Three-in-a-Tree in Near Linear Time*

Kai-Yuan Lai† Hsueh-I Lu‡ Mikkel Thorup§

Abstract

The three-in-a-tree problem is to determine if a simple undirected graph contains an induced subgraph which is a tree connecting three given vertices. Based on a beautiful characterization that is proved in more than twenty pages, Chudnovsky and Seymour [*Combinatorica* 2010] gave the previously only known polynomial-time algorithm, running in $O(mn^2)$ time, to solve the three-in-a-tree problem on an n-vertex m-edge graph. Their three-in-a-tree algorithm has become a critical subroutine in several state-of-the-art graph recognition and detection algorithms.

In this paper we solve the three-in-a-tree problem in $\tilde{O}(m)$ time, leading to improved algorithms for recognizing perfect graphs and detecting thetas, pyramids, beetles, and odd and even holes. Our result is based on a new and more constructive characterization than that of Chudnovsky and Seymour. Our new characterization is stronger than the original, and our proof implies a new simpler proof for the original characterization. The improved characterization gains the first factor n in speed. The remaining improvement is based on dynamic graph algorithms.

1 Introduction

The graphs considered in this paper are all assumed to be undirected. Also, it is convenient to think of them as connected. Let G be such a graph with n vertices and m edges. An *induced* subgraph of G is a subgraph H that contains all edges from G between vertices in H. For the *three-in-a-tree* problem, we are given three specific terminals in G, and we want to decide if G has an induced tree T, that is, a tree T which is an induced subgraph of G, containing these terminals. Chudnovsky and Seymour [28] gave the formerly only known polynomial-time algorithm, running in $O(mn^2)$ time, for the three-in-a-tree problem. In this paper, we reduce the complexity of three-in-a-tree from $O(mn^2)$ to $O(m \log^2 n) = \tilde{O}(m)$ time.

Theorem 1.1. *It takes $O(m \log^2 n)$ time to solve the three-in-a-tree problem on an n-vertex m-edge simple graph.*

To prove Theorem 1.1, we first improve the running time to $O(mn)$ using a simpler algorithm with a simpler correctness proof than that of Chudnovsky and Seymour. The remaining improvement is done employing dynamic graph algorithms.

*An extended abstract to appear in *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing*, 2020.

†Department of Computer Science and Information Engineering, National Taiwan University. A preliminary version of the paper appeared as the master's thesis of this author [63]. baaldiablo3@gmail.com.

‡Corresponding author. Department of Computer Science and Information Engineering, National Taiwan University. Address: 1 Roosevelt Road, Section 4, Taipei 106, Taiwan, ROC. Research of this author is supported by MOST grants 107–2221–E–002–032–MY3 and 104–2221–E–002–044–MY3. hil@csie.ntu.edu.tw.

§Department of Computer Science, University of Copenhagen, Denmark. mikkel2thorup@gmail.com. Research of this author is supported by VILLUM Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC).

	best previously known results	our work
three-in-a-tree	$O(n^4)$ [28]	$\tilde{O}(n^2)$: Theorem 1.1
theta	$O(n^{11})$ [28]	$\tilde{O}(n^6)$: Theorem 1.2
pyramid	$O(n^9)$ [18]	$\tilde{O}(n^5)$: Theorem 1.3
perfect graph	$O(n^9)$ [18]	$O(n^8)$: Theorem 1.4
odd hole	$O(n^9)$ [26]	$O(n^8)$: Theorem 1.4
beetle	$O(n^{11})$ [15]	$\tilde{O}(n^6)$: Theorem 1.5
even hole	$O(n^{11})$ [15]	$O(n^9)$: Theorem 1.6

Figure 1: Comparing our work with the best previously known results for an n-vertex graph.

1.1 Significance of three-in-a-tree

The three-in-a-tree problem may seem like a toy problem, but it has proven to be of general importance because many difficult graph detection and recognition problems reduce to it. The reductions are often highly non-trivial and one-to-many, solving three-in-a-tree on multiple graph instances with different placements of the three terminals. With our near-linear three-in-a-tree algorithm and some improved reductions, we get the results summarized Figure 1. These results will be explained in more detail in Section 1.2.

Showcasing some of the connections, our improved three-in-a-tree algorithm leads to an improved algorithm to detect if a graph has an *odd hole*, that is, an induced cycle of odd length above three. This is via the recent odd-hole algorithm of Chudnovsky, Scott, Seymour, and Spirkl [26]. A highly nontrivial consequence of odd-hole algorithm is that we can use it to recognize if a graph G is *perfect*, that is, if the chromatic number of each induced subgraph H of G equals the clique number of H. The celebrated Strong Perfect Graph Theorem states that a graph is perfect if and only if neither the graph nor its complement has an odd hole. An odd-hole algorithm can therefore trivially test if a graph is perfect. The Strong Perfect Graph Theorem, implying the last reduction was a big challenge to mathematics, conjectured by Berge in 1960 [6, 7, 8] and proved by Chudnovsky, Robertson, Seymour, and Thomas [25], earned them the 2009 Fulkerson prize. Our improved three-in-a-tree algorithm improves the time to recognize if a graph is perfect from $O(n^9)$ to $O(n^8)$. While this is a modest polynomial improvement, the point is that three-in-a-tree is a central sub-problem on the path to solve many other problems.

The next obvious question is why three-in-a-tree? Couldn't we have found a more general subproblem to reduce to? The dream would be to get something like disjoint paths and graph minor theory where we detect a constant sized minor or detect if we have disjoint paths connecting of a constant number of terminal pairs (one path connecting each pair) in $O(n^2)$ time. This is using the algorithm of Kawarabayashi, Kobayashi, and Reed [61], improving the original cubic algorithm of Robertson and Seymour [71].

In light of the above grand achievements, it may seem unambitious for Chudnovsky and Seymour to work on three-in-a-tree as a general tool. The difference is that the above disjoint paths and minors are not necessarily induced subgraphs. Working with induced paths, many of the most basic problems become NP-hard. Obviously, we can decide if there is an induced path between two terminals, but Bienstock [9] has proven that it is NP-hard to decide two-in-a-cycle, that is, if two terminals are in an induced cycle. From this we easily get that it is NP-hard to decide three-in-a-path, that is if there is an induced path containing three given terminals. Both of these problems would be trivial if we could solve the induced disjoint path problem for just two terminal pairs. In connection with the even and odd holes and perfect graphs, Bienstock also proved that it is NP-hard

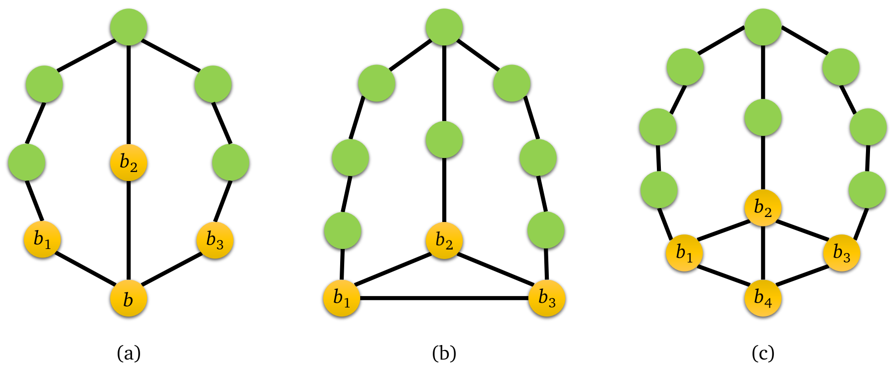


Figure 2: (a) Theta. (b) Pyramid. (c) Beetle.

to decide if there is an even (respectively, odd) hole containing a given terminal.

In light of these NP-hardness results it appears quite lucky that three-in-a-tree is tractable, and of sufficient generality that it can be used as a base for solving other graph detection and recognition problems nestled between NP-hard problems. In fact, three-in-a-tree has become such a dominant tool in graph detection that authors sometimes explained when they think it cannot be used [29, 78], e.g., Trotignon and Vušković [78] wrote "A very powerful tool for solving detection problems is the algorithm three-in-a-tree of Chudnovsky and Seymour [...] But as far as we can see, three-in-a-tree cannot be used to solve $\Pi_{H_{1|1}}$."

While proving that a problem is in P is the first big step in understanding the complexity, there has also been substantial prior work on improving the polynomial complexity for many of the problems considered in this paper. In the next subsection, we will explain in more detail how our near-linear three-in-a-tree algorithm together with some new reductions improve the complexity of different graph detection and recognition problems. In doing so we also hope to inspire more new applications of three-in-a-tree in efficient graph algorithms.

1.2 Implications

We are now going to describe the use of our three-in-a-tree algorithm to improve the complexity of several graph detection and recognition problems. The reader less familiar with structural graph theory may find it interesting to see how the route to solve the big problems takes us through several toy-like subproblems, starting from three-in-a-tree. Often we look for some simple configuration implying an easy answer. If the simple configuration is not present, then this tells us something about the structure of the graph that we can try to exploit.

We first define the big problems in context. A *hole* is an induced simple cycle with four or more vertices. A graph is *chordal* if and only if it has no hole. Rose, Tarjan, and Leuker [72] gave a linear-time algorithm for recognizing chordal graphs. A hole is *odd* (respectively, *even*) if it consists of an odd (respectively, even) number of vertices. G is *Berge* if G and its complement are both odd-hole-free. The celebrated Strong Perfect Graph Theorem, which was conjectured by Berge [6, 7, 8] and proved by Chudnovsky, Robertson, Seymour, and Thomas [25], states that G is Berge if and only if G is perfect, i.e., the chromatic number of each induced subgraph H of G equals the clique number of H.

The big problems considered here are the detection of odd and even holes, but related to this we are going to look for "thetas", "pyramids", and "beetles", as illustrated in Figure 2. These are different induced subdivisions where a *subdivision* of a graph is one where edges are replaced by paths of arbitrary length. A hole is thus an induced subdivision of a length-4 cycle, and a minimal three-in-a-tree is an induced subdivision of a star with two or three leaves that are all prespecified terminals.

The first problem Chudnovsky and Seymour [28] solved using their three-in-tree algorithm was to detect a *theta* which is any induced subdivision of $K_{2,3}$ [5]. Chudnovsky and Seymour are interested in thetas because they trivially imply an even hole. They developed the previously only known polynomial-time algorithm, running in $O(n^{11})$ time, for detecting thetas in G via solving the three-in-a-tree problem on $O(n^7)$ subgraphs of G. Thus, Theorem 1.1 reduces the time to $\tilde{O}(n^9)$. Moreover, we show in Lemma 6.1 that thetas in G can be detected via solving the three-in-a-tree problem on $O(mn^2)$ n-vertex graphs, leading to an $\tilde{O}(n^6)$-time algorithm as stated in Theorem 1.2.

Theorem 1.2. *It takes $O(mn^4 \log^2 n)$ time to detect thetas in an n-vertex m-edge graph.*

The next problem Chudnovsky and Seymour solved using their three-in-tree algorithm was to detect a *pyramid* which is an induced subgraph consisting of an apex vertex u and a triangle $v_1v_2v_3$ and three paths P_1, P_2, and P_3 such that P_i connects u to v_i and touch P_j, $j \neq i$, only in u, and such that at most one of P_1, P_2, and P_3 has only one edge. The point in a pyramid is that it must contain an odd hole. An $O(n^9)$-time algorithm for detecting pyramids was already contained in the perfect graph algorithm of Chudnovsky et al. [18, §2], but Chudnovsky and Seymour use their three-in-a-tree to give a more natural "less miraculous" algorithm for pyramid detection, but with a slower running time of $O(n^{10})$. With our faster three-in-a-tree algorithm, their more natural pyramid detection also becomes the faster algorithm with a running time of $\tilde{O}(n^8)$. Moreover, as for thetas, we improve the reductions to three-in-a-tree. We show (see Lemma 6.2) that pyramids in G can be detected via solving the three-in-a-tree problem on $O(mn)$ n-vertex graphs, leading to an $\tilde{O}(mn^3)$-time algorithm as stated in Theorem 1.3.

Theorem 1.3. *It takes $O(mn^3 \log^2 n)$ time to detect pyramids in an n-vertex m-edge graph.*

We now turn to odd holes and perfect graphs. Since a graph is perfect if and only if it and its complement are both odd-hole-free, an odd-hole algorithm implies a perfect graph algorithm, but not vice versa. Cornuéjols, Liu, and Vušković [39] gave a decomposition-based algorithm for recognizing perfect graphs that runs in $O(n^{18})$ time, which was reduced to $O(n^{15})$ time by Charbit, Habib, Trotignon, and Vušković [17]. The best previously known algorithm, due to Chudnovsky, Cornuéjols, Liu, Seymour, and Vušković [18], runs in $O(n^9)$ time. However, the tractability of detecting odd holes was open for decades [30, 33, 37, 59] until recently. Chudnovsky, Scott, Seymour, and Spirkl [26] announced an $O(n^9)$-time algorithm for detecting odd holes, which also implies a simpler $O(n^9)$-time algorithm for recognizing perfect graphs. An $O(n^9)$-time bottleneck of both of these perfect-graph recognition algorithms was the above mentioned algorithm for detecting pyramids [18, §2].

By Theorem 1.3, the pyramids can now be detected in $\tilde{O}(mn^3)$-time, but Chudnovsky et al.'s odd-hole algorithm has six more $O(n^9)$-time subroutines [26, §4]. By improving all these bottle-neck subroutines, we improve the detection time for odd holes to $O(m^2n^4)$, hence the recognition time for perfect graphs to $O(n^8)$.

Theorem 1.4. *(1) It takes $O(m^2n^4)$ time to detect odd holes in an n-vertex m-edge graph, and hence (2) it takes $O(n^8)$ time to recognize an n-vertex perfect graph.*

Even-hole-free graphs have been extensively studied [2, 34, 35, 40, 41, 52, 62, 74]. Vušković [83] gave a comprehensive survey. Conforti, Cornuéjols, Kapoor, and Vušković [32, 36] gave the first polynomial-time algorithm for detecting even holes, running in $O(n^{40})$ time. Chudnovsky, Kawarabayashi, and Seymour [20] reduced the time to $O(n^{31})$. A *prism* consists of two vertex-disjoint triangles together with three vertex-disjoint paths between the two triangles such that the union of every two of the three paths induces a cycle. Chudnovsky et al. [20] also observed that the time of detecting even holes can be further reduced to $O(n^{15})$ as long as detecting prisms is not too expensive, but this turned out to be NP-hard [68]. However, Chudnovsky and Kapadia [19] and Maffray and Trotignon [68, Algorithm 2] devised $O(n^{35})$-time and $O(n^5)$-time algorithms for detecting prisms in theta-free and pyramid-free graphs G, respectively. Later, da Silva and Vušković [41] improved the time of detecting even holes in G to $O(n^{19})$. The best formerly known algorithm, due to Chang and Lu [15], runs in $O(n^{11})$ time. One of its two $O(n^{11})$-time bottlenecks [15, Lemma 2.3] detects so-called beetles in G via solving the three-in-a-tree problem on $O(n^7)$ subgraphs of G. Theorem 1.1 reduces the time to $\tilde{O}(n^9)$. Moreover, we show in Lemma 6.3 that beetles can be detected via solving the three-in-a-tree problem on $O(m^2)$ n-vertex graphs, leading to an $\tilde{O}(n^6)$-time algorithm as stated in Theorem 1.5.

Theorem 1.5. *It takes $O(m^2 n^2 \log^2 n)$ time to detect beetles in an n-vertex m-edge graph.*

Combining our faster beetle-detection algorithm with our $O(n^9)$-time algorithm in §6.3, which is carefully improved from the other $O(n^{11})$-time bottleneck subroutine [15, Lemma 2.4], we reduce the time of detecting even holes to $O(n^9)$ as stated in Theorem 1.6.

Theorem 1.6. *It takes $O(m^2 n^5)$ time to detect even holes in an n-vertex m-edge graph.*

For other implications of Theorem 1.1, Lévêque, Lin, Maffray, and Trotignon gave $O(n^{13})$-time and $O(n^{14})$-time algorithms for certain properties Π_{B_4} and Π_{B_6}, respectively [66, Theorems 3.1 and 3.2]. By Theorem 1.1 and the technique of §6.2.1, the time can be reduced by a $\Theta(n^5/\log^2 n)$ factor. Theorem 1.1 also improves the algorithms of van 't Hof, Kaminski, and Paulusma [81, Lemmas 4 and 5]. We hope and expect that three-in-a-tree with its new near-optimal efficiency will find many other applications in efficient graph algorithms.

1.3 Other related work

For the general *k-in-a-tree* problem, we are given k specific terminals in G, and we want to decide if G has an induced tree T. The k-in-a-tree problem is NP-complete [43] when k is not fixed. With our Theorem 1.1, it can be solved in near-linear time for $k \leq 3$, and the tractability is unknown for any fixed $k \geq 4$ [54]. Solving it in polynomial time for constant k would be a huge result. It is, however, not clear that k-in-a-tree for $k > 3$ would be as powerful a tool in solving other problems as three-in-a-tree has proven to be.

