A Careful Study of Planar Product Structure Theorems in Relation to Queue Layouts

by

Kaya Gouin

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Computer Science Honours

Supervised by

Dr. Pat Morin

School of Computer Science

Carleton University

1 Introduction

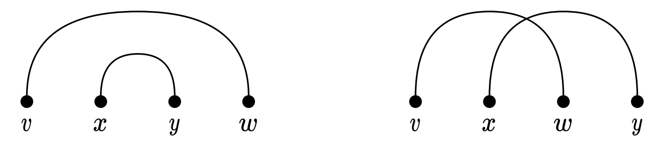
In the past decades, many conjectures essential to our comprehension of mathematical objects have been resolved. From these conjectures, a novel way in which to represent graphs—which lead to a proof of bounded queue-number, among others—emerged.

In 1992, Heath, Leighton, and Rosenberg conjectured that planar graphs have bounded queue-number [16]. Following a proof by [7] that planar graphs have bounded stack number, the conjecture came in an effort to classify the respective potentials of stacks and queues—fundamental data structures in the field of computer science—to represent any given graph. In the ensuing two decades, valuable work has been accomplished, yet no known technique could be used to achieve a proof of the conjecture.

It was in 2020 that Dujmović et al. proved that planar graphs indeed have bounded queue-number [12]. By discovering a way of decomposing planar graphs into simpler components, they were able to resolve Heath, Leighton, and Rosenberg's longstanding conjecture. The decomposition, captured in a class of theorems referred to as *planar product structure theorems*, is at the heart of this thesis and is discussed at length in subsequent sections.

Let us define the stack-number and queue-number of a planar graph G. Let $V(G)$ denote the vertex set of G, and $E(G)$ its edge set. Consider disjoint edges $vw, xy \in E(G)$, and a linear ordering $\preccurlyeq$ of $V(G)$ in which every vertex $u \in V(G)$ is comparable to any vertex $u' \in V(G) \setminus \{u\}$. Given the linearly ordered set $V(G) = \{v, w, x, y\}$, the disjoint edges vw and xy are said to *nest* if $v \prec x \prec y \prec w$, and are said to *cross* if $v \prec x \prec w \prec y$. It is worth noting that edges do not nest if they share a common endpoint. A set of pairwise non-crossing edges is labeled as a stack (with respect to $\preccurlyeq$), whereas a set of pairwise non-nested edges is labeled as a queue (with respect to $\preccurlyeq$). Figure 1 depicts the distinction between stack and queue.

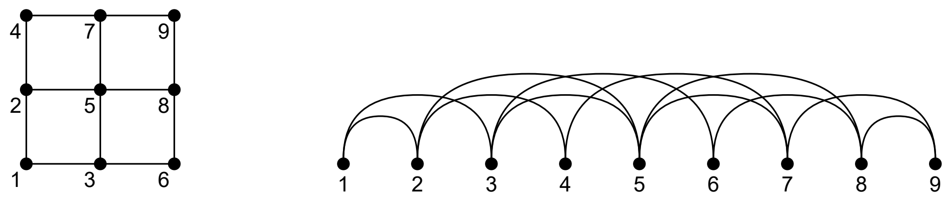
Figure 1: Visual representation of a stack (left) and a queue (right).



In a *k-stack layout*, vertices of a graph are arranged with respect to a linear ordering $\preccurlyeq$, and edges are partitioned into stacks. In a *k-queue layout*, vertices of a graph are arranged with respect to a linear ordering $\preccurlyeq$, and edges are partitioned into queues. The *stack-number* of a graph G, denoted by $\text{sn}(G)$, is the minimum integer k for which a k-stack layout in G exists. Similarly, the *queue-number* of a graph G, denoted by $\text{qn}(G)$, is the minimum integer k for which a k-queue layout in G exists. An example of a 1-queue layout of a grid graph is presented in Figure 2.

We now present a brief overview of the planar product structure theorems. The idea behind planar product

Figure 2: Visual representation of a 1-queue layout.



structure theorems is that we can factor a planar graph into three simpler components, one which is tree-like, one which is a path, and one which is the complete graph on at most three vertices. More formally, any planar graph G is contained in the *strong product*—an extension of the cartesian product—of three graphs H, P, and K_n, where H is a tree-like graph, P is a path, and K_n is the complete graph on n vertices, where $n \in \{1, 2, 3\}$ [12, 20].

Since their discovery in 2020, the planar product structure theorems have been used to efficiently resolve a number of longstanding open problems on any n-vertex planar graph G:

- computing an $O(1)$-queue layout of G [12];
- non-repetitively vertex-colouring G with $O(1)$ colours [11];
- assigning $(1 + o(1)) \log_2 n$-bit labels to the vertices of G so that one can determine from the labels of vertices v and w whether 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 [13];
- 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 [8]; 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 this thesis, we study the precise way in which a planar product structure decomposition is achieved. We begin with a review of relevant definitions in Section 2. Then, in Section 3, we detail the way in which the components of a planar product structure theorem are obtained. In Section 4, we relate the planar product structure theorem to the proof of bounded queue-number by Dujmović et al. Finally, Section 5 concludes by reiterating the significance of planar product structure theorems.

2 Tools

Throughout this thesis, we use standard graph theory terminology as used in the textbook by Diestel [9]. Graphs discussed in this work are simple and finite. For a given graph G, $V(G)$ and $E(G)$ denote the vertex and edge sets of G, respectively.

