On Computing the Vertex Connectivity of 1-Plane Graphs

Therese Biedl
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Karthik Murali
School of Computer Science, Carleton University, Ottawa, Canada

Abstract

A graph is called *1-plane* if it has an embedding in the plane where each edge is crossed at most once by another edge. A crossing of a 1-plane graph is called an $\times$*-crossing* if there are no other edges connecting the endpoints of the crossing (apart from the crossing pair of edges). In this paper, we show how to compute the vertex connectivity of a 1-plane graph G without $\times$-crossings in linear time.

To do so, we show that for any two vertices u, v in a minimum separating set S, the distance between u and v in an auxiliary graph $\Lambda(G)$ (obtained by planarizing G and then inserting into each face a new vertex adjacent to all vertices of the face) is small. It hence suffices to search for a minimum separating set in various subgraphs Λ_i of $\Lambda(G)$ with small diameter. Since $\Lambda(G)$ is planar, the subgraphs Λ_i have small treewidth. Each minimum separating set S then gives rise to a partition of Λ_i into three vertex sets with special properties; such a partition can be found via Courcelle's theorem in linear time.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases 1-Planar Graph, Connectivity, Linear Time, Treewidth

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.23

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2212.06782` [2]

Funding *Therese Biedl*: Supported by NSERC.
Karthik Murali: Research done in part for the author's Master's thesis at University of Waterloo [26].

1 Introduction

The class of *planar graphs*, which are graphs that can be drawn on the plane without crossings, is fundamental to both graph theory and graph algorithms. Many problems can be more efficiently solved in planar graphs than in general graphs. However, real-world graphs, such as social networks and biological networks, are typically non-planar. But they are often *near-planar*, i.e., close to planar in some sense. One such graph class is the *1-planar graphs*, i.e., graphs that can be drawn on the plane such that each edge is crossed at most once. Introduced in 1965 [28], both structural and algorithmic properties of 1-planar graphs have been studied extensively, see [17, 21] for overviews.

In this paper, we look at the problem of vertex connectivity for 1-planar graphs. The problem of *vertex connectivity* is fundamental in graph theory: given a connected graph G, what is the size (denoted by $\kappa(G)$) of the smallest *separating set*, i.e., set of vertices whose removal makes G disconnected? Vertex connectivity has many applications, e.g. in network reliability and for measuring social cohesion.

50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).
Editors: Kousha Etessami, Uriel Feige, and Gabriele Puppis; Article No. 23; pp. 23:1–23:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Known Results. For an n-vertex m-edge graph G, one can test in linear (i.e. $O(m+n)$) time whether $\kappa(G) \geq 1$ with a graph traversal algorithm. In 1969, Kleitman [20] showed how to test $\kappa(G) \leq k$ in time $O(k^2nm)$. Subsequently, [29] and [18] presented linear-time algorithms to decide k-connectivity for $k = 2$ and $k = 3$ respectively. (Some errors in [18] were corrected in [15].) For $\kappa(G) = 4$, the first $O(n^2)$ algorithm was by Kanevsky and Ramachandran [19]. For $\kappa(G) \in O(1)$, the first $O(n^2)$ algorithm was by Nagamochi and Ibaraki [27]. For general k and m, the fastest running times are $\tilde{O}(n^\omega + nk^\omega)$ [25] and $\tilde{O}(kn^2)$ [16]. (Here, $\tilde{O}(g(n)) = O(g(n)\log^c n)$ for some constant c, and $\omega < 2.372$ is the matrix multiplication exponent.) Both algorithms are randomized and are correct with high probability. The fastest deterministic algorithm takes time $O(m \cdot (n + \min\{k^{5/2}, kn^{3/4}\}))$ [12].

Recent breakthroughs brought the run-time to test k-connectivity down to $\widehat{O}(m + \min\{n^{1.75}k^{1+k/2}, n^{1.9}k^{2.5}\})$ when $k < n^{1/8}$ [13], and to $\tilde{O}(m + nk^3)$ for randomized algorithms [11]. (Here $\widehat{O}(g(n)) = O(g(n)^{1+o(1)})$.) Very recently, Li et al. [24] showed that in fact $\widehat{O}(m)$ run-time can be achieved for randomized algorithms, independent of k. On the other hand, the problem of obtaining a deterministic linear time algorithm for deciding vertex connectivity is still open.

Vertex Connectivity in Planar Graphs. Any simple planar graph G has at most $3n - 6$ edges, hence has a vertex with at most five distinct neighbours, so $\kappa(G) \leq 5$. Since $\kappa(G) \leq 3$ can be tested in linear time, it only remains to test whether $\kappa(G) = 4$ or $\kappa(G) = 5$. In 1990, Laumond [23] gave a linear time algorithm to compute $\kappa(G)$ for maximal planar graphs. In 1999, Eppstein gave an algorithm to test vertex connectivity of all planar graphs in linear time [8]. His algorithm inspired the current work, and so we review it briefly here. Given a planar graph G with a fixed planar embedding (a *plane* graph), let the *radialization* be the plane graph obtained by adding a new vertex inside each face of G and connecting this *face vertex* to all vertices on the boundary of the face. The subgraph formed by the newly added edges is called the *radial graph* $R(G)$ [10]. The following is known.

▶ **Theorem 1** (attributed to Nishizeki in [8]). *Let S be a minimal separating set of a plane graph G. Then there is a cycle C in $R(G)$ with $V(C) \cap V(G) = S$ such that there are vertices of G inside and outside C.*

It hence suffices to find a shortest cycle C in $R(G)$ for which $V(C) \cap V(G)$ is a separating set of G. Since $\kappa(G) \leq 5$, this reduces to the problem of testing the existence of a bounded-length cycle (with some separation properties) in a planar graph. Eppstein solves this in linear time by modifying his planar subgraph isomorphism algorithm suitably [8].

Our Results. In this paper, we consider testing vertex connectivity of near-planar graphs, a topic that to our knowledge has not been studied before. We focus on 1-planar graphs that come with a fixed 1-planar embedding (*1-plane graphs*), since testing 1-planarity is NP-hard [14]. Since a simple 1-planar graph G has at most $4n-8$ edges [3], we have $\kappa(G) \leq 7$. For technical reasons we assume that G has no $\times$*-crossing*, i.e., a crossing without other edges among the endpoints of the crossing edges.

Let G be a 1-plane graph without ×-crossings. Inspired by Eppstein's approach, we define a planar auxiliary graph $\Lambda(G)$, and show that G has a separating set of size k if and only if $\Lambda(G)$ has a *co-separating triple* with size- and diameter-restrictions. (Roughly speaking, "co-separating triple" means that the vertices of $\Lambda(G)$ can be partitioned into three sets (A, X, B), such that X separates A, B in $\Lambda(G)$ while simultaneously $X \cap V(G)$ separates $A \cap V(G)$ and $B \cap V(G)$ in G. Detailed definitions are in Section 2.)

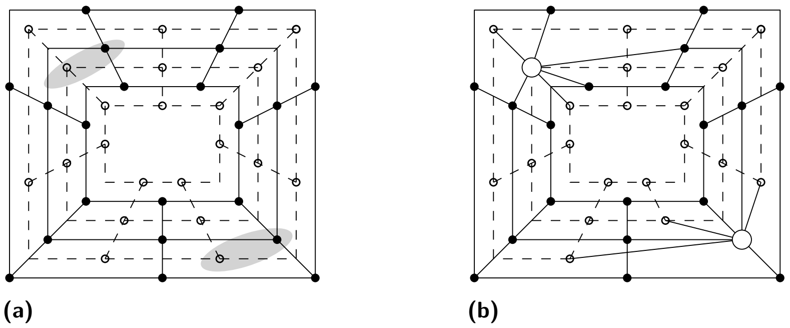

Figure 1 Theorem 2 does not hold for 1-plane graphs with ×-crossings.

