planar, and that given any vertex can answer whether the connected component containing that vertex is planar in worst case  $\mathcal{O}(\log n / \log \log n)$  time.

In fact, our algorithm not only maintains *whether* the graph is presently planar, but also implicitly maintains *how* the graph may be embedded in the plane, in the affirmative case. Specifically, we give a deterministic algorithm that maintains a planar embedding of a fully dynamic planar graph in amortized  $\mathcal{O}(\log^3 n)$  time per edge insertion, and worst case  $\mathcal{O}(\log^2 n)$  time per edge deletion. In this algorithm, attempts to insert edges that would violate planarity are detected and rejected, but may still change the embedding. The algorithm for fully-dynamic general graphs then follows by a simple reduction.

Our main result consists of two parts which may be of independent interest. Our analysis goes via a detailed understanding of *flips*, i.e. local changes to the embedding, to be defined in Section 1.1. Firstly, we consider any algorithm for maintaining an embedding that lazily makes no changes to the embedding upon edge deletion, and that for each (attempted) insertion greedily only does the minimal (or close to minimal) number of flips necessary to accommodate the edge. We prove that any such algorithm will do amortized  $\mathcal{O}(\log n)$  flips. Secondly, we show how to find such a sufficiently small set of flips in worst case  $\mathcal{O}(\log^2 n)$  time per flip.

The idea of focusing on flips is not new: In [11], we use insights from Eppstein [6] to improve upon the data structure by Italiano, La Poutré, and Rauch [15], so that it also facilitates *flips*, i.e. local changes to the embedding, and, so that it may handle edge-insertions that only require one such flip. In [12], we analyze these *flips* further and show that there exists a class of embeddings

where only  $\Theta(\log n)$  flips are needed to accommodate any one edge insertion that preserves planarity.

The core idea of our analysis of the lazy greedy algorithm is to define a potential function based on how far the current embedding is from being in this class. This is heavily inspired by the analysis of Brodal and Fagerberg’s algorithm for fully-dynamic bounded outdegree orientation [4].

## 1.1 Maintaining an embedding if it exists

Before stating our results in detail, we will define some crucial but natural terminology for describing changeable embeddings of dynamic graphs.

**Planar graphs** are graphs that can be drawn in the plane without edge crossings. A planar graph may admit many planar embeddings, and we use the term *plane graph* to denote a planar graph equipped with a given planar embedding. Given a plane graph, its drawing in the plane defines *faces*, and the faces together with the edges form its *dual graph*. Related, one may consider the bipartite *vertex-face (multi-)graph* whose nodes are the vertices and faces, and which has an edge for each time a vertex is incident to a face. Through the paper, we will use the term *corner* to denote an edge in the vertex-face graph, reflecting that it corresponds to a corner of the face in the planar drawing of the graph.

If a planar graph has no vertex cut-sets of size  $\leq 2$ , its embedding is unique up to reflection. On the other hand, if a plane graph has an articulation point (cut vertex) or a separation pair (2-vertex cut), then it may be possible to alter the embedding by *flipping* [21, 11, 12] the embedding in that point or pair (see figure 1). Given two embeddings of the same graph, the flip-distance between them is the minimal number of flips necessary to get from one to the other. Intuitively, a flip can be thought of as cutting out a subgraph by cutting along a 2-cycle or 4-cycle in the vertex-face graph, possibly mirroring its planar embedding, and then doing the inverse operation of cutting along a 2- or 4-cycle in the vertex-face graph. The initial cutting and the final gluing involve the same vertices but not necessarily the same faces, thus, the graph but not the embedding is preserved.

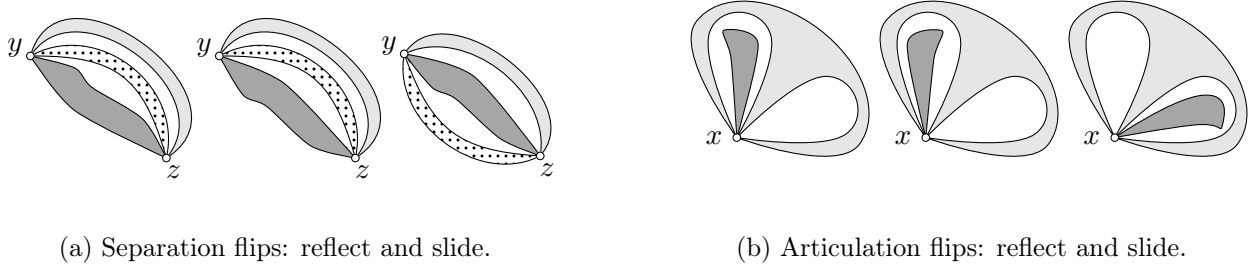


Figure 1: Local changes to the embedding of a graph [12].

We use the following terminology: To distinguish between whether the subgraph being flipped is connected to the rest of the graph by a separation pair or an articulation point, we use the terms *separation flip* and *articulation flip*, respectively. To indicate whether the subgraph was mirrored, moved to a different location, or both, we use the terms *reflect*, *slide*, and *reflect-and-slide*. Note that for articulation flips, only slide and reflect-and-slide change which edges are insertable across a face. For separation flips, note that any slide operation or reflect-and-slide operation may be obtained by doing 3 or 2 reflect operations, respectively.

**Results.** Let  $n$  denote the number of vertices of our fully-dynamic graph.

**Theorem 1.** *There is a data structure for fully-dynamic planarity testing that handles edge-insertions and edge-deletions in amortized  $\mathcal{O}(\log^3 n)$  time, answers queries to planarity-compatibility of an edge in amortized  $\mathcal{O}(\log^3 n)$  time, and answers queries to whether the graph is presently planar in worst case  $\mathcal{O}(1)$  time, or to whether the component of a given vertex is presently planar in worst case  $\mathcal{O}(\log n / \log \log n)$  time. It maintains an implicit representation of an embedding that is planar on each planar connected component, and may answer queries to the neighbors of a given existing edge in this current embedding, in  $\mathcal{O}(\log^2 n)$  time.*

The result follows by applying a simple extension of the reduction by Eppstein et al. [7, Corollary 1] to the following theorem:

**Theorem 2.** *There is a data structure for maintaining a planar embedding of a fully-dynamic planar graph that handles edge-updates and planarity-compatibility queries in amortized  $\mathcal{O}(\log^3 n)$  time, edge deletions in worst-case  $\mathcal{O}(\log^2 n)$  time, and queries to the neighbors of a given existing edge in the current embedding in worst-case  $\mathcal{O}(\log^2 n)$  time.*

The underlying properties in the data structure above include that the queries may change the embedding, but the deletions do not change anything aside from the mere deletion of the edge itself.

To arrive at these theorems, we prove some technical lemmas which may be of independent interest. A weak form of these that is easy to state is the following:

**Lemma 3.** *Any algorithm for maintaining a fully dynamic planar embedding that for each attempted edge-insertion greedily does the minimal number of flips, and that for each edge deletion lazily does nothing, will do amortized  $\mathcal{O}(\log n)$  flips per insertion when starting with an empty graph (or amortized over  $\Omega(n / \log n)$  operations).*

## 1.2 Article outline

In Section 2, we introduce some of the concepts and data structures that we use to prove our result. In Section 3 we give the proof of Lemma 3 conditioned on insights and details deferred to Section 6. In Section 4, we prove Theorem 2 by giving an algorithm (given an edge to insert and an embedded graph) for greedily finding flips that bring us closer to an embedding that is compatible with the edge we are trying to insert. In Section 5, we show how the reduction that extends this result to general graphs, proving Theorem 1.

## 2 Preliminaries

Since each 3-connected component of a planar graph has a unique embedding up to reflection, the structure of 3-connected components play an important role in the analysis of embeddings. Namely, the two-vertex cuts (also known as separation pairs) point to places where there is a choice in how to embed the graph. Similarly, any cutvertex (also known as articulation point) points to a freedom in the choice of embedding. In the following, we will define the BC tree and the SPQR tree which are structures that reflect the 2-connected components of a connected graph and the 3-connected components of a 2-connected graph, respectively. Then, related to the understanding of a combinatorial embedding, we will define *flips* which are local changes to the embedding, and the notion of *flip distance* between embedded graphs.

**BC trees**, as described in [9, p. 36], reflect the 2-connected components and their relations.

**Definition 4.** Let  $x$  be a vertex in a connected loopless multigraph  $G$ . Then  $x$  is an *articulation point* if  $G - \{x\}$  is not connected.

**Definition 5.** A (strict) BC tree for a connected loopless multigraph  $G = (V, E)$  with at least 1 edge is a tree with nodes labelled  $B$  and  $C$ , where each node  $v$  has an associated *skeleton graph*  $\Gamma(v)$  with the following properties:

- For every node  $v$  in the BC tree,  $V(\Gamma(v)) \subseteq V$ .
- For every edge  $e \in E$  there is a unique node  $v = b(e)$  in the BC tree such that  $e \in E(\Gamma(v))$ .
- For every edge  $(u, v)$  in the BC tree,  $V(\Gamma(u)) \cap V(\Gamma(v)) \neq \emptyset$  and either  $u$  or  $v$  is a  $C$  node.
- If  $v$  is a  $B$  node,  $\Gamma(v)$  is either a single edge or a biconnected graph.
- If  $v$  is a  $C$  node,  $\Gamma(v)$  consists of a single vertex, which is an articulation point in  $G$ .
- No two  $B$  nodes are neighbors.
- No two  $C$  nodes are neighbors.

The BC tree for a connected graph is unique. The (skeleton graphs associated with) the  $B$  nodes are sometimes referred to as  $G$ 's biconnected components. In this paper, we use the term *relaxed* BC tree as defined in [12] to denote a tree that satisfies all but the last condition. Unlike the strict BC tree, the relaxed BC tree is not unique.

**SPQR trees** Reflecting the structure of the 3-connected components, we rely on the SPQR tree.

**Definition 6** (Hopcroft and Tarjan [13, p. 6]). Let  $\{a, b\}$  be a pair of vertices in a biconnected multigraph  $G$ . Suppose the edges of  $G$  are divided into equivalence classes  $E_1, E_2, \dots, E_k$ , such that two edges which lie on a common path not containing any vertex of  $\{a, b\}$  except as an end-point are in the same class. The classes  $E_i$  are called the *separation classes* of  $G$  with respect to  $\{a, b\}$ . If there are at least two separation classes, then  $\{a, b\}$  is a *separation pair* of  $G$  unless (i) there are exactly two separation classes, and one class consists of a single edge<sup>1</sup>, or (ii) there are exactly three classes, each consisting of a single edge<sup>2</sup>.

**Definition 7** ([10]). The (*strict*) *SPQR tree* for a biconnected multigraph  $G = (V, E)$  with at least 3 edges is a tree with nodes labeled S, P, or R, where each node  $x$  has an associated *skeleton graph*  $\Gamma(x)$  with the following properties:

- For every node  $x$  in the SPQR tree,  $V(\Gamma(x)) \subseteq V$ .
- For every edge  $e \in E$  there is a unique node  $x = b(e)$  in the SPQR tree such that  $e \in E(\Gamma(x))$ .
- For every edge  $(x, y)$  in the SPQR tree,  $V(\Gamma(x)) \cap V(\Gamma(y))$  is a separation pair  $\{a, b\}$  in  $G$ , and there is a *virtual edge*  $ab$  in each of  $\Gamma(x)$  and  $\Gamma(y)$  that corresponds to  $(x, y)$ .
- For every node  $x$  in the SPQR tree, every edge in  $\Gamma(x)$  is either in  $E$  or a virtual edge.
- If  $x$  is an S node,  $\Gamma(x)$  is a simple cycle with at least 3 edges.
- If  $x$  is a P node,  $\Gamma(x)$  consists of a pair of vertices with at least 3 parallel edges.
- If  $x$  is an R node,  $\Gamma(x)$  is a simple triconnected graph.
- No two S nodes are neighbors, and no two P nodes are neighbors.

The SPQR tree for a biconnected graph is unique (see e.g. [5]). In this paper, we use the term *relaxed SPQR tree* as defined in [12] to denote a tree that satisfies all but the last condition. Unlike the strict SPQR tree, the relaxed SPQR tree is not unique.

<sup>1</sup>So in a triconnected graph, the endpoints of an edge do not constitute a separation pair.

<sup>2</sup>So the graph consisting of two vertices connected by 3 parallel edges is triconnected.

**Pre-split BC trees and SPQR trees.** Once the BC tree or the SPQR is rooted, one may form a path decomposition [18] over them. Given a connected component, one may form its BC tree and an SPQR tree for each block. In [12], we show how to obtain a balanced combined tree for each component, inspired by [3], where the heavy paths reflect not only the local SPQR tree of a block, but also the weight of the many other blocks.

Given a heavy path decomposition, [12] introduced *presplit* versions of BC trees and SPQR trees, in which cutvertices, P nodes and S nodes that lie internal on heavy paths have been split in two, thus transforming the strict BC tree or SPQR tree into a relaxed BC or SPQR tree.

**Flip-finding.** In [11] we give a structure for maintaining a planar embedded graph subject to edge deletions, insertions across a face, and flips changing the embedding. It operates using the tree-cotree decomposition of a connected plane graph; for any spanning tree, the non-tree edges form a spanning tree of the dual graph where faces and vertices swap roles. The data structure allows the following interesting operation: Mark a constant number of faces, and search for vertices along a path in the spanning tree that are incident to all marked faces. Or, dually, mark a number of vertices and search for faces along a path in the cotree. This mark-and-search operation is supported in  $\mathcal{O}(\log^2 n)$  time. Originally, in [11], we use the mark-and-search operation to detect single flips necessary to bring a pair of vertices to the same face. It turns out this operation is more powerful than previously assumed, and we will use it as part of the machinery that finds all the possibly many flips necessary to bring a pair of vertices to the same face.

**Good embeddings** In [12], a class of good embeddings are provided that share the property that any edge that can be added without violating planarity only requires  $\mathcal{O}(\log n)$  flips to the current embedding. The good embeddings relate to the dynamic balanced heavy path decompositions of BC trees and SPQR trees in the following way: For all the articulation points and separation pairs that lie internal on a heavy path, the embedding of the graph should be favorable to the possibility of an incoming edge connecting the endpoints of the heavy path, if possible. No other choices to the embedding matter; the properties of the heavy path decomposition ensure that only  $\mathcal{O}(\log n)$  such choices may be unfavorable to accommodating any planarity-preserving edge insertion.

**Projections and meets** For vertices  $x, y$ , and  $z$  on a tree, we use  $\text{meet}(x, y, z)$  to denote the unique common vertex on all 3 tree paths between  $x, y$ , and  $z$ . Alternatively  $\text{meet}(x, y, z)$  can be seen as the projection of  $x$  on the tree path from  $y$  to  $z$ . For a vertex  $x$  and a fundamental cycle  $C$  in a graph with some implied spanning tree, let  $\pi_C(x)$  denote the projection of  $x$  on  $C$ . Note that if  $(y, z)$  is the non-tree edge closing the cycle  $C$  then  $\pi_C(x) = \text{meet}(x, y, z)$ .

### 3 Analyzing the number of flips in the lazy greedy algorithm

This section is dedicated to the proof of Lemma 3. We define some distance measures and a concept of good embeddings, and single out exactly the properties of these that would be sufficient for the lazy analysis to go through. All proofs that these sufficient conditions indeed are met are deferred to Section 6, thus enabling us to give an overview within the first limited number of pages.

**Definition 8.** We will consider only two kinds of flips:

- An articulation flip at  $a$  takes a single contiguous subsequence of the edges incident to  $a$  out, possibly reverses their order, and inserts them again, possibly in a different position.
- A separation flip at  $s, t$  reverses a contiguous subsequence of the edges incident to each of  $s, t$ .

**Definition 9.** We will distinguish between two types of separation flips at  $s, t$ :

- In a P flip, each of the two subgraphs consist of at least 2  $\{s, t\}$ -separation classes.
- In an SR flip, at least one of the two subgraphs consist of a single  $\{s, t\}$ -separation class.

**Definition 10.** A separation flip at  $s, t$  is *clean* if the first and last edges in each of the subsequences being reversed are biconnected, and *dirty* otherwise. A clean separation flip preserves the sets of edges that could participate in an articulation flip. All articulation flips are considered clean.

**Definition 11.** Given vertices  $u, v$ , a flip is called *critical* (for  $u, v$ ) if exactly one of  $u$  and  $v$  is in the subgraph being flipped.

**Definition 12.** For a planar graph  $G$ , let  $\text{Emb}(G)$  denote the graph whose nodes are planar embeddings of  $G$ , and  $(H, H')$  is an edge if  $H'$  is obtained from  $H$  by applying a single flip. As a slight abuse of notation we will also use  $\text{Emb}(G)$  to denote the set of all planar embeddings of  $G$ . Furthermore, for vertices  $u, v$  in  $G$  let  $\text{Emb}(G; u, v)$  denote the (possibly empty) set of embeddings of  $G$  that admit insertion of  $(u, v)$ .

We will often need to discuss distances in some (pseudo)metric between a particular embedding  $H \in \text{Emb}(G)$  and some particular subset of embeddings  $S \subseteq \text{Emb}(G)$ . For this, we define (for any metric or pseudometric  $\text{dist}$ )

$$\text{dist}(H, S) = \text{dist}(S, H) := \min_{H' \in S} \text{dist}(H, H')$$

**Definition 13.** For any two embeddings  $H, H'$  of the planar graph  $G$ , that is,  $H, H' \in \text{Emb}(G)$ , we say that a path from  $H$  to  $H'$  in  $\text{Emb}(G)$  is clean if every flip on the path is clean.

- Let  $\text{dist}_{\text{clean}}(H, H')$  be the length of a shortest clean path from  $H$  to  $H'$ .
- Let  $\text{dist}_{\text{sep}}(H, H')$  be the minimum number of separation flips on a clean path from  $H$  to  $H'$ .
- Let  $\text{dist}_P(H, H')$  be the minimum number of P flips on a clean path from  $H$  to  $H'$ .

**Observation 14.**  $\text{dist}_{\text{clean}}$  is a metric, and  $\text{dist}_{\text{sep}}$  and  $\text{dist}_P$  are pseudometrics, on  $\text{Emb}(G)$ .

In Section 6, we define two families of functions related to these particular (pseudo) metrics. Intuitively, for each  $\tau \in \{\text{clean}, \text{sep}, P\}$ , any planar graph  $G$  containing vertices  $u, v$ , and any embedding  $H \in \text{Emb}(G)$ :

- $\text{critical-cost}_\tau(H; u, v)$  is the number of flips of type  $\tau$  needed to accommodate  $(u, v)$  (if possible).
- $\text{solid-cost}_\tau(H; u, v)$  is the number of flips of type  $\tau$  needed to reach a “good” embedding that accommodates  $(u, v)$  (if possible).

We will state the properties we need for these functions here, in the form of Lemmas (to be proven once the actual definition has been given).