Strong Product. The *strong product* $G_1 \boxtimes G_2$ of two graphs G_1 and G_2 is a graph whose vertex set is the Cartesian product $V(G_1) \times V(G_2)$, in which the vertices (v, x) and (w, y) are adjacent if and only if $v = w$ and $xy \in E(G_2)$; $vw \in G_1$ and $x = y$; or $vw \in G_1$ and $xy \in E(G_2)$.

Partitions. A *partition* of a graph G is a set $\mathcal{P}$ of non-empty sets of vertices in G such that every vertex of G is in exactly one element of $\mathcal{P}$. Each element of $\mathcal{P}$ is called a *part*.

Quotients. Denoted by $G/\mathcal{P}$, the *quotient* of $\mathcal{P}$ for a given graph G is the graph with vertex set $\mathcal{P}$ where distinct parts $A, B \in \mathcal{P}$ are adjacent in $G/\mathcal{P}$ if and only if some vertex in A is adjacent in G to some vertex in B.

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 $\psi(vw) : (0, 1) \to \mathbb{R}^2$ whose endpoints are $\psi(v)$ and $\psi(w)$. We let $\psi(V(G)) \coloneqq \{\psi(v) : v \in V(G)\}$, $\psi(E(G)) \coloneqq \bigcup_{vw \in E(G)} \psi(vw)$, and $\psi(G) \coloneqq \psi(V(G)) \cup \psi(E(G))$. An embedding ψ of G is *plane* if $\psi(vw) \cap \psi(V(G)) = \emptyset$ and $\psi(vw) \cap \psi(xy) = \emptyset$ for every 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 the pair (G, ψ) is referred to as an *embedded graph*, and we shall not distinguish between a vertex v of G and the point $\psi(v)$, between an edge vw of G and the curve $\psi(vw)$, nor between G and the point set $\psi(G)$. Any cycle in an embedded graph defines a plane simple closed curve. For such a cycle C, $\mathbb{R}^2 \setminus C$ has two components: one which is bounded, and one which is unbounded. We shall 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. We let $F(G)$ denote the set of all faces of G. It is worth mentioning that every embedded graph contains exactly one face which is unbounded—the *outer face*.

Decompositions and Treewidth. For graphs H and G, an *H-decomposition* of G consists of a collection $(B_x \subseteq V(G) : x \in V(H))$ of subsets of $V(G)$, called *bags*, indexed by the vertices of H, and with the following properties: for every vertex v of G, the set $\{x \in V(H) : v \in B_x\}$ induces a non-empty connected subgraph of H; for every edge vw of G, there exists a vertex $x \in V(H)$ for which $v, w \in B_x$. The *width* of such an H-decomposition is

$\max\{|B_x| : x \in V(H)\} - 1$. A *tree-decomposition* is a T-decomposition for some tree T. The *treewidth* of a graph G is the minimum width of a tree-decomposition of G.

Layerings. A *layering* of a graph G is an ordered partition $(V_0, V_1, ...)$ of $V(G)$ such that for every edge $vw \in E(G)$, if $v \in V_i$ and $w \in V_j$, then $|i - j| \leq 1$.

If r is a vertex in a connected graph G and $V_i := \{v \in V(G) : \text{dist}_G(r, v) = i\}$ for all $i \geq 0$, then $(V_0, V_1, ...)$ is called a *BFS layering* of G *rooted* at r (where $\text{dist}_G(r, v)$ denotes the minimum number of edges between vertices r and v in G). Associated with a BFS layering is a *BFS spanning tree* T obtained by choosing, for each non-root vertex $v \in V_i$ with $i \geq 1$, a neighbour $w \in V_{i-1}$, and adding the edge vw to T. Then $\text{dist}_T(r, v) = \text{dist}_G(r, v)$ for every vertex v of G.

Layered Treewidth. The *layered treewidth* of a graph G is the minimum integer k such that G has a tree-decomposition $(B_x : x \in V(T))$ and a layering $(V_0, V_1, ...)$ such that $|B_x \cap V_i| \leq k$ for every bag B_x and layer V_i.

Layered Partitions. *Layered partitions* are partitions with bounded *layered width*, where the layered width of a partition $\mathcal{P}$ of a graph G is the minimum integer ℓ such that for some layering $(V_0, V_1, ...)$ of G, each part in $\mathcal{P}$ has at most ℓ vertices in each layer V_i.

3 Planar Product Structure Theorems

Planar product structure theorems were the key to achieving a proof of bounded queue-number for planar graphs. In this section, we consider the specifics of an algorithm for planar product structure decompositions.

The planar product structure theorem we shall consider in this work—the one which enabled Dujmović et al. to obtain their best bound for queue-number—involves a tree-like graph H, a path P, as well as the complete graph on three vertices K_3. It is formally defined as such:

Theorem 1. *For any planar graph G, there exists a planar graph H of treewidth at most 3 and a path P such that $G \subseteq H \boxtimes P \boxtimes K_3$* [12].

Of the elements of Theorem 1, the planar graph H of bounded treewidth constitutes the most central and complex piece. Our focus shall be on the precise way in which we efficiently compute H in such a way that it is of treewidth at most three.

3.1 Tripod Decompositions

The decomposition process, performed on G, leads to the planar graph H of bounded treewidth at most three, and may be viewed through a recursive lens: locate a specific tripod structure within a planar graph, then recurse on subgraphs bounded by it.