▶ **Theorem 2.** *Let G be a 1-plane graph without $\times$-crossings. Then G has a separating set of size at most k if and only if $\Lambda(G)$ has a co-separating triple (A, X, B) where $|X \cap V(G)| \leq k$ and the subgraph of $\Lambda(G)$ induced by X has diameter at most $4k$.*

Let S be a minimum separating set of G and let (A, X, B) be the co-separating triple derived from S with this theorem. Since vertices of X are close to each other in $\Lambda(G)$, we can project (A, X, B) onto a subgraph $\Lambda_i \subseteq \Lambda(G)$ of diameter $O(|S|)$ to obtain a co-separating triple (A_i, X, B_i) of Λ_i. Conversely, we will show that with a suitable definition of subgraph Λ_i every co-separating triple (A_i, X, B_i) of Λ_i can be extended into a co-separating triple (A, X, B) of $\Lambda(G)$ from which we can obtain $X \cap V(G)$ as a separating set of G. Since Λ_i is planar and has diameter $O(|S|)$, it has treewidth $O(|S|)$. Using standard approaches for graphs of small treewidth, and by $\kappa(G) \leq 7$, we can search for (A_i, X, B_i) in linear time. Therefore we will obtain:

▶ **Theorem 3.** *The vertex connectivity of a 1-plane graph without $\times$-crossings can be computed in linear time.*

Limitations. We briefly discuss here the difficulty with ×-crossings. Figure 1(a) shows two copies of a graph that are interleaved to produce a 1-planar embedding such that each crossing is an ×-crossing. When these two graphs are fused together by identifying two pairs of vertices that are diametrically opposite to each other (shown by grey blobs in Figure 1(a)), we get the graph G in Figure 1(b). The two fused vertices form a separating set of G. Moreover, this is the only minimum separating set since both graphs in Figure 1(a) are 3-connected. This example can be extended (by adding more concentric layers and more vertices within each layer) to show that the distance between the two fused vertices can be made arbitrarily large even in $\Lambda(G)$. Thus Theorem 2 fails to hold for graphs with ×-crossings, and in consequence our techniques to test vertex connectivity cannot be extended to them.

Organization of the Paper. In Section 2, we lay out the preliminaries, defining co-separating triples and $\Lambda(G)$ for 1-plane graphs. In Section 3, we generalize Theorem 1 to the class of *full 1-plane graphs*, which are 1-plane graphs where the endpoints of each crossing induce the complete graph K_4. Using this result we prove Theorem 2 in Section 4, and turn it into a linear-time algorithm in Section 5. We summarize in Section 6.

2 Preliminaries

We assume familiarity with graphs, see e.g. [7]. All graphs in this paper are assumed to be connected and have no loops. A *separating set* of a graph G is a set S of vertices such that $G \setminus S$ is disconnected; we use the term *flap* for a connected component of $G \setminus S$. Set S *separates* two sets A, B if there is no path connecting A and B in $G \setminus S$. The *vertex connectivity* of G, denoted $\kappa(G)$, is the cardinality of a minimum separating set. For any vertex set A, an *A-vertex* is a vertex that belongs to A; we also write *G-vertex* for a vertex of $V(G)$.

A drawing of a graph in the plane is called *good* if edges are simple curves that intersect only if they properly cross or have a common endpoint, any two edges intersect at most once, and no three edges cross in a point. A *1-planar graph* is a graph that has a good drawing in the plane where each edge is crossed at most once; such a drawing is called a *1-planar drawing* and a graph with a given 1-planar drawing is called a *1-plane graph.* Throughout the paper, we assume that we are given a 1-plane graph G.

Let $\{(u,v),(w,x)\}$ be a crossing in G, i.e., edges (u,v) and (w,x) cross each other. The vertices $\{u,v,w,x\}$ are called *endpoints of the crossing*; these are four distinct vertices since the drawing is good. Two endpoints are called *consecutive* if they are not the two ends of (u,v) or (w,x). We distinguish six types of crossings by whether consecutive endpoints are adjacent (see Figure 2). As in [9], we call a crossing *full* if $\{u,v,w,x\}$ induces K_4, and *almost-full* if $\{u,v,w,x\}$ induces K_4 minus one edge.[1] We call it *bowtie* if $\{u,v,w,x\}$ induces a cycle, *arrow* if $\{u,v,w,x\}$ induces $K_{1,3}$ plus one edge, *chair* if $\{u,v,w,x\}$ induces a path of length three (the *length* of a path is its number of edges), and the crossing is an ×-crossing otherwise (no edges connecting consecutive endpoints of the crossing). For an almost-full crossing, there are exactly two consecutive non-adjacent endpoints; we call these the *wing tips* and the other two endpoints the *spine-vertices.* For an arrow crossing, depending on whether an endpoint is adjacent to zero or two of its consecutive endpoints, we call it the *tail* or *tip*; the other two endpoints are the *base vertices.*

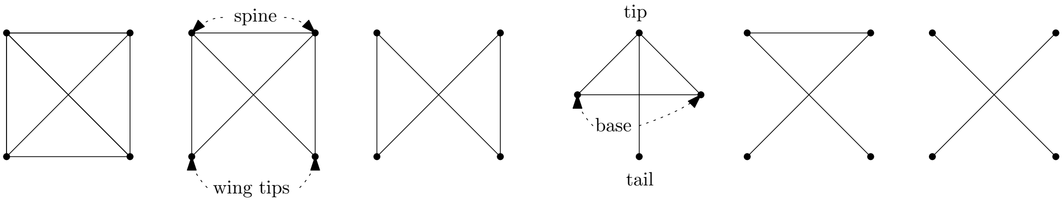

Figure 2 Types of crossings. From left to right: Full, almost-full, bowtie, arrow, chair and ×.

The *planarization* of G, denoted $G^\times$, is obtained by replacing any crossing $\{(u,v),(w,x)\}$ with a *dummy vertex*, i.e., remove the crossing edges and insert (at the point where the crossing used to be) a new vertex adjacent to all of u,v,w,x. The resulting drawing is *planar*, i.e., has no crossings. In a planar drawing Γ, a *face* is a maximal region of $\mathbb{R}^2 \setminus \Gamma$. Drawing Γ defines at each vertex v the *rotation* $\rho(v)$, which is the circular list of incident edges and faces, ordered clockwise.

[1] If G has parallel edges, then "$\{u,v,w,x\}$ induces graph H" is intended to mean "the underlying simple graph of the graph induced by $\{u,v,w,x\}$ is H".

Pre-processing. In Figure 2, we assumed that any edge (u, x) between consecutive endpoints of a crossing is actually drawn near that crossing, i.e., $G^\times$ contains a face incident to (u, x) and the dummy vertex of the crossing. (We call such a face a *kite face* and the edge a *kite edge*.) In general this may not be true. But if (u, x) exists elsewhere in the drawing, then we can duplicate it and insert it as a kite edge. This affects neither 1-planarity nor vertex connectivity nor crossing type, so assume from now on that at any crossing all edges among consecutive endpoints exist as kite edges. Since we never create a face of $G^\times$ that is bounded by two edges, graph G continues to have at most $4n - 8$ edges.

Radial Planarization. Recall that Eppstein [8] used the radialization to compute the vertex connectivity of a planar graph. We now generalize this concept to our 1-plane graph G as follows. The *radial planarization*, denoted $\Lambda(G)$, is obtained by first planarizing G to obtain $G^\times$, and then radializing $G^\times$ (see Figure 3). In other words, we add a *face vertex* f inside each face F of $G^\times$, and for every incidence of F with a vertex v we add an edge (v, f), drawn inside F and inserted in the drawing of $\Lambda(G)$ such that it bisects the occurrence of F in the rotation $\rho(v)$. (Repeated incidences of F with v give rise to parallel edges (f, v).) As in [10], we use the term *radial graph* for the subgraph $R(G)$ of $\Lambda(G)$ formed by the edges incident with face vertices. Note that $\Lambda(G)$ has three types of vertices: G-vertices, dummy vertices that replace crossings of G, and face vertices. For a cycle C in $\Lambda(G)$, we define the shortcut $\mathbf{V}_G(C) := V(G) \cap V(C)$ for the G-vertices of C.