**Lemma 15.** For any planar graph  $G$  with vertices  $u, v$ , and any embedding  $H \in \text{Emb}(G)$ ,

$$\begin{aligned} \text{solid-cost}_\tau(H; u, v) &\geq \text{critical-cost}_\tau(H; u, v) \geq 0 \\ \text{solid-cost}_{\text{clean}}(H; u, v) &\geq \text{solid-cost}_{\text{sep}}(H; u, v) \geq \text{solid-cost}_P(H; u, v) \\ \text{critical-cost}_{\text{clean}}(H; u, v) &\geq \text{critical-cost}_{\text{sep}}(H; u, v) \geq \text{critical-cost}_P(H; u, v) \end{aligned}$$

And if  $G \cup (u, v)$  is planar,

$$\text{critical-cost}_{\text{clean}}(H; u, v) = 0 \iff H \in \text{Emb}(G; x, y)$$

**Lemma 16.** *Let  $u, v$  be vertices in a planar graph  $G$ , let  $H \in \text{Emb}(G)$  and let  $H' \in \text{Emb}(G)$  be the result of a single flip  $\sigma$  in  $H$ . Define*

$$\begin{aligned}\Delta \text{critical-cost}_\tau &:= \text{critical-cost}_\tau(H'; u, v) - \text{critical-cost}_\tau(H; u, v) \\ \Delta \text{solid-cost}_\tau &:= \text{solid-cost}_\tau(H'; u, v) - \text{solid-cost}_\tau(H; u, v)\end{aligned}$$

*then  $\Delta \text{critical-cost}_\tau \in \{-1, 0, 1\}$ ,  $\Delta \text{solid-cost}_\tau \in \{-1, 0, 1\}$ , and*

$$\Delta \text{critical-cost}_\tau \neq 0 \quad \implies \quad \sigma \text{ is a critical flip} \quad \iff \quad \Delta \text{solid-cost}_\tau = \Delta \text{critical-cost}_\tau$$

**Lemma 17.** *Let  $u, v$  be vertices in a planar graph  $G$ , and let  $H \in \text{Emb}(G)$ .*

- *If  $\text{critical-cost}_\tau(H; u, v) > 0$  then there exists a clean flip in  $H$  such that the resulting  $H' \in \text{Emb}(G)$  has  $\text{critical-cost}_\tau(H'; u, v) < \text{critical-cost}_\tau(H; u, v)$ .*
- *If  $\text{solid-cost}_\tau(H; u, v) > 0$  then there exists a clean flip in  $H$  such that the resulting  $H' \in \text{Emb}(G)$  has  $\text{solid-cost}_\tau(H'; u, v) < \text{solid-cost}_\tau(H; u, v)$ .*

The main motivation for defining critical-cost comes from the following

**Corollary 18.** *Let  $G$  be a planar graph, let  $u, v$  be vertices in  $G$  such that  $G \cup (u, v)$  is planar, and let  $H \in \text{Emb}(G)$ . Then*

$$\text{dist}_\tau(H, \text{Emb}(G; u, v)) = \text{critical-cost}_\tau(H; u, v).$$

*Proof.* “ $\geq$ ” follows from Lemmas 15 and 16, and “ $\leq$ ” follows from Lemma 17. □

With these properties in hand, we can now redefine what we mean by a *good* embedding in a quantifiable way:

**Definition 19.** Given a planar graph  $G$  with vertices  $u, v$ , the *good* embeddings of  $G$  with respect to  $u, v$  is the set

$$\text{Emb}^*(G; u, v) := \{H \in \text{Emb}(G) \mid \text{solid-cost}_{\text{clean}}(H; u, v) = 0\}$$

And the set of all good embeddings of  $G$  is

$$\text{Emb}^*(G) := \bigcup_{u, v} \text{Emb}^*(G; u, v) = \{H \in \text{Emb}(G) \mid \min_{u, v} \text{solid-cost}_{\text{clean}}(H; u, v) = 0\}$$

Note that this definition is not the same as in [12], although the underlying ideas are the same. With this definition, we get

**Corollary 20.** *Let  $G$  be a planar graph, let  $u, v$  be vertices in  $G$ , and let  $H \in \text{Emb}(G)$ . Then*

$$\begin{aligned}\text{dist}_\tau(H, \text{Emb}^*(G; u, v)) &= \text{solid-cost}_\tau(H; u, v) \\ \text{dist}_\tau(H, \text{Emb}^*(G)) &= \min_{u, v} \text{solid-cost}_\tau(H; u, v)\end{aligned}$$

*Proof.* “ $\geq$ ” follows from Lemmas 15 and 16, and “ $\leq$ ” follows from Lemma 17. □

The reason we call these embeddings good is the following property



**Lemma 21.** *Given a planar graph  $G$  with vertices  $u, v$  and any  $H \in \text{Emb}^*(G)$ , then*

$$\text{dist}_{\text{clean}}(H, \text{Emb}^*(G; u, v)) \in \mathcal{O}(\log n)$$

**Corollary 22.** *Given a planar graph  $G$  with vertices  $u, v$  and any  $H \in \text{Emb}(G)$ , then*

$$\text{dist}_\tau(H, \text{Emb}^*(G)) \leq \text{dist}_\tau(H, \text{Emb}^*(G; u, v)) \leq \text{dist}_\tau(H, \text{Emb}^*(G)) + \mathcal{O}(\log n)$$

*Proof.* The first inequality follows from  $\text{Emb}^*(G) \supseteq \text{Emb}^*(G; u, v)$ . For the second, let  $H' \in \text{Emb}^*(G)$  minimize  $\text{dist}_\tau(H, H')$ . By Corollary 20 and Lemmas 15 and 21,

$$\text{dist}_\tau(H', \text{Emb}^*(G; u, v)) \leq \text{dist}_{\text{clean}}(H', \text{Emb}^*(G; u, v)) \in \mathcal{O}(\log n)$$

and the result follows by the triangle inequality.  $\square$

**Lemma 23.** *Given a planar graph  $G$  with vertices  $u, v$ , such that  $G \cup (u, v)$  is planar, and any  $H \in \text{Emb}(G; u, v)$ . Then*

$$\text{solid-cost}_\tau(H; u, v) = \text{solid-cost}_\tau(H \cup (u, v); u, v)$$

Everything so far has been stated in terms of clean flips. Most of our results do not depend on this, due to the following lemma (Proved in Section 6.3).

**Lemma 24.** *A dirty separation flip corresponds to a clean separation flip and at most 4 articulation flips. At most one of these 5 flips change  $\text{solid-cost}_\tau$  or  $\text{critical-cost}_\tau$ .*

**Theorem 25.** *Let  $p, q, r \in \mathbb{N}_0$  be nonnegative integer constants. Let  $\mathcal{A}$  be a lazy greedy algorithm for maintaining a planar embedding with the following behavior:*

- $\mathcal{A}$  does no flips during edge deletion; and
- during the attempted insertion of the edge  $(u, v)$  into an embedded graph  $H$ ,  $\mathcal{A}$  only uses critical flips; and
- this sequence of flips can be divided into steps of at most  $r$  flips, such that each step (except possibly the last) decreases the following potential by at least 1:

$$\text{critical-cost}_{\text{clean}}(H; u, v) + p \cdot \text{critical-cost}_{\text{sep}}(H; u, v) + q \cdot \text{critical-cost}_P(H; u, v)$$

*Then,  $\mathcal{A}$  uses amortized  $\mathcal{O}(\log n)$  steps per attempted edge insertion.*

*Proof.* Let  $G$  be a planar graph with vertices  $u, v$ , such that  $G \cup (u, v)$  is planar, let  $H_0 \in \text{Emb}(G)$  be the embedding before inserting  $(u, v)$  and let  $H_1, \dots, H_k \in \text{Emb}(G)$  be the embedded graphs after each “step” until finally  $H_k \in \text{Emb}(G; u, v)$ .

Define  $\text{dist}_{\text{alg}} = \text{dist}_{\text{clean}} + p \cdot \text{dist}_{\text{sep}} + q \cdot \text{dist}_P$ . Then  $\text{dist}_{\text{alg}}$  is a metric on  $\text{Emb}(G)$ , and for each  $H \in \text{Emb}(G)$  we can define

$$\begin{aligned} \Phi(H) &= \text{dist}_{\text{alg}}(H, \text{Emb}^*(G)) \\ \Phi(H; u, v) &= \text{dist}_{\text{alg}}(H, \text{Emb}^*(G; u, v)) \end{aligned}$$

By Corollary 22  $\Phi(H) \leq \Phi(H; u, v) \leq \Phi(H) + \mathcal{O}(\log n)$ .

By assumption,  $\text{dist}_{\text{alg}}(H, \text{Emb}(G; u, v))$  is strictly decreasing in each step, and by Lemma 16,  $\Phi(H; u, v)$  decreases by exactly the same amount. In particular, after  $k$  steps it has decreased by at

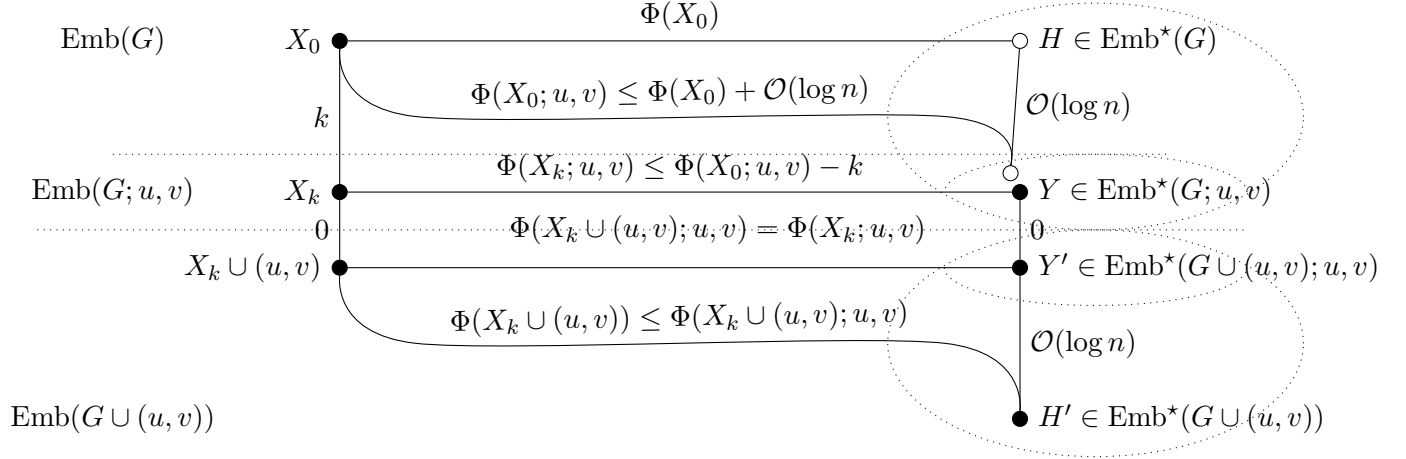


Figure 2: Illustration of the proof of Theorem 25.

least  $k$ , so  $\Phi(H_k; u, v) \leq \Phi(H_0; u, v) - k \leq \Phi(H_0) + \mathcal{O}(\log n) - k$ . By Lemma 23,  $\Phi(H_k \cup (u, v); u, v) = \Phi(H_k; u, v)$ , so

$$\Phi(H_k \cup (u, v)) \leq \Phi(H_k \cup (u, v); u, v) = \Phi(H_k; u, v) \leq \Phi(H_0; u, v) - k \leq \Phi(H_0) + \mathcal{O}(\log n) - k$$

The same argument holds when an attempted insert stops after  $k$  steps because  $G \cup (u, v)$  is not planar. Since the potential  $\Phi(H)$  increases by at most  $\mathcal{O}(\log n)$  and drops by at least the number of steps used, the amortized number of steps for each attempted insert is  $\mathcal{O}(\log n)$ .

For deletion it is even simpler, as

$$\Phi(H - (u, v)) \leq \Phi(H - (u, v); u, v) = \Phi(H; u, v) \leq \Phi(H) + \mathcal{O}(\log n)$$

Thus, each deletion increases the potential by  $\mathcal{O}(\log n)$ . However, as we start with an empty edge set, the number of deletions is upper bounded by the number of insertions, and so each edge can instead pay a cost of  $\mathcal{O}(\log n)$  steps when inserted to cover its own future deletion. In other words, deletions are essentially free.  $\square$

As a direct consequence, our main lemma holds.

**Lemma 3.** *Any algorithm for maintaining a fully dynamic planar embedding that for each attempted edge-insertion greedily does the minimal number of flips, and that for each edge deletion lazily does nothing, will do amortized  $\mathcal{O}(\log n)$  flips per insertion when starting with an empty graph (or amortized over  $\Omega(n/\log n)$  operations).*

*Proof.* Set  $p = q = 0$  and  $r = 1$  in the lemma above, and the result follows.  $\square$

Our algorithm may not decrease critical-cost in every step. To simplify the description of how and when it changes, we associate each type of flip with one of the critical-cost $_{\tau}$  as in Corollary 27:

- A P flip is associated with critical-cost $_p$
- An SR flip is associated with critical-cost $_{\text{sep}}$
- An articulation flip is associated with critical-cost $_{\text{clean}}$

A flip is potential-decreasing/potential-neutral/potential-increasing if it changes its associated cost by  $-1/0/1$  respectively.

Using the following lemma (proved in Section 6)

**Lemma 26.** *Any SR flip or articulation flip leaves  $\text{critical-cost}_P$  and  $\text{solid-cost}_P$  unchanged, and any articulation flip leaves  $\text{critical-cost}_{sep}$  and  $\text{solid-cost}_{sep}$  unchanged.*

we can simplify Theorem 25 to.

**Corollary 27.** *Let  $\mathcal{A}$  be a lazy greedy algorithm for planar embeddings that does no flips during edge deletion, and that during attempted edge insertion only uses critical flips such that*

- *Each flip (except possibly the last) is potential-decreasing or potential-neutral.*
- *For some constant  $r$ , no sequence of  $r$  consecutive flips are potential-neutral.*

*Then  $\mathcal{A}$  uses amortized  $\mathcal{O}(\log n)$  steps per attempted edge insertion.*

*Proof.* Simply use  $p = r + 1, q = r^2 + 2r + 1, r = r$  in Theorem 25, and note that Lemma 26 guarantees the resulting potential is strictly decreasing in each round of at most  $r$  flips.  $\square$

## 4 A Greedy Flip-Finding Algorithm

We use the data structure from [11] to represent the current embedding. This structure maintains *interdigitating spanning trees* (also known as the tree co-tree decomposition) for the primal and dual graphs under flips, admissible edge insertions, and edge deletions in worst case  $\mathcal{O}(\log^2 n)$  time per operation. In particular it supports the  $\text{LINKABLE}(u, v)$  operation, which in worst case  $\mathcal{O}(\log^2 n)$  time either determines that  $u$  and  $v$  has no face in common and returns “no”, or returns some pair of corners  $((u, f), (v, f))$  where  $f$  is a common face.

Furthermore, the structure allows for a mark-and-search operation, in which a constant number of faces may be “marked”, and vertices along a path on the spanning tree that are incident to all marked faces may be sought after in  $\mathcal{O}(\log^2 n)$  time. (Dually, one may mark vertices and search for faces in the same time, ie.  $\mathcal{O}(\log^2 n)$ .)

### 4.1 Algorithm overview

We want to use this structure to search for the flips needed to insert a new edge  $(u, v)$  that is not admissible in the current embedding. For simplicity, we will present an algorithm that fits the framework in Corollary 27, rather than insisting on finding a shortest sequence of clean flips. This is sufficient to get amortized  $\mathcal{O}(\log n)$  flips, and with  $\mathcal{O}(\log^2 n)$  overhead per flip, amortized  $\mathcal{O}(\log^3 n)$  time for (attempted) edge insertion. If desired, the algorithm can be made to detect if it has made non-optimal flips and backtrack to use an optimal sequence of flips without affecting the asymptotic amortized running time.

At the highest possible level of abstraction, our algorithm is just the  $\text{MULTI-FLIP-LINKABLE}$  routine from Algorithm 1. In the following we will go into more detail and provide detailed proofs. We will assume full knowledge of how to use the mark-and-search features from [11] to e.g. search a path in the dual tree for the first face containing a given pair of vertices.

Let  $a_1, \dots, a_{k-1}$  be the articulation points on  $u \cdots v$ , and let  $a_0 = u$  and  $a_k = v$ . For  $1 \leq i \leq k$  let  $B_i$  be the biconnected component (or bridge) containing  $a_{i-1} \cdots a_i$ . Our algorithm “cleans up” this path by sweeping from  $a_0 = u$  to  $a_k = v$ . At all times the algorithm keeps track of a latest

---

**Algorithm 1**


---

```

1: function MULTI-FLIP-LINKABLE( $u, v$ )
2:    $u' \leftarrow u$ 
3:   while  $u' \neq v$  do
4:     if  $u', v$  biconnected then
5:        $v' \leftarrow v$ 
6:     else
7:        $v' \leftarrow$  first articulation point on  $u' \cdots v$ .
8:     if not DO-SEPARATION-FLIPS( $u', v'$ ) then  $\triangleright$  Do all separation flips needed in  $B_{i+1}$ .
9:       return “no”
10:    DO-ARTICULATION-FLIPS( $u, u', v', v$ )  $\triangleright$  Do articulation flips required by  $B_{i+1}$ .
11:     $\triangleright$  Now  $u$  shares a face with  $v'$ , and (if  $v' \neq v$ ) with at least one edge in  $B_{i+2}$ .
12:     $u' \leftarrow$  FIND-NEXT-FLIP-BLOCK( $u, u', v', v$ )  $\triangleright$  Skip to next relevant  $a_i$ 
13:  return “yes”  $\triangleright u, v$  are now in same face

```

---

articulation point  $u' = a_i$  seen on  $u \cdots v$  (initially  $u' = a_0 = u$ ) such that either  $u' = u$  or  $(u, u')$  is admissible in the current embedding. We will further maintain the invariant that (unless  $i = k$ ) there is a common face of  $u$  and  $a_i$  that contains at least one edge from  $B_{i+1}$ . In the round where  $u' = a_i$  the algorithm sets  $v' = a_{i+1}$  and does the following:

1. It finds and applies all separation flips in  $B_{i+1}$  needed to make  $(a_i, a_{i+1})$  admissible, or detects (after some number of flips) that  $B_{i+1} \cup (a_i, a_{i+1})$  — and therefore  $G \cup (u, v)$  — is nonplanar.
2. It finds at most one articulation flip at  $u'$  and at most one articulation flip at  $v'$ , such that afterwards  $u$  shares a face with  $v' = a_{i+1}$ , and (if  $v' \neq v$ ) with at least one edge from  $B_{i+2}$ .
3. It finds the first  $a_j$  with  $j \geq i + 1$  such that either: the next iteration of the loop finds at least one flip; or no more flips are needed and  $a_j = v$ . It then sets  $u' \leftarrow a_j$ .

The algorithm stops when  $u' = v$ . By our invariant  $(u, v)$  is admissible if it reaches this point.

**Lemma 28.** *If  $G \cup (u, v)$  is planar and  $H_0 \in \text{Emb}(G)$ , this algorithm finds a sequence of graphs  $H_1, \dots, H_k \in \text{Emb}(G)$  such that  $H_k \in \text{Emb}(G; u, v)$ . The main loop performs  $\mathcal{O}(k)$  iterations.*

*Proof.* If  $G \cup (u, v)$  is planar, then for every block  $B_{i+1}$  there exists some (possibly empty) set of separation flips such that  $H \cup (a_i, a_{i+1})$  is planar. Thus, after line 8 we know that  $a_i$  and  $a_{i+1}$  share a face. And by our invariant, we also know that  $u = a_0$  and  $u' = a_i$  share a face, incident to at least one edge from  $B_{i+1}$ .

Now the call to DO-ARTICULATION-FLIPS in line 10 uses at most 2 articulation flips to update the invariant so  $u$  shares a face with  $v'$  and (if  $v \neq v'$ ) with at least one edge in  $B_{i+2}$ .

Finally, the call to FIND-NEXT-FLIP-BLOCK in line 12, updates  $u'$  to the largest  $a_j$  so the invariant still holds. In particular, either a separation flip is needed in  $B_{j+1}$ , or an articulation flip is needed in  $a_{j+1}$  before the face shared by  $u = a_0$  and  $a_{j+1}$  is incident to an edge in  $B_{j+2}$ .

Thus, in every iteration after (possibly) the first, at least one flip is performed. Thus if we stop after  $k$  flips, the number of iterations is at most  $k + 1$ .

We stop with  $H \notin \text{Emb}(G; u, v)$  only if there is a block  $B_{i+1}$  where DO-ARTICULATION-FLIPS( $a_i, a_{i+1}$ ) returns “no” because we have detected that  $G \cup (u, v)$  is nonplanar.

Otherwise we keep making progress, and will eventually have  $u' = v$ . By our invariant  $(u, v)$  are now in the same face, and thus  $H_k \in \text{Emb}(G; u, v)$ .  $\square$

## 4.2 FIND-NEXT-FLIP-BLOCK

The simplest part of our algorithm is the FIND-NEXT-FLIP-BLOCK function in Algorithm 2 we use to move to the next “interesting” articulation point, or to vertex  $v$  if we are done.

By interesting is meant the following: it is an articulation point  $a_j$  on the BC-path from  $u$  to  $v$  such that a flip in either  $a_j$ ,  $B_{j+1}$ , or  $a_{j+1}$  is necessary in order to bring  $u$  and  $v$  to the same face.

---

### Algorithm 2

---

```

1: function FIND-NEXT-FLIP-BLOCK( $u, u', v', v$ )
2:    $\triangleright u$  shares a face with  $v'$ , and (if  $v' \neq v$ ) with at least one edge in  $B_{i+2}$ .
3:   if  $v' \neq v$  then
4:      $f_u, c_u^1, c_u^2 \leftarrow \text{FIND-BOUNDING-FACE}(u, v', v)$   $\triangleright f_u$  is incident to  $u$ .
5:      $f_v, c_v^1, c_v^2 \leftarrow \text{FIND-BOUNDING-FACE}(v, v', u)$ 
6:     if  $f_u = f_v$  then
7:       if  $v$  incident to  $f_u$  then
8:         return  $v$ 
9:       return last internal node on  $u \cdots v$  touching  $f_u$  on both sides.
10:  return  $v'$ 

11: function FIND-BOUNDING-FACE( $u, a, v$ )
12:   $c_u \leftarrow$  any corner incident to  $u$ ;  $c_v \leftarrow$  any corner incident to  $v$ 
13:   $f \leftarrow$  first face on the dual path  $c_u \cdots c_v$  touching  $a$  on both sides.
14:   $c_L, c_R \leftarrow$  first corners on left and right side of  $c_u \cdots c_v$  that are incident to both  $a$  and  $f$ .
15:  return  $f, c_L, c_R$ 

```

---

**Lemma 29.** *If  $a_0$  and  $a_{i+1}$  share a face containing at least one edge from  $B_{i+2}$ , then in worst case  $\mathcal{O}(\log^2 n)$  time FIND-NEXT-FLIP-BLOCK( $a_0, a_i, a_{i+1}, a_k$ ) either returns the last  $a_j$  ( $i < j < k$ ) such that  $a_0$  and  $a_j$  share a face containing at least one edge from  $B_{j+1}$ ; or  $a_k$  if  $a_0$  and  $a_k$  share a face.*

*Proof.* First note that the running time is  $\mathcal{O}(\log^2 n)$  because the dominating subroutine is the mark-and-search algorithm from [11] called a constant number of times in FIND-BOUNDING-FACE(...), and once on line 9.