Central to the decomposition process is a well-known variation of Sperner's lemma:

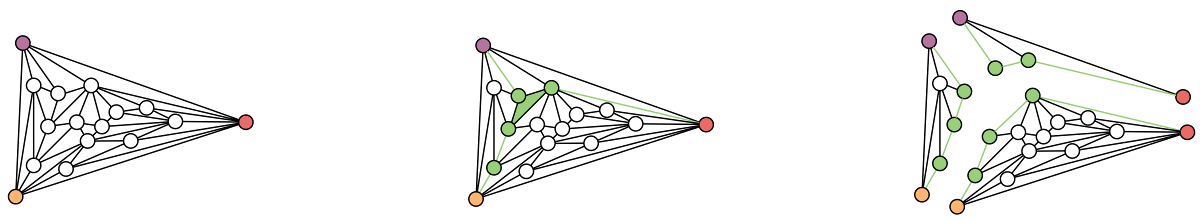
Lemma 1 (Sperner's lemma). *Let G be a near-triangulation whose vertices are coloured $1, 2, 3$, with the outer face $F = [P_1, P_2, P_3]$, where each vertex in P_i is coloured i. Then, G contains an internal face—a Sperner triangle—whose vertices are coloured $1, 2, 3$.*

Let G be a triangulation, and let T be a BFS spanning tree of G. For an internal face of G bounded by vertices coloured u, v, w, a (G, T)-tripod $\mathcal{Y}$ with crotch uvw is the vertex set of three disjoint, and each possibly empty, upward paths in T—the legs of $\mathcal{Y}$—whose lower endpoints are u, v, w. A (G, T)-tripod decomposition is then a partition of the vertices of G into (G, T)-tripods.

In the initial stage of a tripod decomposition on a given triangulation, vertices which make up the exterior face are uniquely coloured. Illustrated in Figure 3, these vertices represent the endpoints of three distinct hypothetical tripods. From there, we locate the Sperner triangle along with its three legs, each of which leads to one of the three initially coloured vertices of the external face. An empty leg incurs if a vertex of the Sperner triangle happens to be one of the three coloured vertices of the external face. The tripod—comprised of the Sperner triangle and associated leg structure—is uniquely coloured and divides the initial triangulation into discrete near-triangulations, each bounded by a different multi-coloured cycle.

In subsequent recursive steps, near-triangulations are similarly decomposed. These recursive branches terminate when empty near-triangulations are reached.

Figure 3: Initial stage of a tripod decomposition.

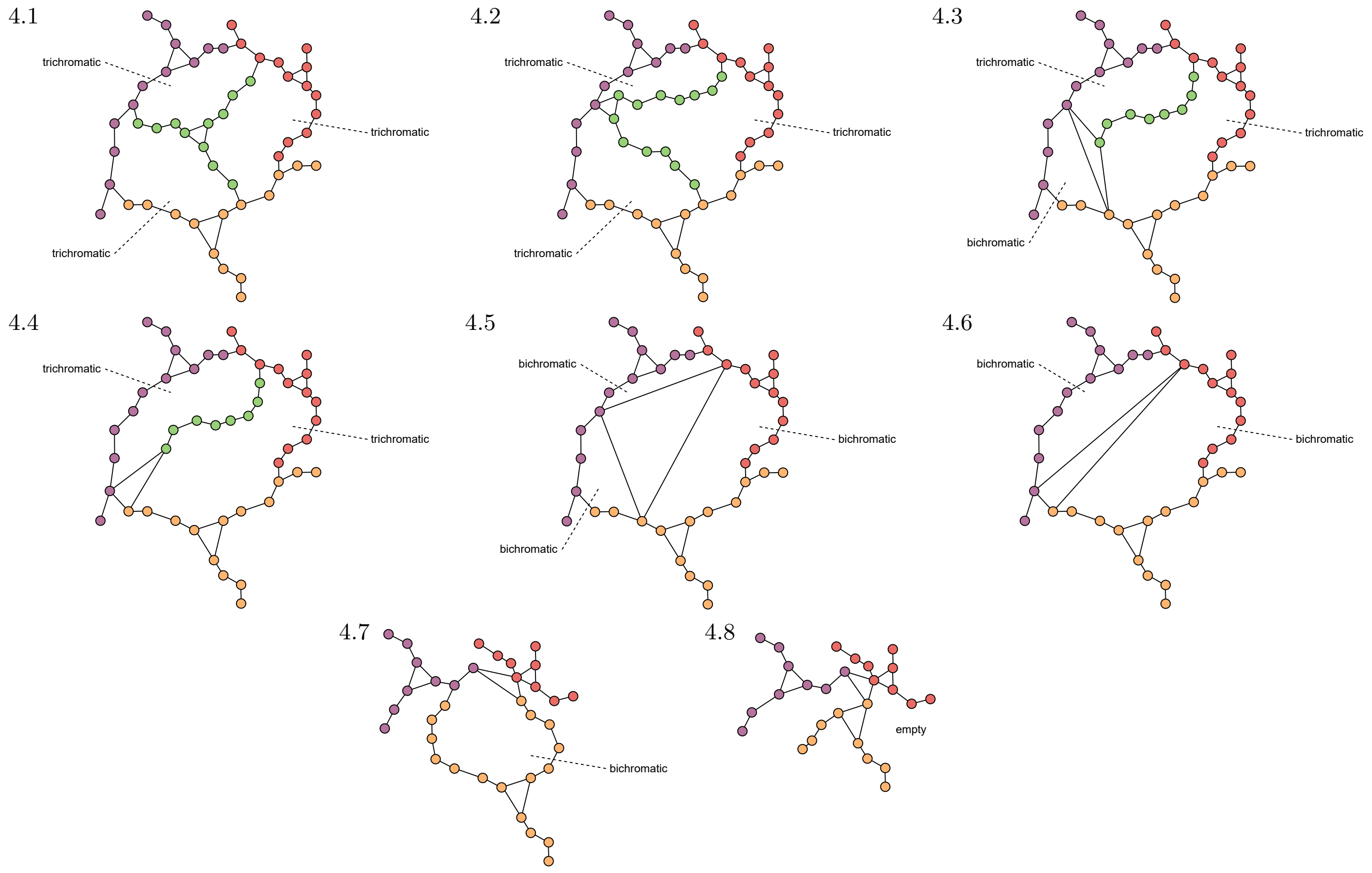


Although we tend to visualize a tripod as dividing a near- or perfect triangulation into three non-empty subgraphs, there are in fact a number of unique cases, all of which need careful consideration.

