

An Optimal Algorithm for Product Structure in Planar Graphs

Prosenjit Bose  

School of Computer Science, Carleton University, Ottawa, Canada

Pat Morin  

School of Computer Science, Carleton University, Ottawa, Canada

Saeed Odak 

Department of Computer Science and Electrical Engineering, University of Ottawa, Canada

Abstract

The *Product Structure Theorem* for planar graphs (Dujmović et al. *JACM*, **67**(4):22) states that any planar graph is contained in the strong product of a planar 3-tree, a path, and a 3-cycle. We give a simple linear-time algorithm for finding this decomposition as well as several related decompositions. This improves on the previous $O(n \log n)$ time algorithm (Morin. *Algorithmica*, **85**(5):1544–1558).

2012 ACM Subject Classification Mathematics of computing → Graph theory; Mathematics of computing → Graph algorithms

Keywords and phrases Planar graphs, product structure

Digital Object Identifier 10.4230/LIPIcs.SWAT.2022.19

Related Version *Full Version*: <https://arxiv.org/abs/2202.08870>

Funding This research was partly funded by NSERC.

Acknowledgements This research was initiated at the BIRS 21w5235 Workshop on Graph Product Structure Theory, held November 21–26, 2021 at the Banff International Research Station. The authors are grateful to the workshop organizers and participants for providing a stimulating research environment. We are especially grateful to Vida Dujmović for sharing Theorem 1.b with us.

1 Introduction

For two graphs G and X , the notation $G \subseteq X$ denotes that G is isomorphic to some subgraph of X . The following *planar product structure theorems* have recently been used as a key tool in resolving a number of longstanding open problems on planar graphs, including queue number [8], nonrepetitive chromatic number [11], adjacency labelling [10], universal graphs [12], p -centered colouring [6], and vertex ranking [5].¹²

► **Theorem 1** (Dujmović et al. [8], Ueckerdt et al. [18]). *For any planar graph G , there exists:*

- (a) *a planar graph H of treewidth at most 3 and a path P such that $G \subseteq H \boxtimes P \boxtimes K_3$ [8];*
- (b) *a planar graph H of treewidth at most 4 and a path P such that $G \subseteq H \boxtimes P \boxtimes K_2$; and*
- (c) *a planar graph H of treewidth at most 6 and a path P such that $G \subseteq H \boxtimes P$ [18].*

¹ For graphs G and X , an X -decomposition of G is a collection $\mathcal{X} := (B_x : x \in V(X))$ of subsets of $V(G)$ called *bags* indexed by the vertices of X and such that (i) for each $v \in V(G)$, $X[\{x \in V(X) : v \in B_x\}]$ is connected; and (ii) for each $vw \in E(G)$, there exists some $x \in V(X)$ such that $\{v, w\} \subseteq B_x$. The *width* of \mathcal{X} is $\max\{|B_x| : x \in V(X)\} - 1$. In the special case where X is a tree, \mathcal{X} is called a *tree decomposition* of G . The *treewidth* $tw(G)$ of G is the minimum width of any tree decomposition of G .

² For two graphs G_1 and G_2 , the *strong graph product* of G_1 and G_2 , denoted $G_1 \boxtimes G_2$, is a graph whose vertex set is $V(G_1 \boxtimes G_2) := V(G_1) \times V(G_2)$ and that contains an edge between distinct vertices $v = (v_1, v_2)$ and $w = (w_1, w_2)$ if and only if (i) $v_1 = w_1$ and $v_2 w_2 \in E(G_2)$; (ii) $v_2 = w_2$ and $v_1 w_1 \in E(G_1)$; or (iii) $v_1 w_1 \in E(G_1)$ and $v_2 w_2 \in E(G_2)$.



© Prosenjit Bose, Pat Morin, and Saeed Odak;
licensed under Creative Commons License CC-BY 4.0

18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022).

Editors: Artur Czumaj and Qin Xin; Article No. 19; pp. 19:1–19:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In each of the applications of Theorem 1, the proofs are constructive and lead to algorithms whose running-time is dominated by the time required to compute the relevant decomposition. The proofs of each part of Theorem 1 are constructive and lead to $O(n^2)$ time algorithms as observed already by Dujmović et al. [8]. Morin [16] later showed that there exists an $O(n \log n)$ time algorithm to find the decomposition in Theorem 1.a. In the current note, we show that there exists a linear time algorithm for finding each of the three decompositions guaranteed by Theorem 1. This immediately gives an $O(n)$ -time algorithm for each of the following problems on any n -vertex planar graph G :

- computing an $O(1)$ -queue layout of G [8];
- nonrepetitively vertex-colouring G with $O(1)$ colours [11];
- assigning $(1 + o(1)) \log n$ -bit labels to the vertices of G so that one can determine from the labels of vertices v and w whether or not v and w are adjacent in G [10];
- mapping the vertices of G into a universal graph U_n that has $n^{1+o(1)}$ vertices and edges so that any pair of vertices that are adjacent in G maps to a pair of vertices that are adjacent in U_n [12];
- colouring the vertices of G with $O(p^3 \log p)$ colours so that each connected subgraph H of G contains a vertex whose colour is unique in H or contains vertices of at least $p + 1$ different colours [6]; and
- colouring the vertices of G with $O(\log n / \log \log \log n)$ integers so that the maximum colour that appears on any path P of length at most ℓ appears at exactly one vertex of P (for any fixed $\ell \geq 2$) [5].

In addition to planar graphs, product structure theorems similar to Theorem 1 exist for k -planar graphs, h -framed graphs, bounded-genus graphs, and graphs from apex-minor-free families [8, 9, 2]. The existence of each of these decompositions relies at some point on Theorem 1. Thus, the algorithm presented here can also serve as an optimal subroutine in the computation of product structure decompositions of graphs from these classes. However, these larger graph classes also require additional machinery that (at the time of writing) makes finding these decompositions largely impractical.³

The remainder of this paper is organized as follows: Section 2 presents some necessary background and notation. Section 3 reviews the proof of Theorem 1.a. Section 4 presents the linear time algorithm for finding the decomposition in Theorem 1.a. Section 5 describes the algorithms for finding the decompositions in Theorem 1.b and Theorem 1.c.

2 Preliminaries

Throughout this paper we use standard graph theory terminology as used in the textbook by Diestel [7]. All graphs discussed here are simple and finite. For a graph G , $V(G)$ and $E(G)$ denote the vertex and edge sets of G , respectively. We use the terms *vertex* and *node* interchangeably, though we typically refer to the vertices of some primary graph G of interest and refer to the nodes of some auxiliary graph (such as a spanning tree) related to G . We say that a subgraph G' of a graph G *spans* a set $S \subseteq V(G)$ if $S \subseteq V(G')$.