For correctness, assume  $a = a_{i+1}$  is an articulation point separating  $u = a_0$  from  $v = a_k$ . Then there is at least one articulation point in the dual graph incident to  $a$ , and all such articulation points lie on a path in the dual tree. By assumption,  $u$  lies in the bounding face of  $a_i$  and  $a_{i+1}$ , that is, in the face  $f$  returned by FIND-BOUNDING-FACE( $u, a_i, a_{i+1}$ ). Now there are four basic cases:

1. If  $a_{i+2}$  does not lie in  $f$ , then we have indeed reached a block where flips are necessary, and the algorithm returns  $a_{i+1}$  as desired. (See Figure 3)
2. On the other hand, if  $a_{i+2}$  lies in  $f$  but only has corners incident to  $f$  on one side, then we are in a case where a flip in  $a_{i+2}$  is necessary to bring  $u$  to the same face as  $v$ , and the algorithm returns  $a_{i+1}$  as desired. (See Figure 4)
3. Thirdly, it may be the case that  $a_{i+2}$  is incident to  $f$  on both sides of the path, in which case no flips in  $a_{i+1}$ ,  $B_{i+1}$ , or  $a_{i+2}$  are necessary. In this case there is a non-trivial segment of articulation points  $a_{i+1}, a_{i+2}, \dots$  that lie in  $f$ . Our algorithm will now return the last such  $a_j$  incident to  $f$  on both sides of the path. Here, we have two sub-cases. If  $a_{j+1}$  is not incident to  $f$ , this indicates that either we have reached a point  $a_j$  where an articulation flip is needed, or,

we have reached a block  $B_j$  where flips are needed. On the other hand, if  $a_{j+1}$  is incident to  $f$  but only on one side, we are in the case where an articulation flip in  $a_{j+1}$  is needed to bring  $u$  and  $v$  to the same face. In both cases, our algorithm returns  $a_j$  as desired. (See Figure 5)

4. Finally, we may be in the case where  $v$  lies in the same face as  $u$  and we are done, but in this case, the face  $f$  must be the shared face: Namely, since  $u$  and  $v$  are separated by at least one articulation point  $a$  in the primal graph with  $a$  incident to some face  $f$  on both sides of the tree-path from  $u$  to  $v$ , then there is a 2-cycle through  $f$  and  $a$  in the vertex-face graph separating  $u$  from  $v$ , and thus,  $f$  must be the unique face shared by all articulation points on any path  $u \cdots v$ . (See Figure 6)  $\square$

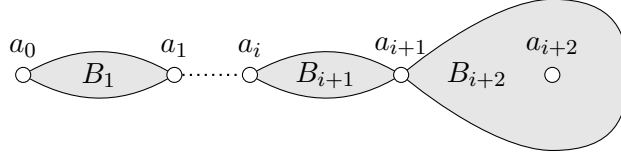


Figure 3: Lemma 29 case 1:  $a_{i+2}$  does not lie in  $f$ .  $\text{FIND-NEXT-FLIP-BLOCK}(a_0, a_i, a_{i+1}, a_k)$  returns  $a_{i+1}$ .

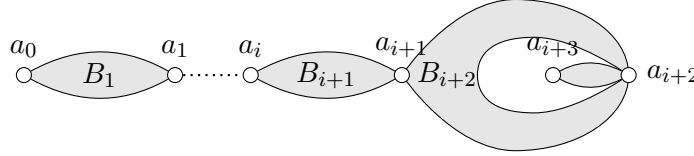


Figure 4: Lemma 29 case 2: the path  $a_0 \cdots a_k$  touches  $f$  on only one side in  $a_{i+2}$ .  $\text{FIND-NEXT-FLIP-BLOCK}(a_0, a_i, a_{i+1}, a_k)$  returns  $a_{i+1}$ .

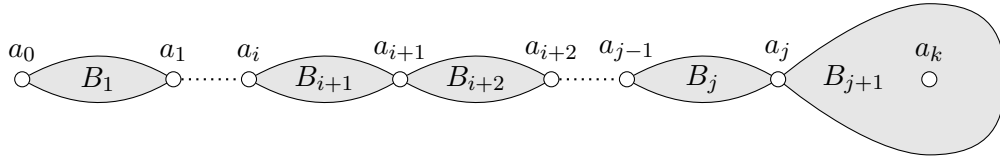


Figure 5: Lemma 29 case 3:  $\text{FIND-NEXT-FLIP-BLOCK}(a_0, a_i, a_{i+1}, a_k)$  returns  $a_j$  with  $i + 1 < j < k$ .

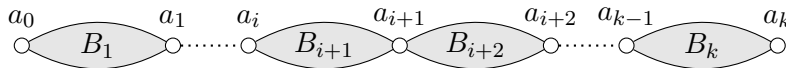


Figure 6: Lemma 29 case 4:  $\text{FIND-NEXT-FLIP-BLOCK}(a_0, a_i, a_{i+1}, a_k)$  returns  $a_k$ .

### 4.3 DO-ARTICULATION-FLIPS

The next piece of our algorithm is the DO-ARTICULATION-FLIPS function in Algorithm 3.

**Lemma 30.** *If  $a_0$  and  $a_i$  share a face containing at least one edge from  $B_{i+1}$ , and  $a_i$  and  $a_{i+1}$  share a face, then in  $\mathcal{O}(\log^2 n)$  time  $\text{DO-ARTICULATION-FLIPS}(a_0, a_i, a_{i+1}, a_k)$  does at most one articulation*

---

**Algorithm 3**

---

**Require:**  $u', v'$  linkable

```
1: function DO-ARTICULATION-FLIPS( $u, u', v', v$ )
2:   if  $u = u'$  then
3:     if  $v' = v$  then
4:        $\triangleright$  Nothing to do
5:     else
6:        $f_v, c_v^1, c_v^2 \leftarrow \text{FIND-BOUNDING-FACE}(v, v', u)$ 
7:       if  $f_v$  not incident to  $u'$  then
8:          $c_u, c_v \leftarrow \text{LINKABLE}(u', v')$ 
9:          $\text{ARTICULATION-FLIP}(c_v^1, c_v^2, c_v)$ 
10:    else
11:      if  $v' = v$  then
12:         $f_u, c_u^1, c_u^2 \leftarrow \text{FIND-BOUNDING-FACE}(u, u', v)$ 
13:        if  $f_u$  not incident to  $v'$  then
14:           $c_u, c_v \leftarrow \text{LINKABLE}(u', v')$ 
15:           $\text{ARTICULATION-FLIP}(c_u^1, c_u^2, c_u)$ 
16:        else  $u \neq u'$  and  $v' \neq v$ 
17:           $f_v, c_v^1, c_v^2 \leftarrow \text{FIND-BOUNDING-FACE}(v, v', u)$ 
18:           $f_u, c_u^1, c_u^2 \leftarrow \text{FIND-BOUNDING-FACE}(u, u', v)$ 
19:          if  $f_u = f_v$  then
20:             $\triangleright$  Nothing to do
21:          else if  $u'$  incident to  $f_v$  then
22:             $c_u \leftarrow$  any corner between  $u'$  and  $f_v$ 
23:             $\text{ARTICULATION-FLIP}(c_u^1, c_u^2, c_u)$ 
24:          else if  $v'$  incident to  $f_u$  then
25:             $c_v \leftarrow$  any corner between  $v'$  and  $f_u$ 
26:             $\text{ARTICULATION-FLIP}(c_v^1, c_v^2, c_v)$ 
27:          else
28:             $c_u, c_v \leftarrow \text{LINKABLE}(u', v')$ 
29:             $\text{ARTICULATION-FLIP}(c_u^1, c_u^2, c_u)$ 
30:             $\text{ARTICULATION-FLIP}(c_v^1, c_v^2, c_v)$ 
```

---

flip at each of  $a_i$  and  $a_{i+1}$ , each of which are critical. Afterwards,  $a_0$  shares a face with  $a_{i+1}$ , and (if  $i \leq k - 2$ ) with at least one edge in  $B_{i+2}$ .

*Proof.* The running time of DO-ARTICULATION-FLIPS( $\dots$ ) is  $\mathcal{O}(\log^2 n)$  because it does a constant number of calls to FIND-BOUNDING-FACE( $\dots$ ), LINKABLE( $\dots$ ), and ARTICULATION-FLIP( $\dots$ ), which each take worst case  $\mathcal{O}(\log^2 n)$  time. The number and location of the articulation flips done is likewise clear from the definition, and since each of the flips move exactly one of  $u$  and  $v$ , any flips done are critical.

- If no articulation flips are done:  $u$  and  $v$  are already in the same face. (See Figure 7)
- If an articulation flip is done at  $v' = a_{i+1}$  but not at  $u' = a_i$ :  $B_{i+2}$  is flipped into a face containing  $u$  in line 9 if  $u = u'$ , or a face containing  $u'$  and  $B_i$  (and hence  $u$ ) in line 26 otherwise. (See Figure 8)

- If an articulation flip is done at  $u' = a_i$  but not at  $v' = a_{i+1}$ :  $B_i$  (and hence  $u$ ) is flipped into a face containing  $v'$  in line 15 if  $v' = v$ , or a face containing  $v'$  and  $B_{i+2}$  in line 23 otherwise. (See Figure 9)
- If articulation flips are done at both  $u' = a_i$  and  $v' = a_{i+1}$ : Both  $B_i$  (and hence  $u$ ) and  $B_{i+2}$  are flipped into the same face in lines 29–30. (See Figure 10)

In each case the postcondition is satisfied.  $\square$



Figure 7: DO-ARTICULATION-FLIPS( $a_0, a_i, a_{i+1}, a_k$ ): If  $f_u = f_v$  there is nothing to do.

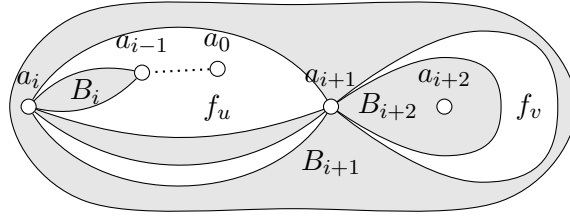


Figure 8: DO-ARTICULATION-FLIPS( $a_0, a_i, a_{i+1}, a_k$ ): When  $f_u \neq f_v$  and  $a_{i+1} \in f_u$ , we may use any corner between  $f_u$  and  $a_{i+1}$  to flip  $B_{i+2}$  into.

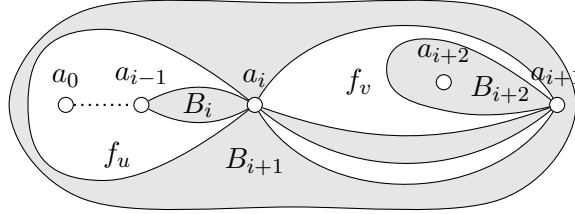


Figure 9: DO-ARTICULATION-FLIPS( $a_0, a_i, a_{i+1}, a_k$ ): When  $f_u \neq f_v$  and  $a_i \in f_v$ , we use any corner between  $f_v$  and  $a_i$  to flip  $B_i$  into.

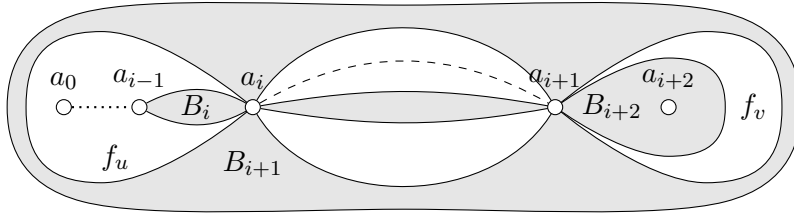


Figure 10: DO-ARTICULATION-FLIPS( $a_0, a_i, a_{i+1}, a_k$ ): When  $a_{i+1} \notin f_u$  and  $a_i \notin f_v$ , use a LINKABLE( $a_i, a_{i+1}$ ) query to find the corners where  $(a_i, a_{i+1})$  could be inserted, and flip into those corners.



**Lemma 31.** *During execution of  $\text{MULTI-FLIP-LINKABLE}(u, v)$ , the pattern of articulation flips is as follows:*

- *If  $\text{DO-ARTICULATION-FLIPS}(a_0, a_i, a_{i+1}, a_k)$  does a flip at  $a_i$ , it is potential-decreasing.*
- *If  $\text{DO-ARTICULATION-FLIPS}(a_0, a_i, a_{i+1}, a_k)$  does a flip at  $a_{i+1}$  that is not potential-decreasing, then it is potential-neutral, the following  $\text{FIND-NEXT-FLIP-BLOCK}(a_0, a_i, a_{i+1}, a_k)$  returns  $a_{i+1}$ , and in the next iteration either:*
  - *$\text{DO-SEPARATION-FLIPS}(a_{i+1}, a_{i+2})$  returns “no”; or*
  - *$\text{DO-SEPARATION-FLIPS}(a_{i+1}, a_{i+2})$  does at least one flip; or*
  - *$\text{DO-ARTICULATION-FLIPS}(a_0, a_{i+1}, a_{i+2}, a_k)$  does a potential-decreasing articulation flip at  $a_{i+1}$ .*

*Proof.* If  $\text{DO-ARTICULATION-FLIPS}(a_0, a_i, a_{i+1}, a_k)$  does a flip at  $u' = a_i$ , it is because  $u$  and  $v'$  does not yet share a face and at least one flip at  $a_i$  is needed to bring them together. By our invariants,  $u$  does share a face with both  $a_i$  and at least one edge of  $B_{i+1}$ . Observe that no amount of clean separation flips can help bringing them together, and thus any clean path in  $\text{Emb}(G)$  from  $H$  to  $H' \in \text{Emb}(G; u, v)$  contains at least one separation flip at  $a_i$ . In particular, any shortest path contains exactly one such flip, and thus this flip is potential-decreasing.

On the other hand, if  $\text{DO-ARTICULATION-FLIPS}(a_0, a_i, a_{i+1}, a_k)$  does a flip at  $u' = a_{i+1}$  because some other block  $B'$  is incident to  $a_{i+1}$  on both sides of  $u \cdots v$ , then it can happen that (after all separation flips in  $B_{i+2}$  are done)  $u$  is still not sharing a face with  $a_{i+2}$  even though  $a_{i+1}$  is (See Figure 11). In this case, a second articulation flip is needed at  $a_{i+1}$  (which will be potential-decreasing). However, if  $a_i$  shares a face with  $B'$ , the first flip at  $a_{i+1}$  could have been skipped, and we would reach the invariant state with one fewer flips. In this case, and only this case, the first flip the algorithm does at  $a_{i+1}$  is potential-neutral. Now observe that if we do such a potential-neutral flip, the following  $\text{FIND-NEXT-FLIP-BLOCK}(a_0, a_i, a_{i+1}, a_k)$  must return  $a_{i+1}$ , because either  $a_{i+1}$  does not share a face with  $a_{i+2}$  and  $\text{DO-SEPARATION-FLIPS}(a_{i+1}, a_{i+2})$  will behave as described, or  $a_{i+1}$  does share a face with  $a_{i+2}$  but  $a_0$  does not share a face with  $a_{i+2}$  so the second articulation flip at  $a_{i+1}$  is done as described.  $\square$

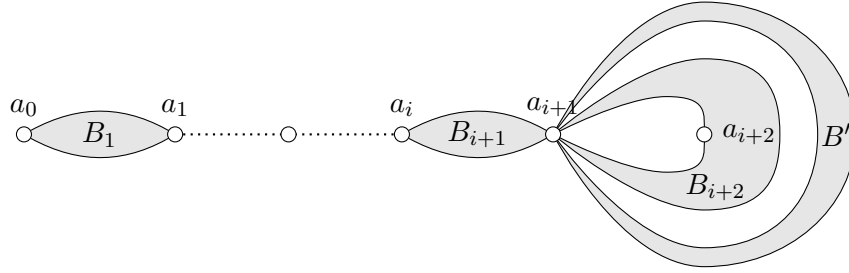


Figure 11: Example where  $\text{DO-ARTICULATION-FLIPS}$  makes a potential-neutral flip at  $a_{i+1}$ , moving  $B_{i+2}$  into the outer face. With or without this flip, we still need to flip  $B_1 \cdots B_{i+1}$  into a face containing  $a_{i+2}$ .

#### 4.4 If $u, v$ are biconnected ( $\text{DO-SEPARATION-FLIPS}$ )

The remaining piece of our algorithm, and by far the most complicated, consists of the  $\text{DO-SEPARATION-FLIPS}$  function in Algorithm 4, and its subroutines.

The general idea in  $\text{DO-SEPARATION-FLIPS}(u, v)$  is to repeatedly find and apply a separation flip  $\sigma$  (uniquely described by a tuple of 4 corners forming a 4-cycle in the vertex-face graph), such that

- The vertices incident to  $\sigma$  form a separation pair  $\{s, t\}$ , separating  $u$  from  $v$ .
- $u$  is incident to at least one of the faces  $f_u, f_v$  incident to  $\sigma$ .
- $\sigma$  partitions (the edges of)  $H$  into two subgraphs  $H_u, H_v$ , with  $u \in H_u \setminus \{s, t\}$  and  $v \in H_v \setminus \{s, t\}$ .

We call such a flip a  $u$ -flip (w.r.t.  $v$ ), and the corresponding  $H_u$  a  $u$ -flip-component (w.r.t.  $v$ ). We call a  $u$ -flip *maximal* if it maximizes the size (e.g. edges plus vertices) of  $H_u$ .

A given  $u$ -flip-component  $H_u$  remains a  $u$ -flip-component if we flip it, so we may require that each step flips a strictly larger subgraph than the previous. If no strictly larger  $H_u$  exists and  $u$  and  $v$  still do not share a face, we conclude that  $G \cup (u, v)$  is nonplanar and stop.

---

#### Algorithm 4

---

```

1: function DO-SEPARATION-FLIPS( $u, v$ )
2:    $s \leftarrow 0$ 
3:   while not LINKABLE( $u, v$ ) do  $\triangleright$  Separation flip needed in block bounded by  $u, v$ 
4:      $s', \sigma \leftarrow \text{FIND-FIRST-SEPARATION-FLIP}(u, v)$ 
5:     if  $s' \leq s$  then return “no”
6:     Execute separation flip  $\sigma$ 
7:      $s \leftarrow s'$ 
8:   return “yes”

```

---

**Lemma 32.** Assume that  $\text{FIND-FIRST-SEPARATION-FLIP}(u, v)$  runs in worst case  $\mathcal{O}(\log^2 n)$  time, and in each step:

- If  $G \cup (u, v)$  is planar it finds a maximal  $u$ -flip.
- If  $G \cup (u, v)$  is non-planar it either:
  - finds a maximal  $u$ -flip; or
  - finds a  $u$ -flip such that the next  $u$ -flip-component found has the same size; or
  - finds no  $u$ -flip.

Then  $\text{DO-SEPARATION-FLIPS}(a_i, a_{i+1})$  does only critical separation flips in worst case  $\mathcal{O}(\log^2 n)$  time per flip, and:

- If  $G \cup (u, v)$  is planar, every flip is potential-decreasing.
- If  $G \cup (u, v)$  is nonplanar every flip except the last is potential-decreasing.

*Proof.* The running time is worst case  $\mathcal{O}(\log^2 n)$  per flip, by our assumption, and because the  $\text{LINKABLE}(u, v)$  query and actually executing the flip takes  $\mathcal{O}(\log^2 n)$  in the underlying data structure from [11]. By definition, any  $u$ -flip is a critical flip, so all the flips performed are critical. By assumption, each tuple of corners  $\sigma$  that we consider (except the last) bound a maximal  $u$ -flip-component. If the flip described by  $\sigma$  is a P flip, there may be two possible choices if the node containing  $u$  on the  $u, v$ -critical path in the SPQR tree is an S node. However, either choice is potential-decreasing, as it brings the two neighbors to the involved P node that are on the critical path together. If the flip described by  $\sigma$  is not a P flip, it is potential-decreasing unless the first node  $X$  that is not included in  $H_u$  on the  $u, v$ -critical path in the SPQR tree is an  $R$  node that is *cross* (i.e. the virtual edges in  $\Gamma(X)$  corresponding to the path do not share a face). If  $X$  is cross,  $G \cup (u, v)$  is nonplanar, and the flip will be the last we execute, since no larger  $H_u$  exists after the flip.  $\square$

Let  $B = (V_B, E_B)$  be the biconnected component of  $G$  that contains  $u, v$ , and suppose  $u, v$  do not share a face in  $G$  (otherwise  $u$  and  $v$  would be linkable and  $\text{FIND-FIRST-SEPARATION-FLIP}(u, v)$  would not be called). Consider the  $u, v$ -critical path  $X_1 \cdots X_k$  with  $u \in X_1, v \in X_k$  in the SPQR tree for  $B$ . Observe that if  $k = 1$  our assumption that  $u, v$  do not share a face means that  $X_1$  is an R node, and no flip exists that can make  $u$  and  $v$  linkable. Assume therefore that  $k > 1$ . Then  $X_1$  and  $X_k$  are distinct, and by definition of  $u, v$ -critical path none of them are P nodes.