3.1.1 Trichromatic Tripod Decompositions

Trichromatic decompositions are characterized by a cycle of exactly three colours bounding a near-triangulation. Eight distinct cases—which may arise depending upon certain combinations of features—are illustrated in Figure 4 and described below.

Figure 4: Trichromatic tripod decompositions.



In the first case, the tripod has three non-empty legs, each with an upper endpoint connected to some differently coloured vertex on the boundary. The tripod splits the near-triangulation into three non-empty sections, each a near-triangulation bounded by a trichromatic cycle.

In the second case, the tripod has two non-empty legs, along with one empty leg. The non-empty legs each have an upper endpoint connected to some differently coloured vertex on the boundary. As well, exactly one of the vertices which make up the tripod crotch—the one which corresponds to the empty leg—is on the cycle defining the near-triangulation. As in the first case, the near-triangulation is separated into three sections by the tripod, each bounded by a trichromatic cycle.

In a third conceivable case, the tripod has one non-empty leg, along with two empty legs. The upper endpoint of the non-empty leg is connected to some coloured vertex on the boundary, and exactly two of the three vertices which constitute the crotch of the tripod—the ones which correspond to the empty legs—are located on the cycle defining the near-triangulation, with their shared edge excluded from the cycle. Here, the interior of the cycle is separated into three sections by the tripod; the two sections whose enclosing cycles include the upper endpoint of the non-empty leg have enclosing cycles which are trichromatic, whereas the section whose enclosing cycle includes the two vertices of the tripod crotch has an enclosing cycle which is bichromatic.

Resembling the third case, a fourth conceivable case is composed of a tripod with one non-empty leg, along with two empty legs. The upper endpoint of the non-empty leg is connected to some coloured vertex on the boundary, and exactly two of the three vertices which constitute the crotch of the tripod, along with their shared edge, are part of the cycle defining the near-triangulation. The interior is separated into two sections by the tripod, with enclosing cycles which are trichromatic, as they include the upper endpoint of the non-empty leg.

A fifth conceivable case involves a tripod with three empty legs, all of which lie on the cycle bounding the near-triangulation. Deprived of legs, the tripod may be characterized solely by its crotch, whose components constitute the vertices of the Sperner triangle. Here, none of the edges of the Sperner triangle are located on the cycle which bounds the near-triangulation. Therefore, the empty-legged tripod decomposes the near-triangulation into three components, each of which is bounded by a bichromatic cycle.

In a sixth case, the tripod consists of three empty legs, all of which lie on the cycle bounding the near-triangulation. Contrasted with the fifth case, exactly one of the three edges of the Sperner triangle is located on the cycle which bounds the interior. The tripod therefore decomposes the interior into two components, each of which is bounded by a bichromatic cycle.

In the seventh case, as in the fifth and the sixth, the tripod consists of three empty legs, all of which lie on the cycle bounding the near-triangulation. Contrasted with both prior cases, exactly two of the three edges of the Sperner triangle are situated on the cycle which bounds the near-triangulation. The tripod therefore decomposes the interior into one component, bounded by a bichromatic cycle.

Resembling the three prior cases, the eighth case involves a tripod which is comprised of three empty legs, all of which lie on the cycle bounding the interior. Every edge of the Sperner triangle is situated on the bounding cycle. Seeing that the tripod occupies all of the cycle's interior, there are no components on which to recurse.

3.1.2 Bichromatic Tripod Decompositions

In the case of bichromatic decompositions, a cycle of exactly two colours bounds a near-triangulation. Five distinct cases are illustrated in Figure 5 and described below.

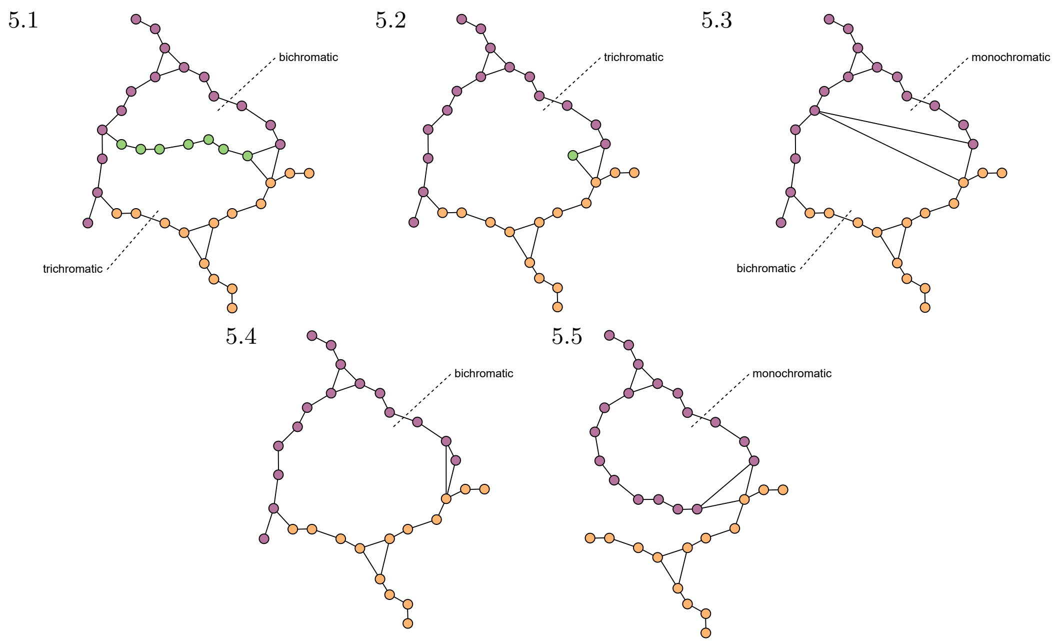


Figure 5: Bichromatic tripod decompositions.