While k-in-a-tree with bounded k is unsolved for general graphs, there has been substantial work devoted to k-in-a-tree for special graph classes. Derhy, Picouleau, and Trotignon [44] and Liu and Trotignon [67] studied k-in-a-tree on graphs with girth at least k for $k = 4$ and general $k \geq 4$, respectively. Dos Santos, da Silva, and Szwarcfiter [48] studied the k-in-a-tree problem on chordal graphs. Golovach, Paulusma, and van Leeuwen [54] studied the k-in-a-tree, k-in-a-cycle, and k-in-a-path problems on AT-free graphs [65]. Bruhn and Saito [13], Fiala, Kaminski, Lidický, and Paulusma [50], and Golovach, Paulusma, and van Leeuwen [55] studied the k-in-a-tree and k-in-a-path problems on claw-free graphs.

See [1, 4, 11, 14, 16, 21, 22, 23, 24, 27, 37, 46, 47, 49, 51, 53, 57, 70, 73] for more work on graph detection, recognition, and characterization. Also see [12, Appendix A] for a survey of the recognition complexity of more than 160 graph classes.

On the hardness side, recall that three-in-a-tree can also be viewed as three in a subdivided star with two or three terminal leaves. However, detecting such a star with 4 terminal leaves is NP-hard. (This follows from Bienstock's NP-hardness of 2-in-a-cycle [9], asking if there exists a hole containing two vertices u and v, which may be assumed to be nonadjacent: Add two new leaves u_1 and u_2 adjacent to u and then, for every two neighbors v_1 and v_2 of v, check if the new graph contains an induced subdivision of a star with exactly four terminal leaves u_1, u_2, v_1, v_2.) Even without terminals, it is NP-hard to detect induced subdivisions of any graph with minimum degree at least four [4, 66]. Finally, we note that if we allow multigraphs with parallel edges, then even 2-in-a-path becomes NP-hard. This NP-hardness is an easy exercise since the induced path cannot contain both end-points of parallel edges.

We note that it is the subdivisions that make induced graph detection hard for constant sized pattern graphs. Without subdivisions, we can trivially check for any induced k-vertex graph in $O(n^k)$ time. Nesetril and Poljak has improved this to roughly $O(n^{k\omega/3})$ where ω is the exponent of matrix multiplication [69]. On the other hand, the ETH hypothesis implies that we cannot detect if a k-clique is a(n induced) subgraph in $n^{o(k)}$ time [60]. A more general understanding of the hardness of detecting induced graphs has been presented recently in [42].

1.4 Techniques

Chudnovsky and Seymour's $O(n^2 m)$-time algorithm for the three-in-a-tree problem is based upon their beautiful characterization for when a graph with three given terminals are contained in some induced tree [28]. The aim is to either find a three-in-a-tree or a witness that it cannot exist. During the course of the algorithm, they develop the witness to cover more and more of the graph. In each iteration, they take some part that is not covered by the current witness and try to add it in, but then some other part of the witness may pop out. They then need a potential function argument to show progress in each iteration.

What we do is to introduce some extra structure to the witness when no three-in-a-tree is found, so that when things are added, nothing pops out. This leads to a simpler more constructive algorithm that is faster by a factor n. Our new witness has more properties than that of Chudnovsky and Seymour, so our characterization of no three-in-a-tree is strictly stronger, yet our overall proof is shorter. Essentially the point is that by strengthening the inductive hypothesis, we get a simpler inductive step. The remaining improvement in speed is based on dynamic graph algorithms.

1.5 Road map

The rest of the paper is organized as follows. Section 2 is a background section where we review Chudnovsky and Seymour's characterization for three-in-a-tree, sketch how it is used algorithmically, as well as the bottleneck for a fast implementation. Section 3 presents our new stronger characterization as well as a high level description of the algorithms and proofs leading to our $\tilde{O}(m)$ implementation. Section 4 proves the correctness of our new characterization. Section 5 provides an efficient implementation. Finally, Section 6 shows how our improved three-in-a-tree algorithm, in tandem with other new ideas, can be used to improve many state-of-the-art graph recognition and detection algorithms. Section 7 concludes the paper.

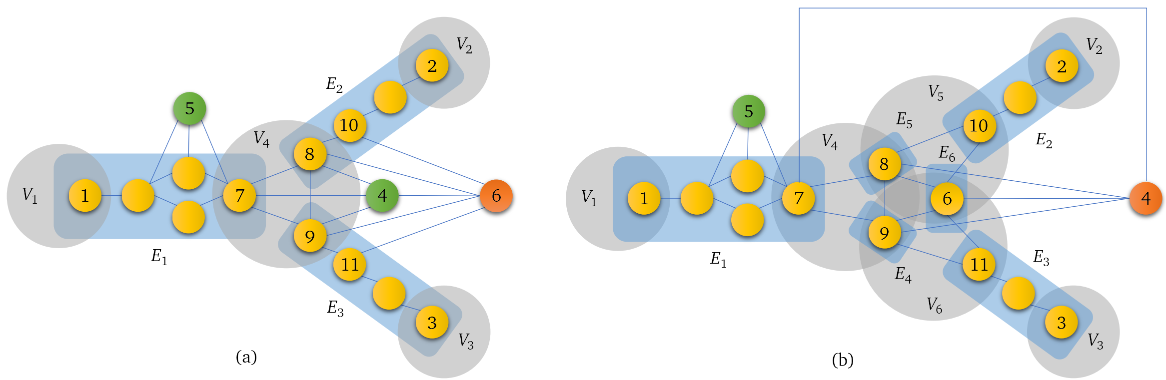


Figure 3: (a) An X-net $\mathbb{H}$ with nodes $V_1, \ldots, V_4$ and arcs E_1, E_2, E_3, where X consists of the vertices other than $4, 5, 6$. Vertices 4 and 5 are $\mathbb{H}$-local. Vertex 6 is $\mathbb{H}$-nonlocal. (b) A nonlocal net $\mathbb{H}$ having a triad $\Delta(V_4, V_5, V_6) = \{6, 8, 9\}$. Vertex 5 is $\mathbb{H}$-local. Vertex 4 is $\mathbb{H}$-nonlocal.

2 Background

2.1 Preliminaries

Let $|S|$ denote the cardinality of set S. Let $R \setminus S$ for sets R and S consist of the elements of R not in S. Let G and H be graphs. Let $V(G)$ (respectively, $E(G)$) consist of the vertices (respectively, edges) of G. Let u and v be vertices. Let U and V be vertex sets. Let $N_G(u)$ consist of the neighbors of u in G. The *degree* of u in G is $|N_G(u)|$. Let $N_G[u] = N_G(u) \cup \{u\}$. Let $N_G(U)$ be the union of $N_G(u) \setminus U$ over all vertices $u \in U$. Let $N_G(u, H) = N_G(u) \cap V(H)$ and $N_G(U, H) = N_G(U) \cap V(H)$. The subscript G in notation N_G may be omitted. A *leaf* of G is a degree-one vertex of G. Let $\nabla(G)$ denote the graph obtained from G by adding an edge between each pair of leaves of G. Let $G[H]$ denote the subgraph of G induced by $V(H)$. Let $G - U = G[V(G) \setminus U]$. Let $G - u = G - \{u\}$. Let uv denote an edge with end-vertices u and v. Graphs H_1 and H_2 are *disjoint* if $V(H_1) \cap V(H_2) = \varnothing$. Graphs H_1 and H_2 are *adjacent* in G if H_1 and H_2 are disjoint and there is an edge uv of G with $u \in V(H_1)$ and $v \in V(H_2)$. A *UV-path* is either a vertex in $U \cap V$ or a path having one end-vertex in U and the other end-vertex in V. A *UV-rung* [28] is a vertex-minimal induced UV-path. If $U = \{u\}$, then a UV-path is also called a *uV-path* and a *Vu-path*. If $U = \{u\}$ and $V = \{v\}$, then a UV-path is also called a *uv-path*. Let *Uv-rung*, *uV-rung*, and *uv-rung* be defined similarly.

For the three-in-a-tree problem, we assume without loss of generality that the three given terminals of the input n-vertex m-edge simple undirected graph G are exactly the leaves of G. A *sapling* of G is an induced tree containing all three leaves of G, so the three-in-a-tree problem is the problem of finding a sapling.

2.2 Chudnovsky and Seymour's characterization

Let $\mathbb{H}$ be a graph such that each member of $V(\mathbb{H})$ and $E(\mathbb{H})$, called *node* and *arc* respectively, is a subset of $X \subseteq V(G)$. $\mathbb{H}$ is an *X-net* of G if the following Conditions N hold (see Figure 3(a)):

N1: Graph $\mathbb{H}$ is connected and graph $\nabla(\mathbb{H})$ is biconnected.
N2: The arcs of $\mathbb{H}$ form a nonempty disjoint partition of the vertex set X.
N3: Graph $\mathbb{H}$ has exactly three leaf nodes, each of which consists of a leaf vertex of G.
N4: For any arc $E = UV$ of $\mathbb{H}$, each vertex of X in E is on a UV-rung of $G[E]$.
N5: For any arc E and node V of $\mathbb{H}$, $E \cap V \neq \varnothing$ if and only if V is an end-node of E in $\mathbb{H}$.

N6: For any vertices u and v in X contained by distinct arcs E and F of $\mathbb{H}$, uv is an edge of G if and only if arcs E and F share a common end-node V in $\mathbb{H}$ with $\{u,v\} \subseteq V$.

An arc $E = UV$ is *simple* if $G[E]$ is a UV-rung. A *net* is an X-net for an X. A *base net* is a net obtained via the next lemma, for which we include a proof to make our paper self-contained.

Lemma 2.1 (Chudnovsky and Seymour [28]). *It takes $O(m)$ time to find a sapling of G or a net of G whose arcs are all simple.*

Proof. Let s_1, s_2, s_3 be the leaves of G. Obtain vertex sets R and S such that $G[S]$ is an s_2s_3-rung of G and $G[R]$ is an s_1S-rung of G. Let $x_1 \in R \setminus S$ be closest to S in $G[R]$. Let each $x_j \in N(x_1, S)$ with $j \in \{2,3\}$ be closest to s_j in $G[S]$. Since s_2 and s_3 are leaves of G, x_2 and x_3 are internal vertices of path $G[S]$. If $x_2 = x_3$, then $G[R \cup S]$ is a sapling of G. If x_2 and x_3 are distinct and nonadjacent, then $G[R \cup S] - I$ is a sapling of G, where I consists of the internal vertices of the x_2x_3-path in $G[S]$. If x_2 and x_3 are adjacent in G, then G admits an $R \cup S$-net having nodes $V_0 = \{x_1, x_2, x_3\}$ and $V_i = \{s_i\}$ with $i \in \{1,2,3\}$ and simple arcs $E_i = V_0V_i$ with $i \in \{1,2,3\}$ consisting of the vertices of the s_ix_i-rung of $G[R \cup S]$. □

The original definition of Chudnovsky et al. only used nets with no parallel arcs, but for our own more efficient construction, we need to use parallel arcs. A *triad* of $\mathbb{H}$ is $\Delta(V_1, V_2, V_3) = (V_1 \cap V_2) \cup (V_2 \cap V_3) \cup (V_3 \cap V_1)$ for nodes V_1, V_2, and V_3 that induce a triangle in graph $\mathbb{H}$. A subset S of X is *$\mathbb{H}$-local* if S is contained by a node, arc, or triad of $\mathbb{H}$ [28]. A set $Y \subseteq V(G - X)$ is *$\mathbb{H}$-local* if $N(Y, X)$ is $\mathbb{H}$-local. $\mathbb{H}$ is *local* if every $Y \subseteq V(G - X)$ with connected $G[Y]$ is $\mathbb{H}$-local. See Figure 3. The following theorem is Chudnovsky and Seymour's characterization.

Theorem 2.2 (Chudnovsky and Seymour [28, 3.2]). *G is sapling-free if and only if G admits a local net with no parallel arcs.*

The proof of Theorem 2.2 in [28] takes up more than 20 pages. We will here present a stronger characterization with a shorter proof, which moreover leads to a much faster implementation. Our results throughout the paper do not rely on Theorem 2.2. Moreover, our paper delivers an alternative self-contained proof for Theorem 2.2.

Chudnovsky and Seymour's proof of Theorem 2.2 is algorithmic maintaining an X-net $\mathbb{H}$ with $X \subseteq V(G)$ having no parallel arcs until a sapling of G is found or $\mathbb{H}$ becomes local, implying that G is sapling-free by the if direction of Theorem 2.2. In each iteration, if $\mathbb{H}$ is not local, they find a minimal set $Y \subseteq V(G - X)$ with connected $G[Y]$ such that Y is $\mathbb{H}$-nonlocal. Their proof for the only-if direction of Theorem 2.2 shows that if $G[X \cup Y]$ is sapling-free, then $\mathbb{H}$ can be updated to an X'-net with $Y \subseteq X' \subseteq X \cup Y$. Although Y joins the resulting X'-net $\mathbb{H}$, a subset of X may have to be moved out of $\mathbb{H}$ to preserve Conditions N for $\mathbb{H}$. To bound the number of iterations, Chudnovsky and Seymour showed that the potential $|X| + (n+1) \cdot |V(\mathbb{H})|$ of $\mathbb{H}$ stays $O(n^2)$ and is increased by each iteration, implying that the total number of iterations is $O(n^2)$. In the next section, we will present a new stronger characterization that using parallel arcs with particular properties avoids the aforementioned in-and-out situation. More precisely, our X will grow in each iteration, reducing the number of iterations to at most n.

3 Our stronger characterization

A base net of G contains only simple arcs. However, we do need other more complex arcs, but we will show that it suffices that all non-simple arcs are "flexible" in the sense defined below. For vertex sets S, V_1, and V_2, an (S, V_1, V_2)-*sprout* is an induced subgraph of G in one of the following *Types S*:

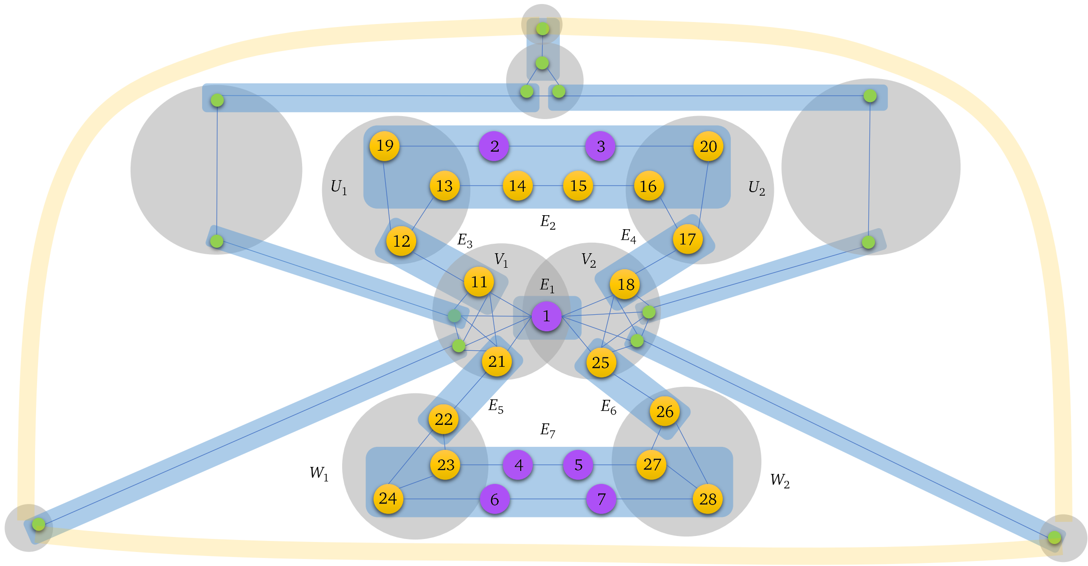
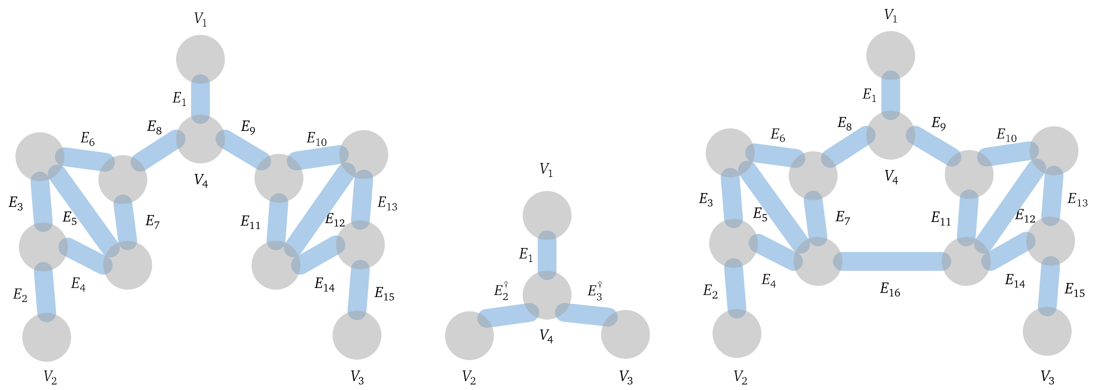


Figure 4: A web $\mathbb{H}$. The arcs of $\nabla(\mathbb{H})$ between the three leaves of $\mathbb{H}$ are in yellow.

S1: A tree intersecting each of S, V_1, and V_2 at exactly one vertex.
S2: An SV_1-rung not intersecting V_2 plus a disjoint SV_2-rung not intersecting V_1.
S3: A V_1V_2-rung not intersecting S plus a disjoint SV-rung with $V = V_1 \cup V_2$.