To aid in our discussion, we need to name certain subsets of the edges of  $E_B$  based on their relationship with the SPQR nodes on the  $u, v$ -critical path  $X_1 \cdots X_k$  in the SPQR tree for  $B$ .

**Definition 33.** Let  $e_u$  and  $e_v$  be the edges incident to  $u$  and  $v$  on the primal spanning tree path  $u \cdots v$ . For each  $i$  with  $1 \leq i \leq k$  define:

$$\begin{aligned} E_{<i} &:= \begin{cases} \emptyset & \text{if } i = 1 \\ \text{the separation class} & \text{otherwise} \\ \text{of } e_u \text{ w.r.t. } X_{i-1} \cap X_i & \end{cases} & E_{\geq i} &:= E_B \setminus E_{<i} \\ E_{>i} &:= \begin{cases} \emptyset & \text{if } i = k \\ \text{the separation class} & \text{otherwise} \\ \text{of } e_v \text{ w.r.t. } X_i \cap X_{i+1} & \end{cases} & E_{\leq i} &:= E_B \setminus E_{>i} \\ E_i &:= E_{\leq i} \cap E_{\geq i} & E_{\neq i} &:= E_B \setminus E_i \end{aligned}$$

Now  $E_1, \dots, E_k$  is a partition of  $E_B$ , so for each  $e \in E_B$  there is a unique *index*  $i = \text{index}(e)$  such that  $e \in E_i$ . Furthermore, for each  $1 \leq i \leq k$  the set  $E_i$  is associated with the node  $X_i$

Since  $u$  and  $v$  are biconnected, there exists two internally vertex-disjoint paths between  $u$  and  $v$ . Let  $p^s, p^t$  be an arbitrary pair of internally vertex-disjoint paths from  $u$  to  $v$ . We will use  $p^s$  and  $p^t$  to define some further concepts, and then argue (e.g. in Lemmas 34 and 35) that these definitions do not depend on the particular choice of  $p^s, p^t$ .

For  $1 \leq i \leq k-1$  let  $\{s_i, t_i\} = X_i \cap X_{i+1}$  such that  $s_i \in p^s$  and  $t_i \in p^t$ , and let  $s_0 = t_0 = u$  and  $s_k = t_k = v$ . Let  $f_u$  be any face maximizing the maximum  $i$  such that  $f_u$  contains all of  $u, s_i$ , and  $t_i$ . Note that the candidates for  $f_u$  do not depend on the specific choice of  $p^s$  and  $p^t$ , but only on the structure of the SPQR tree and the current embedding. Together,  $p^s \cup p^t$  form a simple cycle in  $G$  which we call a  *$u, v$ -critical cycle*. This cycle partitions the plane into two regions. Call the region containing  $f_u$  the  *$f_u$ -side* region and the other the *anti- $f_u$ -side* region.

For each node  $X_i$  on the  $u, v$ -critical path in the SPQR tree, any  $u, v$ -critical cycle corresponds to a unique cycle in  $\Gamma(X_i)$ , and the partition into  $f_u$ -side and anti- $f_u$ -side regions carry over into  $\Gamma(X_i)$ .

- If  $X_i$  is a P node, we say that  $X_i$  is  *$f_u$ -blocking* if the  $f_u$ -side region of  $\Gamma(X_i)$  contains any edges, and *anti- $f_u$ -blocking* if the anti- $f_u$ -side region contains any edges.
- If  $X_i$  is an R node, we say that  $X_i$  is  *$f_u$ -blocking* (resp. *anti- $f_u$ -blocking*) if the  $f_u$ -side (resp. anti- $f_u$ -side) region of  $\Gamma(X_i)$  does not contain a face incident to all of  $s_{i-1}, t_{i-1}, s_i, t_i$ . (Note that this holds even for  $i = 1$  and  $i = k$ ).
- If  $X_i$  is an S node it is neither  $f_u$ -blocking nor anti- $f_u$ -blocking.

A node that is both  $f_u$ -blocking and anti- $f_u$ -blocking is simply called *blocking*. A blocking R node is also called a *cross* node.

**Lemma 34.** *If  $G \cup (u, v)$  is planar let  $r = k$ , otherwise let  $r$  be the minimum index such that  $X_r$  is a cross node. Then  $r$  is well-defined and depends only on  $G$  and the vertices  $u, v$ .*

*Proof.* This is trivial if  $G \cup (u, v)$  is planar, so suppose not. Then  $G \cup (u, v)$  contains either a  $K_5$  subdivision containing  $(u, v)$  or a  $K_{3,3}$  subdivision containing  $(u, v)$ . In particular, the SPQR tree for  $G \cup (u, v)$  contains an  $R$  node whose skeleton graph contains such a subdivision. When deleting  $(u, v)$  from  $G \cup (u, v)$ , this  $R$  node splits into the  $u, v$ -critical path  $X_1 \cdots X_k$  in  $G$ . This path contains at least one  $R$  node containing a  $K_4$  subdivision, and since  $G \cup (u, v)$  is nonplanar, for at least one such  $R$  node  $X_i$  the vertices  $\{s_{i-1}, t_{i-1}, s_i, t_i\} = (X_{i-1} \cap X_i) \cup (X_i \cap X_{i+1})$  do not all share a face in  $\Gamma(X_i)$ .  $\square$

**Lemma 35.** *Let  $r$  be defined as in Lemma 34. If  $u$  and  $v$  do not share a face, then for  $1 \leq i \leq r$  whether  $X_i$  is (anti-)  $f_u$ -blocking depends only on the choice of  $f_u$  and the current embedding, and is independent of the particular choice of  $u, v$ -critical cycle.*

*Proof.* For any S or P node  $X_i$  with  $2 \leq i \leq k$ , the vertices  $s_i$  and  $t_i$  are completely determined by  $s_{i-1}$  and  $t_{i-1}$ , and do not depend on the particular choice of  $p^s, p^t$ . The same is true for every R node  $X_i$  with  $2 \leq i \leq r-1$ , since by definition of  $r$  these are not both  $f_u$ - and anti- $f_u$ -blocking. Thus  $s_2, \dots, s_{r-1}$  and  $t_2, \dots, t_{r-1}$  are completely determined by  $s_1, t_1$ . There are only two possible ways to map  $\{s_1, t_1\}$  to the two vertices in  $X_1 \cap X_2$ , so  $s_1, \dots, s_{r-1}$  and  $t_1, \dots, t_{r-1}$  are uniquely defined up to an exchange of every  $s$  and  $t$  with its opposite. Now consider any two  $u, v$ -critical cycles  $C_1, C_2$ , and a node  $X_i$  with  $1 \leq i \leq r$ .

- If  $X_i$  is an S node, it is by definition neither  $f_u$ - nor anti- $f_u$ -blocking, and this does not depend on the choice of critical cycle.
- If  $X_i$  is a P node, then  $1 < i < r$  and the cycle in  $\Gamma(X_i)$  does not depend on which critical cycle is used, only on which neighbors  $X_i$  has on the  $u, v$ -critical path in the SPQR tree. Thus the definition of  $f_u$ -side and anti- $f_u$ -side region, and hence  $X_i$ 's status as  $f_u$ - or anti- $f_u$ -blocking does not depend on the choice of critical cycle, but only on the choice of  $f_u$  and the current embedding.
- If  $X_i$  is an R node, and is not cross there is a unique face of  $\Gamma(X_i)$  that contains all of  $s_{i-1}, s_i, t_{i-1}, t_i$ , and this face is in the  $f_u$ -side region of  $C_1$  if and only if it is in the  $f_u$ -side region of  $C_2$ . Thus,  $X_i$  is  $f_u$ - or anti- $f_u$ -blocking with respect to  $C_1$  if and only if it is with respect to  $C_2$ .
- If  $X_i$  is a cross node, this does not depend on the choice of  $u, v$ -critical cycle but only on  $G, u, v$ . By definition  $X_i$  is both  $f_u$ - and anti- $f_u$ -blocking in both  $C_1$  and  $C_2$ .  $\square$

**Observation 36.** *Suppose  $u$  and  $v$  do not share a face, let  $f_u$  be given, and let  $r$  be defined as in Lemma 34. Let  $b \in \{1, \dots, r\}$  be the minimum index such that  $X_b$  is  $f_u$ -blocking. Then  $b$  depends only on  $u, v$ , the current embedding, and the choice of  $f_u$ , and is independent of the particular choice of  $u, v$ -critical cycle.*

Since we assume  $k > 1$ , if  $b = 1$  then by definition  $X_1$  is cross and no  $u$ -flip exists. For the remainder of this section, we will therefore assume that  $b > 1$  (and thus  $1 < b \leq r \leq k$ ).

In each step of our algorithm, to maximize the size of  $H_u$  we must find the face  $f_u$  and the separation pair  $\{s_{b-1}, t_{b-1}\}$ . In addition, we must find a second face  $f_v$ , such that  $H_u$  corresponds to the region on one side of the 4-cycle  $f_u, s_{b-1}, f_v, t_{b-1}$  in the vertex-face graph.

#### 4.4.1 CHOOSE-BEST-FLIP

The first thing we observe is that if we can somehow guess (or compute) the faces  $f_u$  and  $f_v$  bounding (a candidate to) the maximal  $u$ -flip-component, then the function  $\text{CHOOSE-BEST-FLIP}(u, v, f_u, f_v)$

from Algorithm 5 can compute the size and corners of the proposed flip. In addition, if a given pair of faces are “obviously not” bounding the maximal  $u$ -flip-component (either because no  $u$ -flip-component is bounded by these faces, or because a larger component can be easily found), this function can detect and report it.

**Definition 37.** Define a  $u$ -flip-component  $H_u$  to be *locally maximal* if, given the corners  $\sigma$ , and the separation pair  $\{s_j, t_j\}$ , and faces  $f'_u, f'_v$  incident to  $\sigma$ :

- No other  $u$ -flip-component bounded by  $f'_u, f'_v$  is larger than  $H_u$ ; and
- No other  $u$ -flip-component incident to  $s_j, t_j$  and  $f'_u$  is larger than  $H_u$ .

Any maximal  $u$ -flip-component is also locally maximal, so any pair of faces that do *not* bound a locally maximal  $u$ -flip-component can be rejected.

**Lemma 38.** *Given vertices  $u, v$  that do not share a face and a 4-tuple of corners  $\sigma$  we can in worst case  $\mathcal{O}(\log^2 n)$  time determine if  $\sigma$  bounds a locally maximal  $u$ -flip-component.*

*Proof.* Let  $f_u, f_v$  be the faces, and  $\{s_x, s_y\}$  the separation pair incident to  $\sigma$ . If  $u \notin f_u$  or  $\{u, v\} \cap \{s_x, s_y\} \neq \emptyset$ ,  $\sigma$  does not bound a  $u$ -flip-component.

Let  $H_u \ni u, H_v$  be the subgraphs of  $H$  on the two sides of  $\sigma$ . If  $v \notin H_v$  then  $\sigma$  does not bound a  $u$ -flip-component, otherwise  $H_u$  is a  $u$ -flip-component and we must determine if it is locally maximal.

Let  $C$  be the fundamental cycle in  $H$  closed by the first primal edge on the dual path  $f_u \cdots f_v$ . Now  $C \cap H_v$  is a path from  $s_x$  to  $s_y$ . If any internal node on this path is incident to both  $f_u$  and  $f_v$ , then  $H_u$  is not locally maximal.

Let  $e^*$  be the dual of the first edge on the primal tree path  $s_x \cdots s_y$ , and let  $C^*$  be the fundamental cycle closed by  $e^*$  in the dual tree. Then  $\sigma$  cuts  $C^*$  into an  $f_u \cdots f_v$  path through each of  $H_u^*$  and  $H_v^*$ . Let  $f'_v$  be the first face on the path from  $f_v$  to  $f_u$  in  $C^* \cap H_v^*$  that is incident to both  $s_x$  and  $s_y$ , and let  $\sigma'$  be any 4 corners between  $(f_v, s_x), (f_v, s_y), (f'_v, s_x), (f'_v, s_y)$ . Let  $H'_u \ni u, H'_v$  be the subgraphs of  $H$  on the two sides of  $\sigma'$ . If  $v \notin H'_v$  then  $H_u$  is not locally maximal.

Otherwise  $H_u$  is locally maximal. Each step of this test can be done in worst case  $\mathcal{O}(\log^2 n)$  time using the data structure from [11].  $\square$

**Lemma 39.** *Let  $f_u, f_v$  be faces. If  $f_u, f_v$  are incident to the corners  $\sigma$  bounding a locally maximal  $u$ -flip-component  $H_u$ , then  $\text{CHOOSE-BEST-FLIP}(u, v, f_u, f_v)$  in Algorithm 5 returns a tuple  $(s, \sigma)$  where  $s$  is the size of  $H_u$ . Otherwise  $(0, \perp)$  is returned. In either case, the total time is worst case  $\mathcal{O}(\log^2 n)$ .*

*Proof.* If  $f_u, f_v$  are valid then by definition  $u \in f_u$  and  $u \notin f_v$  (and thus  $f_u \neq f_v$ ). Let  $e$  be any primal edge on the dual tree path from  $f_u$  to  $f_v$ , and consider the fundamental cycle  $C$  closed by  $e$ . By construction,  $C$  has  $f_u$  and  $f_v$  on opposite sides, so any common vertex of  $f_u$  and  $f_v$  must be on  $C$ . In particular, we must have  $s_i, t_i \in C$ , and if no vertex in  $C$  is incident to both  $f_u$  and  $f_v$  then  $f_u$  and  $f_v$  do not bound a valid  $u$ -flip-component. We want the flip-component containing  $u$  to be as large as possible, which means the flip-component containing  $v$  must be as small as possible. Let  $p_v = \pi_C(v)$  and let  $p_x, p_y$  be the neighbors of  $p_v$  on  $C$ . Then at least one of the edges  $\{(p_x, p_v), (p_v, p_y)\}$  is in the flip-component of  $v$ . Suppose  $(x, y)$  is this edge, then the separation pair  $\{s_x, s_y\}$  we want will consist of the first vertex incident to both  $f_u$  and  $f_v$  in each direction on  $C$  away from  $(x, y)$ . If  $G$  is not biconnected, there may be multiple corners between each of  $s_x, s_y$  and each of  $f_u, f_v$ . To truly minimize the size of the flip-component of  $v$ , we must choose the nearest 4

---

**Algorithm 5**


---

```

1: function CHOOSE-BEST-FLIP( $u, v, f_u, f_v$ )
2:    $\triangleright$  Determine maximum  $u$ -flip in  $f_u, f_v$ 
3:   if  $u \in f_u$  and  $u \notin f_v$  then
4:      $e \leftarrow$  the first primal edge on the dual tree path from  $f_u$  to  $f_v$ .
5:      $C \leftarrow$  fundamental cycle of  $e$ 
6:     if  $\exists w \in C$  incident to both  $f_u$  and  $f_v$  then
7:        $p_v \leftarrow \pi_C(v)$ ;  $p_x, p_y \leftarrow$  neighbors of  $p_v$  on  $C$ 
8:       for  $(x, y) \in \{(p_x, p_v), (p_v, p_y)\}$  do
9:          $s_x, s_y \leftarrow$  vertices on  $C$  incident to  $f_u$  and  $f_v$  closest to  $(x, y)$  in each direction
10:         $c_x^u, c_y^u \leftarrow$  corners at  $s_x, s_y$  nearest to  $(x, y)$  on the  $f_u$  side
11:         $c_x^v, c_y^v \leftarrow$  corners at  $s_x, s_y$  nearest to  $(x, y)$  on the  $f_v$  side
12:         $\sigma \leftarrow (c_x^u, c_y^u, c_x^v, c_y^v)$ 
13:        if  $\sigma$  bounds a locally maximal  $u$ -flip-component then       $\triangleright \sigma$  is a valid  $u, v$ -flip
14:           $s \leftarrow$  size of  $u$ -flip-component of  $\sigma$ 
15:          return  $(s, \sigma)$ 
16:   return  $(0, \perp)$ 

```

---

corners to  $(x, y)$ . This gives us 4 corners  $(c_x^u, c_y^u, c_x^v, c_y^v)$ , and by Lemma 38 we can test if these bound a locally maximal  $u$ -flip-component and compute its size in worst case  $\mathcal{O}(\log^2 n)$  time. Note that if both candidates for  $(x, y)$  are valid, they will give the same answer, so it is ok to stop as soon as we find one that is valid. If none of the two candidates for  $(x, y)$  give a valid flip, then  $f_u, f_v$  did not bound a locally maximal  $u$ -flip-component, and we return  $(0, \perp)$ .  $\square$

#### 4.4.2 FIND-SINGLE-FLIP-CANDIDATES

Our task is thus to compute (candidates for)  $f_u$  and  $f_v$ . In [11] we considered the special case where  $G \cup (u, v)$  is planar and a single flip is needed to admit  $(u, v)$ . The function FIND-SINGLE-FLIP-CANDIDATES in Algorithm 6 is essentially the same as is described in [11], but instrumented to return a bit more information which will become important later.

**Definition 40.** A fundamental cycle  $C$  is *separating* if  $f_u$  and  $f_v$  are on opposite sides of  $C$ . If  $X_b$  is the first  $f_u$ -blocking node on  $X_1 \dots X_r$  then a fundamental cycle  $C$  is *good* when either  $C$  is separating or  $C$  intersects both  $E_{\leq b}$  and  $E_{> b}$ .

**Definition 41.** A tuple  $(f'_u, f'_v, C, e'_u, e'_v)$  is a *candidate tuple* if  $f'_u$  and  $f'_v$  are faces,  $C$  is a fundamental cycle, and  $e'_u, e'_v \in C$ . A candidate tuple is *good* if  $f'_u = f_u$ ,  $C$  is good as defined in Definition 40,  $\text{index}(e'_u) = \min_{e \in C} \text{index}(e)$ , and  $\text{index}(e'_v) = \max_{e \in C} \text{index}(e)$ . A good candidate tuple is *correct* if furthermore  $f'_v = f_v$ .

**Lemma 42.** FIND-SINGLE-FLIP-CANDIDATES( $u, v$ ) from Algorithm 6 runs in worst case  $\mathcal{O}(\log^2 n)$  time and returns a set of at most 20 candidate tuples.

*Proof.* The result set starts empty and changes only by addition of elements, and simple counting shows that either exactly 2, or at most  $2 \cdot 2 \cdot 2 + 2 \cdot 2 \cdot 2 + 2 \cdot 2 = 20$  elements are added, and by construction each element added is trivially a candidate tuple.

For each element added, we use a constant number of elementary operations each taking worst case constant time, and a constant number of the following operations:

- Find the first or last edge on a path in the primal or dual tree.
- Find the faces incident to an edge in the primal tree.
- Compute the meet of 3 faces in the dual tree.
- Determine if two faces are the same.
- Find the primal edge corresponding to an edge in the dual tree.
- Given a fundamental cycle  $C$  and a vertex  $w \in C$ , find the edges on  $C$  incident to  $w$ .
- Given a fundamental cycle  $C$  and faces  $f_1, f_2$  determine if they are on the same side of  $C$ .
- Given a fundamental cycle  $C$  and a vertex  $w \notin C$ , find the projection  $\pi_C(w) \in C$ .

Using the data structure from [11], each of these operations can be done in worst case  $\mathcal{O}(\log n)$  time.  $\square$

**Lemma 43.** *If  $G \cup (u, v)$  is planar and only one flip is needed at  $f_u, f_v$  to admit  $(u, v)$ , then at least one  $(f'_u, f'_v, \cdot, \cdot, \cdot) \in \text{FIND-SINGLE-FLIP-CANDIDATES}(u, v)$  from Algorithm 6 is correct.*

*Proof.* We consider the 4 cases for how the fundamental separating cycles may relate to  $u$  and  $v$ , and show that in each case we find at least one correct candidate tuple.

**If some fundamental separating cycle contains both  $u$  and  $v$ ,** then any such cycle  $C$  separates  $u_L$  from  $u_R$  and  $v_L$  from  $v_R$ . In particular a primal nontree edge  $e$  closes such a cycle if and only if it is on both  $u_L \cdots u_R$  and  $v_L \cdots v_R$ . But then we also have  $f_u^L \neq f_u^R$  and  $e \in f_u^L \cdots f_u^R$ . We therefore execute Line 9–16 exactly when such a separating cycle exists, and  $C$  is such a separating cycle. By definition,  $e_u, e_u^1, e_u^2 \in E_1$  and  $e_v, e_v^1, e_v^2 \in E_k$ , and the cycle  $C$  separates  $u_L$  from  $u_R$ , so there exists an  $f \in \{u_L, u_R\}$  that is on the same side of  $C$  as  $f_u$ . For this value of  $f$ , we must have  $f_u \in f_u^1 \cdots f_u^2$ ,  $f_u \in f_u^1 \cdots f_v^1$ , and  $f_u \in f_u^2 \cdots f_v^1$ , and thus  $f_u = \text{meet}(f_u^1, f_u^2, f_v^1)$  and  $(\text{meet}(f_u^1, f_u^2, f_v^1), \cdot, C, e_u, e_v)$  is a good candidate tuple. Similarly, if only one flip is needed to admit  $(u, v)$  then  $f_v$  is on the opposite side of  $C$  from  $f$ , and we must have that  $f_v \in \bar{f}_v^1 \cdots \bar{f}_v^2$ ,  $f_v \in \bar{f}_v^1 \cdots \bar{f}_u^1$ , and  $f_v \in \bar{f}_v^2 \cdots \bar{f}_u^1$ , and thus  $f_v = \text{meet}(\bar{f}_v^1, \bar{f}_v^2, \bar{f}_u^1)$  and  $(\text{meet}(f_u^1, f_u^2, f_v^1), \text{meet}(\bar{f}_v^1, \bar{f}_v^2, \bar{f}_u^1), C, e_u, e_v)$  is a correct candidate tuple.