In the first case, the tripod consists of one non-empty leg, along with two empty legs. The upper endpoint of the non-empty leg connects to some coloured vertex on the boundary. As well, exactly one of the three edges of the Sperner triangle—the one whose endpoints correspond to the two empty legs of the tripod—is located on the cycle bounding the interior. The tripod therefore divides the interior into two components, each of which is bounded by a bichromatic cycle.

In the second case, as in the first, the tripod consists of one non-empty leg, along with two empty legs. The upper endpoint of the non-empty leg connects to some coloured vertex on the boundary. Rather than reaching any vertex on the boundary, as in the first case, the coloured vertex which is reached by the upward path of the non-empty leg is one of the vertices of the Sperner triangle—one which corresponds to an empty leg. The second case is therefore a special case of the first, where the interior is decomposed not into two components bounded by bichromatic cycles, but into one component bounded by a trichromatic cycle.

In the third case, the tripod consists of three empty legs, all of which lie on the cycle bounding the near-triangulation. Arranged in such a way that exactly one of the edges of the Sperner triangle is located on the cycle which bounds the interior, the empty-legged tripod divides the near-triangulation into two components; one which is bounded by a bichromatic cycle, and one which is bounded by a monochromatic cycle.

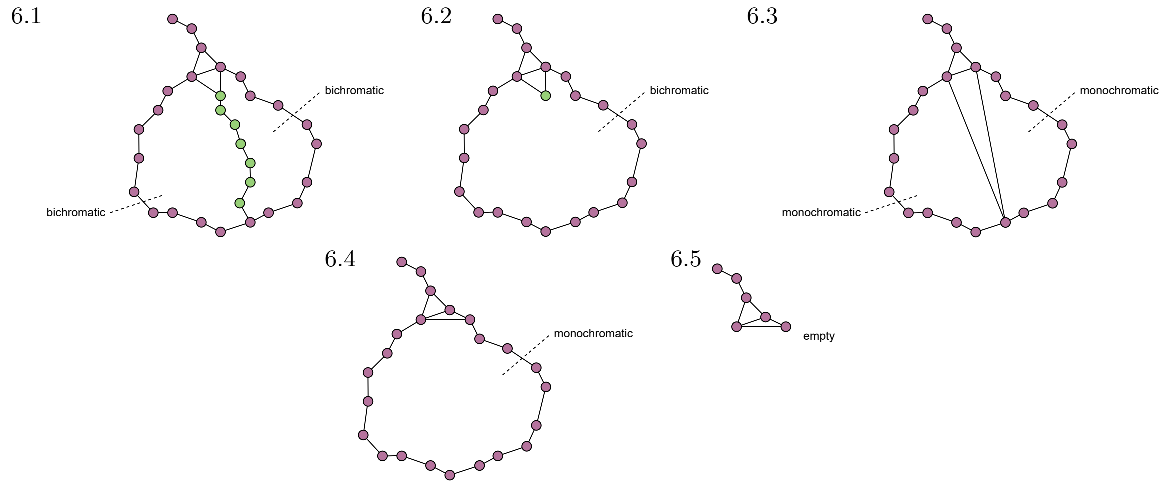
The fourth case, as in the third, consists of a tripod with three empty legs, all of which lie on the cycle bounding the near-triangulation. The tripod is positioned such that exactly two of the edges of the Sperner triangle are along the cycle which bounds the interior. Crucially, the edge of the Sperner triangle which is not along the bordering cycle has endpoints of differing colours. The interior is therefore decomposed into one component contained within a bichromatic cycle.

Closely resembling the fourth, the fifth case involves a tripod with three empty legs, all of which lie on the cycle bounding the interior. The tripod is positioned in such a way that exactly two of the edges of the Sperner triangle are along the bordering cycle. Contrasted with the fourth case, the edge of the Sperner triangle which is completely contained within the cycle has endpoints of the same colour. Therefore, the tripod decomposes the interior into one component bounded by a monochromatic cycle.

3.1.3 Monochromatic Tripod Decompositions

Monochromatic tripod decompositions are characterized by a cycle of exactly one colour bordering a near-triangulation. Five different cases are illustrated in Figure 6 and described below.

Figure 6: Monochromatic tripod decompositions.



The first case, which involves a tripod with one non-empty leg and two empty legs, has the upper endpoint of the non-empty leg connecting to some coloured vertex on the boundary, and is positioned such that exactly one of the edges of the Sperner triangle—the edge with endpoints which correspond to the lower endpoints of both empty legs—is along the bordering cycle. As a consequence, the tripod decomposes the near-triangulation into two components, each bounded by their respective bichromatic cycle.