Let $S = \{1, \ldots, 7\}$ for the example in Figure 4. Vertex 1 is an (S, V_1, V_2)-sprout of Type S1. The set $\{2, 19, 12, 11, 13, 14, 15, 16\}$ induces an (S, V_1, U_2)-sprout of Type S1. The only (S, U_1, U_2)-sprout and (S, W_1, W_2)-sprout of Type S1 contain vertex 1. The set $\{23, 4, 7, 28\}$ induces an (S, W_1, W_2)-sprout of Type S2. The set $\{19, 2, 13, 14, 15, 16\}$ induces an (S, U_1, U_2)-sprout of Type S3. An arc $E = UV$ of $\mathbb{H}$ is *flexible* if $G[E]$ contains an (S, U, V)-sprout for each nonempty vertex set $S \subseteq E$. For the example in Figure 4, arcs E_1, E_3, E_4, E_5, E_6 are simple and arcs E_1, E_2, E_7 are flexible. An X-net $\mathbb{H}$ is an *X-web* if all arcs of $\mathbb{H}$ are simple or flexible. A *web* is an X-web for some X. A base net of G is a web of G. Let $\mathbb{H}$ be a net. A *split component* $\mathbb{G}$ for $\mathbb{H}$ is either an arc UV of $\mathbb{H}$ or a subgraph of $\mathbb{H}$ containing a cutset $\{U, V\}$ of $\nabla(\mathbb{H})$ such that $\mathbb{G}$ is a maximal subgraph of $\nabla(\mathbb{H})$ in which U and V are nonadjacent and do not form a cutset [45]. For both cases, we call $\{U, V\}$ the *split pair* of $\mathbb{G}$ for $\mathbb{H}$. The split components having split pair $\{V_1, V_2\}$ in Figure 4 are (1) the V_1V_2-path with an arc E_1, (2) the V_1V_2-path with arcs E_3, E_2, E_4, and (3) the V_1V_2-path with arcs E_5, E_7, E_6. Thus, even if $\mathbb{H}$ has no parallel arcs, there can be more than one split components sharing a common split pair. One can verify that each split component $\mathbb{G}$ of $\mathbb{H}$ contains at most one leaf node of $\mathbb{H}$ and, if $\mathbb{G}$ contains a leaf node V of $\mathbb{H}$, then V belongs to the split pair of $\mathbb{G}$. A vertex subset C of G is a *chunk* of $\mathbb{H}$ if C is the union of the arcs of one or more split components for $\mathbb{H}$ that share a common split pair $\{U, V\}$ for $\mathbb{H}$. In this case, we call $\{U, V\}$ the *split pair* of C for $\mathbb{H}$ and call C a *UV-chunk* of $\mathbb{H}$. A chunk of $\mathbb{H}$ is *maximal* if it is not properly contained by any chunk of $\mathbb{H}$. A node of $\mathbb{H}$ is a *maximal split node* if it belongs to the split pair of a maximal chunk for $\mathbb{H}$. For the net $\mathbb{H}$ of G in Figure 4, E_1, E_3, $E_3 \cup E_2$, $E_3 \cup E_2 \cup E_4$, and $E_1 \cup E_3 \cup E_2 \cup E_4$ are all chunks of $\mathbb{H}$. If we consider only the subsets of $V(G)$ that intersect the numbered vertices, then $E_1 \cup \cdots \cup E_7$ is the only maximal chunk and V_1 and V_2 are the only maximal split nodes. Given an X-net $\mathbb{H}$, a subset S of X is *$\mathbb{H}$-tamed* if every pair of vertices from S is either in the same arc or together in some node of $\mathbb{H}$. A set $Y \subseteq V(G - X)$ is *$\mathbb{H}$-tamed* if $N(Y, X)$ is $\mathbb{H}$-tamed. $\mathbb{H}$ is *taming* if every $Y \subseteq V(G - X)$ with connected $G[Y]$ is $\mathbb{H}$-tamed. If $S \subseteq X$ is $\mathbb{H}$-local, then S is $\mathbb{H}$-tamed. The converse does not hold: If $\mathbb{H}$ has simple arcs E and F between nodes U and V, $G[E]$ is an edge uv with $u \in U$ and $v \in V$, and $G[F]$ is a vertex $w \in U \cap V$, then



(a) $\mathbb{H}$ (b) $\mathbb{H}^\dagger$ (c) $\mathbb{H} = \mathbb{H}^\dagger$

Figure 5: The aiding net of the $\mathbb{H}$ in (a) is the $\mathbb{H}^\dagger$ in (b) with $E_2^\dagger = E_2 \cup \cdots \cup E_8$ and $E_3^\dagger = E_9 \cup \cdots \cup E_{15}$. The net $\mathbb{H}$ in (c) aids itself.

$\{u, v, w\}$ is $\mathbb{H}$-tamed and $\mathbb{H}$-nonlocal. However, if $\mathbb{H}$ has no parallel arcs, then each $\mathbb{H}$-tamed subset of X is $\mathbb{H}$-local, as shown in Lemma 3.5(2).

A *non-trivial* V_1V_2-chunk C of $\mathbb{H}$ is one that is not an arc in $\mathbb{H}$. We then define the operation MERGE(C) which for a V_1V_2-chunk C of $\mathbb{H}$ replaces all arcs of $\mathbb{H}$ intersecting C by an arc $E = V_1V_2$ with $E = C$ and deletes the nodes whose incident arcs are all deleted. We shall prove that this MERGE operation preserves that $\mathbb{H}$ is a net (see Lemma 3.4). Let $\mathbb{H}^\dagger$ denote the X-net obtained from $\mathbb{H}$ by applying MERGE(C) on $\mathbb{H}$ for each maximal chunk C of $\mathbb{H}$. We call $\mathbb{H}^\dagger$ the X-net that *aids* $\mathbb{H}$. Such an aiding net has no non-trivial chunks and no parallel arcs. See Figure 5 for examples. The simple graph $\nabla(\mathbb{H}^\dagger)$ is triconnected. V is node of $\mathbb{H}^\dagger$ if and only if V is a maximal split node of $\mathbb{H}$. E is an arc of $\mathbb{H}^\dagger$ if and only if E is a maximal chunk of $\mathbb{H}$ (respectively, $\mathbb{H}^\dagger$). The next theorem is our characterization, which is the basis for our much more efficient near-linear time algorithm.

Theorem 3.1. *G is sapling-free if and only if G admits a web $\mathbb{H}$ with a taming aiding net $\mathbb{H}^\dagger$.*

Theorem 3.1 is stronger than Chudnovsky and Seymour's Theorem 2.2 in that our proof of Theorem 3.1 provides as a new shorter proof of Theorem 2.2. To quantify the difference, the proof of Theorem 2.2 in [28] takes up more than 20 pages while our proof of our stronger Theorem 3.1 is self-contained and takes up 13 pages (pages 7–19) including the review of their structure, many more figures, and a simpler $O(mn)$-time algorithm. For the relation between the two structural theorems, we will prove in Lemma 3.5(2) that every taming net of G having no parallel arcs is local. Since the aiding net $\mathbb{H}^\dagger$ in Theorem 3.1 has no parallel arcs, $\mathbb{H}^\dagger$ is local as required by Theorem 2.2. The algorithmic advantage of Theorem 3.1 is that we know that $\mathbb{H}^\dagger$ is the aiding net of a web $\mathbb{H}$ which has more structure than an arbitrary net.

To get a self-contained proof of the easy if-direction of Theorem 3.1, we prove more generally that if G admits a taming net, then G is sapling-free (Lemma 3.5(1)). This proof holds for any net including nets with parallel arcs like our web $\mathbb{H}$. Proving the only-if direction is the hard part for both structural theorems. Our new proof follows the same general pattern as the old one stated after the statement of Theorem 2.2, but with crucial differences to be detailed later.

We grow an X-web $\mathbb{H}$ with $X \subseteq V(G)$ until a sapling of G is found or $\mathbb{H}^\dagger$ becomes taming, implying that G is sapling-free by the if direction of Theorem 3.1. In each iteration, if $\mathbb{H}^\dagger$ is not taming, we find a minimal set $Y \subseteq V(G - X)$ with connected $G[Y]$ such that Y is not $\mathbb{H}^\dagger$-tamed. To prove the only-if direction of Theorem 3.1, we show that if $G[X \cup Y]$ is sapling-free, then $\mathbb{H}$ can be expanded to an X'-web with $X' = X \cup Y$.

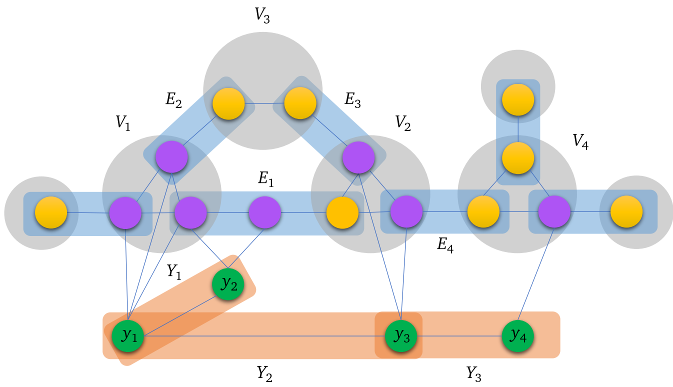


Figure 6: An X-web $\mathbb{H}$, where X consists of the vertices other than y_1, y_2, y_3, y_4. Vertices $y_1, \ldots, y_4$ are all $\mathbb{H}$-tamed and $\mathbb{H}^\dagger$-tamed. Y_1 and Y_2 are $\mathbb{H}$-wild and $\mathbb{H}^\dagger$-nonwild. Y_3 is $\mathbb{H}$-wild and $\mathbb{H}^\dagger$-wild. Y_1 is $\mathbb{H}$-solid. Y_2 and Y_3 are $\mathbb{H}$-nonsolid. E_1, $E_1 \cup E_2 \cup E_3$, and $E_1 \cup E_2 \cup E_3 \cup E_4$ are pods of Y_1 and Y_2 in $\mathbb{H}$. Y_3 is $\mathbb{H}$-unpodded. Y_1 and Y_2 are $\mathbb{H}$-sticky and Y_3 is $\mathbb{H}$-nonsticky.

Comparing with the proof of Chudnovsky and Seymour that we sketched below Theorem 2.2, we note that in their case, their new X'-net would be for some $Y \subseteq X' \subseteq X \cup Y$, whereas we get $X' = X \cup Y$. This is why we can guarantee termination in $O(n)$ rounds while they need a more complicated potential function to demonstrate enough progress in $O(n^2)$ rounds.

Another important difference is that we operate both on a web $\mathbb{H}$ and its aiding net $\mathbb{H}^\dagger$. Recall that the web $\mathbb{H}$ is a net allowing parallel arcs, but with the special structure that all arcs are simple or flexible. This special structure is crucial to our simpler inductive step where we can always add Y as above to get a new web over $X' = X \cup Y$. If we just used $\mathbb{H}$, then we would have too many untamed sets. This is where we use the aiding net $\mathbb{H}^\dagger$ which generally has fewer untamed sets. It is only for the minimally $\mathbb{H}^\dagger$-untamed sets $Y \subseteq V(G - X)$ that we can guarantee progress as above. Thus we need the interplay between the well-structured fine grained web $\mathbb{H}$ and its more coarse grained aiding net $\mathbb{H}^\dagger$ to get our shorter more constructive proof of Theorem 3.1. On its own, our more constructive characterization buys us a factor n in speed. This has to be combined with efficient data structures to get down to near-linear time.

3.1 Two major lemmas and our algorithm for detecting saplings

Let $\mathbb{H}$ be an X-net. An *$\mathbb{H}$-wild* set is a minimally $\mathbb{H}$-untamed $Y \subseteq V(G - X)$ such that $G[Y]$ is a path. In Figure 6, $Y_1 \cup Y_2$ is $\mathbb{H}$-untamed but not $\mathbb{H}$-wild, since $Y_1 \subsetneq Y_1 \cup Y_2$ is $\mathbb{H}$-untamed. $\mathbb{H}$ is not taming if and only if G admits an $\mathbb{H}$-wild set. An $S \subseteq X$ is *$\mathbb{H}$-solid* if S is a node of $\mathbb{H}$ or S is a subset of an arc $E = UV$ of $\mathbb{H}$ such that $G[E]$ contains no (S, U, V)-sprout. If S is a subset of a simple arc of $\mathbb{H}$, then S is $\mathbb{H}$-solid if and only if $G[S]$ is an edge, since a sprout has to be an induced subgraph of G. Let $Y \subseteq V(G - X)$ such that $G[Y]$ is a path. Y is *$\mathbb{H}$-solid* if (1) $N(Y, X)$ is the union of two $\mathbb{H}$-solid sets and (2) $N(y, X) = \varnothing$ for each internal vertex y, if any, of path $G[Y]$. A *pod* of Y in $\mathbb{H}$ is a V_1V_2-chunk C of $\mathbb{H}$ with the following *Conditions P*:

P1: $N(Y, X) \subseteq V_1 \cup C \cup V_2$.
P2: For each $i \in \{1, 2\}$, $N(y, V_i) \subseteq C$ or $V_i \subseteq C \cup N(y)$ holds for an end-vertex y of path $G[Y]$.

Y is *$\mathbb{H}$-podded* if Y admits a pod in $\mathbb{H}$. Y is *$\mathbb{H}$-sticky* if Y is $\mathbb{H}$-solid or $\mathbb{H}$-podded. See Figure 6.

Lemma 3.2. *Let Y be an $\mathbb{H}^\dagger$-wild set for an X-web $\mathbb{H}$. (1) If Y is $\mathbb{H}$-nonsticky, then $G[X \cup Y]$ contains a sapling. (2) If Y is $\mathbb{H}$-sticky, then $\mathbb{H}$ can be expanded to an $X \cup Y$-web.*

By Lemmas 2.1 and 3.2 and Theorem 3.1, the following algorithm detects saplings in G:

Algorithm A

Step A1: If a sapling of G is found (Lemma 2.1), then exit the algorithm.
Step A2: Let X-web $\mathbb{H}$ be the obtained base net of G and then repeat the following steps:

(a) If $\mathbb{H}^\dagger$ is taming, then report that G is sapling-free (if-direction of Theorem 3.1) and exit.
(b) If $\mathbb{H}^\dagger$ is not taming, then obtain an $\mathbb{H}^\dagger$-wild set Y.
(c) If Y is $\mathbb{H}$-nonsticky, then report that $G[X \cup Y]$ contains a sapling (Lemma 3.2(1)) and exit.
(d) If Y is $\mathbb{H}$-sticky, then expand $\mathbb{H}$ to an $X \cup Y$-web (Lemma 3.2(2)).

Lemma 3.3. *Algorithm A can be implemented to run in $O(m \log^2 n)$ time.*

3.2 Reducing Theorems 1.1, 2.2, and 3.1 to Lemmas 3.2 and 3.3 via aiding net

We need a relationship between simple paths in $\mathbb{H}$ and induced paths in G. For any simple UV-path $\mathbb{P}$ of $\mathbb{H}$ (i.e., U and V are the end-nodes of $\mathbb{P}$ in $\mathbb{H}$), we define a *$\mathbb{P}$-rung* of G as a UV-rung of G where all edges are contained in the arcs of $\mathbb{P}$. Such a $\mathbb{P}$-rung always exists by Conditions N4 and N6 of $\mathbb{H}$ as long as $U \neq V$. For the degenerate case $U = V$, let *$\mathbb{P}$-rung* be defined as the empty vertex set. For any distinct nodes U_1 and U_2 of $\mathbb{H}$ intersecting a V_1V_2-chunk C of $\mathbb{H}$, there are disjoint $\mathbb{U}\mathbb{V}$-rungs $\mathbb{P}_1$ and $\mathbb{P}_2$ of $\mathbb{H}$ with $\mathbb{U} = \{U_1, U_2\}$ and $\mathbb{V} = \{V_1, V_2\}$ by Condition N1 of $\mathbb{H}$. Since $\mathbb{P}_1$ and $\mathbb{P}_2$ are disjoint, any $\mathbb{P}_1$-rung and $\mathbb{P}_2$-rung of G are disjoint and nonadjacent by Conditions N2 and N6 of $\mathbb{H}$. Consider the V_1V_2-chunk $C = E_1 \cup \cdots \cup E_7$ in Figure 4. Let $\mathbb{V} = \{V_1, V_2\}$. Let $\mathbb{P}_1$ be the path of $\mathbb{H}$ with arc E_3. Let $\mathbb{P}_2$ be the path of $\mathbb{H}$ with arc E_4. Let $\mathbb{P}_3$ be the path of $\mathbb{H}$ with arcs E_6 and E_7. Let $\mathbb{P}_4$ be the degenerate path of $\mathbb{H}$ consisting of a single node V_1. If $\mathbb{U} = \{U_1, U_2\}$, then $\mathbb{P}_1$ and $\mathbb{P}_2$ are disjoint $\mathbb{U}\mathbb{V}$-rungs. If $\mathbb{U} = \{U_1, W_1\}$, then $\mathbb{P}_1$ and $\mathbb{P}_3$ are disjoint $\mathbb{U}\mathbb{V}$-rungs of $\mathbb{H}$. If $\mathbb{U} = \{V_1, W_1\}$, then $\mathbb{P}_3$ and $\mathbb{P}_4$ are disjoint $\mathbb{U}\mathbb{V}$-rungs of $\mathbb{H}$. The path of G induced by vertex set $\{11, 12\}$ is the unique $\mathbb{P}_1$-rung of G. The path of G induced by vertex set $\{17, 18\}$ is the unique $\mathbb{P}_2$-rung of G. The paths induced by vertex sets $\{25, 26, 27, 5, 4, 23\}$ and $\{25, 26, 28, 7, 6, 24\}$ are the two $\mathbb{P}_3$-rungs of G. The empty vertex set is the unique $\mathbb{P}_4$-rung of G.