Quotient Graphs

Given a graph G and a partition \mathcal{P} of $V(G)$, the *quotient graph* G/\mathcal{P} is the graph with vertex set $V(G/\mathcal{P}) := \mathcal{P}$ and in which two nodes $X, Y \in V(G/\mathcal{P})$ are adjacent if G contains at least one edge xy with $x \in X$ and $y \in Y$.

³ Possible exceptions here are cases in which a bounded-genus graph, k -planar graph, or h -framed graph is given along with its embedding.

Embeddings, Planar Graphs, and (Near-)Triangulations

An *embedding* ψ of a graph G associates each vertex v of G with a point $\psi(v) \in \mathbb{R}^2$ and each edge vw of G with a simple open curve $\psi(vw) : (0, 1) \rightarrow \mathbb{R}^2$ whose endpoints⁴ are $\psi(v)$ and $\psi(w)$. We do not distinguish between such a curve $\psi(vw)$ and the point set $\{\psi(vw)(t) : 0 < t < 1\}$. We let $\psi(V(G)) := \{\psi(v) : v \in V(G)\}$, $\psi(E(G)) := \bigcup_{vw \in E(G)} \psi(vw)$, and $\psi(G) := \psi(V(G)) \cup \psi(E(G))$. An embedding ψ of G is *plane* if $\psi(vw) \cap \psi(V(G)) = \emptyset$ and $\psi(vw) \cap \psi(xy) = \emptyset$ for each distinct pair of edges $vw, xy \in E(G)$. A graph G is *planar* if it has a plane embedding. A *triangulation* is an edge-maximal planar graph.

If ψ is a plane embedding of a planar graph G , then we call the pair (G, ψ) an *embedded graph* and we will not distinguish between a vertex v of G and the point $\psi(v)$ or between an edge vw of G and the curve $\psi(vw)$. Similarly, we will not distinguish between G and the point set $\psi(G)$. Any cycle in an embedded graph defines a Jordan curve. For such a cycle C , $\mathbb{R}^2 \setminus C$ has two components, one bounded and the other unbounded. We will refer to the bounded component as the *interior* of C and the unbounded component as the *exterior* of C . If G is an embedded triangulation, then the subgraph of G consisting of all edges and vertices of G contained in the closure of the interior of C is called a *near-triangulation*.

Each component of $\mathbb{R}^2 \setminus G$ is a *face* of G and we let $F(G)$ denote the set of faces of G . If G is 2-connected then, for any face $f \in F(G)$, the set of vertices and edges of G contained in the boundary of f forms a cycle. We may therefore treat a face f of a 2-connected graph G as a component of $\mathbb{R}^2 \setminus G$ or as the cycle of G on the boundary of f , relying on context to distinguish between the two usages. Note that every embedded graph contains exactly one face – the *outer face* – that is unbounded.

Duals and Cotrees

The *dual* G^* of an embedded graph G is the graph with vertex set $V(G^*) := F(G)$ and edge set $E(G^*) := \{fg \in \binom{F(G)}{2} : E(f) \cap E(g) \neq \emptyset\}$.⁵ If T is a spanning tree of G then the *cotree* \bar{T} of (G, T) is the graph with vertex set $V(\bar{T}) := V(G^*)$ and edge set $E(\bar{T}) := \{ab \in E(G^*) : E(a) \cap E(b) \setminus E(T) \neq \emptyset\}$. It is well known that, if G is connected and T is a spanning tree of G then \bar{T} is a spanning tree of G^* [7, Chapter 4, Exercise 42].

For our purposes, a *binary tree* is a rooted tree of maximum degree 3 whose root has degree at most 2 and in which each child v of a node u is either the unique *left child* or the unique *right child* of u . If G is a triangulation and we root \bar{T} at any face $f_0 \in F(G)$ that contains an edge of T , then \bar{T} is a binary tree, with the classification of left and right children determined by the embedding of G .⁶

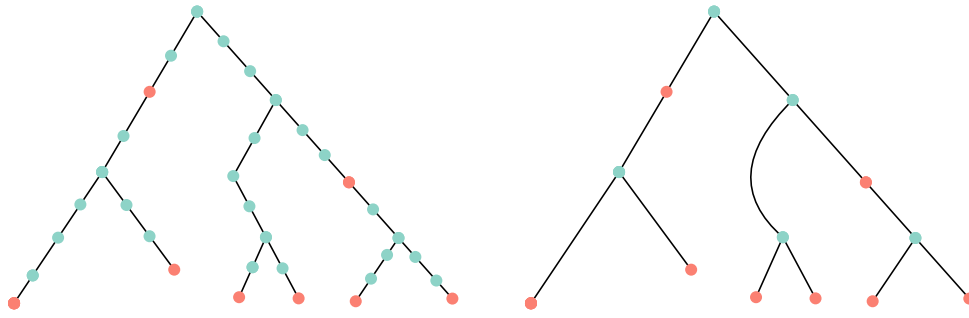
Paths and Distances

A *path* in G is a (possibly empty) sequence of distinct vertices v_0, \dots, v_r with the property that $v_{i-1}v_i \in E(G)$, for each $i \in \{1, \dots, r\}$. The *endpoints* of a path v_0, \dots, v_r are the vertices v_0 and v_r . The *length* of a non-empty path v_0, \dots, v_r is the number, r , of edges in the path.

⁴ The *endpoints* of an open curve $\psi : (0, 1) \rightarrow \mathbb{R}^2$ are the two points $\lim_{\epsilon \downarrow 0} \psi(\epsilon)$ and $\lim_{\epsilon \downarrow 0} \psi(1 - \epsilon)$.

⁵ For a set S , $\binom{S}{2}$ denotes the $\binom{|S|}{2}$ -element set $\binom{S}{2} := \{\{x, y\} : x, y \in S, x \neq y\}$.

⁶ There is a small ambiguity here when T contains two edges of f_0 , in which case the unique child of f_0 in \bar{T} can be treated as the left or right child of f_0 .



■ **Figure 1** A binary tree T with set $S \subseteq V(T)$ depicted in red and the model of T with respect to S .

Trees, Depth, Ancestors, and Descendants

Let T be a tree rooted at a vertex $v_0 \in V(T)$. For any vertex $w \in V(T)$, $P_T(w)$ denotes the path in T from w to v_0 . For any $w_0 \in V(T)$, any prefix w_0, \dots, w_r of $P_T(w_0)$ is called an *upward path* in T ; w_0 is the *lower endpoint* of this path and w_r is the *upper endpoint*. The *T -depth* of a node $w \in V(T)$ is the length of the path $P_T(w)$. The second node in $P_T(v)$ (if any) is the *T -parent* of v . A vertex $a \in V(T)$ is a *T -ancestor* of $w \in V(T)$ if $a \in V(P_T(w))$. If a is a *T -ancestor* of w then w is a *T -descendant* of a .