The second case, which involves a tripod with one non-empty leg and two empty legs, has the upper endpoint of the non-empty leg connecting to some coloured vertex on the boundary, and is positioned such that exactly one of the edges of the Sperner triangle—the edge with endpoints which correspond to the lower endpoints of both empty legs—are along the bordering cycle. Rather than reaching any vertex on the boundary, as in the first case, the coloured vertex which is reached by the upward path of the non-empty leg is one of the vertices of the Sperner triangle—one which corresponds to an empty leg. The second case is therefore a special case of the first, where the interior is decomposed not into two components bounded by bichromatic cycles, but into one component bounded by a bichromatic cycle.

The third case involves a tripod with three empty legs, all of which lie on the cycle bounding the near-triangulation. Here, exactly one of the three edges of the Sperner triangle is located on the cycle bounding the interior component. The tripod therefore divides the near-triangulation into two components, each of which is bounded by a monochromatic cycle.

In a fourth case, the tripod consists of three empty legs, all of which lie on the cycle bounding the near-triangulation. Contrasted with the third case, exactly two of the three edges of the Sperner triangle are situated on the cycle which bounds the interior component. Here, the tripod divides the interior into one component, bounded by a monochromatic cycle.

Resembling the two prior cases, a fifth conceivable case involves a tripod composed of three empty legs, all of which lie on the cycle bounding the near-triangulation. Here, every edge of the Sperner triangle is situated on the cycle which bounds the interior component. Given that the tripod occupies all of the cycle's interior, there are no components on which to recurse.

3.2 Sperner Triangle Specifics

In Section 3.1, we described the high-level mechanics of the H-decomposition, omitting the specifics of the Sperner triangle identification process—which we consider next.

3.2.1 Preliminaries

We begin by defining some key concepts, some of which have been alluded in earlier sections.

Duals and Cotrees. The *dual* $G^\star$ of an embedded graph G is the graph with vertex set $V(G^\star) := F(G)$ and edge set $E(G^\star) := \{fg \in \binom{F(G)}{2} : E(f) \cap E(g) \neq \emptyset\}$. If T is a spanning tree of G, then the *cotree* $\overline{T}$ of (G, T) is the graph with vertex set $V(\overline{T}) := V(G^\star)$ and edge set $E(\overline{T}) := \{ab \in E(G^\star) : E(a) \cap E(b) \setminus E(T) \neq \emptyset\}$. If G is connected, then $\overline{T}$ is a spanning tree of $G^\star$.

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 $\overline{T}$ at any face $f_0 \in F(G)$ that contains an edge of T, then $\overline{T}$ is a binary tree, with the classification of left and right children determined by the embedding of G. It is worth noting that there is a small ambiguity when T contains two edges of f_0, in which case the unique child of f_0 in $\overline{T}$ can be treated as the left or right child of f_0.

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

Trees, Depth, Ancestors, and Descendants. Let T be a tree which is rooted at 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 $w_0, ..., w_r$ 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 vertex $w \in V(T)$ is the length of the path $P_T(w)$. The second vertex 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.

Least Common Ancestors. For any two vertices $v, w \in V(T)$, the *least common ancestor* $\mathrm{lca}_T(v, w)$ of v and w is the vertex a in $P_T(v) \cap P_T(w)$ having maximum T-depth. The *least 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 $\mathrm{lca}_T(v, w)$ for any two vertices $v, w \in V(T)$. A number of optimal solutions to this problem exists, which, after $O(n)$ time preprocessing using $O(n)$ space, can answer queries in $O(1)$ time [2, 3, 4, 14, 15, 19]. The simplest of these solutions is by Bender and Farach-Colton.

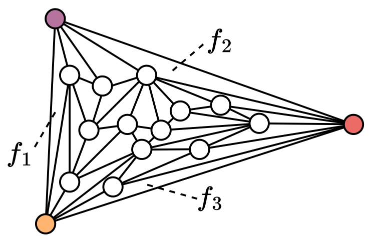
3.2.2 Sperner Triangle Identification

Given a triangulation G, a BFS spanning tree T rooted at one of the three vertices which make up the external face, and the spanning tree's associated cotree $\overline{T}$, we may find a Sperner triangle—as guaranteed by Lemma 1—by employing Bender and Farach-Colton's solution to the least common ancestor problem.

In their approach, Bender and Farach-Colton employ a solution for a different problem—the *Range Minimum Query (RMQ) Problem*—which they show is intimately linked to the least common ancestor problem. More formally, they lay out an optimal solution to a variant of the RMQ problem, known as the $\pm$1RMQ problem, then establish the reduction from $\pm$1RMQ to LCA [3]. While Bender and Farach-Colton's optimal solution to the least common ancestor problem is an invaluable aspect of the H-decomposition process, the specifics of their algorithm are not essential to the comprehension of such a decomposition. With this in mind, we turn our focus to the precise structures that we pass as argument to the optimal solution so as to find the Sperner triangle.

In the initial stage of a tripod decomposition on G, illustrated in Figure 7, three internal faces (f_1, f_2, f_3) are relevant to the identification of the Sperner triangle. These faces, analogous to vertices of the binary cotree $\overline{T}$, are those incident to the external face. Using Bender and Farach-Colton's optimal solution to the LCA problem, we find the least common ancestor of the two vertices which are farthest from the root of $\overline{T}$. The least common ancestor in $\overline{T}$, analogous to an internal face of G, corresponds to the Sperner triangle of G.

Figure 7: Relevant faces to the initial Sperner triangle identification.