Lemma 3.4. *If C is a V_1V_2-chunk of an X-net $\mathbb{H}$, then applying* MERGE(C) *on $\mathbb{H}$ results in an X-net.*

Proof. Let $\mathbb{H}'$ be the resulting $\mathbb{H}$. Since any node cutset of $\mathbb{H}'$ is also a node cutset of $\mathbb{H}$, Conditions N1 of $\mathbb{H}'$ holds. Conditions N2 and N3 of $\mathbb{H}'$ hold trivially. Conditions N5 and N6 of $\mathbb{H}'$ follow from those of $\mathbb{H}$. To see Condition N4 of $\mathbb{H}'$, let x be a vertex in C. Let $E = U_1U_2$ be the arc of $\mathbb{H}$ containing x. There are disjoint $\mathbb{U}\mathbb{V}$-rungs $\mathbb{P}_1$ and $\mathbb{P}_2$ of $\mathbb{H}$ with $\mathbb{U} = \{U_1, U_2\}$ and $\mathbb{V} = \{V_1, V_2\}$. Let each P_i with $i \in \{1, 2\}$ be a $\mathbb{P}_i$-rung of $G[C]$. Let Q be a U_1U_2-rung of $G[E]$ containing x. $G[P_1 \cup Q \cup P_2]$ is a V_1V_2-rung of $G[C]$ containing x. □

Lemma 3.5. *(1) If G admits a taming net, then G is sapling-free. (2) If an X-net $\mathbb{H}$ has no parallel arcs, then every $\mathbb{H}$-tamed subset of X is $\mathbb{H}$-local.*

Since any $\mathbb{H}$-local subset of X for any X-net $\mathbb{H}$ is $\mathbb{H}$-tamed, Lemma 3.5(1) implies the if direction of Chudnovsky et al.'s Theorem 2.2. Moreover, by Lemma 3.5(2), the only-if direction of Theorem 3.1 implies the only-if direction of Theorem 2.2. Thus, our proofs for Lemma 3.5 and the only-if direction of Theorem 3.1 form a self-contained proof for Theorem 2.2.

Proof of Lemma 3.5. Statement 1: Assume a taming net $\mathbb{H}$ and a sapling T of G for contradiction. By Condition N6 of $\mathbb{H}$, any two adjacent vertices in T contained by distinct arcs of $\mathbb{H}$ belong to a node. If $G[Y]$ is a connected component of $T - X$, then vertices u and v of T in $N(Y, X)$ belong to an arc of $\mathbb{H}$: If u and v were in distinct arcs, then $\{u, v\}$ would be contained by a node of $\mathbb{H}$, since $\mathbb{H}$ is taming. By Condition N6 of $\mathbb{H}$, uv is an edge of G, contradicting that T is an induced tree. By Conditions N2, N3, and N5 of $\mathbb{H}$, the nodes and arcs of $\mathbb{H}$ intersecting T form a three-leaf connected subgraph $\mathbb{T}$ of $\mathbb{H}$. Thus, T intersects a node of $\mathbb{T}$ and three of its incident arcs in $\mathbb{T}$. Condition N6 implies a triangle in T, contradiction.

Statement 2: Assume an $\mathbb{H}$-tamed $\mathbb{H}$-nonlocal $S \subseteq X$ for contradiction. Let $E_1, \ldots, E_\ell$ with $\ell \geq 2$ be the arcs of $\mathbb{H}$ intersecting S. Since S is $\mathbb{H}$-tamed, any E_i and E_j with $1 \leq i < j \leq \ell$ share a common end-node. If there is a common end-node V of $E_1, \ldots, E_\ell$, then the other end-nodes of E_i with $i \in \{1, \ldots, \ell\}$ are pairwise distinct, since $\mathbb{H}$ has no parallel arcs. Since S is $\mathbb{H}$-tamed, Condition N6 implies $S \subseteq V$, contradicting that S is $\mathbb{H}$-nonlocal. Thus, $\ell = 3$ and E_1, E_2, E_3 form a triangle of $\mathbb{H}$. Since S is $\mathbb{H}$-nonlocal, there a vertex $x_i \in E_i \setminus V_i$ with $i \in \{1, 2, 3\}$ for an end-node V_i of E_i. Let x_j be a vertex of S in the arc E_j with $j \in \{1, 2, 3\} \setminus \{i\}$ incident to V_i. $\{x_i, x_j\} \subseteq S$ is $\mathbb{H}$-untamed, contradicting that S is $\mathbb{H}$-tamed. □

Proof of Theorems 1.1 and 3.1. The if direction of Theorem 3.1 follows from Lemma 3.5(1). To see the only-if direction of Theorem 3.1, let $\mathbb{H}$ be an X-web with maximum $|X|$ as ensured by Lemma 2.1. If $\mathbb{H}^\dagger$ were not taming, then any $\mathbb{H}^\dagger$-wild Y would be $\mathbb{H}$-sticky by Lemma 3.2(1), which in turn implies an $X \cup Y$-web by Lemma 3.2(2), contradicting the maximality of $\mathbb{H}$. Thus Theorem 3.1 follows. By Lemmas 2.1 and 3.2 and the if direction of Theorem 3.1, Algorithm A correctly detects saplings in G. Thus, Theorem 1.1 follows from Lemma 3.3. □

Lemma 3.3 is not needed in the above reduction of Theorem 3.1 or else our proof of Theorem 2.2 would not be shorter than that in [28]. To complete proving Theorems 2.2 and 3.1, we prove Lemma 3.2 in §4. After that, to complete proving Theorem 1.1, we prove Lemma 3.3 in §5.

4 Proving Lemma 3.2

The following lemma is needed in the proofs of Lemma 3.2(1) in §4.1 and Lemma 3.2(2) in §4.2. For any chunk C of a net $\mathbb{H}$, the *arc set* $\mathbb{C}$ of $\mathbb{H}$ for C consists of the arcs of $\mathbb{H}$ that intersect C.

Lemma 4.1. *Let $\mathbb{H}$ be an X-web. (1) If Y is an $\mathbb{H}^\dagger$-wild set, then Y is $\mathbb{H}^\dagger$-podded if and only if Y is $\mathbb{H}$-podded. (2) Each $\mathbb{H}^\dagger$-solid subset of X is $\mathbb{H}$-solid.*

Proof. Statement 1: The only-if direction is straightforward, since each V_1V_2-chunk of $\mathbb{H}^\dagger$ is a V_1V_2-chunk of $\mathbb{H}$. For the if direction, let C be a V_1V_2-chunk of $\mathbb{H}$ that satisfies all Conditions P for Y. The maximal chunk of $\mathbb{H}$ containing C is an arc $E^\dagger = W_1W_2$ of $\mathbb{H}^\dagger$. By $N(Y, X) \subseteq V_1 \cup C \cup V_2 \subseteq W_1 \cup E^\dagger \cup W_2$, Condition P1 holds for $E^\dagger$ in $\mathbb{H}^\dagger$. Since Y is $\mathbb{H}^\dagger$-untamed, $N(Y, X)$ intersects $(W_1 \cup W_2) \setminus E^\dagger$, implying $\{V_1, V_2\} \cap \{W_1, W_2\} \neq \varnothing$. Let $V_1 = W_1$ and $W_1 \setminus E^\dagger \subseteq V_1 \setminus C \subseteq N(Y, X)$ without loss of generality. If $N(Y, X)$ does not intersect $W_2 \setminus E^\dagger$, then Condition P2 holds for Y in $\mathbb{H}^\dagger$. Otherwise, we have $V_2 = W_2$ and $W_2 \setminus E^\dagger \subseteq V_2 \setminus C \subseteq N(Y, X)$, also implying Condition P2 of Y in $\mathbb{H}^\dagger$. Thus, $E^\dagger$ is a pod of Y in $\mathbb{H}^\dagger$.

Statement 2: It suffices to consider the case that the $\mathbb{H}^\dagger$-solid subset S of X is not a node of $\mathbb{H}$, implying that S is not a node of $\mathbb{H}^\dagger$. Let $\mathbb{W} = \{W_1, W_2\}$ for the arc $C = W_1W_2$ of $\mathbb{H}^\dagger$ with $S \subseteq C$. $G[C]$ contains no (S, W_1, W_2)-sprout. The rest of the proof lets all sprouts be (S, W_1, W_2)-sprouts unless explicitly specified otherwise. Let E_i with $1 \leq i \leq |\mathbb{C}|$ be the arcs in the arc set $\mathbb{C}$ of $\mathbb{H}$ for C. Let $\mathbb{V}_i$ consist of the end-nodes of E_i. For any i and j that may not be distinct, let $\mathbb{P}_{i,j}$ and $\mathbb{Q}_{i,j}$ be disjoint $\mathbb{W}\mathbb{V}_i$-rung and $\mathbb{W}\mathbb{V}_j$-rung of $\mathbb{H}$. Let $P_{i,j}$ be a $\mathbb{P}_{i,j}$-rung of G. Let $Q_{i,j}$ be a $\mathbb{Q}_{i,j}$-rung of G. Let $U_{i,j}$ be the end-node of $\mathbb{P}_{i,j}$ in $\mathbb{V}_i$. Let $V_{i,j}$ be the end-node of $\mathbb{Q}_{i,j}$ in $\mathbb{V}_j$. If $S \subseteq E_i$ for an $i \in \{1, \ldots, |\mathbb{C}|\}$, then $G[E_i]$ contains no $(S, U_{i,i}, V_{i,i})$-sprout T or else $G[T \cup P_{i,i} \cup Q_{i,i}]$ would be a sprout of $G[C]$. Thus, S is $\mathbb{H}$-solid. The rest of the proof assumes for contradiction that S intersects two or more arcs of $\mathbb{C}$.

We first show that S is contained by a node of $\mathbb{H}$. For any distinct i and j such that S intersects both E_i and E_j, let r be an arbitrary vertex in $S \cap E_i$ and s be an arbitrary vertex in $S \cap E_j$. Let $P = G[P_{i,j} \cup P']$ and $Q = G[Q_{i,j} \cup Q']$ for arbitrary $rU_{i,j}$-rung P' of $G[E_i]$ and $sV_{i,j}$-rung Q' of $G[E_j]$. By Conditions N2 and N5, $P - r$ and $Q - s$ are disjoint and nonadjacent, implying that r and s are adjacent or else $G[P \cup Q]$ would contain a sprout of Type S2 in $G[C]$. Since r and s are arbitrary, Condition N6 implies $S \subsetneq U$ for a node U of $\mathbb{H}$: If S is not contained by any node of $\mathbb{H}$, then S is contained by $\Delta(V_1, V_2, V_3)$ and intersects $V_1 \cup V_2$, $V_2 \cap V_3$, and $V_3 \cap V_1$ for nodes V_1, V_2, V_3 of $\mathbb{H}$. Let $\mathbb{V} = \{V_1, V_2, V_3\}$. Let $\mathbb{P}_i$ and $\mathbb{P}_j$ with $\{i, j\} \subseteq \{1, 2, 3\}$ be disjoint $\mathbb{V}\mathbb{W}$-rungs of $\mathbb{C}$ such that V_i and V_j are the end-nodes of $\mathbb{P}_i$ and $\mathbb{P}_j$ in $\mathbb{V}$. $G[P_i \cup \{v\} \cup P_j]$ for $v \in S \cap V_i \cap V_j$ and $\mathbb{P}_i$-rung P_i and $\mathbb{P}_j$-rung P_j of $G[C]$ is a sprout of Type S1, contradiction.

For any arcs $E_i = UV_i$ and $E_j = UV_j$ of $\mathbb{C}$ with $V_i \neq V_j$, we say that E_i *evades* E_j if there are disjoint $\mathbb{V}\mathbb{W}$-rungs $\mathbb{P}$ and $\mathbb{Q}$ of $\mathbb{H}$ with $\mathbb{V} = \{V_i, V_j\}$ such that $\mathbb{P} \cup \mathbb{Q}$ does not intersect U. E_i evades E_j if and only if E_j evades E_i. If E_i evades E_j and E_i intersects S, then $E_j \cap U \subseteq S$ or else $G[C]$ would contain a sprout $G[P_i \cup Q_j \cup P \cup Q]$ of Type S1, where P_i is an E_i-rung intersecting S, Q_j is an E_j-rung intersecting $U \setminus S$, P is a $\mathbb{P}$-rung, and Q is a $\mathbb{Q}$-rung.

By $S \subsetneq U$, $E_j \cap U \nsubseteq S$ holds for an arc $E_j = UV_j$. If each arc $E_i = UV_i$ intersecting S satisfies $V_i = V_j$, then $G[P_{i,i} \cup Q_{i,i} \cup R]$ for any UV_i-rung R of $G[E_i]$ that intersects S is a sprout of Type S1, contradiction. Thus, an arc $E_i = UV_i$ with $V_i \neq V_j$ intersects S. By $E_j \cap U \nsubseteq S$ and $E_i \cap S \neq \varnothing$, E_i does not evade E_j. We show contradiction by identifying an arc $E_k = UV_k$ with $V_k \notin \{V_i, V_j\}$ such that E_i evades E_k, implying $E_k \cap U \subseteq S$, and E_k evades E_j, implying $E_k \cap S = \varnothing$. Let $\mathbb{P}_i$ and $\mathbb{P}_j$ be disjoint $\mathbb{V}\mathbb{W}$-rungs of $\mathbb{H}$ with $\mathbb{V} = \{V_i, V_j\}$. Since E_i does not evade E_j, $\mathbb{P}_i \cup \mathbb{P}_j$ intersects U. Let $\mathbb{P}_j$ intersect U without loss of generality. U is the neighbor of V_j in $\mathbb{P}_j$. Let $E_k = UV_k$ be the incident arc of U in $\mathbb{P}_j$ with $V_k \neq V_j$. Let $\mathbb{Q} = \mathbb{P}_j - \{U, V_j\}$. Since $\mathbb{P}_i$ and $\mathbb{Q}$ are disjoint $\mathbb{V}\mathbb{W}$-rungs of $\mathbb{H}$ with $\mathbb{V} = \{V_i, V_k\}$ and $\mathbb{P}_i \cup \mathbb{Q}$ does not intersect U, E_i evades E_k. Let $\mathbb{R}'$ be a rung of $(\mathbb{C} \cup \{W_1W_2\}) - U$ between V_j and $\mathbb{P}_i$. $\mathbb{R}'$ does not intersect $\mathbb{Q}$ or else E_i would evade E_j. Let $\mathbb{R}$ be the $V_j\mathbb{W}$-rung of $\mathbb{P}_i \cup \mathbb{R}'$. Since $\mathbb{Q}$ and $\mathbb{R}$ are disjoint $\mathbb{V}\mathbb{W}$-rungs of $\mathbb{H}$ with $\mathbb{V} = \{V_k, V_j\}$ and $\mathbb{Q} \cup \mathbb{R}$ does not intersect U, E_k evades E_j. $\square$

4.1 Proving Lemma 3.2(1)

A net *self-aids* if it aids itself. Since the aiding net of any web self-aids, Lemma 3.2(1) is immediate from Lemma 4.2 by Lemma 4.1.

Lemma 4.2. *For self-aiding X-net $\mathbb{H}_0$ and $\mathbb{H}_0$-wild $\mathbb{H}_0$-nonsticky set Y, $G[X \cup Y]$ contains a sapling.*

The rest of the subsection proves Lemma 4.2 using Lemmas 4.3, 4.4, and 4.5. Let $\mathbb{L}$ consist of the leaves of the self-aiding net $\mathbb{H}$ in Lemma 4.3, 4.4, or 4.5. Since $\nabla(\mathbb{H})$ is triconnected, each nonleaf node of $\mathbb{H}$ has degree at least three in $\mathbb{H}$ and any three-node set $\mathbb{U}$ of $\mathbb{H}$ admits pairwise disjoint $\mathbb{U}\mathbb{L}$-rungs $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3$ of $\mathbb{H}$. By Condition N6 of $\mathbb{H}$, any $\mathbb{P}_i$-rungs P_i of G with $i \in \{1, 2, 3\}$ are pairwise disjoint and nonadjacent.

Lemma 4.3. *If Y is an $\mathbb{H}$-wild $\mathbb{H}$-nonsticky set for a self-aiding X-net $\mathbb{H}$ of G such that $N_G(Y, X) = M_1 \cup M_2$ and each of M_1 and M_2 is contained by a node or arc of $\mathbb{H}$, then $G[X \cup Y]$ contains a sapling.*

Proof. Let $N = N_G(Y, X)$. We start with proving the following statement.