Lowest Common Ancestors

For any two vertices $v, w \in V(T)$, the *lowest common ancestor* $\text{lca}_T(v, w)$ of v and w is the node a in $P_T(v) \cap P_T(w)$ having maximum T -depth. The *lowest common ancestor problem* is a well-studied data structuring problem that asks to preprocess a given n -vertex rooted tree so that one can quickly return $\text{lca}_T(v, w)$ for any two nodes $v, w \in V(T)$. A number of optimal solutions to this problem exist that, after $O(n)$ time preprocessing using $O(n)$ space, can answer queries in $O(1)$ time [4, 17, 15, 1, 3, 13]. The most recent work in this area includes simple and practical data structures that achieve this optimal performance [1, 3, 13].

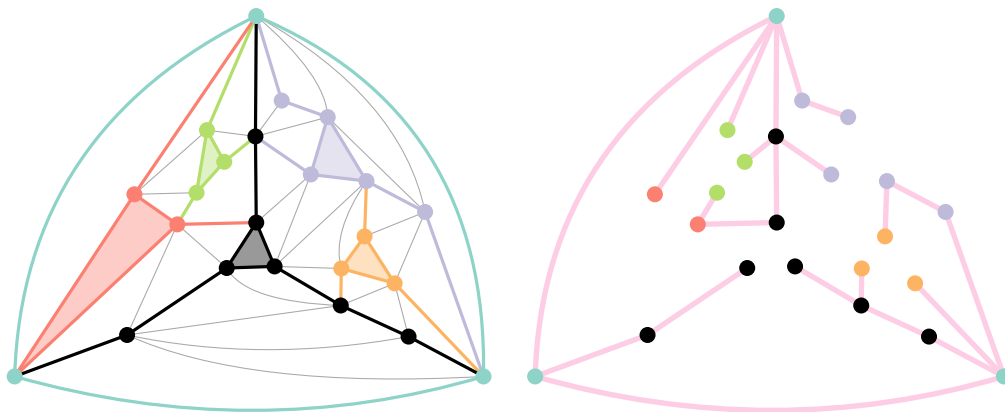
Reconstructing Binary Tree Models

Let T be a binary tree and $S \subseteq V(T)$. An upward path v_0, \dots, v_r in a binary tree T is *S -non-branching* if v_i has degree 2 and $v_i \notin S$ for each $i \in \{1, \dots, r-1\}$. For any binary tree T and set $S \subseteq V(T)$, the *model* T' of T with respect to S is the binary tree obtained by replacing each maximal S -non-branching path v_0, \dots, v_r with the edge v_0v_r ; if v_{r-1} is the left (respectively, right) child of v_r then v_0 becomes the left (respectively, right) child of v_r . See Figure 1.

► **Lemma 2.** *Let T be a binary tree, let $S = \{x_1, \dots, x_d\} \subseteq V(T)$, and let T_0 be the minimal subtree of T that spans S . Then there exists an algorithm that, given an $O(1)$ -query time lowest common ancestor data structure for T , computes the model T'_0 of T_0 with respect to S in $O(d^2)$ time.*

Proof. The proof is by induction on $|S|$. The base case $|S| = 1$ is trivial, since then $T'_0 = T_0$ is the tree with one node, which is the unique element in S .

If $|S| \geq 2$, then the first step is to determine the root r of T_0 , which must also be the root of T'_0 . This is easily done by first setting $r := x_1$ and then repeatedly setting $r := \text{lca}_T(r, x_i)$ for each $i \in \{2, \dots, d\}$. This step takes $O(d)$ time.



■ **Figure 2** A (G, T) -tripod decomposition of a triangulation G (and the underlying spanning tree T).

If r has no left child in T , then we can immediately apply induction on $S \setminus \{r\}$ and make the right child of r in T'_0 the root of the model obtained by induction. The case in which r has no right child can be handled similarly. If r has both a left child r_1 and a right child r_2 , then the next step is to partition $S \setminus \{r\}$ into a set S_1 of descendants of r_1 and a set S_2 of descendants of r_2 . For each $x \in S \setminus \{r\}$ there are only two possibilities for $\text{lca}_T(r_1, x)$

1. If $\text{lca}_T(r_1, x) = r_1$ then $x \in S_1$.
2. If $\text{lca}_T(r_1, x) = r$ then $x \in S_2$.

Therefore, using $O(d)$ lowest common ancestor queries, we can determine the root r of T' and partition $S \setminus \{r\}$ into sets S_1 and S_2 that define the left and right subtrees of r . We can now recurse on S_1 to obtain a tree with root r'_1 and recurse on S_2 to obtain a tree with root r'_2 . We make r'_1 the left child of r and r'_2 the right child of r to obtain the model T'_0 of T_0 . The running-time of this algorithm obeys the recurrence $T(d) \leq O(d) + T(d_1) + T(d_2)$, where $d_1 + d_2 \leq d$ and $d_1, d_2 \leq d - 1$. This recurrence resolves to $T(d) \in O(d^2)$. ◀

3 Tripod Decompositions

Refer to Figure 2. Let G be an n -vertex triangulation and let T be a spanning tree of G . For a face uvw of G , a (G, T) -tripod Y with crotch uvw is the vertex set of three disjoint (and each possibly empty) upward paths in T (the legs of Y) whose lower endpoints are u , v , and w . A (G, T) -tripod decomposition is a partition of $V(G)$ into (G, T) -tripods. Dujmović et al. [8] proved the following result:

► **Theorem 3.** *Let G be a triangulation and T be a spanning tree of G . Then there exists a (G, T) -tripod decomposition \mathcal{Y} such that G/\mathcal{Y} has treewidth at most 3.*

It is straightforward to verify that Theorem 3 implies Theorem 1.a by first triangulating the given graph and then taking T to be a breadth-first spanning tree of the resulting triangulated graph [8, Observation 35].

3.1 Tripod Decompositions from Face Orderings

We now describe how a (G, T) -tripod decompositions can be obtained from a sequence of distinct faces of G . Throughout this section (and for the remainder of the paper):

- G is an embedded triangulation with outer face f_0 and
- T is a spanning-tree of G rooted at a vertex $v_0 \in V(f_0)$.

19:6 An Optimal Algorithm for Product Structure

For any subgraph f of G , we define $Y_T(f) := f \cup \bigcup_{v \in V(f)} P_T(v)$.⁷ In words, $Y_T(f)$ is the subgraph of G that includes all the vertices and edges of f and all the vertices and edges of each path from each vertex of f to the root of T .

Let $\mathcal{F} := f_0, \dots, f_r$ be a sequence of distinct faces of G whose first element is the outer face f_0 . Let G_{-1} denote the graph with no vertices and, for each $i \in \{0, \dots, r\}$, define the graph $G_i := \bigcup_{j=0}^i Y_T(f_j)$ and let $Y_i := V(G_i) \setminus V(G_{i-1})$. Let $\mathcal{G}_{\mathcal{F}} := G_0, \dots, G_r$ and let $\mathcal{Y}_{\mathcal{F}} := Y_0, \dots, Y_r$.