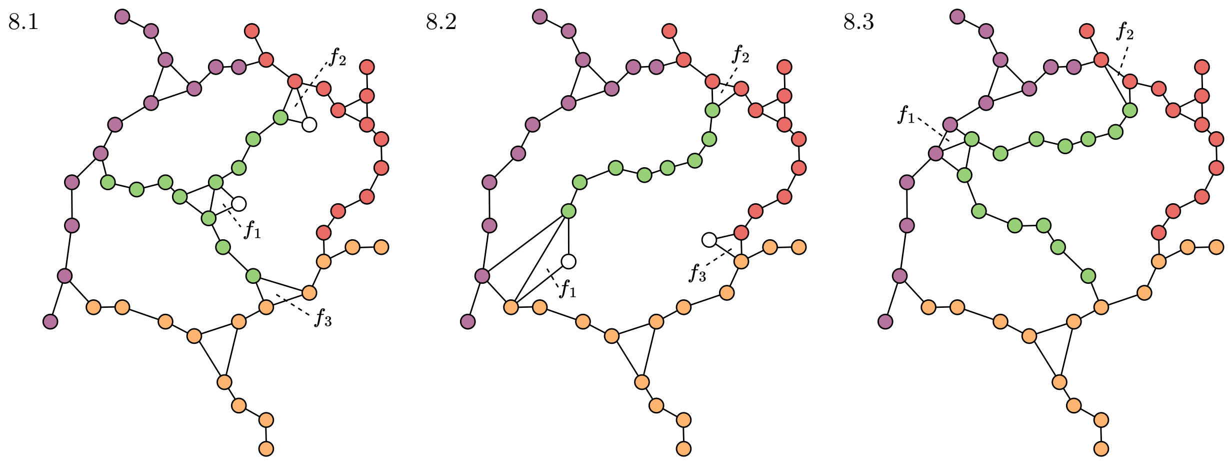
In subsequent steps, upward paths are identified in T, and the interior of G is decomposed into near-triangulations. The recursion begins, and a new Sperner triangle is found based upon the previous enclosing cycle and associated Sperner triangle. These steps are repeated at every level of recursion for which a non-empty near-triangulation exists.

As with tripod decompositions described in Section 3.1, there are a number of unique cases with respect to the faces relevant to the identification of the Sperner triangle, all of which need careful consideration.

3.2.2.1 Trichromatic Cycle Sperner Triangle Identification

In the case where the near-triangulation is enclosed by a trichromatic cycle, at most three faces are relevant to the identification of the Sperner triangle. As illustrated in Figure 8, three cases may arise.

Figure 8: Relevant faces to the Sperner triangle identification for a trichromatic cycle.



The first case, depicted in Figure 8.1, is one with three relevant faces. The first face, denoted as f_1, is the one adjacent to the Sperner triangle of the previous recursive step. The second, denoted as f_2, is the last face we see on an upward walk along one tripod leg before arriving at the defining cycle. The third, f_3, is the last face we see on an upward walk along the other tripod leg before arriving at the defining cycle. Pertinently, faces f_2 and f_3 may each share one or two edges with the defining cycle.

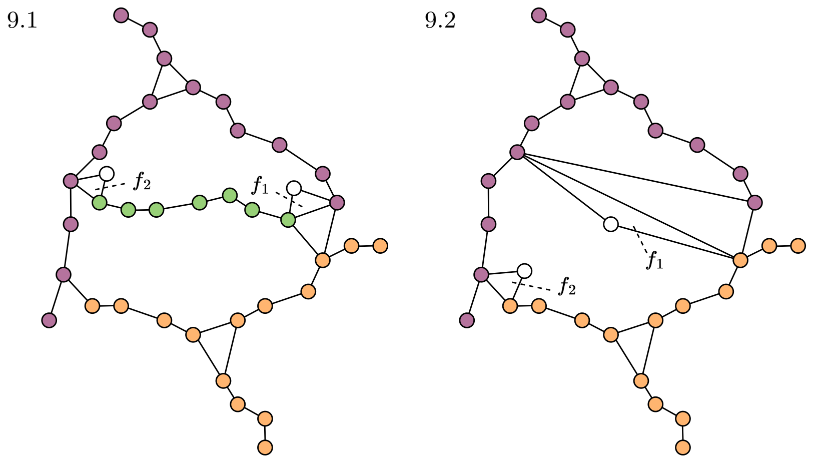
The second case, shown in Figure 8.2, is also one with three relevant faces. The first face, denoted as f_1, is the one adjacent to the Sperner triangle of the previous recursive step. The second, f_2, is the last face we see on an upward walk along one tripod leg before arriving at the defining cycle. Given the tripod's empty leg which falls on the cycle for the present recursive step, the third face, f_3, is chosen to be the input face to the previous recursive step—the face which falls within the defining cycle for the present recursive step.

The third case, shown in Figure 8.3, is one with two relevant faces. The first face, denoted as f_1, is the one adjacent to the Sperner triangle of the previous recursive step. The second, f_2, happens to be both the last face we see on an upward walk along one tripod leg before arriving at the defining cycle, and, given the tripod's empty leg which falls on the cycle for the present recursive step, the input face to the previous recursive step—the face which falls within the defining cycle for the present recursive step. The third case is therefore a special case of the second.

3.2.2.2 Bichromatic Cycle Sperner Triangle Identification

In the case where the near-triangulation is enclosed by a bichromatic cycle, exactly two faces are relevant to the identification of the Sperner triangle. As illustrated in Figure 9, two cases may arise.

Figure 9: Relevant faces to the Sperner triangle identification for a bichromatic cycle.