Claim 1: *If $M_i \subseteq U$ with $\{i, j\} = \{1, 2\}$ holds for a node U and $M_j \subseteq U_1 \cup F$ holds for an end-node U_1 of an arc F with $U_1 \neq U$, $U \setminus F \nsubseteq M_i$, and $M_i \nsubseteq F$, then $G[X \cup Y]$ contains a sapling.*

Let $\mathbb{R}_1 = \{U_1\}$. Since the degree of U is at least three, $U \setminus F \nsubseteq M_i$ and $M_i \nsubseteq F$ imply that the node set consisting of the neighbors of U other than U_1 in $\mathbb{H}$ admits a nonempty disjoint partition $\mathbb{R}_2$ and $\mathbb{R}_3$ such that (a) each arc between U and $\mathbb{R}_2$ intersects M_i and (b) each arc between U and $\mathbb{R}_3$ intersects $U \setminus M_i$. Let $\mathbb{H}'$ be the triconnected graph obtained from $\nabla(\mathbb{H})$ by (1) replacing node U and its incident arcs with a triangle on a set $\mathbb{W} = \{W_1, W_2, W_3\}$ of three new nodes and (2) adding an arc between W_i and each node in $\mathbb{R}_i$ for all $i \in \{1,2,3\}$. There are pairwise disjoint $\mathbb{WL}$-rungs $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3$ of $\mathbb{H}'$ such that each $\mathbb{P}_i$ with $i \in \{1,2,3\}$ is a $W_i L_i$-rung with $L_i \in \mathbb{L}$. Let $\mathbb{Q}_1$ be the path of $\mathbb{H}$ consisting of arc F and path $\mathbb{P}_1 - W_1$. Let Q_1 be the NL_1-rung of a $\mathbb{Q}_1$-rung of G intersecting M_j. Let $\mathbb{Q}_2$ be the L_2L_3-path of $\mathbb{H}$ obtained from $\mathbb{P}_2 \cup \mathbb{P}_3$ by replacing the two arcs W_2U_2 and W_3U_3 with the two arcs UU_2 and UU_3. Let Q_2 be a $\mathbb{Q}_2$-rung of G intersecting exactly one vertex in $M_i \cap U$. N intersects each of Q_1 and Q_2 at exactly one vertex. Thus, $G[Y \cup Q_1 \cup Q_2]$ contains a sapling of $G[X \cup Y]$. Claim 1 is proved.

Claim 2: *If $G[X \cup Y]$ is sapling-free, then each M_i with $i \in \{1,2\}$ is $\mathbb{H}$-solid.*

To prove Claim 2 by Claim 1, let each M_i with $i \in \{1,2\}$ be contained by a node V_i or an arc E_i. We first show that if $M_1 \subseteq V_1$ and $M_2 \subseteq V_2$, then V_1V_2 is not an arc. Assume an arc $E = V_1V_2$ for contradiction. Since Y is $\mathbb{H}$-wild and $\mathbb{H}$-unpodded, we have $V_i \nsubseteq (E \cup M_i)$ and $M_i \nsubseteq E$ for $\{i,j\} = \{1,2\}$, contradicting Claim 1 with $U = V_i$, $U_1 = V_j$, and $F = E$.

To see Claim 2(a): *$M_i \subseteq V_i$ for $\{i,j\} = \{1,2\}$ implies $M_i = V_i$*, assume $V_i \nsubseteq M_i$. If $M_j \subseteq V_j$, then V_iV_j is not an arc, contradicting Claim 1 with $U = V_i$, $U_1 = V_j$, and F being an incident arc of V_j. $M_j \subseteq E_j$ contradicts Claim 1 with $U = V_i$, $F = E_j$, and U_1 being an end-node of E_j that is not V_i. To see Claim 2(b): *$M_i \subseteq E_i = UV$ for $\{i,j\} = \{1,2\}$ implies that M_i is $\mathbb{H}$-solid*, assume an (M_i, U, V)-sprout T of $G[E_i]$. If $M_j \subseteq V_j$, then let $W = V_j$. By Claim 2(a), W is not incident to E_i or else E_i would be a pod of Y in $\mathbb{H}$. If $M_j \subseteq E_j$, then let W be an end-node of E_j not incident to E_i. Let $\mathbb{U} = \{U, V, W\}$. Let $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3$ be pairwise disjoint $\mathbb{UL}$-rungs of $\mathbb{H}$. Let each P_k with $k \in \{1,2,3\}$ be a $\mathbb{P}_k$-rung of G. $G[P_1 \cup P_2 \cup P_3 \cup Y \cup T \cup E_j]$ contains a sapling.

To prove the lemma by Claim 2, assume for contradiction that $G[X \cup Y]$ is sapling-free. Since Y is $\mathbb{H}$-nonsolid, Claim 2 implies an internal vertex y of path $G[Y]$ with nonempty $N_G(y, X) \subseteq M_1 \cap M_2$. By Condition N2, $M_i = V_i$ holds for $\{i,j\} = \{1,2\}$. By Condition N5, if $M_j \subseteq V_j$, then $\mathbb{H}$ has an arc $E = V_iV_j$, which is a pod of Y in $\mathbb{H}$; and if $M_j \subseteq E_j$, then V_i is incident to E_j, which is thus a pod of Y in $\mathbb{H}$. Both cases contradict that Y is $\mathbb{H}$-nonsticky. □

If Y is $\mathbb{H}$-wild for an X-net $\mathbb{H}$, then let $\ell(Y, \mathbb{H}, G)$ denote the minimum number of $\mathbb{H}$-tamed subsets of X whose union is $N_G(Y, X)$. A net is *simple* if all of its arcs are simple. If $\mathbb{H}$ is a simple self-aiding X-net of G, then $G[X]$ is isomorphic to the line graph of a subdivision of $\mathbb{H}$.

Lemma 4.4. *If Y is an $\mathbb{H}$-wild set for a simple self-aiding X-net $\mathbb{H}$ of G with $\ell(Y, \mathbb{H}, G) = 2$ such that $N_G(Y, X)$ contains a triad of $\mathbb{H}$, then $G[X \cup Y]$ contains a sapling.*

Proof. Let $\mathbb{U} = \{U_1, U_2, U_3\}$ for nodes with $\Delta = \Delta(U_1, U_2, U_3) \subseteq N = N_G(Y, X)$. Let $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3$ be pairwise disjoint $\mathbb{UL}$-rungs of $\mathbb{H}$. For each $i \in \{1,2,3\}$, let $L_i \in \mathbb{L}$ such that the $\mathbb{P}_i$-rung P_i of G is a U_iL_i-rung. Since N is untamed, $N \setminus \Delta \neq \varnothing$. Since $\mathbb{H}$ is simple, each arc intersecting $N \setminus \Delta$ is incident to at most one node of $\mathbb{U}$. By $\ell(Y, \mathbb{H}, G) = 2$, $N \setminus \Delta$ intersects at most one of P_1, P_2, and P_3. If N intersects P_i for $\{i,j,k\} = \{1,2,3\}$, then $G[Y \cup Q_i \cup (U_j \cap U_k) \cup P_j \cup P_k]$ contains a sapling for the NL_i-rung Q_i of P_i. It remains to consider $(N \setminus \Delta) \cap V(P_1 \cup P_2 \cup P_3) = \varnothing$.

Case 1: Each arc E intersecting $N \setminus \Delta$ satisfies $|E| = 1$ and is incident to $\mathbb{P}_i$ and $\mathbb{P}_j$ for $\{i,j,k\} = \{1,2,3\}$. Let E be an arc intersecting $N \setminus \Delta$. Let $V_i \in V(\mathbb{P}_i)$ and $V_j \in V(\mathbb{P}_j)$ be end-nodes of E with $U_i \neq V_i$. Let $\mathbb{Q}_1$ be the U_iL_k-path of $\mathbb{H}$ consisting of arc $E_j = U_iU_k$ and $\mathbb{P}_k$. Let Q_1 be the $\mathbb{Q}_1$-rung of G. Let $\mathbb{Q}_2$ be the L_iL_j-path of $\mathbb{H}$ consisting of E, the V_iL_i-rung of $\mathbb{P}_i$, and the V_jL_j-rung of $\mathbb{P}_j$. Let Q_2 be the $\mathbb{Q}_2$-rung of G. By $U_i \neq V_i$, $\mathbb{Q}_1$ and $\mathbb{Q}_2$ are disjoint. By $(N \setminus \Delta) \cap V(P_1 \cup P_2 \cup P_3) = \varnothing$, Q_1

(respectively, Q_2) intersects N exactly at the vertex in arc E_j (respectively, E). Thus, $G[Y \cup Q_1 \cup Q_2]$ contains a sapling of $G[X \cup Y]$.

Case 2: An arc E intersecting $N \setminus \Delta$ violates the condition of Case 1. Let $\mathbb{Q}$ be a shortest path of $\mathbb{H}$ between $V(E)$ and $V(\mathbb{P}_1 \cup \mathbb{P}_2 \cup \mathbb{P}_3)$. Since E violates the condition of Case 1, we may require that if $U \in V(E)$ and $V_i \in V(\mathbb{P}_i)$ with $\{i, j, k\} = \{1, 2, 3\}$ are the end-nodes of $\mathbb{Q}$, then the NU-rung Q_i of $G[E]$ is not adjacent to $P_j \cup P_k$. Let $E_i = U_j U_k$. Let Q be the $\mathbb{Q}$-rung of G. Let R_i be the $V_i L_i$-rung of P_i. $G[P_j \cup P_k \cup E_i \cup Q_i \cup Q \cup R_i]$ contains a sapling of $G[X \cup Y]$. □

Lemma 4.5. *Let Y be an $\mathbb{H}$-wild set for a simple self-aiding X-net $\mathbb{H}$ of graph G with $\ell(Y, \mathbb{H}, G) \geq 3$. If $G[X \cup Y]$ is sapling-free, then Y is $\mathbb{H}$-podded for G.*

Proof. Since Y is $\mathbb{H}$-wild with $\ell = \ell(Y, \mathbb{H}, G) \geq 3$, Y consists of a vertex y. Let $N_1, \ldots, N_\ell$ be pairwise disjoint $\mathbb{H}$-tamed subsets of X whose union is $N = N_G(Y, X)$. Let L consist of the leaves of $H = G[X \cup Y]$. Let each graph $H_{i,j}$ with $1 \leq i < j \leq \ell$ be obtained from H by deleting the edges between y and $N \setminus (N_i \cup N_j)$. We claim that *each $H_{i,j}$ is sapling-free*. Since $\mathbb{H}$ is a simple self-aiding X-net of $H_{i,j}$ with $\ell(Y, \mathbb{H}, H_{i,j}) = 2$, Y is $\mathbb{H}$-sticky for $H_{i,j}$ by Lemmas 4.3 and 4.4. Thus, each N_i with $i \in \{1, \ldots, \ell\}$ is either contained by a node or arc of $\mathbb{H}$. Assume that $N_1, \ldots, N_k$ are $\mathbb{H}$-solid and $N_{k+1}, \ldots, N_\ell$ are not. If $k < \ell$, then Y is $\mathbb{H}$-podded for all $H_{i,\ell}$ with $i \in \{1, \ldots, \ell - 1\}$. If N_ℓ is contained by a node U, then there is exactly one vertex u in $U \setminus N_\ell$, implying $\ell = 3$ and that the arc containing u is a pod of Y in $\mathbb{H}$ for G. If N_ℓ is not contained by a node, then $\ell = 3$ and the arc containing N_ℓ is a pod of Y in $\mathbb{H}$ for G. As for $k = \ell$, observe that there cannot be a 3-node set $\mathbb{U} = \{U_{i_1}, U_{i_2}, U_{i_3}\}$ with $\{i_1, i_2, i_3\} \subseteq \{1, \ldots, \ell\}$ such that each node U_{i_j} with $j \in \{1, 2, 3\}$ is either a solid set N_{i_j} or an end-node of the arc E_{i_j} containing a solid set N_{i_j}: Assume for contradiction that such a $\mathbb{U}$ exists. Let vertex set E be the union of the arcs E_{i_j} with $N_{i_j} \subseteq E_{i_j}$. Let $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}_3$ be pairwise disjoint $\mathbb{U}\mathbb{L}$-rungs of $\mathbb{H}$. For each $j \in \{1, 2, 3\}$, let P_j be a $\mathbb{P}_j$-rung of G. $G[Y \cup P_1 \cup P_2 \cup P_3 \cup E]$ contains a sapling, contradiction. The observation implies $\ell = 3$ and that Y is $\mathbb{H}$-podded for G.

To prove the claim, assume a sapling T of $H_{i,j}$. Since any edge in $H[T] \setminus T$ is between y and $N_{i,j}$, the following statements hold or else H would contain a sapling in which y is the degree-3 vertex: (1) The degree of y in T is two. (2) $H[T] \setminus T$ has exactly one edge e, implying that y and a vertex $u_1 \in N_{i,j}$ are the end-vertices of e. (3) The degree-3 vertex u_2 of T is adjacent to y and u_1 in T, implying $u_2 \in N_i \cup N_j$. That is, $H[T]$ consists of a triangle on $U = \{y, u_1, u_2\}$ and pairwise disjoint UL-rungs P_1, P_2, P_3 of T with $N \cap V(P_1) = \{u_1\} \subseteq N_{i_1}$, $N \cap V(P_2) = \{u_2\} \subseteq N_{i_2}$, and $y \in V(P_3)$ for distinct i_1 and i_2 in $\{1, \ldots, \ell\}$. By Lemma 3.5(2), each N_{i_k} with $k \in \{1, 2\}$ is contained by a node, arc, or triad S_k. $N \cap (S_1 \cup S_2)$ is $\mathbb{H}$-untamed or else there would be $\ell - 1$ pairwise disjoint $\mathbb{H}$-tamed subsets of X whose union is N. Let each E_k with $k \in \{1, 2\}$ be the simple arc with $u_k \in E_k$. We show that H contains a sapling in which y is the degree-3 vertex.

Case 1: If $E_1 = E_2$. Since $N \cap (S_1 \cup S_2)$ is $\mathbb{H}$-untamed, a vertex $v_k \in S_k \cap (N \setminus E_k)$ with $k \in \{1, 2\}$. Since $\mathbb{H}$ is simple and $\{u_k, v_k\} \subseteq S_k$, S_k is not a triad. S_k is not an arc or else $E_k = S_k$ would intersect $N \setminus E_k$. Thus, S_k is an end-node of E_k with $\{u_k, v_k\} \subseteq S_k$ and $u_{3-k} \notin S_k$. By $u_k \in S_k$ and Condition N6, S_k is not adjacent to $(P_3 - y) \cup (P_{3-k} - u_{3-k})$ in H. Since $\mathbb{H}$ is simple, $v_k \in N \cap S_k$ implies that $H[(T - u_k) \cup \{v_k\}]$ is a sapling of H.

Case 2: $E_1 \neq E_2$. By Condition N5, $\{u_1, u_2\} \subseteq V$ for a common end-node V of arcs E_1 and E_2. By Condition N6, $E_k \subseteq V(P_k)$ for each $k \in \{1, 2\}$. Since $N \cap (S_1 \cup S_2)$ is $\mathbb{H}$-untamed, a vertex $v_k \in S_k \cap (N \setminus V)$ with $k \in \{1, 2\}$. Since $\mathbb{H}$ is simple and $\{u_k, v_k\} \subseteq S_k$, S_k is not a triad. S_k is not an arc or else $\{u_k, v_k\} \subseteq S_k = E_k \subseteq V(P_k)$ would contradict $N \cap V(P_k) = \{u_k\}$. Thus, S_k is a node. By $u_k \in E_k \cap S_k$, S_k is an end-node of E_k containing u_k. By $v_k \in S_k \setminus V$, $S_k \neq V$. By $u_k \in S_k$ and Condition N6, S_k is not adjacent to $V(P_3 - y) \cup V(P_{3-k} - u_{3-k})$ in H. Since $\mathbb{H}$ is simple, $v_k \in N \cap S_k$ implies that $H[(T - u_k) \cup \{v_k\}]$ is a sapling of H. □

Proof of Lemma 4.2. Assume for contradiction that $G[X \cup Y]$ is sapling-free. A vertex set $D \subseteq X$ is an *inducing* set of $\mathbb{H}_0$ if $G[E_0 \cap D]$ for each arc $E_0 = U_0V_0$ is an U_0V_0-rung of $G[E_0]$. For any inducing set D of $\mathbb{H}_0$, let $\mathbb{H}_0(D)$ denote the simple self-aiding D-net of graph $H_0(D) = G[Y \cup D]$ obtained from $\mathbb{H}_0$ by replacing each arc E_0 of $\mathbb{H}_0$ with the arc $E = E_0 \cap D$ and replacing each node V_0 of $\mathbb{H}_0$ with the node $V = V_0 \cap D$. Let $N = N_G(Y, X)$. Let $\ell = \ell(Y, \mathbb{H}_0, G)$. If $\ell = 2$, then Lemma 4.3 implies $N \nsubseteq S_1 \cup S_2$ for any node or arc S_i of $\mathbb{H}_0$ with $i \in \{1,2\}$. Thus, N contains a triad Δ and $N \setminus \Delta$ is not contained by any arc of $\mathbb{H}_0$ between two nodes of Δ. By $\ell = 2$, there is an inducing set D of $\mathbb{H}_0$ with $\ell(Y, \mathbb{H}_0(D), H_0(D)) = 2$ and $\Delta \subseteq N_{H_0(D)}(Y, D)$, contradicting Lemma 4.4. Thus, $\ell \geq 3$, implying a three-vertex set $S \subseteq N$ such that every two-vertex subset of S is $\mathbb{H}_0$-untamed. Let D be an inducing set of $\mathbb{H}_0$ with $S \subseteq D$, implying $\ell(Y, \mathbb{H}_0(D), H_0(D)) \geq 3$. By Lemma 4.5, there is a pod $E = UV$ of Y in $\mathbb{H}_0(D)$ for $H_0(D)$ such that $N_{H_0(D)}(Y)$ intersects $E \setminus (U \cup V)$, $U \setminus E$, and $V \setminus E$. Let $E_0 = U_0V_0$ be the arc of $\mathbb{H}_0$ with $E = E_0 \cap D$, $U = U_0 \cap D$, and $V = V_0 \cap D$. Since E_0 is not a pod of Y in $\mathbb{H}_0$ and N intersects $E_0 \setminus (U_0 \cup V_0)$, $U_0 \setminus E_0$, and $V_0 \setminus E_0$, a vertex x belongs to $N \setminus (U_0 \cup E_0 \cup V_0)$ or $(U_0 \cup V_0) \setminus (E_0 \cup N)$. Let D' be an inducing set $(D \setminus E_0) \cup V(P)$ of $\mathbb{H}_0$, where $E_0 = U_0V_0$ is the arc of $\mathbb{H}_0$ containing x and P is a U_0V_0-rung of $G[E_0]$ containing x. One can verify that Y is $\mathbb{H}_0(D')$-unpodded for $H_0(D')$ with $\ell(Y, \mathbb{H}_0(D'), H_0(D')) \geq 3$, contradicting Lemma 4.5. □

4.2 Proving Lemma 3.2(2)

This subsection shows that if Y is $\mathbb{H}$-sticky for an X-web $\mathbb{H}$, then $\mathbb{H}$ can be expanded to an $X \cup Y$-web via Subroutine B below. Let $\mathbb{H}$ be an X-net. For any $\mathbb{H}$-solid subset S of X contained by a simple arc $F = U_1U_2$ of $\mathbb{H}$, define *Operation* SUBDIVIDE(S) to (1) create a new node S and (2) replace the simple arc by new simple arcs SU_i with $i \in \{1,2\}$ consisting of the vertices of the SU_i-rung of $G[F]$. Define *Subroutine B* with $N = N(Y, X)$ as follows (see Figure 7):

Subroutine B

Step B1: Y is $\mathbb{H}$-solid. Let S_1 and S_2 be $\mathbb{H}$-solid sets with $N = S_1 \cup S_2$.