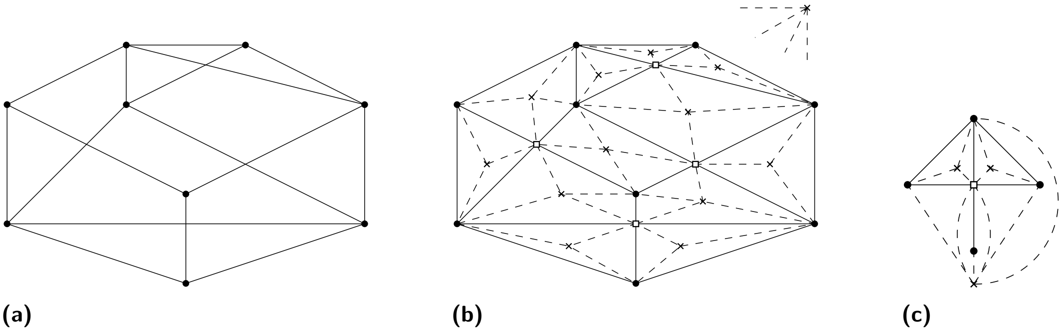

Figure 3 (a) A 1-plane graph G. (b) Its radial planarization $\Lambda(G)$. (c) $\Lambda(G)$ if G is an arrow crossing. Face vertices are crosses, dummy vertices are white squares, edges of $R(G)$ are dashed.

Co-separating Triple. We now clarify what it means to be separating in G and $\Lambda(G)$ simultaneously. We will actually give this definition for an arbitrary graph Λ that shares some vertices with G (since it will later be needed for graphs derived from $\Lambda(G)$).

▶ **Definition 4** (Co-separating triple). *Let Λ be a graph that shares some vertices with G. A partition of the vertices of Λ into three sets (A, X, B) is called a* co-separating triple of Λ *if it satisfies the following properties:*

1. *Each of A, X and B contains at least one G-vertex.*
2. *For any two vertices $a \in A$ and $b \in B$, there is no edge (a, b) in either $E(\Lambda)$ or $E(G)$.*

We say that a co-separating triple of Λ has *diameter d* if any two vertices of X have distance at most d in Λ. When $\Lambda = \Lambda(G)$, then all G-vertices belong to Λ, and since A and B both contain G-vertices the following is immediate:

▶ **Observation 5.** *If (A, X, B) is a co-separating triple of $\Lambda(G)$, then X is a separating set of $\Lambda(G)$ and $X \cap V(G)$ is a separating set of G.*

3 Full 1-Plane Graphs

In this section, we study *full 1-plane graphs*, i.e., 1-plane graphs where all crossings are full. In this case, we will find a co-separating triple (A, X, B) that has a special form (this will be needed in Section 4): Vertex set X contains no dummy vertices, forms a cycle C in $R(G)$, and A and B are exactly the vertices inside and outside this cycle in the planar drawing of $\Lambda(G)$. (In other words, we generalize Theorem 1.)

▶ **Theorem 6.** *Let G be a full 1-plane graph and S be a minimal separating set of G. Then there is a cycle C in $R(G)$ such that C does not visit dummy vertices, $\mathbf{V}_G(C) = S$, and there are vertices of G inside and outside C.*

Proof. The broad idea is to take a maximal path in $R(G)$ that alternates between S-vertices and face vertices with suitable properties, and then close it up into a cycle C of $R(G)$ that separates two vertices of G. Hence C automatically does not visit dummy vertices and $\mathbf{V}_G(C) = S$ since S is minimal. We explain the details now.

Call a face F of $G^\times$ a *transition face* if F is either incident to an edge between two S-vertices, or F is incident to vertices from different flaps of $G \setminus S$. The corresponding face vertex in $\Lambda(G)$ is called a *transition-face vertex*. By walking along the boundary of a transition face, one can easily show the following (see the full version [2]):

▷ Claim 7. Every transition face F contains at least two vertices of S or two incidences with the same vertex of S.

In consequence, any transition-face vertex has (in $R(G)$) two edges to vertices of S. Vice versa, any S-vertex v has (in $R(G)$) two transition-face neighbours, because in the planarization $G^\times$ we transition at v from edges leading to one flap of $G \setminus S$ to edges leading to another flap and back; this can happen only at transition faces. See the full version [2] for a proof of the following claim (and for the formal definition of "clockwise between").

▷ Claim 8. Let (v, t_1) and (v, t_2) be two edges of G such that $v \in S$ and t_1 and t_2 are in different flaps of $G \setminus S$. Then there exists a transition face incident to v that is clockwise between (v, t_1) and (v, t_2).

With this, it is obvious that we can find a simple cycle C that alternates between transition-face vertices and S-vertices, but we need to ensure that C has vertices of G inside and outside, and for this, we choose C more carefully. Formally, let v_1 be an arbitrary S-vertex. Let $P = v_1 \dots v_k$ be a simple path that alternates between transition-face vertices and S-vertices and that is maximal in the following sense: $v_k \in S$, and for any transition-face vertex v_{k+1} adjacent to v_k and any S-vertex v_{k+2} adjacent to v_{k+1}, at least one of v_{k+1}, v_{k+2} already belongs to P. Since $v_k \in S$ and S is minimal, v_k has neighbours t_1, t_2 in different flaps of $G \setminus S$. Applying Claim 8 twice gives a transition face F clockwise between (v_k, t_1) and (v_k, t_2), and a transition face F' clockwise between (v_k, t_2) and (v_k, t_1). If $k > 1$, then (up to renaming of t_1, t_2, F') we may assume that v_{k-1} is the face vertex of F'. Hence for $k > 1$, edge (v_k, f) (where f is the face vertex of F) is not on P, and the same holds vacuously also if $k = 1$. We have cases:

- In the first case, $f \in P$, say $f = v_i$ for some $1 \leq i \leq k-1$ (Figure 4(a)). Then $C := v_i v_{i+1} \dots v_k v_i$ is a cycle with t_1 and t_2 on opposite sides.
- In the second case $f \notin P$. By Claim 7, $R(G)$ contains at least two edges e, e' that connect f to S-vertices. Up to renaming we may assume that $e = (v_k, f)$. Consider extending P via e and e'. By maximality of P the result is non-simple, and by $f \notin P$ therefore $e' = (f, v_i)$ for some $1 \leq i \leq k$.

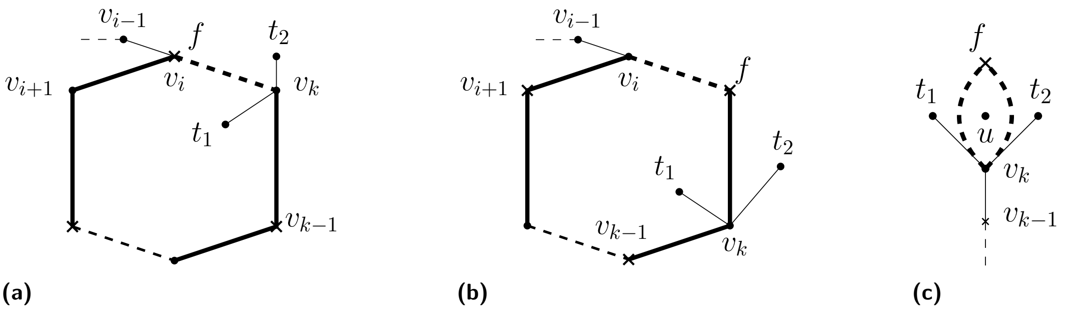

Figure 4 Constructing the cycle C (bold) in $R(G)$.

If $1 \le i \le k-1$ (Figure 4(b)), then define simple cycle $C := v_i v_{i+1} \dots v_k f v_i$ and observe that it has t_1 and t_2 on opposite sides. Otherwise ($i = k$) both edges e, e' connect v_k to f (Figure 4(c)), and we let C be the cycle consisting of e and e'. There can be parallel edges incident to f only because face F of $G^\times$ was incident to v_k repeatedly. Edges e, e' were then added to $\Lambda(G)$ in such a way that at least one other G-vertex on the boundary of F lies between those incidences on either side of cycle C.