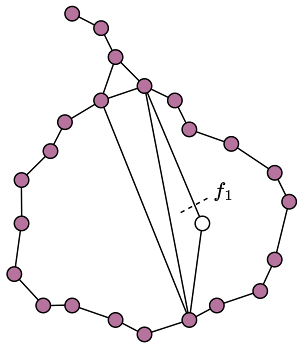
The first case is depicted in Figure 9.1. The first face, denoted as f_1, is the one adjacent to the Sperner triangle of the previous recursive step. The second, f_2, is the last face we see on an upward walk along the non-empty tripod leg before arriving at the defining cycle. Face f_2 may share one or two edges with the defining cycle.

The second case is depicted in Figure 9.2. The first face, denoted as f_1, is the one adjacent to the Sperner triangle of the previous recursive step. Given the tripod's empty legs which fall on the cycle for the present recursive step, the second face, f_2, is chosen to be the input face to the previous recursive step—the face which falls within the defining cycle for the present recursive step.

3.2.2.3 Monochromatic Cycle Sperner Triangle Identification

In the case where the near-triangulation is enclosed by a monochromatic cycle, as illustrated in Figure 10, only one face need be found for the identification of the Sperner triangle; f_1, the face adjacent to the Sperner triangle of the previous recursive step.

Figure 10: Relevant face to the Sperner triangle identification for a monochromatic cycle.



3.3 Assembling the pieces

From the decomposition described throughout Section 3, we construct the graph H by converting every respective Sperner triangle into a vertex, then converting every respective Sperner triangle leg into an edge with endpoints corresponding to the tripods on which it is incident. H is then a planar graph of treewidth at most three.

The dissection of a (triangulated) planar graph G into a graph H of treewidth at most three is a meticulous process. However, it is not one without reward. By Theorem 1, for any planar graph G, there exists a planar graph H of treewidth at most three and a path P such that $G \subseteq H \boxtimes P \boxtimes K_3$ [12]. With the described graph H and a path P taken to be of length equivalent to the height of the BFS spanning tree, we may convert any planar graph into three simple structures on which problem solving requires minimal time and effort.

4 Queue layouts via H-decompositions

The planar product structure theorems, one of which is carefully outlined in Section 3, enabled Dujmović et al. to resolve the longstanding conjecture of planar graphs having bounded queue-number. The next lemma is at the heart of their proof.

Lemma 2 ([12]). *For all graphs H and G, if H has a k-queue layout and G has an H-partition of layered width ℓ with respect to some layering $(V_0, V_1, ...)$ of G, then G has a $3\ell k + \lfloor \frac{3}{2}\ell \rfloor$-queue layout using vertex ordering $\overrightarrow{V_0}, \overrightarrow{V_1}, ...,$ where $\overrightarrow{V_i}$ is some ordering of V_i. In particular,*

$$\mathrm{qn}(G) \leq 3\ell \mathrm{qn}(H) + \lfloor \tfrac{3}{2}\ell \rfloor.$$

With reference to Lemma 2, the 'H-partition of layered width ℓ with respect to some layering $(V_0, V_1, ...)$ of G' is the structure which we have been calling the H-decomposition. Given the precise way in which we decompose G, its associated H-partition has layered width 3.

To arrive at their lowest constant in their proof of bounded queue-number, Dujmović et al. made use of the next lemma—a contribution made years before, which, until recently, could not be extended to graphs of unbounded treewidth.

Lemma 3 ([1, 17]). *Every planar graph with treewidth at most 3 has queue-number at most 5.*

Together, Lemma 2 and Lemma 3 imply that planar graphs have bounded queue-number, with an upper bound of $3 \cdot 3 \cdot 5 + \lfloor \frac{3}{2} \cdot 3 \rfloor = 49$.

5 Conclusion

Since their discovery in 2020, planar product structure theorems have served as invaluable instruments in the field of Graph Theory. With their applications surpassing those of queue-layouts and other problems in the class of planar graphs, carefully dissecting their components is of great value.

References

[1] Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. Queue layouts of planar 3-trees. In Therese C. Biedl and Andreas Kerren, eds., *Proc. 26th International Symposium on Graph Drawing and Network Visualization* (GD '18), vol. 11282 of *Lecture Notes in Comput. Sci.*, pages 213-226. Springer, 2018. doi:10.1007/978-3-030-04414-5_15

[2] 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.

[3] Michael A. Bender and Martín 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. 2007.06455.

[6] Prosenjit Bose, Pat Morin, and Saeed Odak. An optimal algorithm for product structure in planar graphs. *CoRR*, abs/2202.08870, 2022. 2202.08870.

[7] Jonathan F. Buss and Peter Shor. On the pagenumber of planar graphs. In *Proc. 16th ACM Symp. on Theory of Computing* (STOC '84), pages 98-100. ACM, 1984. doi:10.1145/800057.808670.

[8] 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.

[9] Reinhard Diestel. *Graph Theory, Fifth Edition,* volume 173 of *Graduate texts in mathematics*. Springer, 2017. doi:10.1007/978-3-662-53622-3.

[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. 1904.05269.

[12] 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.

[13] Louis Esperet, Gwenaël Joret, and Pat Morin. Sparse universal graphs for planarity. *CoRR*, abs/2010.05779, 2020. 2010.05779.

[14] 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.

[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.

[16] Lenwood S. Heath, F. Thomson Leighton, and Arnold L. Rosenberg. Comparing queues and stacks as mechanisms for laying out graphs. *SIAM J. Discrete Math.*, 5(3):398-412, 1992. doi:10.1137/0405031.

[17] Jan Kratochvíl and Michal Vaner. A note on planar partial 3-trees, 2012. arXiv:1210.8113.

[18] 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.

[19] 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.

[20] Torsten Ueckerdt, David R. Wood, and Wendy Yi. An improved planar graph product structure theorem. *CoRR*, abs/2108.00198, 2021. 2108.00198.