(a) For each $i \in \{1,2\}$, if S_i is contained by a simple arc, then create node S_i by SUBDIVIDE(S_i).
(b) Add each end-vertex y of path $G[Y]$ into the nodes S_i with $i \in \{1,2\}$ and $S_i \subseteq N(y)$.
(c) Make a simple arc $Y = S_1S_2$.

Step B2: Y is $\mathbb{H}$-nonsolid. Thus, Y is $\mathbb{H}$-podded. Let V_1V_2-chunk C of $\mathbb{H}$ be a minimal pod of Y in $\mathbb{H}$. Since Y is $\mathbb{H}^\dagger$-wild, assume $V_1 \in V(\mathbb{H}^\dagger)$ and $V_1 \subseteq C \cup N$ without loss of generality.

(a) If V_2 is incident to exactly one arc $F = VV_2$ in the arc set for C, $N \cap V_2 \subseteq F$, and F is simple, then N intersects $F \setminus V$ by the minimality of C. Let v_2 be the end-vertex of the NV_2-rung P of $G[F]$ in N. Let v be the neighbor of v_2 not in P. Call SUBDIVIDE$(\{v, v_2\})$ to create a new node $V_2 = \{v, v_2\}$. Delete $V(P)$ from C to preserve that C is a V_1V_2-chunk that is a minimal pod of Y in $\mathbb{H}$.
(b) Update $\mathbb{H}$ by MERGE(C). Let $E = V_1V_2$ be the arc of $\mathbb{H}$ with $E = C$.
(c) Add Y to arc E and add each end-vertex y of path $G[Y]$ to the nodes V_i with $V_i \subseteq C \cup N(y)$.

Proof of Lemma 3.2(2). The resulting $\mathbb{H}$ of Step B1 is an $X \cup Y$-web, since all steps preserve Conditions N and all new arcs are simple. The rest of the proof shows that the resulting $\mathbb{H}$ of Step B2 is also an $X \cup Y$-web. At the beginning of Step B2(b) one can verify that, no matter whether $\mathbb{H}$ is updated by Step B2(a) or not, Y is $\mathbb{H}$-nonsolid and $\mathbb{H}$-podded and $\mathbb{H}$ is an X-web with the following *Condition F*: If V_2 is incident to exactly one arc F in the arc set for the minimal pod C of Y in $\mathbb{H}$ and F is simple, then $N(Y, V_2)$ intersects $V_2 \setminus C$. By Lemma 3.4, $\mathbb{H}$ is an X-net (respectively, $X \cup Y$-net) at the end of Step B2(b) (respectively, Step B2(c)). It remains to show that $E = C \cup Y$ is a flexible arc by identifying an (S, V_1, V_2)-sprout of $G[E]$ for any nonempty subset S of E. The rest of the proof lets

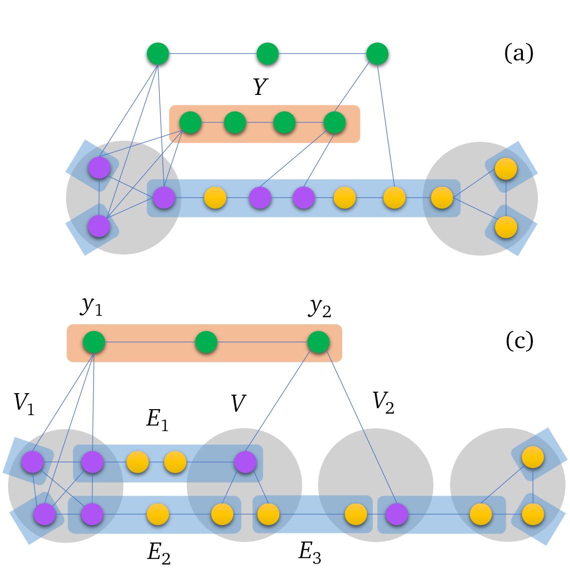


Figure 7: Applying Step B1 on the example in (a) results in the example in (b), in which $E_1 \cup E_2 \cup F$ is a minimal pod of the green y_1y_2-rung. Applying Step B2(a) on the example in (b) results in the example in (c), in which $E_1 \cup E_2 \cup E_3$ is a minimal pod of the green y_1y_2-rung. Applying Steps B2(b) and B2(c) on the example in (c) results in the example in (d).

$\mathbb{H}$ denote the X-web at the beginning of Step B2(b) and lets all sprouts be (S, V_1, V_2)-sprouts of $G[E]$ unless specified otherwise. Let y_1 and y_2 be the end-vertices of path $G[Y]$ with $V_1 \subseteq C \cup N(y_1, X)$. If $|Y| = 1$, then $y_1 = y_2$. If $|Y| \geq 2$, then let $N_i = N(y_i, X)$ and $M_i = N(Y \setminus \{y_{3-i}\}, X)$ for each $i \in \{1, 2\}$ and let $M = M_1 \cap M_2$. Let $S_C = S \cap C$ and $S_Y = S \cap Y$. If $S_C \neq \varnothing$, then S_C is assumed to be $\mathbb{H}$-solid, since any (S_C, V_1, V_2)-sprout of $G[C]$ is a sprout. If $S_Y \neq \varnothing$, then let each P_i with $i \in \{1, 2\}$ be the Sy_i-rung of $G[Y]$. Let $C^* = W_1W_2$ with $W_1 = V_1$ be the arc of $\mathbb{H}^\dagger$ containing C.

Case 1: $S_C = \varnothing$. $G[S]$ is an edge in $G[Y]$ or else a V_1V_2-rung of $G[E]$ containing Y contains a sprout of Type S1 or S2. By $|S| = 2$, $|Y| \geq 2$. Since Y is $\mathbb{H}^\dagger$-wild, $M_1 \subseteq V_1$. We may assume $M_1 = V_1$, since otherwise $G[P_1 \cup Q]$ is a spout of Type S3 for a V_1V_2-rung Q of $G[C]$ intersecting $V_1 \setminus M_1$. Case 1(a): *M_2 is $\mathbb{H}$-nonsolid*. Lemma 4.1(2) implies an (M_2, W_1, W_2)-sprout T^* of $G[C^*]$. Let $T = G[T^*[C] \cup P_2]$. If T^* is of Type S1 or S2, then T contains a sprout of Type S1. If T^* is of Type S3, then T is a sprout of Type S3. Case 1(b): *M_2 is $\mathbb{H}$-solid*. Since M_1 is $\mathbb{H}$-solid and Y is $\mathbb{H}$-nonsolid, we have $M \neq \varnothing$ and $M \subseteq V_1 \cap M_2$. If M_2 were contained by a simple arc F of $\mathbb{H}$, then $F = V_1V_2$ by $V_1 \cap M_2 \neq \varnothing$ and minimality of C, contradicting Condition F. Thus, M_2 is a node of $\mathbb{H}$. By $V_1 \cap M_2 \neq \varnothing$, $F = V_1M_2$ is an arc of $\mathbb{H}$. By $M \subseteq V_1$, we have $M_2 \subseteq F \cup N_2$. By minimality of C, $M_2 = V_2$. By $M \neq \varnothing$ and $|Y| \geq 3$, $G[Y \cup M]$ contains a sprout of Type S1.

Case 2: $S_Y = \varnothing$. $S = S_C$ is $\mathbb{H}$-solid. Let $v_1 \in V_1 \setminus C$, $v_2 \in V_2 \setminus C$, and a set of new vertices $B = \{r, s, u_1, u_2, w\}$. Define an X_0-net $\mathbb{H}_0$ of a graph G_0 on $X_0 \cup Y$ with $X_0 = B \cup C \cup \{v_1, v_2\}$ as follows (see Figure 8): Initially, let $G_0 = G[C \cup Y \cup \{v_1, v_2\}]$ and let $\mathbb{H}_0$ consist of the nodes and arcs of $\mathbb{H}$ that intersect C. For each $i \in \{1, 2\}$, update V_i by deleting all vertices not in C except for v_i and then adding w. Make a new simple arc V_1V_2 consisting of w. Add a minimum number of edges to make $N_{G_0}(w) = (\{y_1\} \cup V_1 \cup V_2) \setminus \{w\}$. Make new nodes $R = \{r\}$, $U_1 = \{u_1\}$, and $U_2 = \{u_2\}$. If S is a node, then let $S_0 = S$; otherwise, make a new node S_0 via SUBDIVIDE(S). Add s into S_0. Make a simple arc RS_0 consisting of r and s. For each $i \in \{1, 2\}$, make a simple arc U_iV_i consisting of vertices u_i and v_i. Add a minimum number of edges to make $N_{G_0}(s) = R \cup S$, $N_{G_0}(r) = \{s\}$, $N_{G_0}(u_1) = \{v_1\}$, and $N_{G_0}(u_2) = \{v_2\}$.

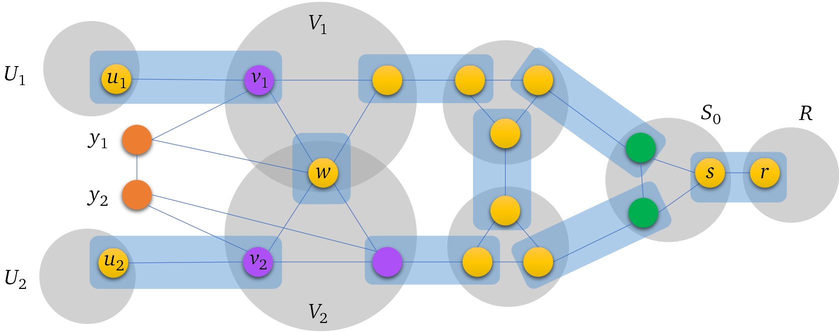


Figure 8: An example of $\mathbb{H}_0$.

$\mathbb{H}_0$ is an X_0-net of G_0 with leaf nodes R, U_1, and U_2 and leaf vertices r, u_1, and u_2. Since $\mathbb{H}$ is an X-web of G and all new arcs of $\mathbb{H}_0$ are simple, $\mathbb{H}_0$ is an X_0-web of G_0. Since each V_i with $i \in \{1,2\}$ is the neighbor of U_i and V_1V_2 is an arc of $\mathbb{H}_0$, V_1 is a maximal split node of $\mathbb{H}_0$. Since Y is $\mathbb{H}^\dagger$-wild, Y is $\mathbb{H}_0^\dagger$-wild. Since Y is $\mathbb{H}$-nonsolid, $\{w\}$ is not a pod of Y in $\mathbb{H}_0$ and Y is $\mathbb{H}_0$-nonsolid. Since node S_0 is adjacent to leaf R in $\mathbb{H}_0$, no V_1V_2-chunk of $\mathbb{H}_0$ intersects S_0, implying no pod of Y in $\mathbb{H}_0$ that is a superset of C. The minimality of C implies no pod of Y in $\mathbb{H}_0$ that is a proper subset of C. Thus, Y is $\mathbb{H}_0$-unpodded. Lemma 3.2(1) implies a sapling T_0 of G_0. $T_0 - (B \cup \{v_1, v_2\})$ is a sprout.

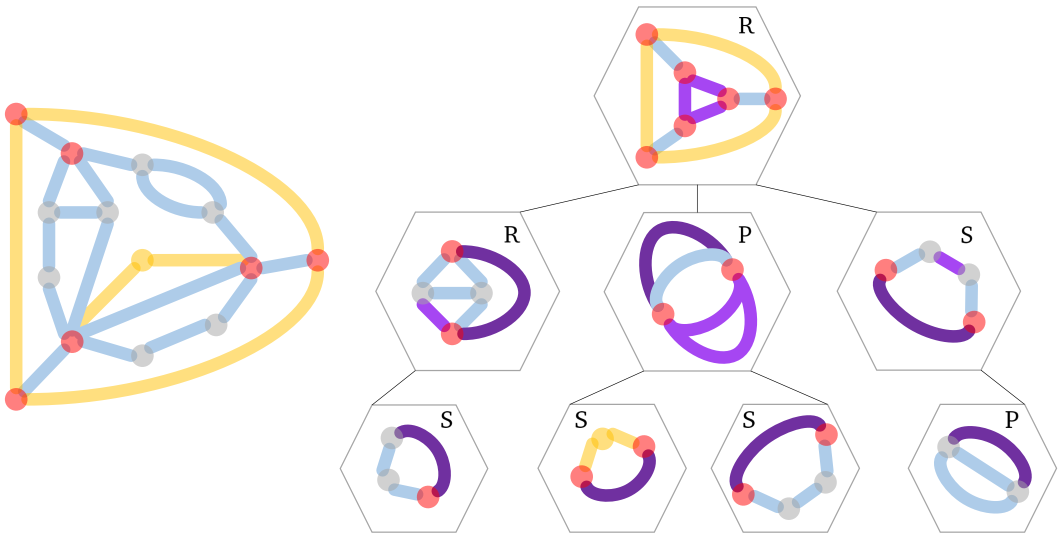
Case 3: $S_Y \neq \varnothing$ and $S_C \neq \varnothing$. S_C is $\mathbb{H}$-solid. Assume $S_Y = \{y_2\}$, since otherwise $G[P_1 \cup Q]$ for an S_CV_2-rung Q of $G[C]$ not intersecting V_1 is a sprout of Type S2. Assume that any N_2V_2-rung Q of $G[C]$ intersects S_C exactly at its end-vertex in N_2, since otherwise $G[P_1 \cup Q]$ contains a sprout of Type S1 or S2. Thus, each vertex $v \in C$ admits a vV_2-rung $Q(v)$ of $G[C]$ with $(V_1 \cup S_C) \cap V(Q(v)) \subseteq \{v\}$: Assume for contradiction a $v \in C$ such that each vV_2-rung $Q(v)$ with $V_1 \cap V(Q(v)) \subseteq \{v\}$ intersects $S_C \setminus \{v\}$. If S_C is a node V of $\mathbb{H}$, then graph $\mathbb{G}(\mathbb{C}) - V$ is disconnected. If S_C is contained by a simple arc F of $\mathbb{H}$, then graph $\mathbb{H}[\mathbb{C}] - \{V_1, V\}$ is disconnected. Either way, the minimality of C implies that N_2 intersects the connected component of $G[C] - S_C$ that intersects V_2, implying an $N_2(V_2 \setminus S_C)$-rung of $G[C]$, contradicting the above assumption.