So in either case we have constructed a cycle C in $R(G)$ with $\mathbf{V}_G(C) \subseteq S$ that does not visit dummy vertices and with two G-vertices (say a and b) inside and outside C. To show that $\mathbf{V}_G(C) = S$, it is sufficient (by minimality of S) to show that $\mathbf{V}_G(C)$ is separating in G. To see this, fix any path π from a to b in G and let π_Λ be the corresponding path in $\Lambda(G)$ obtained by replacing crossings by dummy vertices. Path π_Λ must intersect C, but it uses only G-vertices and dummy vertices while C uses only G-vertices and face vertices, so π_Λ intersects C in a vertex of G and $\mathbf{V}_G(C)$ hence separates a from b in G. ◀

4 1-Plane Graphs Without ×-Crossings

In this section, we prove Theorem 2: Minimal separating sets correspond to co-separating triples with small diameter. One direction is easy: If $\Lambda(G)$ has a co-separating triple (A, X, B) with $|V(G) \cap X| \le k$, then from Observation 5, G has a separating set of size at most k.

Proving the other direction is harder, and we first give an outline. For the rest of this section, fix a minimal separating set S, and two arbitrary flaps ϕ_1, ϕ_2 of $G \setminus S$. We first augment graph G to G_{aug} by adding more edges; this is done to reduce the types of crossings that can exist and thereby the number of cases. (Augmenting the graph is *only* used as a tool to prove Theorem 2; the vertex connectivity algorithm does not use it.) We then find a cycle C for $\Lambda(G_{\text{aug}})$ with vertices of G inside and outside C such that all vertices of S are in the neighbourhood of C. To do so we temporarily modify G_{aug} further to make it a full 1-plane graph G^+_{aug}, appeal to Theorem 6, and show that the resulting cycle can be used for C. By setting $X_{\text{aug}} = V(C) \cup S$, this cycle gives a co-separating triple $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$ of $\Lambda(G_{\text{aug}})$, and using C we can argue that the diameter of the graph induced by X_{aug} is small. Finally we undo the edge-additions to transfer the co-separating triple from $\Lambda(G_{\text{aug}})$ to $\Lambda(G)$.

Augmentation. We define the *augmentation* of G with respect to S, ϕ_1, ϕ_2 to be the graph $G_{\text{aug}} := G_{\text{aug}}(S, \phi_1, \phi_2)$ obtained as the result of the following iterative process:

For any two consecutive endpoints u, x of a crossing, if there is no kite edge (u, x) and it could be added without connecting flaps ϕ_1, ϕ_2, then add the kite edge, update the flaps ϕ_1 and ϕ_2 (because they may have grown by merging with other flaps), and repeat.

By construction S remains a separating set with flaps ϕ_1, ϕ_2 in G_{aug}, and it is minimal since adding edges cannot decrease connectivity. Also, one can easily show the following properties of crossings in G_{aug} (here not having ×-crossings is crucial), see Figure 5 for an illustration, and the full version [2] for details.

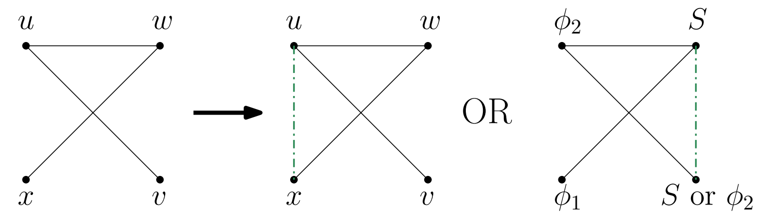

Figure 5 At a chair crossing, we can always add an edge to create an arrow-crossing.

▶ **Observation 9.** *The crossings of $G_{\text{aug}} = G_{\text{aug}}(S, \phi_1, \phi_2)$ have the following properties:*

1. *Any crossing is full, almost-full or an arrow crossing.*
2. *At any almost-full crossing, the spine vertices belong to S and the wing tips belong to ϕ_1 and ϕ_2.*
3. *At any arrow crossing, the tip belongs to S, the tail belongs to one of ϕ_1, ϕ_2, and the base vertices belong to the other of ϕ_1, ϕ_2.*

Extending Theorem 1? Note that we expanded Theorem 1 (for plane graphs) to Theorem 6 (for full 1-plane graphs), but as we illustrate now, it cannot be expanded to 1-plane graphs without ×-crossings. One example for this is the graph that exists of exactly one arrow crossing (see Figure 3(c)), because the tip is a separating set, but there is no 2-cycle in $R(G)$ that contains the tip. For an example with higher connectivity, consider Figure 6. The figure shows a 1-plane graph where each crossing is an arrow crossing or a chair crossing. The graph is 4-connected and a minimum separating set S is shown by vertices marked with white disks. One can verify that in the radial planarization of the graph, there is no 8-cycle in $R(G)$ that contains all vertices in S. (This example will also be used later as running example for our approach.)

Cycle C in $\Lambda(G_{\text{aug}})$. So we cannot hope to find a cycle C in $\Lambda(G_{\text{aug}})$ with G-vertices on both sides that goes *exactly* through S. But we can find a cycle C that is "adjacent" to all of S. To make this formal, define for a cycle C in $\Lambda(G_{\text{aug}})$ the set $\mathbf{V}_\times(C)$ to be the set of all vertices of C that are vertices of $G^\times_{\text{aug}}$, i.e., they are G-vertices or dummy vertices of $\Lambda(G_{\text{aug}})$.

▶ **Lemma 10.** *There is a cycle C in $\Lambda(G_{\text{aug}})$ that uses only edges of $R(G_{\text{aug}})$ and such that*

(1) *every vertex in $\mathbf{V}_\times(C)$ is either in S or is a dummy vertex adjacent to an S-vertex,*
(2) *every S-vertex is either in $\mathbf{V}_\times(C)$ or adjacent to a dummy vertex in $\mathbf{V}_\times(C)$,*
(3) *there are G-vertices that are not in S both inside and outside C, and*
(4) *S separates G-vertices inside C from G-vertices outside C in G_{aug}.*

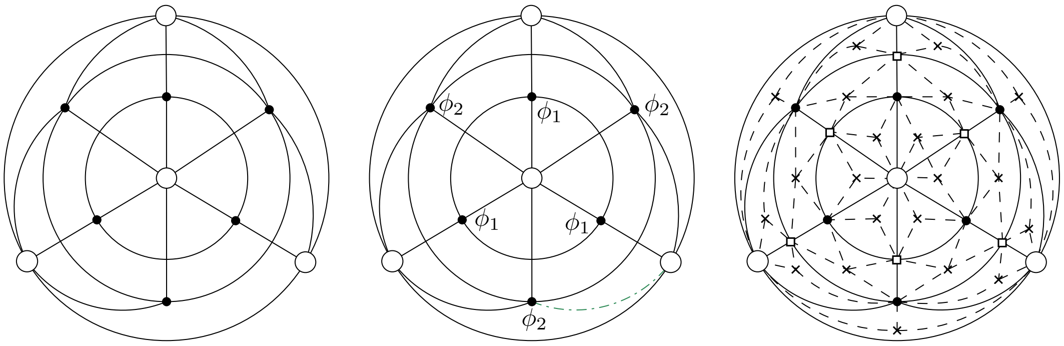

Figure 6 A 4-connected 1-plane graph G, its augmentation G_{aug} with respect to the minimum separating set S (white disks, the added edge is green/dot-dashed), and its radial planarization $\Lambda(G_{\text{aug}})$. Note that the unique chair crossing in G becomes an arrow crossing in G_{aug}. The radial graph does not have an 8-cycle passing through S.

Proof. As outlined, we first convert G_{aug} to a full 1-plane graph G^+_{aug} as follows (see Figure 7 for the abstract construction and Figure 8(a) for the running example): At every almost-full crossing and every arrow crossing, replace the crossing with a dummy vertex. At every arrow crossing, furthermore add a *base edge*, which connects the base vertices and is inserted so that it forms a full crossing. Since every crossing of G_{aug} is full, almost-full or arrow, all crossings of G^+_{aug} are full. We use $D := V(G^+_{\text{aug}}) \setminus V(G)$ for the new vertices and note that every vertex in D is adjacent to an S-vertex and corresponds to a dummy vertex in $\Lambda(G_{\text{aug}})$.