**If no fundamental separating cycle contains both  $u$  and  $v$ , but some contains  $u$ ,** then  $f_u^L = f_u^R$  and  $f_v^L = f_v^R$  and Line 19–27 gets executed. Any such cycle must separate either  $u_L$  or  $u_R$  from the rest of  $\{u_L, u_R, v_L, v_R\}$ , and in particular it must separate  $u_L$  or  $u_R$  from  $f_u^*$ . In this case there must exist an  $f \in \{u_L, u_R\} \setminus \{f_u^*\}$  that is on the opposite side of such a cycle from  $f_u^*$ , and in fact the first primal edge on the dual tree path from  $f_u^*$  to  $f$  closes such a separating cycle  $C$ . By definition,  $e_u, e_u^1, e_u^2 \in E_1$  and there exists  $e'_v \in \{e_v^1, e_v^2\}$  with  $\text{index}(e'_v) = \max_{e \in C} \text{index}(e)$ . Exactly one face  $f'_v$  incident to  $e'_v$  is on the same side of  $C$  as  $f_u$ . Given this face  $f'_v$  and the corresponding faces  $f_u^1, f_u^2$  incident to  $e_u^1, e_u^2$  on the same side of  $C$  as  $f'_v$ , we have  $f_u \in f_u^1 \cdots f_u^2$ ,  $f_u \in f_u^1 \cdots f'_v$ , and  $f_u \in f_u^2 \cdots f'_v$ , and thus  $f_u = \text{meet}(f_u^1, f_u^2, f'_v)$  and  $(\text{meet}(f_u^1, f_u^2, f'_v), \cdot, C, e_u, e'_v)$  is a good candidate tuple. If only one flip is needed to admit  $(u, v)$  then  $f_v$  is on all of  $v_L \cdots v_R$ ,  $v_L \cdots f_u^*$ ,  $v_R \cdots f_u^*$ , and thus  $f_v = f_v^*$  and  $(\text{meet}(f_u^1, f_u^2, f'_v), f_v^*, C, e_u, e'_v)$  is a correct candidate tuple.

**If no fundamental separating cycle contains both  $u$  and  $v$ , but some contains  $v$ ,** then  $f_u^L = f_u^R$  and  $f_v^L = f_v^R$  and Line 28–36 gets executed. Any such cycle must separate either  $v_L$  or  $v_R$  from the rest of  $\{u_L, u_R, v_L, v_R\}$ , and in particular it must separate  $v_L$  or  $v_R$  from  $f_v^*$ . In this case there must exist an  $f \in \{v_L, v_R\} \setminus \{f_v^*\}$  that is on the opposite side of such a cycle from  $f_v^*$ , and in fact the first primal edge on the dual tree path from  $f_v^*$  to  $f$  closes

such a separating cycle  $C$ . By definition,  $e_v, e_v^1, e_v^2 \in E_k$  and there exists  $e'_u \in \{e_u^1, e_u^2\}$  with  $\text{index}(e'_u) = \min_{e \in C} \text{index}(e)$ . In this case,  $f_u$  is on all of  $u_L \cdots u_R$ ,  $u_L \cdots f_v^*$ ,  $u_R \cdots f_v^*$ , and thus  $f_u = f_u^*$  and  $(f_u^*, \cdot, C, e'_u, e_v)$  is a good candidate tuple. Exactly one face  $f'_u$  incident to  $e'_u$  is on the same side of  $C$  as  $f_v$ . Given this face  $f'_u$  and the corresponding faces  $f_v^1, f_v^2$  incident to  $e_v^1, e_v^2$  on the same side of  $C$  as  $f'_u$ , we have  $f_v \in f_v^1 \cdots f_v^2$ ,  $f_v \in f_v^1 \cdots f'_u$ , and  $f_v \in f_v^2 \cdots f'_u$ , and thus  $f_v = \text{meet}(f_v^1, f_v^2, f'_u)$  and  $(f_u^*, \text{meet}(f_v^1, f_v^2, f'_u), C, e'_u, e_v)$  is a correct candidate tuple.

**If no fundamental separating cycle contains  $u$  or  $v$ ,** then  $f_u^L = f_u^R$  and  $f_v^L = f_v^R$  and Line 37–44 gets executed. Assuming only one flip is needed to admit  $(u, v)$ , then  $f_u$  and  $f_v$  are simply the  $f_u^*$  and  $f_v^*$  found in the algorithm. Specifically, since  $u$  and  $v$  are not already linkable,  $f_u^* \neq f_v^*$ . Since there exists  $e'_u \in \{e_u^1, e_u^2\}$  with  $\text{index}(e'_u) = \min_{e \in C} \text{index}(e)$  and  $e'_v \in \{e_v^1, e_v^2\}$  with  $\text{index}(e'_v) = \max_{e \in C} \text{index}(e)$ , at least one of the candidates is correct.  $\square$

**Lemma 44.** *If  $X_b$  is the first  $f_u$ -blocking node on  $X_1 \dots X_r$  and  $1 < b < r$ , then at least one of the candidates  $(f'_u, \cdot, C, e'_u, e'_v) \in \text{FIND-SINGLE-FLIP-CANDIDATES}(u, v)$  from Algorithm 6 is good.*

*Proof.* Since, for every cycle  $C$ , we try all relevant edges near  $\pi_C(u)$  and  $\pi_C(v)$ , we need only argue that  $f'_u$  is  $f_u$  and  $C$  is good; if one of the tuples contains a good cycle and the correct  $f_u$ , then one of the tuples will be good.

If there exists a fundamental separating cycle through at least one of  $u, v$  then (by the same arguments as in Lemma 43) for some such cycle  $C$ ,  $(f_u, \cdot, C, e'_u, e'_v)$  is among the returned candidates. Since  $C$  is separating, it is by definition good, and thus this tuple is clearly good.

Otherwise no fundamental separating cycle contains any of  $u$  or  $v$ . Let  $f_u^* = f_u^L = f_u^R$  and  $f_v^* = f_v^L = f_v^R$ . Then, because  $1 < b < r$ , it must be the case that  $f_u^* \neq f_v^*$ . Thus, let  $e$  be the first primal edge on the dual tree path from  $f_u^*$  to  $f_v^*$ . Note that if  $f_u^* \notin f_v \cdots f_v^*$ , then that edge  $e$  closes a fundamental cycle that is separating and therefore good. Suppose therefore that  $f_u^* \in f_v \cdots f_v^*$ . Then  $e \in E_b$ . Furthermore the path  $f_v \cdots f_v^*$  cuts  $E_{\leq b}$  such that not all of  $s_{b-1}, t_{b-1}, s_b$ , and  $t_b$  are connected in  $E_{\leq b} \setminus (f_v \cdots f_v^*)$ . Thus, the fundamental cycle closed by any primal edge on  $f_v \cdots f_v^*$  must go through  $E_{>b}$ . In particular the edge  $e$  closes a cycle  $C$  intersecting  $E_{>b}$ . Thus  $C$  is good, since it intersects both  $E_b$  and  $E_{>b}$ .  $\square$



---

**Algorithm 6**


---

```

1: function FIND-SINGLE-FLIP-CANDIDATES( $u, v$ )
2:    $e_u, e_v \leftarrow$  the first and last edge on the primal tree path from  $u$  to  $v$ .
3:    $u_L, u_R \leftarrow$  left and right face incident to the first edge on the primal tree path from  $u$  to  $v$ .
4:    $v_L, v_R \leftarrow$  left and right face incident to the first edge on the primal tree path from  $v$  to  $u$ .
5:    $f_u^L \leftarrow \text{meet}(u_L, u_R, v_L)$ ;  $f_u^R \leftarrow \text{meet}(u_L, u_R, v_R)$ 
6:    $f_v^L \leftarrow \text{meet}(v_L, v_R, u_L)$ ;  $f_v^R \leftarrow \text{meet}(v_L, v_R, u_R)$ 
7:    $\text{result} \leftarrow \emptyset$ 
8:   if  $f_u^L \neq f_u^R$  then
9:      $\triangleright$  Handles case of a fundamental cycle through  $u$  and  $v$ .
10:     $e \leftarrow$  primal edge corresponding to first edge on dual tree path  $f_u^L \cdots f_u^R$ 
11:     $C \leftarrow$  fundamental cycle closed by  $e$ 
12:     $e_u^1, e_u^2, e_v^1, e_v^2 \leftarrow$  the edges incident to  $u, v$  on  $C$ 
13:    for  $f \in \{u_L, u_R\}$  do
14:       $f_u^1, f_u^2, f_v^1, f_v^2 \leftarrow$  faces incident to  $e_u^1, e_u^2, e_v^1, e_v^2$  on the same side of  $C$  as  $f$ .
15:       $\bar{f}_u^1, \bar{f}_u^2, \bar{f}_v^1, \bar{f}_v^2 \leftarrow$  faces incident to  $e_u^1, e_u^2, e_v^1, e_v^2$  on the opposite side of  $C$  from  $f$ .
16:       $\text{result} \leftarrow \text{result} \cup \{(\text{meet}(f_u^1, f_u^2, f_v^1), \text{meet}(\bar{f}_v^1, \bar{f}_v^2, \bar{f}_u^1), C, e_u, e_v)\}$ 
17:   else  $f_u^L = f_u^R$  and  $f_v^L = f_v^R$ 
18:      $f_u^* \leftarrow f_u^L$ ;  $f_v^* \leftarrow f_v^L$ 
19:      $\triangleright$  Handles case of a fundamental separating cycle through  $u$  but none through  $u, v$ .
20:     for  $f \in \{u_L, u_R\} \setminus \{f_u^*\}$  do
21:        $e \leftarrow$  first primal edge on dual tree path from  $f_u^*$  to  $f$ 
22:        $C \leftarrow$  fundamental cycle closed by  $e$ 
23:        $e_u^1, e_u^2, e_v^1, e_v^2 \leftarrow$  the edges incident to  $u, \pi_C(v)$  on  $C$ 
24:       for  $e'_v \in \{e_v^1, e_v^2\}$  do
25:         for  $f'_v \in$  faces incident to  $e'_v$  do
26:            $f_u^1, f_u^2 \leftarrow$  faces incident to  $e_u^1, e_u^2$  on the same side of  $C$  as  $f'_v$ .
27:            $\text{result} \leftarrow \text{result} \cup \{(\text{meet}(f_u^1, f_u^2, f'_v), f_v^*, C, e_u, e'_v)\}$ 
28:      $\triangleright$  Handles case of a fundamental separating cycle through  $v$  but none through  $u, v$ .
29:     for  $f \in \{v_L, v_R\} \setminus \{f_v^*\}$  do
30:        $e \leftarrow$  first primal edge on dual tree path from  $f_v^*$  to  $f$ 
31:        $C \leftarrow$  fundamental cycle closed by  $e$ 
32:        $e_u^1, e_u^2, e_v^1, e_v^2 \leftarrow$  the edges incident to  $\pi_C(u), v$  on  $C$ 
33:       for  $e'_u \in \{e_u^1, e_u^2\}$  do
34:         for  $f'_u \in$  faces incident to  $e'_u$  do
35:            $f_v^1, f_v^2 \leftarrow$  faces incident to  $e_v^1, e_v^2$  on the same side of  $C$  as  $f'_u$ .
36:            $\text{result} \leftarrow \text{result} \cup \{(f_u^*, \text{meet}(f_v^1, f_v^2, f'_u), C, e'_u, e_v)\}$ 
37:      $\triangleright$  Handles case of no fundamental separating cycle through  $u$  or  $v$ .
38:     if  $f_u^* \neq f_v^*$  then
39:        $e \leftarrow$  first primal edge on dual tree path from  $f_u^*$  to  $f_v^*$ 
40:        $C \leftarrow$  fundamental cycle closed by  $e$ 
41:        $e_u^1, e_u^2, e_v^1, e_v^2 \leftarrow$  the edges incident to  $\pi_C(u), \pi_C(v)$  on  $C$ 
42:       for  $e'_u \in \{e_u^1, e_u^2\}$  do
43:         for  $e'_v \in \{e_v^1, e_v^2\}$  do
44:            $\text{result} \leftarrow \text{result} \cup \{(f_u^*, f_v^*, C, e'_u, e'_v)\}$ 
45:   return  $\text{result}$ 

```

---

#### 4.4.3 FIND-FIRST-SEPARATION-FLIP

---

##### Algorithm 7

---

```

1: function FIND-FIRST-SEPARATION-FLIP( $u, v$ )
2:    $\triangleright$  Assumes  $u, v$  are biconnected and do not share a face.
3:   candidates  $\leftarrow$  FIND-SINGLE-FLIP-CANDIDATES( $u, v$ )
4:    $\triangleright$  Test for single flip
5:   for  $(f_u, f_v, \cdot, \cdot) \in$  candidates do
6:     if  $v \in f_v$  then
7:        $(s, \sigma) \leftarrow$  CHOOSE-BEST-FLIP( $u, v, f_u, f_v$ )
8:       if  $s > 0$  then
9:         return  $(s, \sigma)$ 
10:   $\triangleright$  Not single flip. If first flip exists, then first  $f_u$ -blocking node  $X_b$  has  $1 < b < r$ .
11:  result  $\leftarrow (0, \perp)$ 
12:  for  $(f_u, \cdot, C, e'_u, e'_v) \in$  candidates do
13:     $\triangleright$  Assume  $f_u$  is correct,  $C$  is good, and  $e'_u, e'_v \in C$  are in the first/last  $E_j$  with  $E_j \cap C \neq \emptyset$ .

14:     $\triangleright$  Handle cases where  $E_{>b} \cap C \neq \emptyset$ .
15:     $f'_v \leftarrow$  face incident to  $e'_v$  on same side of  $C$  as  $f_u$ .
16:    if  $f_u \neq f'_v$  then  $\triangleright$  If  $f_u = f'_v$  we are not in this case.
17:       $(x, y) \leftarrow$  first primal edge on dual tree path  $f_u \cdots f'_v$ .
18:      result  $\leftarrow$  max{result,
        FIND-SEP-P11( $u, v, f_u, C, e'_u, e'_v, x, y$ ),  $\triangleright C$  separating and  $P$ 
        FIND-SEP-R11( $u, v, f_u, C, e'_u, e'_v, x, y$ ),  $\triangleright C$  separating and  $R$ 
        FIND-SEP-P10( $u, v, f_u, C, e'_u, e'_v, x, y$ ),  $\triangleright C$  not separating and  $P$ 
        FIND-SEP-R10( $u, v, f_u, C, e'_u, e'_v, x, y$ )  $\triangleright C$  not separating and  $R$ 
      }

19:     $\triangleright$  Handle cases where  $E_{>b} \cap C = \emptyset$ .
20:     $e_v^1, e_v^2 \leftarrow$  the edges on  $C$  incident to  $\pi_C(v)$ 
21:     $f_v^1, f_v^2 \leftarrow$  the faces incident to  $e_v^1, e_v^2$  on the opposite side of  $C$  from  $f_u$ .
22:     $f_v^3 \leftarrow$  any face incident to  $v$  e.g.  $v_L$  or  $v_R$ 
23:     $\bar{f}_1 \leftarrow$  meet( $f_v^1, f_v^2, f_v^3$ )
24:    result  $\leftarrow$  max{result, CHOOSE-BEST-FLIP( $u, v, f_u, \bar{f}_1$ )}
25:    for  $\bar{f}_2 \in \{\bar{f}_v^1, \bar{f}_v^2\} \setminus \{\bar{f}_1\}$  do  $\triangleright$  Otherwise  $\bar{f}_2$  can not be the right face
26:       $(x, y) \leftarrow$  first primal edge on the dual tree path from  $\bar{f}_1$  to  $\bar{f}_2$ 
27:      result  $\leftarrow$  max{result,
        FIND-SEP-P0X( $u, v, f_u, C, e'_u, e'_v, x, y$ ),  $\triangleright P$ 
        FIND-SEP-R01( $u, v, f_u, C, e'_u, e'_v, x, y$ )  $\triangleright C$  separating and  $R$ 
      }  $\triangleright$  In this case it is impossible to have  $C$  not separating and  $R$ 

28:  return result

```

---

**Lemma 45.** Consider FIND-FIRST-SEPARATION-FLIP in Algorithm 7. Suppose  $(f_u, \cdot, C, e'_u, e'_v)$  is good in Line 13, and  $E_{>b} \cap C \neq \emptyset$  and  $C$  is separating. Then in Line 16,  $f_u \neq f'_v$  and the first primal

edge  $(x, y)$  on the dual tree path from  $f_u$  to  $f'_v$  is in  $E_b$  and closes a good fundamental cycle  $C'$  such that  $C' \setminus C$  is a path  $\pi_C(x) \cdots \pi_C(y)$  in  $E_b$ . Furthermore, if  $X_b$  is a P node,  $\{\pi_C(x), \pi_C(y)\} = \{s_{b-1}, t_{b-1}\}$ .

*Proof.* Since  $(f_u, \cdot, C, e'_u, e'_v)$  is good and  $E_{>b} \cap C \neq \emptyset$  we specifically have  $e'_v \in E_{>b}$ . Since the first  $f_u$ -blocking node on  $X_1 \cdots X_k$  has index  $b < \text{index}(e'_v)$ , that means  $f_u \neq f'_v$ . Let  $(x, y)$  be the first primal edge on the dual tree path from  $f_u$  to  $f'_v$ . Since  $C$  is a fundamental cycle, the path  $f_u \cdots f'_v$  will stay on one side of  $C$ , thus  $(x, y) \in E_b$  and

- if  $X_b$  is a P node the dual tree path  $f_u \cdots f'_v$  cuts every separation class w.r.t.  $\{s_{b-1}, t_{b-1}\}$  on the  $f_u$  side of  $C$ , separating  $s_{b-1} = s_b$  from  $t_{b-1} = t_b$ . Thus,  $(x, y)$  is in a different separation class w.r.t.  $\{s_{b-1}, t_{b-1}\}$  from any edge on  $C$  and we must have  $\{s_{b-1}, t_{b-1}\} = \{\pi_C(x), \pi_C(y)\}$ . The cycle  $C'$  closed by  $(x, y)$  is either separating (and contains  $e'_u \in E_{\leq b}$ ), or contains  $e'_v \in E_{>b}$ .
- if  $X_b$  is an R node the dual tree path  $f_u \cdots f'_v$  cuts  $E_b$ , separating  $s_{b-1}$  and  $s_b$  from  $t_{b-1}$  and  $t_b$ . Thus, the fundamental cycle  $C'$  closed by  $(x, y)$  must connect  $x$  to  $y$  via primal tree paths  $x \cdots \pi_C(x) \subseteq E_b$  and  $y \cdots \pi_C(y) \subseteq E_b$ , and the primal tree path  $\pi_C(x) \cdots \pi_C(y)$  which (assuming  $X_b$  is not a cross node) must go via either  $s_{b-1} \cdots t_{b-1} \subseteq E_{<b}$  (thus  $C'$  is separating and contains  $e'_u$ ) or  $s_b \cdots t_b \subseteq E_{>b}$  (and contains  $e'_v$ ).

In either case,  $C'$  is good.  $\square$

**Lemma 46.** Consider FIND-FIRST-SEPARATION-FLIP in Algorithm 7. Suppose  $(f_u, \cdot, C, e'_u, e'_v)$  is good in Line 13, and  $E_{>b} \cap C \neq \emptyset$  and  $C$  is not separating. Then in Line 16,  $f_u \neq f'_v$  and the first primal edge  $(x, y)$  on the dual tree path from  $f_u$  to  $f'_v$  is in  $E_{\leq b}$  and closes a good fundamental cycle  $C'$  intersecting  $E_{<b}$ .

*Proof.* Since  $f_u$  and  $f'_v$  are on the same side of  $C$ , by construction, the dual tree path  $f_u \cdots f'_v$  must contain  $f_v$ . But then, the first edge  $(x, y)$  closes a separating fundamental cycle. If  $(x, y)$  is in  $E_{<b}$  we are done, so suppose  $(x, y) \in E_b$  which is the only remaining case. Then, either  $x$  or  $y$  is separated from  $C$  in  $E_b \setminus (f_u \cdots f'_v)$ . But then, the primal tree path  $x \cdots y$  must go via  $s_{b-1} \cdots t_{b-1} \subseteq E_{<b}$ .  $\square$

**Lemma 47.** Let  $C$  be a good cycle that is not separating, let  $e_1, e_2$  be the edges incident to  $\pi_C(u)$  on  $C$ , and let  $f_v^1, f_v^2$  be the faces incident to  $e_1, e_2$  on the same side of  $C$  as  $f_u$ , then  $f_v^1 \neq f_v^2$  and  $f_v \in f_v^1 \cdots f_v^2$ .