Informally, we require that each of the *legs* of each tripod Y_i have a *foot* on a different vertex of G_{i-1} and that the tripods Y_1, \dots, Y_r cover all the vertices and edges of G . Formally, we say that the sequence \mathcal{F} is *proper* if, for each $i \in \{1, \dots, r\}$, and each distinct $v, w \in V(f_i)$, $V(Y_T(v) \cap G_{i-1}) \neq V(Y_T(w) \cap G_{i-1})$. The sequence \mathcal{F} is *complete* for G if $G_r = G$. Note that, if \mathcal{F} is complete, then $\{Y_0, \dots, Y_r\}$ is a tripod decomposition of G .

From the preceding definitions it follows that, if \mathcal{F} is proper, then G_i is 2-connected for each $i \in \{0, \dots, r\}$. For any $i \in \{0, \dots, r\}$, consider any face f of G_i , that we now treat as a cycle in G . An easy proof by induction shows that, for any $j \in \{0, \dots, i\}$, the induced graph $f[Y_j]$ is connected. We are interested in keeping the number of tripods in Y_0, \dots, Y_i that contribute to $V(f)$ as small as possible, which motivates our next definition.

The sequence \mathcal{F} is *good* if the resulting sequence of graphs $\mathcal{G}_{\mathcal{F}} := G_0, \dots, G_r$ and tripods $\mathcal{Y}_{\mathcal{F}} := Y_0, \dots, Y_r$ satisfy the following condition: For each $i \in \{0, \dots, r\}$ and each face f of G_i ,

$$|\{\ell \in \{0, \dots, i\} : V(f) \cap Y_\ell \neq \emptyset\}| \leq 3 .$$

In words, each face of each graph G_i has vertices from at most three tripods of Y_0, \dots, Y_i on its boundary. Even more, the vertices of f can be partitioned into at most three paths where the vertices of each path belong to a single tripod. Dujmović et al. [8] prove Theorem 3 by proving the next lemma.

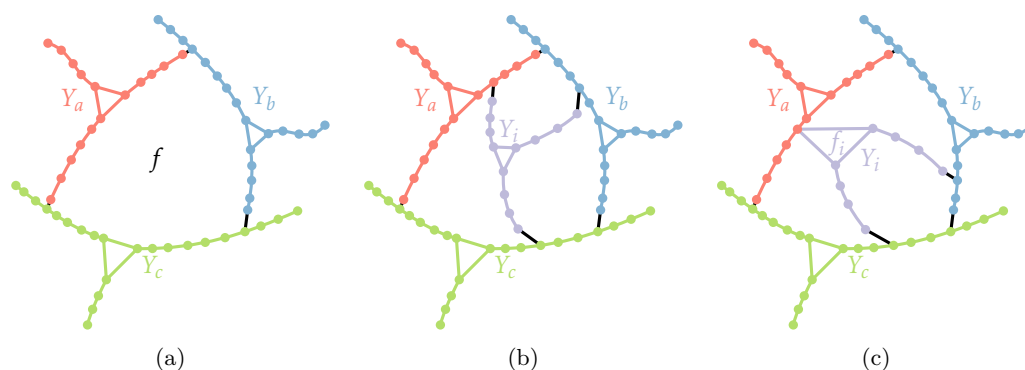
► **Lemma 4.** *Let G be a triangulation with a vertex v_0 on its outer face f_0 and let T be a spanning tree of G rooted at v_0 . Then there exists a sequence $\mathcal{F} := f_0, \dots, f_r$ of distinct faces of G that is proper, good, and complete.*

► **Remark 5.** Lemma 4 is stated in terms of sequences only for convenience and could be rephrased in terms of partial orders. Indeed, consider the partial order \prec defined as follows: For each $i \in \{1, \dots, r\}$ let f'_i be the face of G_{i-1} that contains f_i ; then $f_\ell \prec f_i$ for each $\ell \in \{0, \dots, i-1\}$ such that $V(f'_i) \cap Y_\ell \neq \emptyset$. It is straightforward to check that any linearization of this partial order will result in the same tripod decomposition $\mathcal{Y}_{\mathcal{F}} := \{Y_0, \dots, Y_r\}$.

Dujmović et al. [8] prove Lemma 4 by giving a recursive algorithm that constructs the face sequence \mathcal{F} . For a face f of G_i , define the set $I_f := \{\ell \in \{0, \dots, i\} : V(f) \cap Y_\ell \neq \emptyset\}$. They begin with the outer face f_0 of G . To find the face f_i , $i > 0$, they consider some face $f \notin \{f_0, \dots, f_{i-1}\}$ of G_{i-1} and use Sperner's Lemma to show that there is an appropriate face f_i of G (called a *Sperner triangle*) that is contained in f . In particular, f_i is chosen so that the three upward paths in $Y_T(f_i)$ lead back to each of the (at most 3) tripods in $\{Y_j : j \in I_f\}$. See Figure 3.

This proof leads to a divide-and-conquer algorithm: After finding f_i , the algorithm recursively decomposes each of the near-triangulations that are bounded by the at most three new faces in $S_i := F(G_i) \setminus F(G_{i-1}) \setminus \{f_i\}$. The Sperner triangle f_i can easily be found

⁷ In all of our examples, the subgraph f will always be a single edge or single face of G .



■ **Figure 3** Each face f in G_{i-1} is bounded by at most three tripods Y_{a_f} , Y_{b_f} , and Y_{c_f} and the tripod Y_i is chosen so that it connects each of these.

in time proportional to the number of faces of G in the interior of f . However, because the resulting recursion is not necessarily balanced, a straightforward implementation of this yields an algorithm with $\Theta(n^2)$ worst-case running time.

Morin [16] later showed that, using an appropriate data structure for T , this approach can be implemented in such a way that the resulting algorithm runs in $O(n \log n)$ time. Essentially, Morin's algorithm works by finding the Sperner triangle f_i in time proportional to the minimum number of faces of G contained in any of the faces in S_i . In the next section, we will show that, by using a lowest common ancestor data structure for the cotree \bar{T} along with Lemma 2, the Sperner triangle f_i can be found in constant time, yielding an $O(n)$ time algorithm.