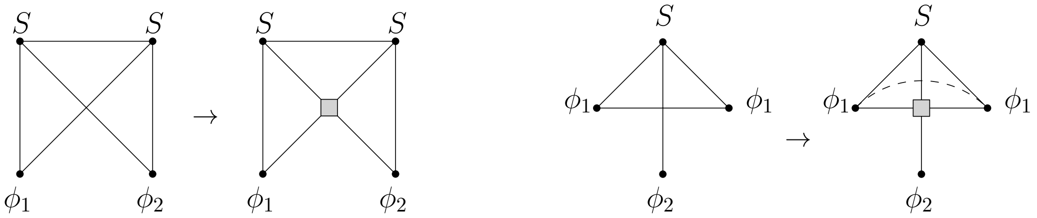

Figure 7 From G_{aug} to G^+_{aug}. Vertices in D are grey squares, the base edge is dashed.

Define $S^+ := S \cup D$ and observe that this is a separating set of G^+_{aug} since no edge of G^+_{aug} connects ϕ_1 with ϕ_2. Apply Theorem 6 to G^+_{aug} and a subset of S^+ that is minimally separating. This gives a cycle C in $R(G^+_{\text{aug}})$ such that $\mathbf{V}_{G^+_{\text{aug}}}(C) \subseteq S^+$, C does not visit dummy vertices of G^+_{aug}, and there are G^+_{aug}-vertices inside and outside C. See Figure 8(b). We claim that C satisfies all conditions, for which we first need to show that it actually is a cycle in $\Lambda(G_{\text{aug}})$. The only difference between $\Lambda(G_{\text{aug}})$ and $\Lambda(G^+_{\text{aug}})$ is at each base edge: Here $\Lambda(G^+_{\text{aug}})$ has an extra vertex c (the dummy vertex for the crossing created by the base edge) and the four incident face vertices, while $\Lambda(G_{\text{aug}})$ has only the two face vertices of the kite faces at the arrow crossing. But by Theorem 6 cycle C does not visit c, so C also is a cycle of $\Lambda(G_{\text{aug}})$.

To see that C satisfies (1), observe that $\mathbf{V}_{\times}(C) = \mathbf{V}_{G^+_{\text{aug}}}(C) \subseteq S^+ = S \cup D$, and every vertex of D is a dummy vertex of $\Lambda(G_{\text{aug}})$ that is adjacent to an S-vertex. Next we show (3). We know that there exist two vertices $a, b \in V(G^+_{\text{aug}})$ inside and outside C. If $a \in S \cup D$, then inspection of Figure 7 shows that a has a neighbour a' in $\phi_1 \cup \phi_2$ (hence $a' \in V(G)$ but $a' \notin S \cup D$). By $\mathbf{V}_{\times}(C) \subseteq S \cup D$ therefore a' is on the same side of C as a. Up to renaming hence $a \notin S \cup D$, and likewise $b \notin S \cup D$. This proves (3).

Before proving (2) and (4), we first show that the same vertices a and b are separated (in G_{aug}) by the set S' consisting of all S-vertices that are in $\mathbf{V}_\times(C)$ or adjacent to $\mathbf{V}_\times(C) \cap D$. To do so, pick an arbitrary path π from a to b in G_{aug}. We define a path π^+ in G^+_{aug} that corresponds to π as follows: use the same set of edges of π, except if π used an edge (r, s) that is part of an almost-full or an arrow crossing. At an almost-full crossing, we replace (r, s) by a path r-d-s where $d \in D$ is the dummy vertex. At an arrow crossing we have two cases. If r, s were the base vertices, then we replace (r, s) by the base edge. If r, s were tip and tail, then we replace (r, s) by r-d-s where $d \in D$ is the dummy vertex.

Since π^+ is a path from inside C to outside C in G_{aug}, it contains a vertex $w \in \mathbf{V}_\times(C) \subseteq S \cup D$. If $w \in S$, then define $t := w$. If $w \in D$, then near w path π^+ must have had the form r-w-s for some $(r, s) \in \pi$, due to our construction of π^+. Furthermore, (r, s) either belongs to an almost-full crossing, or to an arrow crossing with (r, s) connecting the tip and tail. For both types of crossings, one of r, s belongs to S, and we define t to be this vertex. So we have found a vertex $t \in S$ on π that is either on $\mathbf{V}_\times(C)$ or adjacent to a dummy vertex $d \in D \cap \mathbf{V}_\times(C)$. Therefore $t \in S'$ and so any path from a to b intersects S'. So $S' \subseteq S$ is a separating set of G_{aug}, hence by minimality $S' = S$, which proves (2). Also the G-vertices a and b are inside and outside C and separated by S, which proves (4). ◀

Notice that this lemma immediately implies a co-separating triple $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$ of G_{aug}: Fix such a cycle C, let $X_{\text{aug}} = V(C) \cup S$ and let A_{aug} and B_{aug} be the sets of vertices of $\Lambda(G_{\text{aug}}) \setminus X_{\text{aug}}$ inside and outside C respectively. See Figure 8(c). Clearly this is a partition, and by Lemma 10(3), both A_{aug} and B_{aug} have a G-vertex. As A_{aug} and B_{aug} are on opposite sides of cycle C, $V(C) \subseteq X_{\text{aug}}$ separates A_{aug} and B_{aug} in $\Lambda(G_{\text{aug}})$, and by Lemma 10(4), $S \subseteq X_{\text{aug}}$ separates $A_{\text{aug}} \cap V(G)$ and $B_{\text{aug}} \cap V(G)$ in G_{aug}. Therefore there can be no edge (a, b) with $a \in A_{\text{aug}}$ and $b \in B_{\text{aug}}$ in either $\Lambda(G_{\text{aug}})$ or G_{aug}.

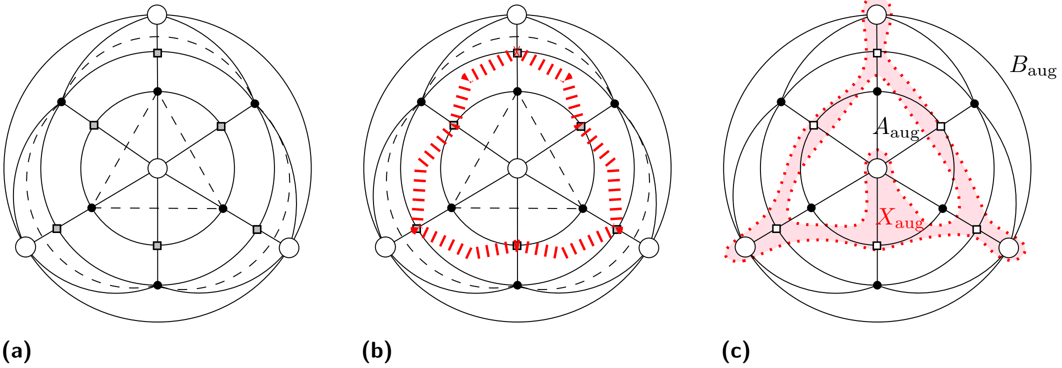

Figure 8 Finding a co-separating triple for the graph G from Figure 6. (a) Graph G^+_{aug}; vertices in D are grey squares and base edges are dashed. (b) Cycle C for the minimal cutting set D; we do not show the face-vertices. Note that every vertex of S is adjacent (in G_{aug}) to a dummy-vertex on C. (c) The resulting co-separating triples $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$.

Small Diameter. In order to prove Theorem 2, we first argue that the subgraph of $\Lambda(G_{\text{aug}})$ induced by X_{aug} has small diameter. Clearly the diameter of this graph is in $O(|C|)$ since all vertices of X_{aug} are on C or adjacent to it by Lemma 10(2). However, $|C|$ may not be in $O(|S|)$, which is why we need a more careful analysis to bound the length of a walk connecting two vertices of X_{aug}. Furthermore, to transfer the diameter-bound to $\Lambda(G)$ later, we need to exclude the edges that were added in G_{aug} from such walks. Write $E_{\text{aug}} := E(G_{\text{aug}}) \setminus E(G)$

(in Figure 6 the unique edge in E_{aug} is green/dash-dotted). Recall that edges in E_{aug} are kite edges, hence connect two vertices of G and have no crossing, therefore these edges also exist in $\Lambda(G_{\text{aug}})$.