*Proof.* Since  $C$  is good and not separating, then by definition  $C$  intersects both  $E_{\leq b}$  and  $E_{>b}$  but not  $E_{<b}$ , and in particular  $E_b \cap C$  consists of a path  $s_b \cdots t_b$ . Since  $u, v$  are assumed to be biconnected,  $f_v^1$  and  $f_v^2$  are distinct. Furthermore let  $s'$  be the first vertex on the primal tree path from  $\pi_C(u)$  to  $u$  such that the remaining path  $s' \cdots u \subseteq E_{<b}$ , then  $s' \in \{s_{b-1}, t_{b-1}\}$ .

Since  $s_b$  and  $t_b$  are distinct, at least one of them is different from  $\pi_C(u)$ , and thus at least one of  $f_v^1$  and  $f_v^2$  is incident to an edge on a path in  $E_b \cap (C \cup T)$  from  $s_b$  or  $t_b$  to  $s'$ . Suppose without loss of generality that  $t_b \neq \pi_C(u)$  and that  $f_v^2$  is incident to the first edge on  $\pi_C(u) \cdots t_b$ . Since  $t_b \in f_v$  and  $s' \in f_v$ , the path  $s' \cdots t_b$  in  $C \cup T$  together with (an imaginary edge  $(t_b, s')$  through)  $f_v$  form a closed curve separating  $f_v^1$  from  $f_v^2$ . Since  $s' \cdots t_b$  is in  $C \cup T$ , the path  $f_v^1 \cdots f_v^2$  does not cross it, and thus it must contain  $f_v$ .  $\square$

**Lemma 48.** Let  $C$  be a good cycle with  $E_{>b} \cap C = \emptyset$ , let  $e_1, e_2$  be the edges incident to  $\pi_C(v)$  on  $C$ , and let  $f_v^1, f_v^2$  be the faces incident to  $e_1, e_2$  on the opposite side of  $C$  from  $f_u$ , then  $f_v^1 \neq f_v^2$  and  $f_v \in f_v^1 \cdots f_v^2$ .

*Proof.* Since  $C$  is good and does not intersect  $E_{>b}$ , we must have that  $C$  is separating and  $E_b \cap C$  contains a path  $s_{b-1} \cdots t_{b-1}$ . Since  $u, v$  are assumed to be biconnected  $\bar{f}_v^1$  and  $\bar{f}_v^2$  are distinct. Furthermore let  $s'$  be the first vertex on the primal tree path from  $\pi_C(v)$  to  $v$  such that the remaining path  $s' \cdots v \subseteq E_{>b}$ , then  $s' \in \{s_b, t_b\}$ .

Since  $s_{b-1}$  and  $t_{b-1}$  are distinct, at least one of them is different from  $\pi_C(v)$ , and thus at least one of  $\bar{f}_v^1$  and  $\bar{f}_v^2$  is incident to an edge on a path in  $E_b \cap (C \cup T)$  from  $s_{b-1}$  or  $t_{b-1}$  to  $s'$ . Suppose without loss of generality that  $t_{b-1} \neq \pi_C(v)$  and that  $\bar{f}_v^2$  is incident to the first edge on  $\pi_C(v) \cdots t_{b-1}$ . Since  $t_{b-1} \in f_v$  and  $s' \in f_v$ , the path  $s' \cdots t_{b-1}$  in  $C \cup T$  together with (an imaginary edge  $(t_{b-1}, s')$  through)  $f_v$  forms a closed curve separating  $\bar{f}_v^1$  from  $\bar{f}_v^2$ . Since  $s' \cdots t_{b-1}$  is in  $C \cup T$ , the path  $\bar{f}_v^1 \cdots \bar{f}_v^2$  does not cross it and so must contain  $f_v$ .  $\square$

**Lemma 49.** *Consider FIND-FIRST-SEPARATION-FLIP in Algorithm 7. If  $(f_u, \cdot, C, e'_u, e'_v)$  is good in Line 13, and  $E_{>b} \cap C = \emptyset$ . Then in Line 23 if  $\bar{f}_1 \neq f_v$  then for at least one  $\bar{f}_2 \in \{\bar{f}_v^1, \bar{f}_v^2\} \setminus \{\bar{f}_1\}$ , the first primal edge  $(x, y)$  on the dual tree path from  $\bar{f}_1$  to  $\bar{f}_2$  closes a good fundamental cycle  $C'$  intersecting  $E_{>b}$ .*

*Proof.* By Lemma 48  $\bar{f}_v^1 \neq \bar{f}_v^2$  and  $f_v \in \bar{f}_v^1 \cdots \bar{f}_v^2$ . Furthermore, assuming  $v \notin f_v$  also  $f_v$  and  $\bar{f}_v^3$  are distinct. If  $\bar{f}_1 = f_v$  we are done, so suppose  $\bar{f}_1 \neq f_v$ . Then either  $f_v \notin \bar{f}_v^1 \cdots \bar{f}_1$  or  $f_v \notin \bar{f}_v^2 \cdots \bar{f}_1$ . Choose  $j \in \{0, 1\}$  such that  $f_v \notin \bar{f}_v^{1+j} \cdots \bar{f}_1$ , then the first primal edge  $(x, y)$  on the dual tree path from  $\bar{f}_1$  to  $\bar{f}_v^{2-j}$  closes a cycle  $C'$  separating  $\bar{f}_v^{1+j} \cdots \bar{f}_1 \cdots \bar{f}_v^3$  from  $f_v \cdots \bar{f}_v^{2-j}$ .

If  $(x, y) \in E_{\leq b}$  then the path  $\bar{f}_v^3 \cdots \bar{f}_1 \cdots f_v$  cuts  $E_{\leq b}$ , separating  $s_b$  from  $t_b$ . But then the primal tree path from  $s_b$  to  $t_b$  is contained in  $E_{>b}$ , and is part of  $C'$ . Thus  $C'$  intersects both  $E_{\leq b}$  and  $E_{>b}$  and is therefore good.

Otherwise  $(x, y) \in E_{>b}$ , and  $\bar{f}_v^1 \cdots \bar{f}_v^2$  cuts  $E_{>b}$ , separating  $s_b$  from  $t_b$ . But then the primal tree path from  $s_b$  to  $t_b$  is contained in  $E_{\leq b}$ , and is part of  $C'$ . Thus  $C'$  intersects both  $E_{\leq b}$  and  $E_{>b}$  and is therefore good.  $\square$

---

### Algorithm 8

---

```

1: function FIND-SEP-P11( $u, v, f_u, C, e'_u, e'_v, x, y$ )
2:    $\triangleright$  Handle cases where  $X_b$  is a P node and  $E_{>b} \cap C \neq \emptyset$  and  $C$  is separating.
3:    $\bar{f}_u, \bar{f}_v \leftarrow$  face incident to  $e'_u, e'_v$  on opposite side of  $C$  from  $f_u$ .
4:    $p_x \leftarrow \pi_C(x); p_y \leftarrow \pi_C(y)$ 
5:   if  $p_x \neq p_y$  then  $\triangleright$  If  $p_x = p_y$  we are not in this case.
6:      $f_v \leftarrow$  first face on  $\bar{f}_v \cdots \bar{f}_u$  incident to both  $p_x$  and  $p_y$ 
7:     return CHOOSE-BEST-FLIP( $u, v, f_u, f_v$ )
8:   return  $(0, \perp)$ 

```

---

**Lemma 50.** *If  $(f_u, \cdot, C, e'_u, e'_v)$  is good,  $E_{>b} \cap C \neq \emptyset$ ,  $C$  is separating,  $(x, y)$  closes a fundamental cycle  $C'$  such that  $C' \setminus C$  is a path  $\pi_C(x) \cdots \pi_C(y)$  in  $E_b$  with  $\{\pi_C(x), \pi_C(y)\} = \{s_{b-1}, t_{b-1}\}$ , and  $X_b$  is a P node, then FIND-SEP-P11( $u, v, f_u, C, e'_u, e'_v, x, y$ ) in Algorithm 8 returns the size and corners of a maximal  $u$ -flip.*

*Proof.* Since  $C$  is separating,  $f_v$  is on the opposite side of  $C$  from  $f_u$ , and thus by construction  $\bar{f}_u$  and  $\bar{f}_v$  are on the same side of  $C$  as  $f_v$ . The path  $s_{b-1} \cdots t_{b-1}$  together with (an imaginary

edge  $(s_{b-1}, t_{b-1})$  through)  $f_v$  forms a closed curve separating  $\bar{f}_u$  from  $\bar{f}_v$ , so  $f_v \in \bar{f}_u \cdots \bar{f}_v$ . Since  $f_v$  maximizes the size of the  $u$ -flip-component, it must be the first face on this path that defines a valid flip, which it does if and only if it contains  $s_{b-1}, t_{b-1}$ . Given the correct  $f_v$ , the rest follows from Lemma 39.  $\square$

---

**Algorithm 9**


---

```

1: function FIND-SEP-P10( $u, v, f_u, C, e'_u, e'_v, x, y$ )
2:    $\triangleright$  Handle cases where  $X_b$  is a  $P$  node and  $E_{>b} \cap C \neq \emptyset$  and  $C$  is not separating.
3:    $f'_v \leftarrow$  the face incident to  $e'_v$  on the same side of  $C$  as  $f_u$ .
4:    $p_x \leftarrow \pi_C(x); p_y \leftarrow \pi_C(y)$ 
5:   if  $p_x \neq p_y$  then  $\triangleright$  If  $p_x = p_y$  we are not in this case.
6:      $f_v \leftarrow$  first face on  $f'_v \cdots f_u$  incident to both  $p_x$  and  $p_y$ 
7:     return CHOOSE-BEST-FLIP( $u, v, f_u, f_v$ )
8:   return  $(0, \perp)$ 

```

---

**Lemma 51.** *If  $(f_u, \cdot, C, e'_u, e'_v)$  is good,  $E_{>b} \cap C \neq \emptyset$ ,  $C$  is not separating,  $(x, y)$  closes a good fundamental cycle intersecting  $E_{<b}$ , and  $X_b$  is a  $P$  node, then FIND-SEP-P10( $u, v, f_u, C, e'_u, e'_v, x, y$ ) in Algorithm 9 returns the size and corners of a maximal  $u$ -flip.*

*Proof.* Since  $C$  is good but not separating,  $C \cap E_{<b} = \emptyset$ ,  $C \cap E_b \neq \emptyset$ , and  $C \cap E_{>b} \neq \emptyset$ . Thus,  $C$  contains  $s_{b-1}, t_{b-1}$  and since  $C$  is a fundamental cycle  $C$  contains the tree path  $s_{b-1} \cdots t_{b-1}$ . Since  $C'$  is a good fundamental cycle through  $E_{<b}$  it must also contain that tree path, and we must have  $(x, y) \in E_{<b}$  and thus  $\{\pi_C(x), \pi_C(y)\} = \{s_{b-1}, t_{b-1}\}$ .

Furthermore, since  $C$  is not separating  $f_v$  is on the same side of  $C$  as  $f_u$ , and thus by construction  $f'_v$  is on the same side of  $C$  as  $f_v$ . The path  $s_{b-1} \cdots t_{b-1}$  together with (an imaginary edge  $(s_{b-1}, t_{b-1})$  through)  $f_v$  forms a closed curve separating  $f_u$  from  $f'_v$ , so  $f_v \in f_u \cdots f'_v$ . Since  $f_v$  maximizes the size of the  $u$ -flip-component, it must be the first face on this path that defines a valid flip, which it does if and only if it contains  $s_{b-1}, t_{b-1}$ . Given the correct  $f_v$ , the rest follows from Lemma 39.  $\square$

**Lemma 52.** *Suppose  $X_b$  is an  $R$  node. Let  $C$  and  $C'$  be good fundamental cycles such that  $C \cup C'$  consists of 3 internally vertex-disjoint paths  $P_{<}, P_{=}, P_{>}$  between a pair of distinct vertices  $p_x, p_y$ , where  $P_{<} \cap E_{<b} \neq \emptyset$ ,  $P_{=} \subseteq E_b$ , and  $P_{>} \cap E_{>b} \neq \emptyset$ . Let  $C'' = P_{<} \cup P_{>}$ , let  $e \in C'' \setminus E_b$ , and let  $e_x^1, e_x^2, e_y^1, e_y^2$  be the edges incident to  $p_x, p_y$  on  $C''$ . Let  $f, f_x^1, f_x^2, f_y^1, f_y^2$  be the faces incident to  $e, e_x^1, e_x^2, e_y^1, e_y^2$  on the same side of  $C''$  as  $f_v$ . Then there exists  $f_x \in \{f_x^1, f_x^2\}$  and  $f_y \in \{f_y^1, f_y^2\}$  such that  $f_v = \text{meet}(f, f_x, f_y)$ .*

*Proof.* Suppose that  $e \in E_{<b}$  (the case  $e \in E_{>b}$  is symmetric). Since  $X_b$  is an  $R$  node, at least one of  $s_{b-1} \neq s_b$  and  $t_{b-1} \neq t_b$  holds. We may assume without loss of generality that  $s_{b-1} \neq s_b$  and that  $p_x$  is on the  $s_{b-1} \cdots s_b$  path in  $C'' \cap E_b$ . Then there exists at least one  $e_x \in \{e_x^1, e_x^2\} \cap E_b$ , and at least one  $e_y \in \{e_y^1, e_y^2\} \cap E_{\geq b}$ . Let  $f_x \in \{f_x^1, f_x^2\}, f_y \in \{f_y^1, f_y^2\}$  be the corresponding faces.

Now  $f_v \in f \cdots f_x$  and  $f_v \in f \cdots f_y$  because  $C'' \cap E_{<b}$  consists of a path  $s_{b-1} \cdots t_{b-1}$  which together with (an imaginary edge  $(s_{b-1}, t_{b-1})$  through)  $f_v$  forms a closed curve separating  $f$  from  $f_x$  and  $f_y$ .

Similarly  $f_v \in f_x \cdots f_y$  because the path  $s_{b-1} \cdots s_b$  in  $C \cap E_b$  together with (an imaginary edge  $(s_{b-1}, s_b)$  through)  $f_v$  forms a closed curve separating  $f_x$  from  $f_y$ .

---

**Algorithm 10**


---

```

1: function FIND-SEP-R11( $u, v, f_u, C, e'_u, e'_v, x, y$ )
2:    $\triangleright$  Handle cases where  $X_b$  is an  $R$  node and  $E_{>b} \cap C \neq \emptyset$  and  $C$  is separating.
3:    $\text{result} \leftarrow (0, \perp)$ 
4:    $\bar{f}_u, \bar{f}_v \leftarrow$  face incident to  $e'_u, e'_v$  on opposite side of  $C$  from  $f_u$ .
5:    $p_x \leftarrow \pi_C(x); p_y \leftarrow \pi_C(y)$ 
6:   if  $p_x \neq p_y$  then  $\triangleright$  If  $p_x = p_y$  we are not in this case.
7:      $e_x^1, e_x^2, e_y^1, e_y^2 \leftarrow$  edges on  $C$  incident to  $p_x, p_y$ , with  $e_x^1, e_y^1$  closest to  $e'_u$ 
8:      $f_x^1, f_x^2, f_y^1, f_y^2 \leftarrow$  faces incident to  $e_x^1, e_x^2, e_y^1, e_y^2$  on the same side of  $C$  as  $\bar{f}_u$ 
9:     for  $\bar{f}_x \in \{f_x^1, f_x^2\}$  do
10:      for  $\bar{f}_y \in \{f_y^1, f_y^2\}$  do
11:         $f_v \leftarrow \text{meet}(\bar{f}_u, \bar{f}_x, \bar{f}_y)$ 
12:         $\text{result} \leftarrow \max\{\text{result}, \text{CHOOSE-BEST-FLIP}(u, v, f_u, f_v)\}$ 
13:   return result

```

---

Since  $f_v$  is on all 3 paths  $f \cdots f_x$ ,  $f \cdots f_y$ , and  $f_x \cdots f_y$ , we have  $f_v = \text{meet}(f, f_x, f_y)$ .  $\square$

**Lemma 53.** Suppose  $X_b$  is an  $R$  node. Let  $C_{\leq} \subseteq E_{\leq b}$  and  $C_{\geq} \subseteq E_{\geq b}$  be good fundamental cycles with at most one vertex in common. Let  $e \in (C_{\leq} \cup C_{\geq}) \setminus E_b$ , and let  $e_x^1, e_x^2, e_y^1, e_y^2$  be the edges incident to  $p_x = \pi_{C_{\leq}}(v), p_y = \pi_{C_{\geq}}(u)$  on  $C_{\leq}, C_{\geq}$ . Let  $f, f_x^1, f_x^2, f_y^1, f_y^2$  be the faces incident to  $e, e_x^1, e_x^2, e_y^1, e_y^2$  on the same side of  $C_{\leq}, C_{\geq}$  as  $f_v$ . Then there exists  $f_x \in \{f_x^1, f_x^2\}$  and  $f_y \in \{f_y^1, f_y^2\}$  such that  $f_v = \text{meet}(f, f_x, f_y)$ .

*Proof.* Suppose that  $e \in E_{<b}$  (the case  $e \in E_{>b}$  is symmetric). Then there exists at least one  $e_x \in \{e_x^1, e_x^2\} \cap E_b$ . Let  $f_x \in \{f_x^1, f_x^2\}$  be the corresponding face. By Lemma 47 or 48  $f_y^1 \neq f_y^2$  and  $f_v \in f_y^1 \cdots f_y^2$ , so there exists  $f_y \in \{f_y^1, f_y^2\}$  such that  $f_v \in f_x \cdots f_y$ .

Now  $f_v \in f \cdots f_x$  and  $f_v \in f \cdots f_y$  because  $C_{\leq} \cap E_{<b}$  consists of a path  $s_{b-1} \cdots t_{b-1}$  which together with (an imaginary edge  $(s_{b-1}, t_{b-1})$  through)  $f_v$  forms a closed curve separating  $f$  from  $f_x$  and  $f_y$ .

Since  $f_v$  is on all 3 paths  $f \cdots f_x$ ,  $f \cdots f_y$ , and  $f_x \cdots f_y$ , we have  $f_v = \text{meet}(f, f_x, f_y)$ .  $\square$

**Lemma 54.** If  $(f_u, \cdot, C, e'_u, e'_v)$  is good,  $E_{>b} \cap C \neq \emptyset$ ,  $C$  is separating,  $(x, y)$  closes a fundamental cycle  $C'$  such that  $C' \setminus C$  is a path  $\pi_C(x) \cdots \pi_C(y)$  in  $E_b$ , and  $X_b$  is an  $R$  node, then FIND-SEP-R11( $u, v, f_u, C, e'_u, e'_v, x, y$ ) in Algorithm 10 returns the size and corners of a maximal  $u$ -flip.

*Proof.* By Lemma 39 it is sufficient to show that at least one of the candidates to  $f_v$  used in the calls to CHOOSE-BEST-FLIP( $u, v, f_u, f_v$ ) is correct. This follows directly from Lemma 52 with  $e = e'_u$  and the fact that  $\bar{f}_u$  is on the same side of  $C$  as  $f_v$ .  $\square$

**Lemma 55.** If  $(f_u, \cdot, C, e'_u, e'_v)$  is good,  $E_{>b} \cap C \neq \emptyset$ ,  $C$  is not separating,  $(x, y)$  closes a good fundamental cycle intersecting  $E_{<b}$ , and  $X_b$  is an  $R$  node, then FIND-SEP-R10( $u, v, f_u, C, e'_u, e'_v, x, y$ ) in Algorithm 11 returns the size and corners of a maximal  $u$ -flip.

*Proof.* By Lemma 39 it is sufficient to show that at least one of the candidates to  $f_v$  used in the calls to CHOOSE-BEST-FLIP( $u, v, f_u, f_v$ ) is correct.