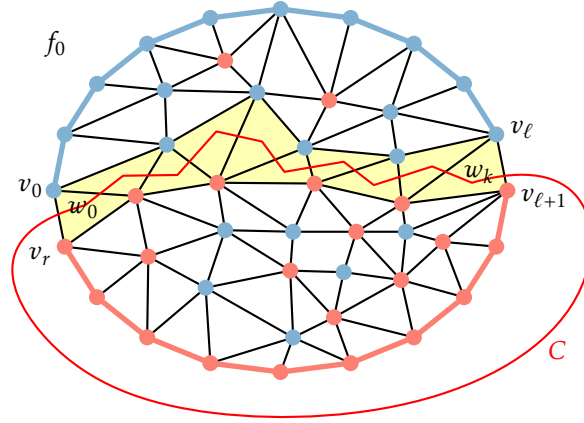
By now, our presentation of this material differs somewhat from that in [8, 18]. Therefore, we now pause to explain how Lemma 4 implies Theorem 3. Let G be a triangulation, let T be spanning tree of G , let $\mathcal{F} := f_0, \dots, f_r$ be the proper good face sequence guaranteed by Lemma 4, and let $\mathcal{Y}_{\mathcal{F}} := \{Y_0, \dots, Y_r\}$ be the resulting tripod decomposition. We now show that there exists a chordal graph H whose largest clique has size at most 4 and that contains $G/\mathcal{Y}_{\mathcal{F}}$. We construct the graph H so that for each $i \in \{0, \dots, r\}$ and each face f of G_i , H contains a clique on $\{Y_j : j \in I_f\}$. To accomplish this, for each $i \in \{1, \dots, r\}$ let f be the face of G_{i-1} that contains f_i and form a clique on $\{Y_i\} \cup \{Y_j : j \in I_f\}$. Inductively, the elements of $\{Y_j : j \in I_f\}$ already form a clique, so this operation is equivalent to attaching Y_i to all the vertices of an existing clique of size at most 3. Therefore, this results in a chordal graph H whose largest clique has size at most 4 and therefore H has treewidth at most 3 [14].

4 An $O(n)$ -Time Algorithm

Refer to Figure 4 for an illustration of the following (probably well-known) simple lemma, which is closely related to Sperner's Lemma:

► **Lemma 6.** *Let N be a near-triangulation with outer face v_0, \dots, v_r and colour each vertex of N red or blue in such a way that v_0, \dots, v_ℓ are coloured red for some $\ell \in \{0, \dots, r-1\}$ and $v_{\ell+1}, \dots, v_r$ are coloured blue. Then there exists a path w_0, \dots, w_k in N^* such that*

1. w_0 is the inner face of N with $v_0 v_r$ on its boundary;
2. w_k is the inner face of N with $v_\ell v_{\ell+1}$ on its boundary; and
3. for each $i \in \{1, \dots, k\}$, the single edge in $E(w_{i-1}) \cap E(w_i)$ has an endpoint of each colour.



■ **Figure 4** Lemma 6.

Proof. If $w_0 = w_k$, the lemma is immediately true, so assume $w_0 \neq w_k$. Say that an edge of N is *bichromatic* if one of its endpoints is red and the other is blue. Any edge that is not bichromatic is *monochromatic*. The outer face f_0 of N has exactly two bichromatic edges v_0v_r and v_lv_{l+1} and any inner face of N has either zero or two bichromatic edges. Consider the subgraph H of N^* obtained removing each edge $fg \in E(N^*)$ such that the edge in $E(f) \cap E(g)$ is monochromatic. Every vertex in H has degree 0 or 2, so each connected component of H is either an isolated vertex or a cycle. The face f_0 has degree 2 so it is contained in a cycle C of H . The two neighbours of f_0 in H are w_0 and w_k . Therefore C contains a path w_0, \dots, w_k that satisfies the conditions of the lemma. ◀

The next lemma, which is the main new insight in this paper, allows us to use Lemma 2 to find Sperner triangles in constant time.

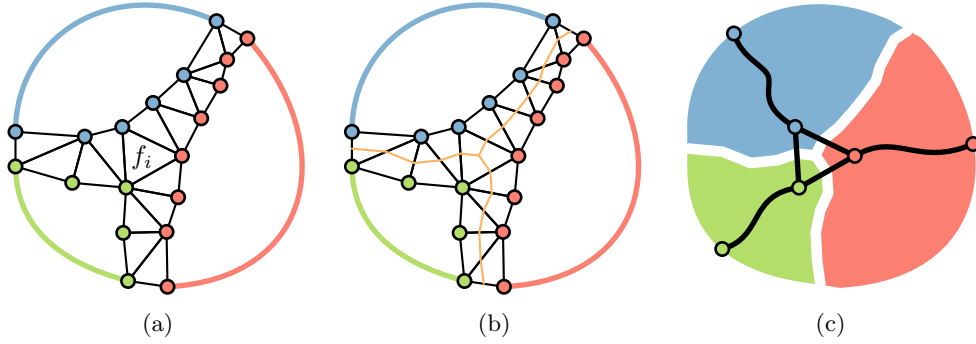
► **Lemma 7.** *Let G be a triangulation with a vertex v_0 on its outer face f_0 ; let T be a spanning tree of G rooted at v_0 ; let \bar{T} be the cotree of (G, T) rooted at f_0 ; let f_0, \dots, f_{i-1} be a good proper sequence of faces of G that yields a sequence $\mathcal{G}_{\mathcal{F}} := G_0, \dots, G_{i-1}$ of graphs and a sequence $\mathcal{Y}_{\mathcal{F}} := Y_0, \dots, Y_{i-1}$ of tripods; let $f \notin \{f_0, \dots, f_{i-1}\}$ be a face of G_{i-1} , and let $S \subseteq F(G)$ contain exactly the (at most three) faces $g \in F(G)$ such that*

- (i) g is contained in the interior of f ;
- (ii) g contains an edge $vw \in E(f)$ with $v \in Y_a$ and $w \in Y_b$ for some distinct $a, b \in I_f$.

Let \bar{T}_0 be the minimal subtree of \bar{T} that spans S . Then, if S is non-empty and $f_i \in V(\bar{T}_0)$ is such that each component of $\bar{T}_0 - f_i$ contains at most one element of S , Then f_0, \dots, f_i is good.

Proof. Let N be the near-triangulation consisting of all vertices and edges of G contained in the closure of the interior of f . Recall that $I_f := \{j \in \{0, \dots, i-1\} : Y_j \cap V(f) \neq \emptyset\}$. Since f_0, \dots, f_{i-1} is good, $|I_f| \leq 3$. Since S is non-empty $|I_f| \geq 2$. For each $j \in I_f$, colour each vertex v of N with the colour j if the first vertex of $P_T(v)$ in $V(f)$ is contained in Y_j . Say that an edge or face of N is *monochromatic*, *bichromatic*, or *trichromatic* if it contains vertices of one, two, or three colours, respectively.