▶ **Lemma 11.** *For any two vertices $u, v \in X_{aug}$, there is a walk W from u to v in $\Lambda(G_{\text{aug}})$ that has length at most $4|S|$ and does not use edges of E_{aug}.*

Proof. Since $u, v \in X_{\text{aug}}$, they are either in $\mathbf{V}_\times(C)$ (recall that this includes dummy vertices of $\Lambda(G_{\text{aug}})$ on C), or face vertices on C, or in S. In the latter two cases they are within distance one of some vertex in $\mathbf{V}_\times(C)$. So there exist vertices $u', v' \in \mathbf{V}_\times(C)$ that are within distance at most one of u and v, respectively. Enumerate one of the paths between u', v' along cycle C as $x_0, \dots, x_{2t}$ with $x_0 = u'$ and $x_{2t} = v'$. (Observe that the vertices in $\mathbf{V}_\times(C)$ are exactly the even-indexed ones since C uses edges of $R(G_{\text{aug}})$.) Each vertex x_{2i} for $i = 0, \dots, t$ is either in S (then set $s_{2i} := x_{2i}$), or by Lemma 10(1) it is a dummy vertex that has a neighbour $s_{2i} \in S$. Define π to be the walk

$$u, u'{=}x_0, s_0, x_0, x_1, x_2, s_2, x_2, x_3, \dots, x_{2i-1}, x_{2i}, s_{2i}, x_{2i}, x_{2i+1}, \dots, s_{2t}, x_{2t}{=}v', v,$$

i.e., it is the walk from u to v via C with detours at even-indexed vertices to reach an S-vertex. Observe that π has two properties: (1) At most three consecutive vertices in π do not belong to S, and at the ends there are at most two consecutive vertices not in S; (2) if y, z are two consecutive vertices of π that are different, then at least one of them is a face vertex or a dummy vertex, hence $(y, z) \notin E_{\text{aug}}$. We call a walk that satisfies (1) and (2) an *S-hopping walk.*

Let W be the shortest S-hopping walk from u to v. Observe that W can visit any S-vertex at most once, for otherwise we could find a shorter S-hopping walk by omitting the part between a repeated S-vertex. Since W contains at most three vertices between any two S-vertices, and at most two vertices not in S at the beginning and end, it has length at most $4|S|$. ◀

From G_{aug} to G. We now show how to transform $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$ into a co-separating triple (A, X, B) of $\Lambda(G)$ of small diameter. Recall that $G_{\text{aug}} = G \cup E_{\text{aug}}$, so $\Lambda(G_{\text{aug}})$ is obtained from $\Lambda(G)$ by inserting the edges E_{aug} and splitting any face vertex of a face that was divided by an edge in E_{aug}. We undo this in two parts. First, remove the edges of E_{aug} from $\Lambda(G_{\text{aug}})$. This does not affect the separation properties of $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$ since we only remove edges, and it maintains the diameter since the walks of Lemma 11 do not use E_{aug}. The second step is to identify face vertices that belong to the same face of G. Define sets A, B, X to be the same as $A_{\text{aug}}, B_{\text{aug}}, X_{\text{aug}}$ except that face vertices that were identified need to be replaced. To do so, observe that A_{aug} and B_{aug} are on opposite sides of C, and so no face vertex of A_{aug} can get identified with a face vertex of B_{aug} unless they both get identified with a face vertex of C. Thus the resulting face vertices come in three kinds: entirely composed of face vertices of A_{aug} (add these to A), entirely composed of face vertices of B_{aug} (add these to B), and containing a face vertex of C (add these to X).

Clearly A, B, X contain vertices of G since $A_{\text{aug}}, B_{\text{aug}}, X_{\text{aug}}$ did and we only identified face vertices. Assume for contradiction that (a, b) is an edge of G or $\Lambda(G)$ for some $a \in A$ and $b \in B$. Then (a, b) is not an edge in G_{aug} or $\Lambda(G_{\text{aug}})$. Thus at least one of a, b must be a face vertex that resulted from identifications. But with our choice of A and B then there was some edge (a', b') in $\Lambda(G_{\text{aug}})$ connecting vertices in A_{aug} and B_{aug}, a contradiction. Thus (A, X, B) is the desired co-separating triple and Theorem 2 holds.

5 Computing Vertex Connectivity in Linear Time

In this section, we show how to use Theorem 2 to obtain a linear time algorithm for finding the vertex connectivity of 1-plane graphs without ×-crossings. The crucial insight is that we only need to find the smallest k for which there is a co-separating triple (A, X, B) with $|X \cap V(G)| = k$. Moreover, the subgraph of $\Lambda(G)$ induced by X has small diameter. Therefore we can create some subgraphs $\Lambda_1, \Lambda_2, \dots$ of $\Lambda(G)$ that have small treewidth and X belongs to at least one subgraph. (We assume that the reader is familiar with treewidth and its implications for algorithms; see for example [4] or the full version [2].) We can search for a co-separating triple within these subgraphs using standard approaches for graphs of bounded treewidth, quite similar to the planar subgraph isomorphism algorithm by Eppstein [8].

The Graphs Λ_i. As a first step, we perform a breadth-first search in $\Lambda(G)$ starting at an arbitrary vertex (the *root*); let T be the resulting BFS-tree. For $j = 1, 2, \dots$ let V_j (the *jth layer*) be the vertices at distance $j-1$ from the root, and let d be the largest index where V_j is non-empty. Define $V_j := \emptyset$ for any index $j < 1$ or $j > d$. For any $a \leq b$, the notation $\Lambda[V_a \cup \dots \cup V_b]$ will be a shortcut for the subgraph of $\Lambda(G)$ induced by $V_a \cup \dots \cup V_b$.

Assume that we know the size $k \leq 7$ of the separating set that we seek (we will simply try all possibilities for k later). Define $w = 4k + 2 \leq 30$, so we know that any two vertices in X (of some putative co-separating triple (A, X, B) that satisfies the conclusion of Theorem 2) have distance at most $w - 2$ in $\Lambda(G)$. Hence X belongs to $V_{i+1} \cup \dots \cup V_{i+w-2}$ for some $i \in \{0, \dots, d-w+2\}$. Thus we will search for X within $\Lambda[V_{i+1} \cup \dots \cup V_{i+w-2}]$, but to guarantee that there are vertices representing A and B we also keep two extra layers above and below (i.e., layers $V_{i-1}, V_i, V_{i+w-1}, V_{i+w}$). Furthermore, we add an edge set U_{i-1} ('upper edges') within V_{i-1} and an edge set L_{i+w} ('lower edges') within V_{i+w} that have some special properties.

▷ Claim 12. For $i \in \{0, \dots, d-w+2\}$, there exist sets of edges U_{i-1} (connecting vertices of V_{i-1}) and L_{i+w} (connecting vertices of V_{i+w}) such that the following holds:

1. Two vertices $u, v \in V_{i-1}$ can be connected via edges of U_{i-1} if and only if there exists a path in $\Lambda[V_1 \cup \dots \cup V_{i-1}]$ that connects u and v.
2. Two vertices $u, v \in V_{i+w}$ can be connected via edges of L_{i+w} if and only if there exists a path in $\Lambda[V_{i+w} \cup \dots \cup V_d]$ that connects u and v.
3. The graph $\Lambda_i := \Lambda[V_{i-1} \cup \dots \cup V_{i+w}] \cup U_{i-1} \cup L_{i+w}$ is planar and has radius at most $w+2$.

Furthermore, $\sum_{j=0}^{d-w+2} |U_j| + |L_j| \in O(n)$ and we can compute these sets in time $O(n)$.

Proof. For U_{i-1}, this is easy. If $i = 0, 1$ then V_{i-1} is empty and $U_{-1} = \emptyset$ works. Otherwise, pick an arbitrary vertex r_{i-1} in V_{i-1}, and let U_{i-1} be the edges that connect r_{i-1} to all other vertices of V_{i-1}. In consequence, all vertices of V_{i-1} can be connected within U_{i-1}, but this is appropriate since they can all be connected within the BFS-tree T, using only layers $1, \dots, i-1$ of T. Graph $\Lambda[V_{i-1} \cup \dots \cup V_{i+w}] \cup U_{i-1}$ is planar, because it can be obtained from the planar graph $\Lambda[V_1 \cup \dots \cup V_{i+w}]$ by first contracting every vertex in layers $2, \dots, i-2$ into its parent in T (yielding one super-node at the root), and then contracting this super-node into r_{i-1}.