---

**Algorithm 11**


---

```

1: function FIND-SEP-R10( $u, v, f_u, C, e'_u, e'_v, x, y$ )
2:    $\triangleright$  Handle cases where  $X_b$  is an  $R$  node and  $E_{>b} \cap C \neq \emptyset$  and  $C$  is not separating.
3:    $\text{result} \leftarrow (0, \perp)$ 
4:    $f'_v \leftarrow$  the face incident to  $e'_v$  on the same side of  $C$  as  $f_u$ .
5:    $C' \leftarrow$  fundamental cycle of  $(x, y)$ 
6:    $p_x \leftarrow \pi_C(x); p_y \leftarrow \pi_C(y)$ 
7:   if  $p_x \neq p_y$  then  $\triangleright$  The two cycles have common edges (case I)
8:      $e_x^1, e_y^1 \leftarrow$  edges on  $C' \setminus C$  incident to  $p_x, p_y$ 
9:     if  $e'_u \in C'$  then
10:       $e_x^2, e_y^2 \leftarrow$  edges on  $C \setminus C'$  incident to  $p_x, p_y$ 
11:    else  $e'_u \notin C'$ 
12:       $e_x^2, e_y^2 \leftarrow$  edges on  $C \cap C'$  incident to  $p_x, p_y$ 
13:     $f_x^1, f_x^2, f_y^1, f_y^2 \leftarrow$  faces incident to  $e_x^1, e_x^2, e_y^1, e_y^2$  on the same side of  $C, C'$  as  $f'_v$ 
14:    for  $f_x \in \{f_x^1, f_x^2\}$  do
15:      for  $f_y \in \{f_y^1, f_y^2\}$  do
16:         $f_v \leftarrow \text{meet}(f'_v, f_x, f_y)$ 
17:         $\text{result} \leftarrow \max\{\text{result}, \text{CHOOSE-BEST-FLIP}(u, v, f_u, f_v)\}$ 
18:  else  $p_x = p_y$   $\triangleright$  The two cycles have no common edges (case X or H)
19:     $e_u^1, e_u^2 \leftarrow$  edges incident to  $\pi_{C'}(v)$  on  $C'$ 
20:     $e_v^1, e_v^2 \leftarrow$  edges incident to  $\pi_C(u)$  on  $C$ 
21:     $f_u^1, f_u^2, f_v^1, f_v^2 \leftarrow$  faces incident to  $e_u^1, e_u^2, e_v^1, e_v^2$  on the same side of  $C, C'$  as  $f'_v$ 
22:    for  $f_x \in \{f_u^1, f_u^2\}$  do
23:      for  $f_y \in \{f_v^1, f_v^2\}$  do
24:         $f_v \leftarrow \text{meet}(f'_v, f_x, f_y)$ 
25:         $\text{result} \leftarrow \max\{\text{result}, \text{CHOOSE-BEST-FLIP}(u, v, f_u, f_v)\}$ 
26:  return  $\text{result}$ 

```

---

If  $p_x \neq p_y$  then  $C \cup C'$  consists of 3 internally vertex-disjoint paths from  $p_x$  to  $p_y$ , and by Lemma 52 with  $e = e'_v$  there exists  $f_x \in \{f_x^1, f_x^2\}$  and  $f_y \in \{f_y^1, f_y^2\}$  such that  $f_v = \text{meet}(f'_v, f_x, f_y)$ .

Otherwise  $p_x = p_y$ , and  $C$  and  $C'$  has at most one vertex in common, so by Lemma 53 with  $e = e'_v$  there exists  $f_x \in \{f_x^1, f_x^2\}$  and  $f_y \in \{f_y^1, f_y^2\}$  such that  $f_v = \text{meet}(f'_v, f_x, f_y)$ .  $\square$

**Lemma 56.** *If  $(f_u, \cdot, C, e'_u, e'_v)$  is good,  $E_{>b} \cap C = \emptyset$ ,  $(x, y)$  closes a good fundamental cycle intersecting  $E_{>b}$ , and  $X_b$  is a  $P$  node, then  $\text{FIND-SEP-P0X}(u, v, f_u, C, e'_u, e'_v, x, y)$  in Algorithm 12 returns the size and corners of a maximal  $u$ -flip.*

*Proof.* Since  $C$  is good and  $E_{>b} \cap C = \emptyset$ ,  $C$  is separating and contains  $s_{b-1}, t_{b-1}$  as well as the tree path  $s_{b-1} \cdots t_{b-1}$ . Since  $C'$  is a good fundamental cycle through  $E_{>b}$  it must also contain that tree path, and we must have  $(x, y) \in E_{>b}$  and thus  $\{\pi_C(x), \pi_C(y)\} = \{s_{b-1}, t_{b-1}\}$ .

Furthermore, since  $C$  is separating  $f_v$  is on the opposite side of  $C$  from  $f_u$ , and thus by construction  $\bar{f}_u$  and  $\bar{f}$  are on the same side of  $C$  as  $f_v$ . The path  $s_{b-1} \cdots t_{b-1}$  together with (an imaginary edge  $(s_{b-1}, t_{b-1})$  through)  $f_v$  forms a closed curve separating  $\bar{f}$  from  $\bar{f}_u$ , so  $f_v \in \bar{f} \cdots \bar{f}_u$ . Since  $f_v$  maximizes the size of the  $u$ -flip-component, it must be the first face on this path that

---

**Algorithm 12**

---

```
1: function FIND-SEP-P0X( $u, v, f_u, C, e'_u, e'_v, x, y$ )
2:    $\triangleright$  Handle cases where  $X_b$  is a  $P$  node and  $E_{>b} \cap C = \emptyset$ .
3:    $\bar{f}_u, \bar{f}_v \leftarrow$  face incident to  $e'_u, e'_v$  on opposite side of  $C$  from  $f_u$ .
4:    $C' \leftarrow$  fundamental cycle of  $(x, y)$ 
5:    $p_x \leftarrow \pi_C(x); p_y \leftarrow \pi_C(y)$ 
6:   if  $p_x \neq p_y$  then  $\triangleright$  If  $p_x = p_y$  we are not in this case.
7:      $\bar{f} \leftarrow$  the face incident to  $(x, y)$  on the same side of  $C'$  as  $\bar{f}_u$ 
8:      $f_v \leftarrow$  first face on  $\bar{f} \cdots \bar{f}_u$  incident to both  $p_x$  and  $p_y$ 
9:     return CHOOSE-BEST-FLIP( $u, v, f_u, f_v$ )
10:  return  $(0, \perp)$ 
```

---

defines a valid flip, which it does if and only if it contains  $s_{b-1}, t_{b-1}$ . Given the correct  $f_v$ , the rest follows from Lemma 39.  $\square$

---

**Algorithm 13**

---

```
1: function FIND-SEP-R01( $u, v, f_u, C, e'_u, e'_v, x, y$ )
2:    $\triangleright$  Handle cases where  $X_b$  is an  $R$  node and  $E_{>b} \cap C = \emptyset$  and  $C$  is separating.
3:   result  $\leftarrow (0, \perp)$ 
4:    $\bar{f}_u, \bar{f}_v \leftarrow$  face incident to  $e'_u, e'_v$  on opposite side of  $C$  from  $f_u$ .
5:    $C' \leftarrow$  fundamental cycle of  $(x, y)$ 
6:    $p_x \leftarrow \pi_C(x); p_y \leftarrow \pi_C(y)$ 
7:   if  $p_x \neq p_y$  then  $\triangleright$  The two cycles have common edges (case I)
8:     if  $e'_v \in C'$  then
9:        $e_x^1, e_y^1 \leftarrow$  edges on  $C \setminus C'$  incident to  $p_x, p_y$ 
10:    else  $e'_v \notin C'$ 
11:       $e_x^1, e_y^1 \leftarrow$  edges on  $C \cap C'$  incident to  $p_x, p_y$ 
12:       $e_x^2, e_y^2 \leftarrow$  edges on  $C' \setminus C$  incident to  $p_x, p_y$ 
13:       $\bar{f}_x^1, \bar{f}_x^2, \bar{f}_y^1, \bar{f}_y^2 \leftarrow$  faces incident to  $e_x^1, e_x^2, e_y^1, e_y^2$  on the same side of  $C, C'$  as  $\bar{f}_u$ 
14:      for  $\bar{f}_x \in \{\bar{f}_x^1, \bar{f}_x^2\}$  do
15:        for  $\bar{f}_y \in \{\bar{f}_y^1, \bar{f}_y^2\}$  do
16:           $f_v \leftarrow \text{meet}(\bar{f}_u, \bar{f}_x, \bar{f}_y)$ 
17:          result  $\leftarrow \max\{\text{result}, \text{CHOOSE-BEST-FLIP}(u, v, f_u, f_v)\}$ 
18:    else  $p_x = p_y$   $\triangleright$  The two cycles have no common edges (case X or H)
19:       $e_u^1, e_u^2 \leftarrow$  edges incident to  $\pi_C(v)$  on  $C$ 
20:       $e_v^1, e_v^2 \leftarrow$  edges incident to  $\pi_{C'}(u)$  on  $C'$ 
21:       $\bar{f}_u^1, \bar{f}_u^2, \bar{f}_v^1, \bar{f}_v^2 \leftarrow$  faces incident to  $e_u^1, e_u^2, e_v^1, e_v^2$  on the same side of  $C, C'$  as  $\bar{f}_u$ 
22:      for  $\bar{f}_x \in \{\bar{f}_u^1, \bar{f}_u^2\}$  do
23:        for  $\bar{f}_y \in \{\bar{f}_v^1, \bar{f}_v^2\}$  do
24:           $f_v \leftarrow \text{meet}(\bar{f}_u, \bar{f}_x, \bar{f}_y)$ 
25:          result  $\leftarrow \max\{\text{result}, \text{CHOOSE-BEST-FLIP}(u, v, f_u, f_v)\}$ 
26:  return result
```

---



**Lemma 57.** *If  $(f_u, \cdot, C, e'_u, e'_v)$  is good,  $E_{>b} \cap C = \emptyset$ ,  $E_{<b} \cap C \neq \emptyset$ ,  $(x, y)$  closes a good fundamental cycle intersecting  $E_{>b}$ , and  $X_b$  is an R node, then  $\text{FIND-SEP-R01}(u, v, f_u, C, e'_u, e'_v, x, y)$  in Algorithm 13 returns the size and corners of a maximal  $u$ -flip.*

*Proof.* By Lemma 39 it is sufficient to show that at least one of the candidates to  $f_v$  used in the calls to  $\text{CHOOSE-BEST-FLIP}(u, v, f_u, f_v)$  is correct.

If  $p_x \neq p_y$  then  $C \cup C'$  consists of 3 internally vertex-disjoint paths from  $p_x$  to  $p_y$ , and by Lemma 52 with  $e = e'_u$  there exists  $\bar{f}_x \in \{\bar{f}_x^1, \bar{f}_x^2\}$  and  $\bar{f}_y \in \{\bar{f}_y^1, \bar{f}_y^2\}$  such that  $f_v = \text{meet}(\bar{f}_u, \bar{f}_x, \bar{f}_y)$ .

Otherwise  $p_x = p_y$ , and  $C$  and  $C'$  has at most one vertex in common, so by Lemma 53 with  $e = e'_u$  there exists  $\bar{f}_x \in \{\bar{f}_x^1, \bar{f}_x^2\}$  and  $\bar{f}_y \in \{\bar{f}_y^1, \bar{f}_y^2\}$  such that  $f_v = \text{meet}(\bar{f}_u, \bar{f}_x, \bar{f}_y)$ .  $\square$

**Lemma 58.** *If  $(f_u, \cdot, C, e'_u, e'_v)$  is good,  $E_{>b} \cap C = \emptyset$ , and  $E_{<b} \cap C = \emptyset$ , then  $X_b$  is a P node.*

*Proof.* Follows directly from the definition of  $f_u, C, e'_u, e'_v$  being correct.  $\square$

**Lemma 59.**  $\text{FIND-FIRST-SEPARATION-FLIP}(u, v)$  in Algorithm 7 runs in worst case  $\mathcal{O}(\log^2 n)$  time, and:

- If  $G \cup (u, v)$  is planar it always finds a maximal  $u$ -flip.
- If  $G \cup (u, v)$  is non-planar it either:
  - finds a maximal  $u$ -flip; or
  - finds a  $u$ -flip  $\sigma$  such that immediately calling  $\text{FIND-FIRST-SEPARATION-FLIP}(u, v)$  again after executing  $\sigma$  will return a  $u$ -flip  $\sigma'$  of the same size; or
  - finds no  $u$ -flip.

*Proof.* If  $G \cup (u, v)$  is planar and only one flip is needed to admit  $(u, v)$ , then by Lemma 43 our  $\text{FIND-SINGLE-FLIP-CANDIDATES}$  algorithm will find a correct candidate tuple, and then by Lemma 39  $\text{CHOOSE-BEST-FLIP}$  will select the corresponding maximal  $u$ -flip, and this will be returned.

Otherwise, let  $b$  be the minimum index of an  $f_u$ -blocking node. If  $b = 1$  then no  $u$ -flip exists and so no  $u$ -flip is found. If  $1 < b < r$ , by Lemma 44 at least one of the candidate tuples is good. Let  $(f_u, \cdot, C, e'_u, e'_v)$  be the good candidate tuple. Let  $E_1, \dots, E_k$  be as in Definition 33. We will case by how  $C$  intersects  $E_{>b}$ , and then by whether  $C$  separates  $f_u$  from  $f_v$ , and then, finally, by whether the  $X_b$  is a P or an R node.

- If  $C \cap E_{>b} \neq \emptyset$ 
  - if  $C$  is separating, then by Lemma 45,  $(x, y)$  closes a good fundamental cycle  $C'$ , such that  $C' \setminus C$  is a path in  $E_b$ , and:
    - \* If  $X_b$  is a P node,  $\{s_i, t_i\} = \{\pi_C(x), \pi_C(y)\}$  so by Lemma 50, we return the maximal  $u$ -flip.
    - \* If  $X_b$  is an R node, then by Lemma 54, we return the maximal  $u$ -flip.
  - if  $C$  is not separating, then by Lemma 46,  $(x, y)$  closes a good fundamental cycle  $C'$  intersecting  $E_{<b}$ , and:
    - \* If  $X_b$  is a P node, then by Lemma 51, we return the maximal  $u$ -flip.
    - \* If  $X_b$  is an R node, then by Lemma 55, we return the maximal  $u$ -flip.
- If  $C \cap E_{>b} = \emptyset$  then by Lemma 49 at least one of the  $(x, y)$  we try closes a cycle  $C'$  intersecting  $E_{>b}$ , and:
  - If  $X_b$  is a P node, then by Lemma 56, we return the maximal  $u$ -flip.
  - If  $X_b$  is an R node, then, by Lemma 58,  $C'$  is separating. Then, by Lemma 57, we return the maximal  $u$ -flip.

Finally, if  $b = r$  and  $G \cup (u, v)$  is nonplanar then we may find a non-maximal  $u$ -flip  $\sigma$ . This flip will be in some separation pair  $\{s_j, t_j\}$  with  $j < b - 1$ , and since every flip is chosen by a call  $\text{CHOOSE-BEST-FLIP}(u, v, f_u, f'_v)$  to be maximal for  $f_u, f'_v$ ,  $X_j$  is not an S node. But then after performing the flip,  $X_j$  will be the first  $f_u$ -blocking node, and since  $\sigma$  was locally maximal it will not be anti- $f_u$ -blocking. By the previous argument we are guaranteed that the next call to  $\text{FIND-FIRST-SEPARATION-FLIP}(u, v)$  finds the unique (since  $X_j$  is not anti- $f_u$ -blocking) maximal  $u$ -flip, which is the inverse of  $\sigma$  and therefore has the same size.  $\square$

We are finally ready to prove a main theorem:

**Theorem 2.** *There is a data structure for maintaining a planar embedding of a fully-dynamic planar graph that handles edge-updates and planarity-compatibility queries in amortized  $\mathcal{O}(\log^3 n)$  time, edge deletions in worst-case  $\mathcal{O}(\log^2 n)$  time, and queries to the neighbors of a given existing edge in the current embedding in worst-case  $\mathcal{O}(\log^2 n)$  time.*

*Proof.* Consider the function  $\text{MULTI-FLIP-LINKABLE}(u, v)$  from Algorithm 1. By Lemma 59 and Lemma 32 each  $\text{DO-SEPARATION-FLIPS}(u, v)$  runs in  $\mathcal{O}(\log^2 n)$  time per flip, and only does critical potential-decreasing flips (and at most one critical flip that is not potential-decreasing). Similarly, by Lemma 30 each  $\text{DO-ARTICULATION-FLIPS}(u, u', v', v)$  runs in  $\mathcal{O}(\log^2 n)$  time and only does critical potential-decreasing or potential-neutral flips. By Lemma 31 any such potential-neutral flip is immediately followed by either a potential-decreasing flip (either a separation flip or an articulation flip) or by the final flip. Thus by Corollary 27 with  $r = 2$ ,  $\text{MULTI-FLIP-LINKABLE}(u, v)$  does amortized  $\mathcal{O}(\log n)$  flips.

By Lemma 29, each  $\text{FIND-NEXT-FLIP-BLOCK}(u, u', v', v)$  call also runs in  $\mathcal{O}(\log^2 n)$  time. By Lemma 28 the main loop in  $\text{MULTI-FLIP-LINKABLE}(u, v)$  iterates amortized  $\mathcal{O}(\log n)$  times, and we have shown that each iteration takes  $\mathcal{O}(\log^2 n)$  time (in addition to the time taken by the separation flips). Thus the total amortized time for  $\text{MULTI-FLIP-LINKABLE}(u, v)$  is  $\mathcal{O}(\log^3 n)$ .

Using that, the remaining edge insertion and deletion is trivial. And the queries to the embedding are handled directly by the underlying data structure from [11].  $\square$

## 5 Allowing non-planar insertions

In [7, p.12, proof of Corollary 1], Eppstein et al. give a reduction from any data structure that maintains a planar graph subject to deletions and planarity-preserving insertions and answers queries to the planarity-compatibility of edges, to a data structure that allows the graph to be non-planar and furthermore maintains whether the graph is presently planar, at the same amortized time. The reduction uses the following simple and elegant argument: If some component of the graph is non-planar, keep a pile of not-yet inserted edges, and upon a deletion, add edges from the pile until either the pile is empty or a new planarity-blocking certificate edge is found.

To maintain not only whether the graph is planar but maintain for each component whether it is planar, it becomes necessary to keep a pile of not-yet inserted edges for each nonplanar connected component. To maintain the connected components, we run an auxiliary fully-dynamic connectivity structure for the entire graph [22], maintaining a spanning forest and the non-tree edges. We may mark the edges indicating whether they are inserted or not-yet inserted in the planar subgraph, and we may mark vertices indicating whether they are incident to not-yet inserted edges. When an edge deletion causes a non-planar component to break into two, a spanning tree breaks into two, say,  $T_u$

and  $T_v$ . For each  $i \in \{u, v\}$ , we may efficiently find not-yet inserted edges incident to  $T_i$ , and insert them into the planar subgraph of the component spanned by  $T_i$ . We may continue until either all edges for that component are handled or until we find the first planarity violating edge. The method for efficiently finding not-yet inserted edges follows the exact same outline as the method for finding candidate replacement edges in the connectivity structure [22]. Thus, the time spent on each edge becomes  $\mathcal{O}(\log n / \log \log n)$  worst-case for finding it, plus  $\mathcal{O}(\log^3 n)$  amortized time for inserting it. We have thus shown

**Theorem 1.** *There is a data structure for fully-dynamic planarity testing that handles edge-insertions and edge-deletions in amortized  $\mathcal{O}(\log^3 n)$  time, answers queries to planarity-compatibility of an edge in amortized  $\mathcal{O}(\log^3 n)$  time, and answers queries to whether the graph is presently planar in worst case  $\mathcal{O}(1)$  time, or to whether the component of a given vertex is presently planar in worst case  $\mathcal{O}(\log n / \log \log n)$  time. It maintains an implicit representation of an embedding that is planar on each planar connected component, and may answer queries to the neighbors of a given existing edge in this current embedding, in  $\mathcal{O}(\log^2 n)$  time.*

## 6 Defining critical-cost and solid-cost

The goal of this section is to properly define the two function families  $\text{critical-cost}_\tau$  and  $\text{solid-cost}_\tau$  mentioned earlier so we can prove the claimed properties.

The general idea is for each function to define a set of *struts*, which are edges that can be inserted in  $G$  without violating planarity, and then measure the total number of flips needed to accommodate all of them.

$$\begin{aligned} \text{critical-cost}_\tau(H; u, v) &= \sum_{(x,y) \in \text{critical-struts}(G; u, v)} \text{dist}_\tau(H, \text{Emb}(G; x, y)) \\ \text{solid-cost}_\tau(H; u, v) &= \sum_{(x,y) \in \text{solid-struts}(G; u, v)} \text{dist}_\tau(H, \text{Emb}(G; x, y)) \end{aligned}$$

We want our struts to have the following properties for any planar graph  $G$  with vertices  $u, v$ :

- S1)  $G \cup \text{solid-struts}(G; u, v)$  is simple and planar.
- S2) For any  $H, H' \in \text{Emb}(G)$  with  $\text{dist}_{\text{clean}}(H, H') = 1$ , there is at most one strut  $(x, y) \in \text{solid-struts}(G; u, v)$  such that  $\text{dist}_{\text{clean}}(H, \text{Emb}(G; x, y)) \neq \text{dist}_{\text{clean}}(H', \text{Emb}(G; x, y))$ .
- S3) If  $H \in \text{Emb}(G)$  admits  $\text{solid-struts}(G; u, v)$  then for any  $u', v'$  there exists  $H' \in \text{Emb}(G)$  that admits  $\text{solid-struts}(G; u', v')$  such that  $\text{dist}_{\text{clean}}(H, H') \in \mathcal{O}(\log n)$ .
- S4) If  $G \cup (u, v)$  is simple and planar, then  $\text{solid-struts}(G; u, v) = \text{solid-struts}(G \cup (u, v); u, v) \cup \{(u, v)\}$ .
- S5)  $\text{critical-struts}(G; u, v) \subseteq \text{solid-struts}(G; u, v)$ .
- S6) For  $(u, v) \in G$ ,  $\text{critical-struts}(G; u, v) = \emptyset$ .
- S7) For  $(u, v) \notin G$ ,  $\text{critical-struts}(G; u, v) = \{(u, v)\}$  if and only if  $G \cup (u, v)$  is planar.
- S8) If  $G \cup (u, v)$  is nonplanar, and  $v = a_0, B_1, a_1, \dots, B_k, a_k = v$  are the endpoints, biconnected components/bridges, and articulation points on  $u \cdots v$  in  $G$ , then
  - (a) For each  $B_i$  such that  $B_i \cup (a_{i-1}, a_i)$  is nonplanar,  $\text{critical-struts}(G; u, v)$  contains a set  $S_i$  of struts with both endpoints in  $B_i$  such that  $a_{i-1}$  and  $a_i$  are triconnected in  $B_i \cup S_i$  and  $B_i \cup S_i$  is planar.