$E(f)$ contains exactly $|I_f|$ bichromatic edges. Since each element of S is an inner face of N that contains a bichromatic edge of f , $|S| \leq |I_f| \leq 3$. Let X be the subgraph of N^* that contains an edge $fg \in E(N^*)$ if and only if f and g are inner faces of N and the edge



■ **Figure 5** The proof of Lemma 7.

in $E(f) \cap E(g)$ is bichromatic. We claim that X is a subgraph of \bar{T} . In order to show this, we need only argue that each edge uv of T in the interior of f is monochromatic. Consider any $uv \in E(N) \setminus E(f)$ where u is the T -parent of v . If $v \notin V(f)$ then, by definition, v has the same colour as u , so uv is monochromatic. The case where $v \in V(f)$ and $u \notin V(f)$ can not occur since $v \in V(f)$ implies that $P_T(v) \subseteq G_{i-1}$, but $u \notin V(G_{i-1})$. Similarly, the case in which $u \in V(f)$ and $v \in V(f)$ can not occur since this implies that $P_T(v) \subseteq G_{i-1}$, but $uv \notin E(G_{i-1})$.

Next we claim that all the elements of S are in a single connected component of X . If $|I_f| = 2$, then this follows immediately from Lemma 6. If $|I_f| = 3$, then let $\{a, b, c\} := I_f$ and consider a pair $g_1, g_2 \in S$ where (without loss of generality) g_1 contains a bichromatic edge of f with colours a and b and g_2 contains a bichromatic edge of f with colours b and c . By treating a and c as a single colour we may again apply Lemma 6 to conclude that g_1 and g_2 are in the same component of X .

Refer to Figure 5(a). Therefore X is a subgraph of \bar{T} that has a component containing all the elements of S . Therefore X contains \bar{T}_0 . By choice, \bar{T}_0 contains a path from f_i to each $g \in S$ and each of these paths is disjoint except for their shared starting location f_i .

Refer to Figure 5(b). Now, consider the embedded graph X_0 obtained as follows: For each $g \in V(\bar{T}_0)$, place a vertex on the center of each bichromatic edge of g and, if g is trichromatic, then place a vertex in the center of g . Next,

1. add an edge joining the center of each trichromatic triangle to each of the centers of its bichromatic edges; and
2. add an edge (embedded as a straight line segment) joining the centers of each pair of bichromatic edges that are on a common bichromatic face $g \in V(\bar{T}_0)$.

The graph X_0 is a tree of maximum-degree 3 that has $|I_f|$ leaves. (Each leaf in X_0 is the center of a bichromatic edge in $E(f)$). With the exception of these three leaves, every point in the embedding of X_0 is contained in the interior of f .

Refer to Figure 5(c). Now treat X_0 as a point set and consider the point set f' obtained by removing X_0 from the closure of f . Now f' has $|I_f|$ connected components and each vertex of f_i is in a different component. Each of the components of f' contains vertices of Y_j for exactly one $j \in I_f$; call this the *colour* of the component. Since no edge of T crosses X_0 , the colour of each vertex in f_i is equal to the colour the component of f' that contains it.

Finally, to see that f_0, \dots, f_i is good first observe that we need only be concerned with the at most three faces in $F(G_i) \setminus F(G_{i-1}) \setminus \{f_i\}$ and each of these shares a bichromatic edge with f_i . If g is a face in $F(G_i) \setminus F(G_{i-1}) \setminus \{f_i\}$ with $E(g) \cap E(f_i) = \{uv\}$ and uv is coloured with a and b , then $V(g) \cap Y_j = \emptyset$ for any $j \in \{0, \dots, i\} \setminus \{a, b, i\}$. This completes the proof. ◀

► **Theorem 8.** *There exists an $O(n)$ time algorithm that, given any n -vertex triangulation G and any rooted spanning tree T of G , produces a (G, T) -tripod decomposition \mathcal{Y} such that $\text{tw}(G/\mathcal{Y}) \leq 3$.*

Proof. Let v_0 be the root of T and let f_0 be a face of G incident to v_0 that contains an edge of T incident to v_0 . In a preprocessing step, we compute the cotree \bar{T} of (G, T) and construct a lowest common ancestor data structure for \bar{T} in $O(n)$ time that allows us to compute $\text{lca}_{\bar{T}}(f, g)$ for any two faces $f, g \in F(G)$ in $O(1)$ time.

After this preprocessing, we construct the good sequence f_0, \dots, f_r recursively. Conceptually, during any recursive invocation, the input is a near-triangulation N bounded by a cycle C in G whose vertices belong to at most three tripods computed in previous steps. Each vertex of G starts initially *unmarked* and we *mark* a vertex once we have placed it in a tripod. The precise input to a recursive invocation is defined as follows:

1. If C intersects three tripods then the input consists of the three inner faces g_1, g_2 , and g_3 of N that contain bichromatic edges of C . Lemma 7 characterizes the face f_i in terms of the minimum subtree \bar{T}_0 of \bar{T} that contains g_1, g_2 , and g_3 . Indeed, f_i is either the unique degree-3 node of \bar{T}_0 (if g_1, g_2 , and g_3 are all leaves of \bar{T}_0) or f_i is the unique node among g_1, g_2 , or g_3 that has degree 2. By Lemma 2 we can construct the model \bar{T}'_0 of \bar{T}_0 in constant time and find the node f_i .
2. If C intersects two tripods, then the input consists of two inner faces g_1, g_2 , of N with bichromatic edges of C on their boundary. In this case, we let $f_i := g_1$ or $f_i = g_2$, either choice satisfies our requirements.
3. If C intersects only one tripod, then the input consists of any inner face g_1 of N that contains an edge in $E(f)$. In this case $f_i := g_1$ satisfies our requirements.

Once we have found the Sperner triangle f_i , we can compute the tripod Y_i and mark its vertices by following the path in T from each vertex of f_i to its nearest marked ancestor in T . This takes $O(1 + |Y_i|)$ time. Once we have done this, we have also found the at most three bichromatic edges of G_i that are needed to perform the at most three recursive invocations on the near triangulations whose outer faces coincide with each of the new faces in $F(G_i) \setminus F(G_{i-1}) \setminus \{f_i\}$.

After setting f_0 , the initial recursive call falls into the third case above, so its input is any of the three inner faces that shares an edge with the outer face, f_0 . Each recursive invocation adds a new face f_i to the good face sequence f_0, \dots, f_r and takes $O(1 + |Y_i|)$ time. Since Y_0, \dots, Y_r is a partition of $V(G)$, the running time of this algorithm is therefore $\sum_{i=0}^r O(1 + |Y_i|) = O(n)$. ◀

5 Variations

In this section we show that there are $O(n)$ time algorithms for computing the decompositions in Theorem 1.b and Theorem 1.c. In the same way that Theorem 1.a follows from the tripod decomposition of Theorem 3, Theorem 1.b follows from a bipod decomposition given by Theorem 10 and Theorem 1.c follows from a monopod decomposition given by Theorem 11.