For L_{i+w}, existence likewise is easy (and was argued by Eppstein [8]): Simply contract any edge of $\Lambda[V_{i+w} \cup \dots \cup V_d]$ that has at least one endpoint not in V_{i+w}, and let L_{i+w} be the edges within V_{i+w} that remain at the end. However, it is not obvious how one could implement contraction in overall linear time; we give an alternate approach for this in the full version [2].

Since both U_{i-1} and L_{i+w} can be seen as obtained via contractions, graph Λ_i is planar. To prove the radius-bound, define r to be r_{i-1} if $i \geq 2$ and to be the root of T otherwise. Any vertex $v \in \Lambda_i$ has distance at most $w{+}2$ from r, because we can go upward from v in T at most $w{+}1$ times until we either reach a vertex v in V_{i-1} (which is $r{=}r_{i-1}$ or adjacent to it due to U_{i-1}), or $i \in \{0, 1\}$ and we reach the root of T (which is r). $\lhd$

An example of graph Λ_i is given in the full version [2].

Co-separating Triples in $\boldsymbol{\Lambda_i}$. We continue to assume that we know $k \leq 7$ (and hence $w = 4k + 2 \leq 30$). Crucial for the correctness of our search for a co-separating triple in $\Lambda(G)$ is that it suffices to search in Λ_i for all i.

▶ **Lemma 13.** *There exists a co-separating triple (A, X, B) of $\Lambda(G)$ with diameter k if and only if there exists an index $i \in \{0, \dots, w{-}d{+}2\}$ and a co-separating triple (A_i, X, B_i) of Λ_i with diameter k for which $X \subseteq \Lambda[V_{i+1} \cup \dots \cup V_{i+w-2}]$.*

Proof. Let (A, X, B) be a co-separating triple of $\Lambda(G)$. Since X has diameter at most $w - 2$, all vertices of X lie in the layers $i{+}1, \dots, i{+}w{-}2$ for some index i. Let A_i and B_i be subsets of A and B restricted to the vertices of Λ_i. We now show that (A_i, X, B_i) is a co-separating triple of Λ_i. Clearly these sets partition Λ_i. Condition 1 ('each set contains a G-vertex') clearly holds for X. To see that it holds for A_i, consider graph G in which $X \cap V(G)$ separates non-empty sets $A \cap V(G)$ and $B \cap V(G)$. Since G is connected, there exists an edge (a, x) with $a \in A \cap V(G)$ and $x \in X \cap V(G)$. This edge may or may not exist in $\Lambda(G)$, but if it does not then it got replaced by a-c-x with a dummy vertex c. So a has distance at most two from a vertex in $V_{i+1} \cup \dots \cup V_{i+w-2}$ and hence belongs to $V_{i-1} \cup \dots \cup V_{i+w}$, and so to Λ_i and to A_i. The argument is symmetric for B_i.

Now we argue Condition 2 ('no edges between A_i and B_i in Λ_i or G'). Fix two vertices $a \in A_i$ and $b \in B_i$. Since $A_i \subseteq A$ and $B_i \subseteq B$, there is no edge (a, b) in either $\Lambda(G)$ or G. So we are done unless (a, b) is an edge of U_{i-1} or L_{i+w}. Assume $(a, b) \in L_{i+w}$, the other case is similar. By Claim 12(2), there exists a path π in $\Lambda[V_{i+w} \cup \dots \cup V_d]$ connecting a and b. No vertex of π belongs to X, and so the vertices of π either all belong to A or all belong to B since (A, X, B) is co-separating. This contradicts $a \in A$ and $b \in B$.

We now prove the other direction. Let (A_i, X, B_i) be a co-separating triple of some Λ_i with $X \subseteq \Lambda[V_{i+1} \cup \dots \cup V_{i+w-2}]$. We define A and B as follows. Begin with all vertices in A_i and B_i, respectively; with this all vertices in Λ_i belong to one of A, X, B. Now consider any vertex v that does not belong to Λ_i, so either $v \in V_1 \cup \dots \cup V_{i-2}$ or $v \in V_{i+w+1} \cup \dots \cup V_d$. Assume the latter (the other case is similar), and let K be the component of $\Lambda[V_{i+w} \cup \dots \cup V_d]$ that contains v. By Claim 12(2), there exists a component K' of graph (V_{i+w}, L_{i+w}) that contains exactly the vertices of $K \cap V_{i+w}$. The vertices of K' must either all be in A_i, or they must all be in B_i, because they are in layer V_{i+w} (so not in X) and they are connected via L_{i+w}. Assign v (and actually all vertices of K) to A if $V(K') \subseteq A_i$, and to B otherwise.

We now show that partition (A, X, B) is a co-separating triple of $\Lambda(G)$. Clearly Condition 1 holds since already A_i and B_i contain G-vertices. To show Condition 2, consider two vertices $a \in A$ and $b \in B$ and assume for contradiction that there is an edge (a, b) in either G or $\Lambda(G)$. This means that at least one of a, b is not in $V_{i-1} \cup \dots \cup V_{i+w}$, else edge (a, b) would contradict that (A_i, X, B_i) was co-separating in Λ_i. Say $a \in V_{i+w+1} \cup \dots \cup V_d$, all other cases are similar. With this it is impossible that (a, b) is an edge of $\Lambda(G)$: Such an edge would put a, b into the same component of $\Lambda[V_{i+w} \cup \dots \cup V_d]$, but by construction of A and B we know that all vertices of such a component are put into the same set of A and B.

So (a, b) must be an edge of $G \setminus \Lambda(G)$, which means that it is crossed. Let c be the dummy vertex on (a, b). If $c \in A$ then (c, b) is an edge of $\Lambda(G)$ with endpoints in A and B, which we proved impossible already. Likewise $c \in B$ is impossible, so we must have $c \in X$. But then $c \in V_{i+1} \cup \cdots \cup V_{i+w-2}$, which puts its neighbour a into $V_i \cup \cdots \cup V_{i+w-1}$, a contradiction. ◀

Subroutine (To Find a Separating Set of Size k). We continue to assume that we know $k \leq 7$ (and hence $w = 4k + 2 \leq 30$). We also assume that edge-sets U_j and L_j have been computed already for all possible indices j, and that the edges of $\Lambda(G)$ have been split into $2d + 1$ sets $E_0, E_{0,1}, \ldots, E_{d-1,d}, E_d$ where E_j (for $j = 0, \ldots, d$) are all edges within layer V_j while $E_{j-1,j}$ (for $j = 1, \ldots, d$) are all edges connecting V_j to V_{j+1}. Perform the following for $i = 0, \ldots, d{-}w{+}2$:

1. Compute Λ_i. This takes time $O(|E(\Lambda_i)|)$ time: The vertices are $V_{i-1} \cup \cdots \cup V_{i+w}$, the edges are $U_{i-1} \cup E_{i-1,i} \cup E_i \cup \cdots \cup E_{i+w-1,i+w} \cup L_{i+w}$, and all these sets are pre-computed.
2. Since Λ_i is a planar graph with radius at most $w + 2$, it has treewidth $O(w)$ [8] and a corresponding tree decomposition $\mathcal{T}$ can be found in $O(|E(\Lambda_i)|)$ time. We may also assume that $\mathcal{T}$ has $O(|\Lambda_i|)$ bags.
3. We want to express Condition 2 of a co-separating triple as a condition in a single graph, and so define Λ_i^+ as follows: Begin with graph Λ_i, and add to it any edge (v, w) of G that is crossed (so is replaced in $\Lambda(G)$ by a path v-c-w via dummy vertex c) and for which v, w, c all belong to Λ_i.
4. Create a tree decomposition $\mathcal{T}^+$ of Λ_i^+ as follows. Begin with $\mathcal{T}$. For any bag Y and any dummy vertex $c \in Y$, add to Y all neighbours of c that belong to Λ_i. One can argue that this is a tree decomposition of Λ_i^+ of width $O(5w) = O(1)$, and can be computed in $O(|\mathcal{T}|) = O(|\Lambda_i|)$ time, see the full version [2].
5. Test whether Λ_i has a co-separating triple (A, X, B) for which X contains exactly k vertices of G and lies within $V_{i+1} \cup \cdots \cup V_{i+w-2}$. One can show (see the full version [2]) that this can be expressed in *monadic second-order logic*, using graph Λ_i^+ for defining adjacencies. By Courcelle's famous theorem [6], since Λ_i^+ has a tree decomposition of constant width, therefore the test can be done in $O(|\mathcal{T}^+|) = O(|\Lambda_i|)$ time.
6. If we find such a co-separating triple, then break (and output $X \cap V(G)$ as a separating set of size k), else try the next i.