Case 3(a): *a vertex* $v \in N_2 \setminus S_C$. $Q(v)$ does not intersect S_C, so $G[Y \cup Q(v)]$ is a sprout of Type S1.
Case 3(b): *a vertex* $v \in S_C \setminus N_2$. $Q(v)$ does not intersect N_2 or else the N_2V_2-rung of $Q(v)$ does not intersect S_C at its end-vertex in N_2, contradiction. Thus, $G[Y \cup Q(v)]$ is a sprout of Type S2.
Case 3(c): $N_2 = S_C$. If $M_1 \neq V_1$, then $G[P_1 \cup Q(v_1)]$ for a $v_1 \in V_1 \setminus M_1$ contains a sprout of Type S3. If $M_1 = V_1$, then M contains a v_1, since Y is $\mathbb{H}$-nonsolid. We have $N = M_1 \cup N_2$. M_1 and N_2 are both $\mathbb{H}$-solid. Thus, $G[Y \cup Q(v_1)]$ contains a sprout of Type S1. □

This completes the proof of our characterization in Theorem 3.1 as well as Chudnovsky and Seymour's characterization in Theorem 2.2. Subroutine B can be implemented to run in $O(m)$ time, so Steps A2(c) and A2(d) take $O(m)$ time. Steps A1, A2(a), and A2(b) take $O(m)$ time. Since the set of vertices of G in $\mathbb{H}$ is enlarged by Step A2(d) and not affected elsewhere, Step A2 halts in $O(n)$ iterations. Thus, Algorithm A can be implemented to run in $O(mn)$ time. To complete proving Theorem 1.1, it remains to implement Algorithm A to run in $O(m \log^2 n)$ time in §5 using dynamic graph algorithms and other data structures.

5 Proving Lemma 3.3

Let G be represented by a static adjacency list. We use a dynamic adjacency list to represent an incremental biconnected multigraph $\mathbb{H}^*$ with $V(\mathbb{H}^*) = V(\mathbb{H})$ that is a supergraph of $\nabla(\mathbb{H})$. An



(a) $\mathbb{H}^*$ (b) $\mathbb{T}$

Figure 9: An example of $\mathbb{H}^*$ and $\mathbb{T}$. The Q-knots are omitted for brevity. The virtual arc in dark purple in a nonroot knot K matches a light purple arc in the parent of K in $\mathbb{T}$. They form the pair of virtual arcs between the poles of K. Each non-purple arc in a knot K is a virtual arc whose corresponding arc of $\mathbb{H}^*$ is contained by a child Q-knot of K. A non-purple arc is in yellow if and only if its corresponding arc of $\mathbb{H}^*$ is dummy. The dummy nodes of $\mathbb{H}^*$ are in yellow. $\mathbb{H}$ is the multigraph obtained from $\mathbb{H}^*$ by deleting the yellow nodes and arcs. $\mathbb{H}^\dagger$ is the simple graph obtained from the one in the root of $\mathbb{T}$ by deleting the yellow arcs. The maximal split nodes of $\mathbb{H}$, i.e., the nodes of $\mathbb{H}^\dagger$ are in red.

arc or node of $\mathbb{H}^*$ is *dummy* if it is an empty vertex set of G. For instance, the three arcs of $\nabla(\mathbb{H})$ between the leaves of $\mathbb{H}$ are dummy in $\mathbb{H}^*$. Other dummy nodes and arcs are created only via operation MERGE. The X-web $\mathbb{H}$ maintained by Algorithm A is exactly $\mathbb{H}^*$ excluding its dummy arcs and nodes. See Figure 9(a) for an example of $\mathbb{H}^*$. Each node and arc of $\mathbb{H}$ and $\mathbb{H}^\dagger$ is associated with a distinct *color* that is a positive integer such that two vertices share a common *arc color* (respectively, *node color*) for $\mathbb{H}$ and $\mathbb{H}^\dagger$ if and only if they are contained by a common arc (respectively, node) of $\mathbb{H}$ and $\mathbb{H}^\dagger$. For each vertex v of G, we maintain a set of at most six colors indicating the arc, maximal chunk, nodes, and maximal split nodes of $\mathbb{H}$ that contain v, which are called the *$\mathbb{H}$-arc*, *$\mathbb{H}^\dagger$-arc*, *$\mathbb{H}$-node*, and *$\mathbb{H}^\dagger$-node colors* of vertex v. For each color c, we store its corresponding arc or node for $\mathbb{H}$ or $\mathbb{H}^\dagger$ and maintain the number of the vertices having the color c without keeping an explicit list of these vertices. For each node V and each incident arc E of V in $\mathbb{H}$, we maintain the cardinality of the vertex set $E \cap V$. Thus, it takes $O(1)$ time to (1) update and query the colors of a vertex and (2) add a vertex to an arc or node of $\mathbb{H}$. For each arc of $\mathbb{H}^*$, we mark whether it is dummy, simple, or flexible and, for each simple arc $E = V_1V_2$ of $\mathbb{H}^*$, we use a doubly linked list to store the V_1V_2-rung $G[E]$. For any vertex v and vertex set Y of G, let $d(v) = |N(v)|$ and $d(Y) = \sum_{y \in Y} d(y)$ throughout the section.

Based on Lemma 5.1, to be proved in §5.4, Steps A2(a) and A2(b) are implemented in §5.1 to run in overall $O(m \log^2 n)$ time throughout Algorithm A. Step A2(c) is implemented in §5.2 to run in overall $O(m)$ time throughout Algorithm A. Step A2(d), i.e., Subroutine B is implemented in §5.3 to run in overall $O(m \log n \cdot \alpha(n,n))$ time throughout Algorithm A, where $\alpha(n,n)$ is the inverse Ackermann function.

5.1 Steps A2(a) and A2(b) of Algorithm A

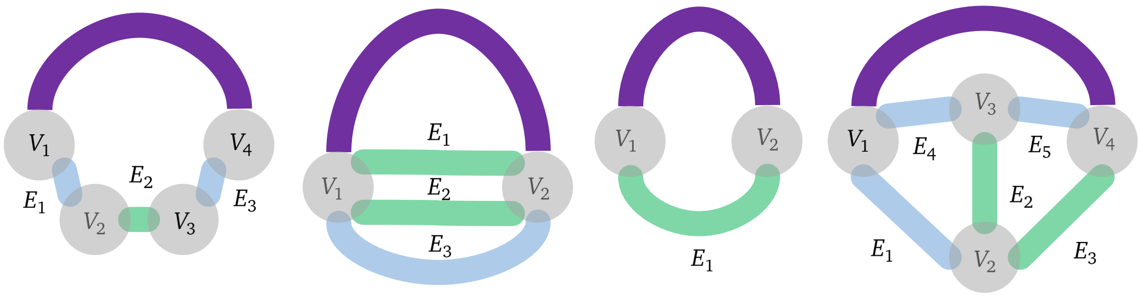
Although vertex colors change only in Step A2(d), the overall number of changes of the $\mathbb{H}^\dagger$-arc and $\mathbb{H}^\dagger$-node colors affects the analysis of our implementation of Steps A2(a) and A2(b). Therefore, this subsection analyzes the time for the change of $\mathbb{H}^\dagger$-arc and $\mathbb{H}^\dagger$-node colors. The time for the change of $\mathbb{H}$-arc and $\mathbb{H}$-node colors will be analyzed for Step A2(d) in §5.3. A vertex of G stays uncolored until it is added into X. Each vertex of X has exactly one $\mathbb{H}^\dagger$-arc color and at most two $\mathbb{H}^\dagger$-node colors. Each node V of $\mathbb{H}^\dagger$ stays a node of $\mathbb{H}^\dagger$ and each vertex in V stays in V for the rest the algorithm. Thus, the $\mathbb{H}^\dagger$-node colors of each vertex are updated $O(1)$ times throughout the algorithm, implying that the overall time for updating $\mathbb{H}^\dagger$-node colors of all vertices is $O(n)$. Although the $\mathbb{H}^\dagger$-arc color of a vertex may change many times, the overall time for updating the $\mathbb{H}^\dagger$-node colors of all vertices can be bounded by $O(n \log n)$. Observe that $\mathbb{H}$ is updated by Subroutine B only via (1) subdividing a simple arc of $\mathbb{H}$, (2) merging an $\mathbb{H}$-podded Y into a minimal pod of Y in $\mathbb{H}$, and (3) creating an arc $E = Y$ for an $\mathbb{H}$-solid Y. If the simple graph $\mathbb{H}^\dagger$ does not change, then each of these updates takes $O(d(Y))$ time. If the simple graph $\mathbb{H}^\dagger$ changes, then Y is $\mathbb{H}$-solid. For instance, let $\mathbb{H}$ be as in Figure 5(a), implying that $\mathbb{H}^\dagger$ is as in Figure 5(b). If an $\mathbb{H}$-solid Y joins $\mathbb{H}$ as the arc E_{16} in Figure 5(c), then all nodes and arcs of $\mathbb{H}$ become nodes and arcs of $\mathbb{H}^\dagger$. However, once two vertices of X have distinct $\mathbb{H}^\dagger$-arc colors, they can no longer share a common arc color for $\mathbb{H}^\dagger$ for the rest of the algorithm. Thus, one can bound the overall number of changes of $\mathbb{H}^\dagger$-arc colors of all vertices by $O(n \log n)$ as follows: If E is an arc of the original $\mathbb{H}^\dagger$ and $E_1, \ldots, E_k$ are the arcs of the updated $\mathbb{H}^\dagger$ with $E_1 \cup \cdots \cup E_k \subseteq E$ and $|E_1| \le \cdots \le |E_k|$, then let the vertices in E_k keep their original $\mathbb{H}^\dagger$-arc color and assign a distinct new $\mathbb{H}^\dagger$-arc color to the vertices in each E_i with $i \in \{1, \ldots, k-1\}$. Since the cardinality of the arc of $\mathbb{H}^\dagger$ containing a specific vertex of X is halved each time its $\mathbb{H}^\dagger$-arc color changes, its $\mathbb{H}^\dagger$-arc color changes $O(\log n)$ times, implying that the $\mathbb{H}^\dagger$-arc colors of all vertices change $O(n \log n)$ times throughout the algorithm. With the data structure of Lemma 5.1, to be proved in §5.4, the overall time for Steps A2(a) and A2(b) throughout the algorithm is $O(m \log^2 n)$.

Lemma 5.1. *If X is an incremental subset of $V(G)$ such that each $x \in X$ has exactly one $\mathbb{H}^\dagger$-arc color a and a set of at most two $\mathbb{H}^\dagger$-node colors corresponding to a subset of the two end-vertices of a, then there is an $O(m+n)$-time obtainable data structure supporting the following queries and updates:*

1. *Move a vertex v of $G - X$ to X in amortized $O(d(v) \cdot \log^2 n)$ time.*
2. *Update the colors of a vertex $v \in X$ in amortized $O(d(v) \cdot \log n)$ time.*
3. *Determine if there is a set $Y \subseteq V(G - X)$ with connected $G[Y]$ such that two vertices of $N(Y, X)$ share no color and, for the positive case, report a minimal such Y in amortized $O(d(Y) \cdot \log^2 n)$ time.*

5.2 Step A2(c) of Algorithm A

Let $\mathbb{S}$ be the $O(d(Y))$-time obtainable set consisting of the nodes V of $\mathbb{H}$ with $V \subseteq N(Y, X)$ and the simple arcs E of $\mathbb{H}$ with $G[E \cap N(Y, X)]$ being an edge. Y is $\mathbb{H}$-solid if and only if $|\mathbb{S}| = 2$, $N(y, X) = \varnothing$ for each internal node y of path $G[Y]$, and $N(Y, X)$ is contained by the union of the nodes or arcs in $\mathbb{S}$. Therefore, it takes $O(d(Y))$ time to determine whether Y is $\mathbb{H}$-solid. Lemma 4.1(1) implies that Y is $\mathbb{H}$-podded if and only if both of the following conditions hold: (a) $N(Y, X)$ is contained by the union of an arc E of $\mathbb{H}^\dagger$ and its end-nodes V_1 and V_2 in $\mathbb{H}^\dagger$ and (b) E is a pod of Y in $\mathbb{H}^\dagger$. Both conditions can be checked in $O(d(Y))$ time via the $\mathbb{H}^\dagger$-arc and $\mathbb{H}^\dagger$-node colors of each vertex in $N(Y, X)$ and the cardinalities of $V_1 \setminus E$ and $V_2 \setminus E$. Therefore, it takes $O(d(Y))$ time to determine whether Y is $\mathbb{H}$-podded. Since the $\mathbb{H}^\dagger$-wild sets Y in all iterations of the algorithm are pairwise disjoint, it takes overall $O(m)$ time for Step A2(c) to determine whether Y is $\mathbb{H}$-sticky throughout the algorithm.



(a) S-knot (b) P-knot (c) Q-knot (d) R-knot

Figure 10: Four examples of the lowest common ancestor K of the Q-knots containing the arcs of $\mathbb{H}$ in $\mathbb{C}_1 \cup \mathbb{C}_2$, which equals E_2 in (a), $E_1 \cup E_2$ in (b), E_1 in (c), and $E_2 \cup E_3$ in (d).

5.3 Step A2(d) of Algorithm A, i.e., Subroutine B

This subsection shows how to implement Subroutine B so that the overall time of Step A2(d) throughout Algorithm A is $O(m \log n \cdot \alpha(n, n))$. Although we may delete nodes and arcs from $\mathbb{H}$ via MERGE(C) for a minimal pod C of Y in $\mathbb{H}$, they stay as dummy nodes and arcs in $\mathbb{H}^*$ in order to make the multigraph $\mathbb{H}^*$ incremental. One can verify that $\mathbb{H}^\dagger$ aids $\mathbb{H}^*$, even though $\mathbb{H}^*$ is not an X-net due to its dummy arcs and nodes. Although Step B1(b) may change $\mathbb{H}^\dagger$, the overall time for updating the $\mathbb{H}^\dagger$-colors has been accounted for in §5.1. Therefore, this subsection only analyzes the time required by the change of $\mathbb{H}$-arc and $\mathbb{H}$-node colors and the cardinalities of $E \cap V_1$ and $E \cap V_2$ for each arc $E = V_1V_2$ of $\mathbb{H}$.

The *SPQR-tree* $\mathbb{T}$ of the incremental multigraph $\mathbb{H}^*$ is an $O(n)$-time obtainable $O(n)$-space tree structure representing the triconnected components of $\mathbb{H}^*$ [45, 56]. Each member of $V(\mathbb{T})$, which we call a *knot*, is a graph homeomorphic to a subgraph of $\mathbb{H}^*$ [45, Lemma 3] such that the knots induce a disjoint partition of the arcs of $\mathbb{H}^*$. Specifically, there is a supergraph $\mathbb{G}$ of $\mathbb{H}^*$ with $V(\mathbb{G}) = V(\mathbb{H}^*)$, where each arc of $\mathbb{G} \setminus \mathbb{H}^*$ is called *virtual* [80], and there are four types of knots of $\mathbb{T}$: (1) *S-knot*: a simple cycle on three or more nodes. (2) *P-knot*: three or more parallel arcs. (3) *Q-knot*: two parallel arcs, exactly one of which is virtual. (4) *R-knot*: a triconnected simple graph that is not a cycle. The Q-knots are the leaves of $\mathbb{T}$ and each arc of $\mathbb{H}^*$ is contained by a Q-knot. No two S-knots (respectively, P-knots) are adjacent in $\mathbb{T}$. Each virtual arc is contained by exactly two adjacent knots. Since $\mathbb{H}$ has $O(n)$ arcs by Condition N2, $\mathbb{T}$ has $O(n)$ knots. If U and V are nonleaf nodes of $\mathbb{H}$ such that UV is a virtual arc, then $\{U, V\}$ is a split pair of $\mathbb{H}$. If distinct nodes U and V admit three internally disjoint UV-paths in $\mathbb{H}^*$, then U and V are contained by a common P-knot or R-knot of $\mathbb{T}$ [45]. By Condition N1 of $\mathbb{H}$, there are three internally disjoint paths in $\nabla(\mathbb{H})$ between each pair of leaves of $\mathbb{H}^*$, implying an R-knot of $\mathbb{T}$ containing the leaves of $\mathbb{H}$. Let $\mathbb{T}$ be rooted at this unique R-knot. Figure 9(b) is the $\mathbb{T}$ for the $\mathbb{H}^*$ in Figure 9(a). Let K be a nonroot knot of $\mathbb{T}$. The *poles* [56] of K are the end-nodes of the unique virtual arc contained by K and its parent knot in $\mathbb{T}$. For the four nonroot knots K in Figure 10, V_1 and V_4 (respectively, V_2) are the poles of the knots in (a) and (d) (respectively, (b) and (c)). Let $\mathbb{C}(K)$ consist of the arcs of $\mathbb{H}$ in the descendant Q-knots of K in $\mathbb{T}$. Let $C(K)$ consist of the vertices of G contained by the arcs of $\mathbb{C}(K)$. If U and V are the poles of a nonroot knot K of $\mathbb{T}$, then $C(K)$ is a UV-chunk and $\mathbb{C}(K)$ is the arc set for $C(K)$. A nonempty vertex set C is a maximal chunk of $\mathbb{H}$ if and only if $C = C(K)$ holds for a child knot K of the root of $\mathbb{T}$. For instance, the X-net $\mathbb{H}$ in Figure 9(a) has six maximal chunks. One of them is $C(K)$ for the child R-knot (respectively, P-knot and S-knot) K of the root of $\mathbb{T}$. The remaining three are $C(K)$ for three omitted child Q-knots K of the root of $\mathbb{T}$. For any nonroot knot K of $\mathbb{T}$ with $C(K) \neq \varnothing$, if K is a P-knot, then $\mathbb{C}(K)$ is the union of the arc sets of all split components of $\{U, V\}$ (e.g., three splits components of $\{V_1, V_2\}$ in the example in Figure 10(b)); otherwise, $\mathbb{C}(K)$ is the arc set of a single split component of $\{U, V\}$, where U and V are the poles of K (e.g., exactly one split component for $\{V_1, V_4\}$ in the examples in Figures 10(a) and 10(d) and exactly one split component for $\{V_1, V_2\}$ in the example in Figure 10(c)).