5.1 Bipod Decompositions

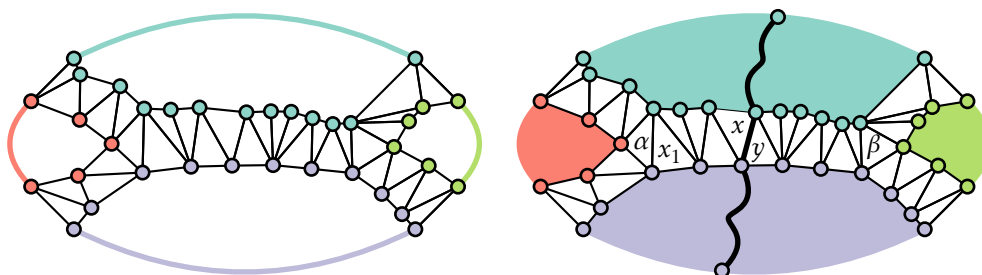
We begin with the decomposition in Theorem 1.b, which was communicated to us by Vida Dujmović, and has not appeared before. This decomposition is obtained by selecting a proper sequence $\mathcal{E} := e_0, \dots, e_k$ of distinct edges of G , which define a sequence of graphs $\mathcal{G}_{\mathcal{E}} := G_0, \dots, G_k$ where $G_i := \bigcup_{j=0}^i P_T(e_j)$ and a sequence of *bipods* $\mathcal{I}_{\mathcal{E}} := \Lambda_0, \dots, \Lambda_k$ where $\Lambda_i = V(G_i) \setminus V(G_{i-1})$. We call \mathcal{E} *good* if, for each $i \in \{0, \dots, k\}$ and each face $f \in F(G_i)$, $V(f)$ has a non-empty intersection with at most 4 bipods in $\Lambda_0, \dots, \Lambda_i$.

Exactly the same argument used in Section 3.1 to show that $G/\mathcal{Y}_{\mathcal{F}}$ is contained in a chordal graph of maximum clique size 4 also shows that if \mathcal{E} is a good edge sequence that produces a bipod partition $\mathcal{I}_{\mathcal{E}}$ of $V(G)$, then $G/\mathcal{I}_{\mathcal{E}}$ is contained in a chordal graph of maximum clique size 5, so $G/\mathcal{I}_{\mathcal{E}}$ has treewidth at most 4.

We now explain why a good edge sequence e_0, \dots, e_r exists.⁸ As before, we set f_0 to be any face of G such that $E(f_0)$ contains an edge of T incident to the root v_0 of T . The edge e_0 is any edge of $E(f_0) \setminus E(T)$. Next we take special care to ensure that G_i is biconnected for $i \geq 1$. In particular, if G_0 contains only two edges of f_0 , then we take e_1 to be the edge of f_0 that does not appear in G_0 . Otherwise, we choose e_1 using the general strategy for choosing e_i , described next.

Refer to Figure 6. Now we may assume that G_{i-1} is biconnected. To choose the edge e_i , we consider any face $f \in F(G_{i-1}) \setminus F(G)$. Inductively, $V(f)$ contains vertices from at most four bipods in $\Lambda_0, \dots, \Lambda_{i-1}$. Let $I_f := \{j \in \{0, \dots, i-1\} : \Lambda_j \cap V(f) \neq \emptyset\}$. If $|I_f| < 4$ then we can select e_i to be any edge in the interior of f . Therefore, we focus on the case $|I_f| = 4$. As before we colour vertices in the near triangulation N using colours in the set I_f ; we let S be the set of inner faces in N that contain a bichromatic edge in $E(f)$; and let \bar{T}_0 be the minimal subtree of \bar{T} that spans S . The same argument in the proof of Lemma 7 shows that every node of \bar{T}_0 is contained in f .

Claim 9, below, shows that \bar{T}_0 contains an edge xy such that each component of $\bar{T}_0 - xy$ contains at most two elements of S . It is straightforward to verify that, if we choose e_i to be the edge in $E(x) \cap E(y)$ then we obtain a graph G_i in which each of the two new faces containing vertices from Λ_i contains vertices from at most three bipods in $\{\Lambda_j : j \in I_f\}$, as required.



■ **Figure 6** Choosing the next e_i in a good edge sequence.

▷ **Claim 9.** \bar{T}_0 contains an edge xy such that each component of $\bar{T}_0 - xy$ contains at most two nodes of S .

Proof. Direct each edge xy of \bar{T}_0 in the direction \overrightarrow{xy} if the component of $\bar{T}_0 - xy$ that contains y contains three or more nodes of S . It is sufficient to show that this process leaves some edge xy of \bar{T}_0 undirected. Assume for the sake of contradiction that every edge of \bar{T}_0 is directed. Then some node x of \bar{T}_0 has only incoming edges. Certainly x does not have degree 1 in \bar{T}_0 .

If x has degree 2 in \bar{T}_0 then \bar{T}_0 contains two subtrees T_1 and T_2 that have only the node x in common and such that $|V(T_1) \cap S| \geq 3$ and $|V(T_2) \cap S| \geq 3$, which implies that $|S| \geq 3 + 3 - 1 > 4$, a contradiction.

⁸ The existence of this edge sequence is more easily proven using Sperner's Lemma, but we want a proof that lends itself to a linear time algorithm.

Suppose therefore that x has degree 3 in \bar{T}_0 . Each face in S contains an edge in $E(f)$, so each face in S has degree at most 2 in \bar{T}_0 . Therefore $x \notin S$. Therefore $\bar{T}_0 - x$ contains three components T_1, T_2, T_3 such that each pair of components contains at least 3 elements of S . But this implies that $|S| \geq (3 \times 3)/2 > 4$, a contradiction. \triangleleft

Algorithmically, using Lemma 2, we can construct the model \bar{T}'_0 of \bar{T}_0 in constant time given the set S . The model \bar{T}'_0 will also contain an edge $\alpha\beta$ such that each component of $\bar{T}'_0 - \alpha\beta$ contains at most two nodes in S . We claim that $E(\alpha)$ contains an edge that makes a suitable choice for e_i , and this edge can be found in constant time. Indeed, the edge $\alpha\beta$ in \bar{T}'_0 corresponds to a path $\alpha, x_1, \dots, x_k, \beta$ in \bar{T}_0 and the unique edge in $E(\alpha) \cap E(x_1)$ is a suitable choice for e_i .

The rest of the details of the algorithm are similar to those given in the proof of Theorem 8: Each subproblem is a near-triangulation N bounded by a cycle C and the input that defines the subproblem consists of the (at most four) faces $S \subseteq F(N)$ incident to bichromatic edges of C .⁹

► **Theorem 10.** *There exists an $O(n)$ time algorithm that, given any n -vertex triangulation G and any rooted spanning tree T of G , produces a (G, T) -bipod decomposition \mathcal{I} such that $\text{tw}(G/\mathcal{I}) \leq 4$.*

5.2 Monopod Decompositions

Finally we consider the decomposition described in Theorem 1.c. This decomposition is obtained from a tripod decomposition $\mathcal{Y} := Y_0, \dots, Y_r$, obtained by a sequence $\mathcal{F} := f_0, \dots, f_r$ of faces of G in the same manner described in Section 3.1. However in this setting, the sequence f_0, \dots, f_r is *good* if, for each $i \in \{0, \dots, r\}$ and each face f of $G_i := \bigcup_{j=0}^i Y_T(f_j)$, $V(f)$ contains vertices from at most 5 legs of tripods in Y_0, \dots, Y_i . Under these conditions, Ueckerdt et al. [18] are able to show that the *monopod decomposition* \mathcal{I} obtained by splitting each tripod Y_i into three upward paths yields a quotient graph G/\mathcal{I} of treewidth at most 6.