The run-time for one index i is hence $O(|E(\Lambda_i)|)$. To bound the total run-time, we must bound $\sum_{i=0}^{d} |E(\Lambda_i)|$. Since each Λ_i uses $w+2$ consecutive layers, any edge of $\Lambda(G)$ belong to at most $w + 2$ sets in $E(\Lambda_0), \ldots, E(\Lambda_{d-w+1})$. Any edge in $U_0, \ldots, U_d, L_0, \ldots, L_d$ belongs to exactly one set in $E(\Lambda_0), \ldots, E(\Lambda_{d-w+1})$. Therefore $\sum_{i=0}^{d-w+1} |E(\Lambda_i)| \leq (w + 2)|E(\Lambda(G)| + \sum_{i=0}^{d-w+1}(|U_i| + |L_i|) \in O(w|E(\Lambda(G))|) + O(n)$, which by $w \in O(1)$ and $|E(\Lambda(G)| \in O(n)$ shows that the total run-time is linear.

The Final Algorithm. The algorithm for testing vertex-connectivity hence proceeds as follows. First pre-process G and duplicate edges to become kite edges where required. Then compute $\Lambda(G)$, the BFS-tree and the layers, and the edge-sets $E_j, E_{j,j+1}$, U_j and L_j for $j = 0, \ldots, d{+}2$. All this takes $O(n)$ time since $\Lambda(G)$ has $O(n)$ edges. For $k = 1, \ldots, 7$, run the sub-routine to test whether there exists a separating set of size k; this will necessarily find the minimum such set. Each such run takes time $O(n)$, and since there is a constant number of them the overall time is linear and Theorem 3 holds.

6 Outlook

In this paper, we showed that the vertex connectivity of a 1-plane graph G without ×-crossings can be computed in linear time. The main insight is that the distance (in an auxiliary graph) between any two vertices of a minimum separating set of G must be bounded. We close with some open questions. First, can we deal with ×-crossings?

▶ **Open problem 1.** *Can the vertex connectivity of an arbitrary 1-plane graph be computed in linear time?*

In our 'bad example' (Figure 1), *all* crossings were ×-crossings. As a first step towards Problem 1, could we at least compute the vertex connectivity in linear time if the number of ×-crossings is bounded by a constant?

Throughout the paper, we assumed that the input came with a fixed 1-planar embedding. We did this since testing 1-planarity is NP-hard [14]. However, it might be easier to test whether there exists a 1-planar embedding without ×-crossing; all the existing NP-hardness proofs of 1-planarity that we are aware of [14, 22, 1, 5] have ×-crossings in the 1-planar drawings.

▶ **Open problem 2.** *Is it NP-hard to test whether a given graph has a 1-planar drawing without ×-crossing?*

The crucial ingredient for our result was the structural property that vertices of a separating set are close in some sense. Are there similar structural properties for edge connectivity or bisections? Are there similar results for other classes of near-planar graphs?

References

1 Christopher Auer, Franz J Brandenburg, Andreas Gleißner, and Josef Reislhuber. 1-planarity of graphs with a rotation system. *Journal of Graph Algorithms and Applocations*, 19(1):67–86, 2015. doi:10.7155/jgaa.00347.

2 Therese Biedl and Karthik Murali. On computing the vertex connectivity of 1-plane graphs. *CoRR*, abs/2212.06782, 2022. doi:10.48550/arXiv.2212.06782.

3 Rainer Bodendiek, Heinz Schumacher, and Klaus Wagner. Bemerkungen zu einem Sechsfarbenproblem von G Ringel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 53:41–52, 1983. doi:10.1007/BF02941309.

4 Hans Bodlaender and Arie Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.

5 Sergio Cabello and Bojan Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM Journal on Computing*, 42(5):1803–1829, 2013. doi:10.1137/120872310.

6 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

7 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

8 David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999. doi:10.7155/jgaa.00014.

9 Igor Fabrici, Jochen Harant, Tomás Madaras, Samuel Mohr, Roman Soták, and Carol T Zamfirescu. Long cycles and spanning subgraphs of locally maximal 1-planar graphs. *Journal of Graph Theory*, 95(1):125–137, 2020. doi:10.1002/jgt.22542.

10 Fedor V Fomin and Dimitrios M Thilikos. New upper bounds on the decomposability of planar graphs. *Journal of Graph Theory*, 51(1):53–81, 2006. doi:10.1002/jgt.20121.

11 Sebastian Forster, Danupon Nanongkai, Liu Yang, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Computing and testing small connectivity in near-linear time and queries via fast local cut algorithms. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 2046–2065. SIAM, 2020. doi:10.1137/1.9781611975994.126.

12 Harold N Gabow. Using expander graphs to find vertex connectivity. *Journal of the ACM*, 53(5):800–844, 2006. doi:10.1145/1183907.1183912.

13 Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Deterministic graph cuts in subquadratic time: Sparse, balanced, and k-vertex. *CoRR*, abs/1910.07950, 2019. arXiv:1910.07950.

14 Alexander Grigoriev and Hans L Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007. doi:10.1007/s00453-007-0010-x.

15 Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

16 Monika R Henzinger, Satish Rao, and Harold N Gabow. Computing vertex connectivity: new bounds from old techniques. *Journal of Algorithms*, 34(2):222–250, 2000. doi:10.1006/jagm.1999.1055.

17 Seok-Hee Hong and Takeshi Tokuyama, editors. *Beyond Planar Graphs, Communications of NII Shonan Meetings*. Springer, 2020. doi:10.1007/978-981-15-6533-5.

18 John E Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973. doi:10.1137/0202012.

19 Arkady Kanevsky and Vijaya Ramachandran. Improved algorithms for graph four-connectivity. *Journal of Computer and System Sciences*, 42(3):288–306, 1991. doi:10.1016/0022-0000(91)90004-O.

20 Daniel Kleitman. Methods for investigating connectivity of large graphs. *IEEE Transactions on Circuit Theory*, 16(2):232–233, 1969. doi:10.1109/TCT.1969.1082941.

21 Stephen Kobourov, Giuseppe Liotta, and Fabrizio Montecchiani. An annotated bibliography on 1-planarity. *Computer Science Review*, 25:49–67, 2017. doi:10.1016/j.cosrev.2017.06.002.

22 Vladimir P Korzhik and Bojan Mohar. Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *Journal of Graph Theory*, 72(1):30–71, 2013. doi:10.1002/jgt.21630.

23 Jean-Paul Laumond. Connectivity of plane triangulations. *Information Processing Letters*, 34(2):87–96, 1990. doi:10.1016/0020-0190(90)90142-K.

24 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 317–329, 2021. doi:10.1145/3406325.3451088.

25 Nathan Linial, László Lovász, and Avi Wigderson. Rubber bands, convex embeddings and graph connectivity. *Combinatorica*, 8(1):91–102, 1988. doi:10.1007/BF02122557.

26 Karthik Murali. Testing vertex connectivity of bowtie 1-plane graphs. Master's thesis, David R. Cheriton School of Computer Science, University of Waterloo, 2022. Available at https://uwspace.uwaterloo.ca/.

27 Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(5&6):583–596, 1992. doi:10.1007/BF01758778.

28 Gerhard Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 29(1):107–117, 1965. doi:10.1007/BF02996313.

29 Robert E Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.