- (b) For each maximal subsequence  $a_{\ell-1}, B_\ell, a_\ell, \dots, B_h, a_h$  such that  $G \cup (a_{\ell-1}, a_h)$  is planar, and such that at least one  $B_i$  with  $\ell \leq i \leq h$  is not a bridge,  $\text{critical-struts}(G; u, v)$  contains a strut  $(a_{\ell-1}, a_h)$ .

Assuming our struts have these properties, we can now prove our Lemmas about the costs.

*Proof of Lemma 15.* The inequality  $\text{solid-cost}_\tau(G; u, v) \geq \text{critical-cost}(G; u, v) \geq 0$  follows from property S5 and the definition of  $\text{solid-cost}_\tau$  and  $\text{critical-cost}_\tau$ . The inequalities  $\text{solid-cost}_{\text{clean}} \geq \text{solid-cost}_{\text{sep}} \geq \text{solid-cost}_P$  and  $\text{critical-cost}_{\text{clean}} \geq \text{critical-cost}_{\text{sep}} \geq \text{critical-cost}_P$  also follow, because in general  $\text{dist}_{\text{clean}} \geq \text{dist}_{\text{sep}} \geq \text{dist}_P$ . Finally, if  $G \cup (u, v)$  is planar, then by property S7,  $\text{critical-struts}(G; u, v) = \{(u, v)\}$  so  $\text{critical-cost}_{\text{clean}}(H; u, v) = \text{dist}_{\text{clean}}(H; \text{Emb}(G; u, v))$ , which is 0 if and only if  $H \in \text{Emb}(G; u, v)$ .  $\square$

*Proof of Lemma 16.* Follows directly from properties S2 and S5, and the definition of critical flip.  $\square$

*Proof of Lemma 17.* From the definition of  $\text{critical-cost}_\tau$  and  $\text{solid-cost}_\tau$  as a sum over  $\text{dist}_\tau$ , it is clear that if the cost is nonzero there exists a flip of type  $\tau$  that decreases at least one of the terms. But by property S2, this is then the only term that changes so this flip also decreases the sum.  $\square$

*Proof of Lemma 21.* By definition of  $H \in \text{Emb}^*(G)$ , there exists vertices  $u_{\min}, v_{\min}$  such that  $\text{solid-cost}(H; u_{\min}, v_{\min}) = 0$ . By definition of  $\text{solid-cost}_{\text{clean}}$  that means  $H$  admits  $\text{solid-struts}_{\text{clean}}(G; u_{\min}, v_{\min})$ . Then by Property S3 there exists  $H' \in \text{Emb}(G)$  that admits  $\text{solid-struts}(G; u, v)$  and has  $\text{dist}_{\text{clean}}(H, H') \in \mathcal{O}(\log n)$ . Since  $H'$  admits  $\text{solid-struts}(G; u, v)$ ,  $H' \in \text{Emb}^*(G; u, v)$ , and thus  $\text{dist}_{\text{clean}}(H, \text{Emb}^*(G; u, v)) \leq \text{dist}_{\text{clean}}(H, H') \in \mathcal{O}(\log n)$ .  $\square$

*Proof of Lemma 23.* By Property S4, the only difference between  $\text{solid-struts}(G; u, v)$  and  $\text{solid-struts}(G \cup (u, v); u, v)$  is that the first includes the strut  $(u, v)$ . Since  $H \in \text{Emb}(G; u, v)$ ,  $\text{dist}_{\text{clean}}(H, \text{Emb}(G; u, v)) = 0$  so removing this term does not change the sum. By Property S2, the possible flips for every other strut  $(u', v')$  is unaffected, meaning that  $\text{dist}(H, \text{Emb}(G; u', v')) = \text{dist}(H \cup (u, v), \text{Emb}(G \cup (u, v); u', v'))$ . In other words,  $\text{solid-cost}(H; u, v) = \text{solid-cost}(H \cup (u, v); u, v)$ .  $\square$

*Proof of Lemma 26.* Follows directly from the definition of  $\text{critical-cost}_\tau$  and  $\text{solid-cost}_\tau$  as a sum  $\sum_{(x,y)} \text{dist}_\tau(H; x, y)$ , and the definition of  $\text{dist}_\tau$ . By definition an SR flip can not change any  $\text{dist}_P$ , and an articulation flip can not change any  $\text{dist}_P$  or  $\text{dist}_{\text{sep}}$ .  $\square$

## 6.1 Biconnected planar graphs

Let  $\text{SPQR}(B; u, v)$  denote the solid paths in the pre-split SPQR tree for the biconnected component  $B$  with critical vertices  $u, v$ .

Let  $\beta$  be a solid path in  $\text{SPQR}(B; u, v)$ . The *relevant part* of  $\beta$  is the maximal subpath that does not end in a P node.

**Single solid SPQR path, simple case** If  $\beta$  consists only of a P node, or the relevant part of  $\beta$  is only a single node, define the struts relevant for  $\beta$  as:

$$\text{struts}(\beta) = \begin{cases} \{(u, v)\} & \text{if } \beta \text{ is the critical path, } (u, v) \notin B, \text{ and } G \cup (u, v) \text{ is planar} \\ \emptyset & \text{otherwise} \end{cases}$$

**Single solid SPQR path, general case** Otherwise let  $X_1, \dots, X_d$  be the relevant part of  $\beta$ . For  $1 \leq j < d$  let  $\{s_j, t_j\} = X_j \cap X_{j+1}$  be the separation pair that separates  $X_j$  from  $X_{j+1}$ . For  $1 < j < d$  we call the node  $X_j$  *cross* if  $X_j$  is an R node, and the virtual edges  $(s_{j-1}, t_{j-1})$  and  $(s_j, t_j)$  do not share a face in  $\Gamma(X_j)$ . If  $\beta$  is the critical path, we can assume without loss of generality that  $u \in X_1$  and  $v \in X_d$ , and we say that  $X_1$  (resp.  $X_d$ ) is cross if it is an R node and  $u$  (resp.  $v$ ) does not share a face with  $(s_1, t_1)$  (resp.  $(s_{d-1}, t_{d-1})$ ).

Let  $\gamma = X_\ell, \dots, X_h$  be a maximal subpath of  $X_1, \dots, X_d$  such that  $X_j$  is not a cross node for  $\ell < j < h$ . Let  $u_\gamma$  be the smallest-labelled vertex in  $\Gamma(X_\ell) - \{s_\ell, t_\ell\}$  that shares a face with  $(s_\ell, t_\ell)$  (counting  $u$  as having label  $-\infty$ ). Similarly let  $v_\gamma$  be the smallest-labelled vertex in  $\Gamma(X_h) - \{s_h, t_h\}$  that shares a face with  $(s_h, t_h)$  (counting  $v$  as having label  $-\infty$ ).

We can now define the struts relevant for  $\beta$  as

$$\text{struts}(\beta) = \{(u_\gamma, v_\gamma) \mid \gamma \text{ is such a maximal subpath}\}$$

### Combining the struts for a SPQR tree

$$\begin{aligned} \text{critical-struts}(B; u, v) &= \bigcup_{\substack{\beta \in \text{SPQR}(B; u, v) \\ \beta \text{ is critical}}} \text{struts}(\beta) \\ \text{off-critical-struts}(B; u, v) &= \bigcup_{\substack{\beta \in \text{SPQR}(B; u, v) \\ \beta \text{ is not critical}}} \text{struts}(\beta) \\ \text{solid-struts}(B; u, v) &= \text{critical-struts}(B; u, v) \cup \text{off-critical-struts}(B; u, v) \end{aligned}$$

## 6.2 General planar graphs

Let  $\text{BC}(G; u, v)$  denote the set of solid paths in the forest of pre-split BC trees for a planar graph  $G$  with critical vertices  $u, v$ .

Let  $\alpha$  be a solid path in  $\text{BC}(G; u, v)$ . The *relevant part* of  $\alpha$  is the maximal subpath that does not end in a C node.

**Single solid BC path, simple case** If  $\alpha$  consists only of a C node, define

$$\begin{aligned} \text{critical-struts}(\alpha) &= \emptyset \\ \text{off-critical-struts}(\alpha) &= \emptyset \\ \text{solid-struts}(\alpha) &= \emptyset \end{aligned}$$

**Single solid BC path, general case** In [12] we defined a set of struts for each such path and used it to choose a critical path in the SPQR tree for each biconnected component. Since those struts might make the graph non-planar, we need a new set; we want to substitute every non-planar strut with a family of “maximal” planar struts in the following sense. Let  $B_1, \dots, B_k$  be the B nodes on  $\alpha$ , for  $1 < i < k$  let  $a_i = B_i \cap B_{i+1}$ , and let  $(a_0, a_k)$  be the strut defined for  $\alpha$  in [12]. Note that if  $\alpha$  is the critical path in  $\text{BC}(G; u, v)$  then we may assume without loss of generality that  $a_0 = u$  and  $a_k = v$ .

Now define

$$\begin{aligned}
\text{critical-struts}(\alpha) &= \{ \text{critical-struts}(B_i; a_{i-1}, a_i) \mid G \cup (a_{i-1}, a_i) \text{ is nonplanar} \} \\
&\cup \left\{ (a_{\ell-1}, a_h) \mid \begin{array}{l} (a_{\ell-1}, a_h) \notin G \text{ and} \\ B_\ell, \dots, B_h \text{ is a maximal subpath of } \alpha \text{ such that} \\ G \cup (a_{\ell-1}, a_h) \text{ is planar} \end{array} \right\} \\
\text{off-critical-struts}(\alpha) &= \bigcup_{i=1}^k \text{off-critical-struts}(B_i; a_{i-1}, a_i) \\
\text{solid-struts}(\alpha) &= \text{critical-struts}(\alpha) \cup \text{off-critical-struts}(\alpha)
\end{aligned}$$

**Combining the struts from a forest of BC trees** Now define:

$$\begin{aligned}
\text{critical-struts}(G; u, v) &= \bigcup_{\substack{\alpha \in \text{BC}(G; u, v) \\ \alpha \text{ is critical}}} \text{critical-struts}(\alpha) \\
\text{off-critical-struts}(G; u, v) &= \left( \bigcup_{\substack{\alpha \in \text{BC}(G; u, v) \\ \alpha \text{ is critical}}} \text{off-critical-struts}(\alpha) \right) \cup \left( \bigcup_{\substack{\alpha \in \text{BC}(G; u, v) \\ \alpha \text{ is not critical}}} \text{solid-struts}(\alpha) \right) \\
\text{solid-struts}(G; u, v) &= \text{critical-struts}(G; u, v) \cup \text{off-critical-struts}(G; u, v)
\end{aligned}$$

### 6.3 Proving the required properties

**Lemma 60.** *The definition of  $\text{critical-struts}(G; u, v)$  and  $\text{solid-struts}(G; u, v)$  in Section 6.1 and 6.2 have property S1–S8.*

*Proof.* Every clean separation flip in  $\text{Emb}(G)$  corresponds to an edge or a P node in a SPQR tree for a biconnected component of  $G$ . The way the struts are chosen for biconnected graphs, means that no two struts “cover” the same edge or P node. Similarly, each possible articulation flip in  $\text{Emb}(G)$  correspond to a C node, and no two struts cover the same C node. Thus any flip in  $H \in \text{Emb}(G)$  can affect  $\text{dist}_\tau(H, \text{Emb}(G; x, y))$  for at most 1 strut  $(x, y)$  proving Property S2.

Since each strut  $(x, y)$  is chosen so  $G \cup (x, y)$  is simple and planar, and inserting one strut can not prevent the insertion of another,  $G \cup \text{solid-struts}(G; u, v)$  is planar and Property S1 holds.

In [12], we showed that it takes only  $\mathcal{O}(\log n)$  simple operations (merging or splitting S,P and C nodes, and changing edges between solid and dashed) to get from the pre-split BC/SPQR trees for  $G$  with respect to  $u, v$  to the trees with respect to  $u', v'$ . Each of these operations affect only at most 2 solid paths, and by our definition, at most one strut on each of these paths. The total change in  $\text{solid-cost}_\tau$  for each of these  $\mathcal{O}(\log n)$  operations is at most a constant, so  $|\text{solid-cost}_\tau(H; u, v) - \text{solid-cost}_\tau(H; u', v')| \in \mathcal{O}(\log n)$ . If  $H$  admits  $\text{solid-struts}(G; u, v)$  then  $\text{solid-cost}_{\text{clean}}(H; u, v) = 0$ , so  $\text{solid-cost}_{\text{clean}}(H; u', v') \in \mathcal{O}(\log n)$ . For each strut  $(x, y) \in \text{solid-struts}(G; u', v')$  with  $\text{dist}_{\text{clean}}(H, \text{Emb}(x, y)) > 0$  there exists some flip that will reduce this distance. After  $\mathcal{O}(\log n)$  such flips we arrive at a new embedding  $H' \in \text{Emb}(G)$  with  $\text{solid-cost}_{\text{clean}}(H'; u', v') = 0$ , which means  $H'$  admits  $\text{solid-struts}(G; u', v')$ . By construction  $\text{dist}_{\text{clean}}(H, H') \in \mathcal{O}(\log n)$ , so this proves Property S3.

Property S5 follows trivially from the definition of  $\text{solid-struts}(G; u, v)$  as the (disjoint) union of  $\text{critical-struts}(G; u, v)$  and  $\text{off-critical-struts}(G; u, v)$ .

Property S6 follows by noting that in the definition, every edge added as a strut is conditioned on the edge not being in the graph.

Property S7 can be seen by considering the definition of  $\text{critical-struts}(\alpha)$ . If  $(u, v) \notin G$  and  $G \cup (u, v)$  is planar, then for the critical path  $\alpha \in \text{BC}(G; u, v)$ , the maximal subpath found in the definition is exactly the one that starts in  $u$  and ends in  $v$ . So in this case,  $\text{critical-struts} = \{(u, v)\}$ . If  $G \cup (u, v)$  is not planar  $(u, v) \notin \text{solid-struts}(G; u, v)$  by Property S1, and hence  $(u, v) \notin \text{critical-struts}(G; u, v)$  by Property S5, and in particular  $\text{critical-struts}(G; u, v) \neq \{(u, v)\}$ .

If  $G \cup (u, v)$  is simple and planar, our definition guarantees that  $(u, v) \in \text{solid-struts}(G; u, v)$ . By definition of the pre-split BC/SPQR trees, the only change to the solid paths when inserting  $(u, v)$  is the contraction of the critical path in the BC tree, and in all the SPQR trees on the critical path. All other solid paths in the pre-split BC/SPQR tree remain unchanged. In particular  $\text{off-critical-struts}(G; u, v) = \text{off-critical-struts}(G \cup (u, v); u, v)$ , and together with Property S6 and S7 and the fact that  $\text{solid-struts}(G; u, v)$  is the disjoint union of  $\text{critical-struts}(G; u, v)$  and  $\text{off-critical-struts}(G; u, v)$  this proves Property S4.

Property S8 follows by simple inspection. Case S8a follows from using maximal subpaths in the definition in Section 6.1. Similarly, Case S8b follows from using maximal subpaths in the definition in Section 6.2.  $\square$

*Proof of Lemma 24.* If the clean separation flip at  $s, t$  changes  $\text{solid-cost}_\tau$  or  $\text{critical-cost}_\tau$ , it must be internal to a unique solid path  $\beta$  in some SPQR tree. Since it is internal, even if  $\beta$  is the critical path for some  $x, y$ , then  $\{x, y\} \cap \{s, t\} = \emptyset$ . Thus,  $s, t$  are not internal to any solid path in the BC tree, and do not have any contribution to  $\text{solid-cost}_\tau$  or  $\text{critical-cost}_\tau$  that can change.

If the clean separation flip does not change  $\text{solid-cost}_\tau$  or  $\text{critical-cost}_\tau$ , it may be incident to the end of a critical path  $m(x, y)$ . However, at most 2 of the at most 4 articulation flips contain neighboring blocks on the solid path in the BC tree. If 2 of them do, then the flip does not change whether or not they share a face, and so  $\text{solid-cost}_\tau$  and  $\text{critical-cost}_\tau$  are unchanged. Otherwise at most 1 of the at most 4 articulation flips change  $\text{solid-cost}_\tau$  or  $\text{critical-cost}_\tau$ , as desired.  $\square$

## 7 Conclusion

We have given an amortized  $\mathcal{O}(\log^3 n)$  time algorithm for updating whether a graph is still planar after the insertion or deletion of an edge. This is close but not equal to the theoretical lower bound of  $\Omega(\log n)$  [16]. An interesting open question is whether this time bound can be improved, or whether an algorithm with worst-case polylogarithmic update time exists.

**Acknowledgements.** The authors would like to thank Mikkel Thorup, Kristian de Lichtenberg, and Christian Wulff-Nilsen for their interest and encouragement.

## References

- [1] Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 477–486, 2016. doi:10.1109/FOCS.2016.58. 1

- [2] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014. doi:10.1109/FOCS.2014.53. 1
- [3] Samuel W. Bent, Daniel Dominic Sleator, and Robert Endre Tarjan. Biased search trees. *SIAM J. Comput.*, 14(3):545–568, 1985. doi:10.1137/0214041. 5
- [4] Gerth Stølting Brodal and Rolf Fagerberg. Dynamic representation of sparse graphs. In Frank K. H. A. Dehne, Arvind Gupta, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, volume 1663 of *Lecture Notes in Computer Science*, pages 342–351. Springer, 1999. doi:10.1007/3-540-48447-7\_34. 2
- [5] Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/BF01961541. 1, 4
- [6] David Eppstein. Dynamic generators of topologically embedded graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 599–608. ACM/SIAM, 2003. URL: <http://dl.acm.org/citation.cfm?id=644108.644208>. 1
- [7] David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Thomas H. Spencer. Separator based sparsification: I. planarity testing and minimum spanning trees. *Journal of Computer and Systems Sciences*, 52(1):3–27, February 1996. doi:10.1006/jcss.1996.0002. 1, 3, 33
- [8] Zvi Galil, Giuseppe F. Italiano, and Neil Sarnak. Fully dynamic planarity testing with applications. *J. ACM*, 46(1):28–91, 1999. doi:10.1145/300515.300517. 1
- [9] Frank Harary. *Graph Theory*. Addison-Wesley Series in Mathematics. Addison Wesley, 1969. 4
- [10] Jacob Holm, Giuseppe F. Italiano, Adam Karczmarz, Jakub Lacki, and Eva Rotenberg. Decremental spqr-trees for planar graphs. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, volume 112 of *LIPIcs*, pages 46:1–46:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. URL: <http://www.dagstuhl.de/dagpub/978-3-95977-081-1>, doi:10.4230/LIPIcs.ESA.2018.46. 4
- [11] Jacob Holm and Eva Rotenberg. Dynamic planar embeddings of dynamic graphs. *Theory Comput. Syst.*, 61(4):1054–1083, 2017. doi:10.1007/s00224-017-9768-7. 1, 2, 5, 10, 12, 17, 20, 21, 22, 33
- [12] Jacob Holm and Eva Rotenberg. Worst-case polylog incremental spqr-trees: Embeddings, planarity, and triconnectivity, 2019. Accepted for publication at SODA 2020. arXiv:1910.09005. 1, 2, 4, 5, 7, 36, 37
- [13] John E. Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973. doi:10.1137/0202012. 4



- [14] John E. Hopcroft and Robert Endre Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974. doi:10.1145/321850.321852. 1
- [15] Giuseppe F. Italiano, Johannes A. La Poutré, and Monika Rauch. Fully dynamic planarity testing in planar embedded graphs (extended abstract). In Thomas Lengauer, editor, *Algorithms - ESA '93, First Annual European Symposium, Bad Honnef, Germany, September 30 - October 2, 1993, Proceedings*, volume 726 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 1993. doi:10.1007/3-540-57273-2\_57. 1
- [16] Mihai Patrascu and Erik D. Demaine. Lower bounds for dynamic connectivity. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 546–553. ACM, 2004. doi:10.1145/1007352.1007435. 1, 38
- [17] Johannes A. La Poutré. Alpha-algorithms for incremental planarity testing (preliminary version). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 706–715. ACM, 1994. doi:10.1145/195058.195439. 1
- [18] Daniel Dominic Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983. doi:10.1016/0022-0000(83)90006-5. 5
- [19] Roberto Tamassia. On-line planar graph embedding. *J. Algorithms*, 21(2):201–239, 1996. doi:10.1006/jagm.1996.0044. 1
- [20] Jeffery Westbrook. Fast incremental planarity testing. In Werner Kuich, editor, *Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings*, volume 623 of *Lecture Notes in Computer Science*, pages 342–353. Springer, 1992. doi:10.1007/3-540-55719-9\_86. 1
- [21] Hassler Whitney. 2-isomorphic graphs. *American Journal of Mathematics*, 1933. 2
- [22] Christian Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In *Encyclopedia of Algorithms*, pages 738–741. 2016. doi:10.1007/978-1-4939-2864-4\_569. 33, 34