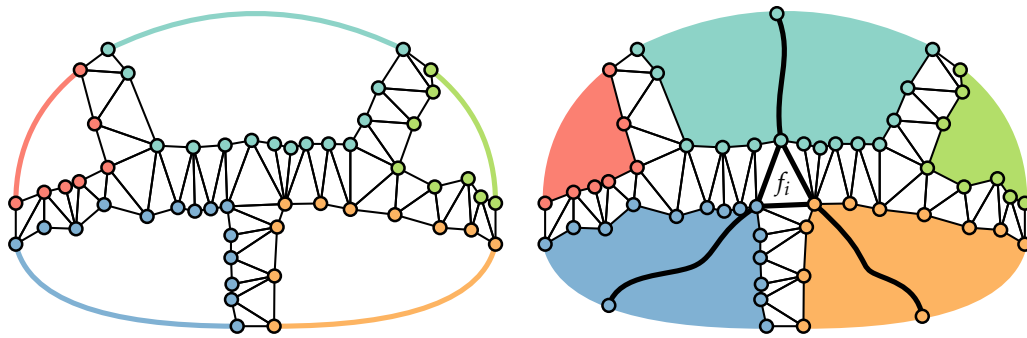
As before we focus on the extreme case when $V(f)$ contains vertices from exactly 5 legs of tripods. Refer to Figure 7. Following the same strategy used for the previous two decompositions, the set S in this case has size at most 5 and the face f_i corresponds to a node of \bar{T}_0 such that each component of $\bar{T}_0 - f_i$ contains at most 2 nodes in S . (This is always possible because $\lfloor 5/2 \rfloor = 2$.) Again, a suitable choice for f_i can be found in the model \bar{T}'_0 of \mathcal{T}_0 in constant time.

► **Theorem 11.** *There exists an $O(n)$ time algorithm that, given any n -vertex triangulation G and any rooted spanning tree T of G , produces a (G, T) -monopod decomposition \mathcal{I} such that $\text{tw}(G/\mathcal{I}) \leq 6$.*

References

- 1 Stephen Alstrup, Cyril Gavoille, Haim Kaplan, and Theis Rauhe. Nearest common ancestors: A survey and a new algorithm for a distributed environment. *Theory Comput. Syst.*, 37(3):441–456, 2004. doi:10.1007/s00224-004-1155-5.
- 2 Michael A. Bekos, Giordano Da Lozzo, Petr Hliněný, and Michael Kaufmann. Graph product structure for h -framed graphs. *CoRR*, abs/2204.11495, 2019. arXiv:2204.11495.

⁹ In the degenerate case where C has no bichromatic edges, the input is any face of N incident to an edge of C .



■ **Figure 7** The selection of a tripod by Ueckerdt et al. [18].

- 3 Michael A. Bender and Martin Farach-Colton. The LCA problem revisited. In Gaston H. Gonnet, Daniel Panario, and Alfredo Viola, editors, *LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings*, volume 1776 of *Lecture Notes in Computer Science*, pages 88–94. Springer, 2000. doi:10.1007/10719839_9.
- 4 Omer Berkman and Uzi Vishkin. Recursive star-tree parallel data structure. *SIAM J. Comput.*, 22(2):221–242, 1993. doi:10.1137/0222017.
- 5 Prosenjit Bose, Vida Dujmović, Mehrnoosh Javarsineh, and Pat Morin. Asymptotically optimal vertex ranking of planar graphs. *CoRR*, abs/2007.06455, 2020. arXiv:2007.06455.
- 6 Michal Debski, Stefan Felsner, Piotr Micek, and Felix Schröder. Improved bounds for centered colorings. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2212–2226. SIAM, 2020. doi:10.1137/1.9781611975994.136.
- 7 Reinhard Diestel. *Graph Theory, Fifth Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2017. doi:10.1007/978-3-662-53622-3.
- 8 Vida Dujmović, Gwenaël Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R. Wood. Planar graphs have bounded queue-number. *J. ACM*, 67(4):22:1–22:38, 2020. URL: <https://dl.acm.org/doi/10.1145/3385731>.
- 9 Vida Dujmovic, Pat Morin, and David R. Wood. Graph product structure for non-minor-closed classes. *CoRR*, abs/1907.05168, 2019. arXiv:1907.05168.
- 10 Vida Dujmović, Louis Esperet, Cyril Gavoille, Gwenaël Joret, Piotr Micek, and Pat Morin. Adjacency labelling for planar graphs (and beyond). *J. ACM*, 68(6):42:1–42:33, 2021. doi:10.1145/3477542.
- 11 Vida Dujmović, Louis Esperet, Gwenaël Joret, Bartosz Walczak, and David R. Wood. Planar graphs have bounded nonrepetitive chromatic number. *CoRR*, abs/1904.05269, 2019. arXiv:1904.05269.
- 12 Louis Esperet, Gwenaël Joret, and Pat Morin. Sparse universal graphs for planarity. *CoRR*, abs/2010.05779, 2020. arXiv:2010.05779.
- 13 Johannes Fischer and Volker Heun. Theoretical and practical improvements on the rmq-problem, with applications to LCA and LCE. In Moshe Lewenstein and Gabriel Valiente, editors, *Combinatorial Pattern Matching, 17th Annual Symposium, CPM 2006, Barcelona, Spain, July 5-7, 2006, Proceedings*, volume 4009 of *Lecture Notes in Computer Science*, pages 36–48. Springer, 2006. doi:10.1007/11780441_5.
- 14 Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16:47–56, 1974. doi:doi:10.1016/0095-8956(74)90094-X.
- 15 Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.*, 13(2):338–355, 1984. doi:10.1137/0213024.

19:14 An Optimal Algorithm for Product Structure

- 16 Pat Morin. A fast algorithm for the product structure of planar graphs. *Algorithmica*, 83(5):1544–1558, 2021. doi:10.1007/s00453-020-00793-5.
- 17 Baruch Schieber and Uzi Vishkin. On finding lowest common ancestors: Simplification and parallelization. *SIAM J. Comput.*, 17(6):1253–1262, 1988. doi:10.1137/0217079.
- 18 Torsten Ueckerdt, David R. Wood, and Wendy Yi. An improved planar graph product structure theorem. *CoRR*, abs/2108.00198, 2021. arXiv:2108.00198.