Lemma 5.2 (Di Battista and Tamassia [45]). *Each update to $\mathbb{T}$ corresponding to the following operations on the incremental biconnected multigraph $\mathbb{H}^*$ can be implemented to run in amortized $\alpha(n,n)$ time: (1) Add a new node V to subdivide an arc V_1V_2 of $\mathbb{H}^*$ into two arcs $E_1 = VV_1$ and $E_2 = VV_2$. (2) Add an arc UV between two nodes U and V of $\mathbb{H}$.*

We first show that, given a vertex set S contained by a simple arc $E = V_1V_2$ such that $G[S]$ is an edge, Operation SUBDIVIDE(S) in Steps B1(a) and B2(a) can be implemented to run in amortized $O(\log n)$ time: Let each P_i with $i \in \{1,2\}$ be the V_iS-rung of $G[E]$. Let j be an index in $\{1,2\}$ with $|V(P_j)| \le |V(P_{3-j})|$. Using the doubly linked list for the V_1V_2-rung $G[E]$, it takes $O(|V(P_j)|)$ time to (1) create a new node $V = S$ with a new $\mathbb{H}$-node color assigned to both vertices in S, (2) create a new simple arc $E_j = VV_j$ consisting of the vertices of P_j, (3) assign a new $\mathbb{H}$-arc color for each vertex in E_j, (4) let arc E_{3-j} take over the $\mathbb{H}$-arc color of E, and (5) obtain the doubly linked lists of $G[E_1]$ and $G[E_2]$ from that of $G[E]$. Each time a vertex x is recolored this way, the cardinality of the simple arc of $\mathbb{H}$ containing x is halved. Therefore, the overall time for Operation SUBDIVIDE(S) in Steps B1(a) and B2(a) is $O(n \log n)$.

Step B1: By the above analysis for SUBDIVIDE, Step B1(a) runs in amortized $O(\log n)$ time. As for Steps B1(b) and B1(c), a new $\mathbb{H}$-arc color is created for the new arc of $\mathbb{H}$. The $\mathbb{H}$-arc and $\mathbb{H}$-node colors of the vertices in Y and the cardinality of each vertex set that is a node, arc, or the intersection of a node and its incident arc can be updated in $O(d(Y))$ time. By Lemma 5.2 and the fact that Subroutine B is executed $O(n)$ times, the overall time for Step B1 is $O(m \log n)$.

Step B2: We first assume that we are given a set $\mathbb{C}$ of arcs of $\mathbb{H}$ whose union is a minimal pod C of Y in $\mathbb{H}$ and show how to implement Steps B2(a), B2(b), and B2(c) to run in overall $O(m \log n)$ time throughout Algorithm A. Let C be a V_1V_2-chunk of $\mathbb{H}$.

Step B2(a): It takes $O(|\mathbb{C}|)$ time to determine whether V_2 is incident to exactly one arc $F = VV_2$ in $\mathbb{C}$ and F is simple. We start from V to traverse the VV_2-rung $G[F]$ to obtain the node $v_2 \in N(Y, F)$ that is closest to V_2 in $G[F]$. The required time is linear in the number of traversed edges plus $d(Y)$. Observe that Step B2(a) in any remaining iteration of Algorithm A does not traverse these edges again. Moreover, the sum of $|\mathbb{C}|$ over all iterations of Algorithm A is $O(n)$. Thus, the overall time of Step B2(a) including that of calling SUBDIVIDE($\{v, v_2\}$) is $O(m \log n)$.

Step B2(b): Let $E_1, \ldots, E_k$ with $|E_1| \le \cdots \le |E_k|$ be the arcs of $\mathbb{H}$ in $\mathbb{C}$. We show how to implement Operation MERGE(C) in Step B2(b) to run in amortized $O(\log n)$ time: We create a new arc $E = V_1V_2$ in $\mathbb{H}^*$ consisting of all vertices in C and mark the original arcs $E_1, \ldots, E_k$ of $\mathbb{H}^*$ intersecting C dummy so that $\mathbb{H}^*$ is incremental as required by Lemma 5.2. The nodes of $\mathbb{H}$ whose incident arcs are all dummy are also marked dummy. The cardinalities of E, V_1, V_2, $E \cap V_1$, and $E \cap V_2$ can be obtained in $O(k)$ time. Since we do not keep an explicit list of the vertices in C, we simply let all vertices in C adopt the $\mathbb{H}$-color of the vertices in E_k. Each time a vertex v is recolored this way, the cardinality of the arc of $\mathbb{H}$ containing v is doubled. Observe that once a vertex in X loses its $\mathbb{H}$-node colors, it stays without any $\mathbb{H}$-node color for the rest of the algorithm. Combining with Lemma 5.2(2), Step B2(b) takes overall $O(n \log n)$ time throughout Algorithm A.

Step B2(c): The $\mathbb{H}$-arc and $\mathbb{H}$-node colors of the vertices of Y and the cardinalities of $E \cap V_1$ and $E \cap V_2$ can be updated in $O(d(Y))$ time.

Lemma 5.3 (Alstrup, Holm, Lichtenberg, and Thorup [3, §3.3]). *For any dynamic rooted n-knot tree, there is an $O(n)$-time obtainable data structure supporting the following operations and queries on $\mathbb{T}$ in amortized $O(\log n)$ time for any given distinct knots K_1 and K_2 of $\mathbb{T}$:*

1. *If K_2 is not a descendant of K_1, then make the subtree rooted at K_1 a subtree of K_2 such that K_2 becomes the parent of K_1.*
2. *Obtain the lowest common ancestor of K_1 and K_2.*
3. *If K_2 is a descendant of K_1, then obtain the child knot of K_1 that is an ancestor of K_2 in $\mathbb{T}$.*

It remains to show that it takes overall $O(m \log n \cdot \alpha(n,n))$ time to obtain the arc set $\mathbb{C}$ of a minimal pod C of an $\mathbb{H}$-podded Y in all iterations of Algorithm A. We additionally construct a data structure for $\mathbb{T}$ ensured by Lemma 5.3. By Lemmas 5.2 and 5.3(1), the overall time for updating the data structure reflecting the updates to $\mathbb{T}$ throughout algorithm A is $O(n \log n \cdot \alpha(n,n))$. Let $C^* = W_1W_2$ be the arc of $\mathbb{H}^\dagger$ with $V_1 = W_1 \subseteq N(Y,W_1) \cup C^*$. By Conditions P, C has to contain all arcs E of $\mathbb{H}$ with (1) $(E \setminus V_1) \cap N(Y,X) \neq \varnothing$ or (2) $(E \cap V_1) \setminus N(Y,X) \neq \varnothing$. Let $\mathbb{C}_1$ and $\mathbb{C}_2$ consist of the arcs of Types (1) and (2), respectively. It takes $O(d(Y))$ time to obtain $\mathbb{C}_1$ and the incident arcs of V_1 that are not of Type (1) or (2). It then takes $O(|\mathbb{C}_2|)$ time to obtain $\mathbb{C}_2$. By Lemma 5.3(2), it takes $O(|\mathbb{C}_1 \cup \mathbb{C}_2| \cdot \log n)$ time to obtain the lowest knot K of $\mathbb{T}$ with $\mathbb{C}_1 \cup \mathbb{C}_2 \subseteq \mathbb{C}(K)$. Since all arcs in $\mathbb{C}_1 \cup \mathbb{C}_2$ are merged into a single arc of $\mathbb{H}$ via $\textsc{merge}(C)$ at the end of the current iteration, the overall time for obtaining K throughout Algorithm A is $O(m \log n \cdot \alpha(n,n))$. It remains to show that $\mathbb{C}$ can be obtained from K in overall $O(m \log n \cdot \alpha(n,n))$ time throughout Algorithm A.

Case 1: K is an S-knot. Let $V_1V_2 \cdots V_\ell V_1$ with $\ell \geq 3$ be the cycle of K such that V_1 and V_ℓ are the poles of K. For each $i \in \{1,\ldots,\ell-1\}$, let K_i be the child knot of K with poles V_i and V_{i+1}, $\mathbb{C}_i = \mathbb{C}(K_1) \cup \cdots \cup \mathbb{C}(K_i)$, and let C_i be the union of the arcs in $\mathbb{C}_i$. Let j be the smallest index in $\{2,\ldots,\ell-1\}$ with $\mathbb{C}_1 \cup \mathbb{C}_2 \subseteq \mathbb{C}_j$. If $N(Y,X) \setminus (V_1 \cup C_{j-1}) = V_j \setminus C_{j-1}$, then $\mathbb{C} = \mathbb{C}_{j-1}$; otherwise, $\mathbb{C} = \mathbb{C}_j$. For the example in Figure 10(a), if $N(X,Y) \setminus (V_1 \cup E_1) = V_2 \setminus E_1$, then E_1 is a minimal pod of Y in $\mathbb{H}$; otherwise, $E_1 \cup E_2$ is a minimal pod of Y in $\mathbb{H}$. By Lemma 5.3(3), the time required to obtain the index j and determine whether $\mathbb{C} = \mathbb{C}_{j-1}$ or $\mathbb{C} = \mathbb{C}_j$ is dominated by the time of obtaining K plus the time of $\textsc{merge}(C)$.

Case 2: K is a P-knot. $\mathbb{C}$ equals the union of $\mathbb{C}(K')$ over all child knots K' of K in $\mathbb{T}$ with $(\mathbb{C}_1 \cup \mathbb{C}_2) \cap \mathbb{C}(K') \neq \varnothing$. For the example in Figure 10(b), $E_1 \cup E_2$ is a minimal pod of Y in $\mathbb{C}$. By Lemma 5.3(3), the time needed to obtain $\mathbb{C}$ is dominated by that of obtaining K.

Case 3: K is a Q-knot. As illustrated by Figure 10(c), $\mathbb{C} = \mathbb{C}(K)$ can be obtained in $O(1)$ time.

Case 4: K is an R-knot. If there is child knot K' of K in $\mathbb{T}$ with poles V_1 and V_2 such that all arcs of K intersecting $\mathbb{C}_1 \cup \mathbb{C}_2$ are incident to V_2 and $N(Y,X) \setminus (V_1 \cup C(K')) = V_2 \setminus C(K')$, then $\mathbb{C} = \mathbb{C}(K')$; otherwise, $\mathbb{C} = \mathbb{C}(K)$. For the example in Figure 10(d), if $N(Y,X) \setminus (V_1 \cup E_1) = V_2 \setminus E_1$, then E_1 is a minimal pod of Y in $\mathbb{H}$; otherwise, $E_1 \cup \cdots \cup E_5$ is a minimal pod of Y in $\mathbb{H}$. By Lemma 5.3(3), the time required to identify all possible vertices V_2, which can be at most two, is dominated by the time of identifying K. If there are no possible V_2, then we have $\mathbb{C} = \mathbb{C}(K)$. Otherwise, for each of the at most two vertices V_2, we spend $O(d(Y))$ time to determine whether the child knot K' with poles V_1 and V_2 satisfies $N(Y,X) \setminus (V_1 \cup C(K')) = V_2 \setminus C(K')$. For the positive (respectively, negative) case, we have $\mathbb{C} = \mathbb{C}(K')$ (respectively, $\mathbb{C} = \mathbb{C}(K)$).

Therefore, the overall time for obtaining the arc set of a minimal pod of Y in $\mathbb{H}$ is $O(m \log n \cdot \alpha(n,n))$. To complete our proof of Lemma 3.3, it remains to prove Lemma 5.1 in §5.4.

5.4 Proving Lemma 5.1

The subsection omits $\mathbb{H}^\dagger$ from the terms $\mathbb{H}^\dagger$-wild, $\mathbb{H}^\dagger$-tamed, $\mathbb{H}^\dagger$-untamed, and $\mathbb{H}^\dagger$-node and $\mathbb{H}^\dagger$-arc colors. Recall that each vertex x of X is associated with exactly one arc color and at most two node colors from which we know which arc E of $\mathbb{H}^\dagger$ contains x and whether $x \in E \cap V$ holds for each end-node V of E. For any nonempty $S \subseteq X$, we say that an $R \subseteq S$ *represents* S and call R a *representative set* of S if $|R| \leq 3$ and, for any $V \subseteq X$, $R \cup V$ is tamed if and only if $S \cup V$ is tamed.

If S is untamed, then each untamed two-vertex subset of S represents S. If R_1 represents S_1, R_2 represents S_2, and R represents $R_1 \cup R_2$, then R represents $S_1 \cup S_2$.

Lemma 5.4. *Any nonempty $S \subseteq X$ admits a representative set obtainable from the colors of the vertices of S in $O(|S|)$ time.*

Proof. Let $E_1, \ldots, E_\ell$ be the arcs of $\mathbb{H}^\dagger$ intersecting S. If $\ell = 1$, then S is tamed. Let V_1 and V_2 be the end-nodes of E_1. Choose an arbitrary vertex from each of the sets $S \cap V_1$, $S \cap V_2$, and $S \setminus (V_1 \cup V_2)$ that are nonempty to form a representative set of S. The rest of the proof assumes $\ell \geq 2$. It takes $O(|S|)$ time to either (1) identify distinct i and j in $\{1, \ldots, \ell\}$ such that E_i and E_j do not share a common end-node or (2) ensure that E_i and E_j for any distinct i and j in $\{1, \ldots, \ell\}$ share a common end-node. Case 1 implies that S is untamed and any two-vertex subset of S intersecting both E_i and E_j represents S.

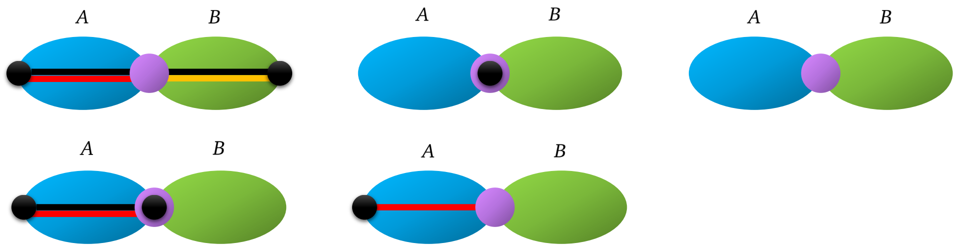
Case 2(a): $E_1, \ldots, E_\ell$ have a common end-node V. If $S \not\subseteq V$, then S is untamed and any $\{u, v\} \subseteq S$ with $u \notin V$ intersecting distinct arcs represents S. If $S \subseteq V$, then S is tamed. If $\ell = 2$, then any two-vertex subset of S intersecting both of E_1 and E_2 represents S. If $\ell \geq 3$, then any three-vertex subset of S intersecting all of E_1, E_2, and E_3 represents S.

Case 2(b): $E_1, \ldots, E_\ell$ have no common end-node. Therefore, $\ell = 3$ and E_1, E_2, and E_3 form a triangle. For indices i, j, k with $\{i, j, k\} = \{1, 2, 3\}$, let V_i and V_j be the end-nodes of E_k. If $S \subseteq \Delta(V_1, V_2, V_3)$, then S is tamed and any three-vertex subset of S intersecting all of E_1, E_2, and E_3 represents S. If $S \not\subseteq \Delta(V_1, V_2, V_3)$, then S is untamed and $\{u, v\}$ with $u \in (S \cap E_i) \setminus V_j$ and $v \in S \cap E_k$ for $\{i, j, k\} = \{1, 2, 3\}$ represents S. $\square$

For each $y \in V(G - X)$, we maintain a balanced binary search tree T_y on $N(y, X)$. For each vertex x of T_y, we maintain a representative set $R_y(x)$ of the vertices in the subtree of T_y rooted at x. Thus, $R_y = R_y(\text{root}(T_y))$ represents $N(y, X)$. We also maintain a doubly linked list D_1 for the vertices $y \in V(G - X)$ with untamed $N(y, X)$. When a vertex joins $N(y, X)$ or a vertex in $N(y, X)$ changes color, R_y and D_1 can be updated in $O(\log n)$ time by Lemma 5.4. Thus, as long as $D_1 \neq \varnothing$, $\mathbb{H}^\dagger$ is not taming and an $\mathbb{H}^\dagger$-wild set consisting of a single vertex can be obtained from D_1 in $O(1)$ time, implying Lemmas 5.1(1), 5.1(2), and 5.1(3). The rest of the subsection handles the case $D_1 = \varnothing$.

Lemma 5.5 (Holm, de Lichtenberg, and Thorup [58]). *A spanning forest of an n-vertex dynamic graph can be maintained in amortized $O(\log^2 n)$ time per edge insertion and deletion such that each update to the graph only adds and deletes at most one edge in the spanning forest.*

We maintain a spanning forest F of the decremental graph $G - X$ by Lemma 5.5. For each maximal connected $U \subseteq V(F)$, we maintain a balanced binary search tree T_U on U. For each $y \in U$, we maintain a representative set $R_U(y)$ for the union of R_z over all vertices z in the subtree of T_U rooted at y. It takes $O(1)$ time to determine if U is tamed from $R_U = R_U(\text{root}(T_U))$. We also maintain a doubly linked list D_2 for the untamed maximal connected subsets U of $V(F)$. When R_y for a vertex $y \in V(G - X)$ changes, D_2 and R_U for the maximal connected $U \subseteq V(F)$ containing y can be updated in $O(\log n)$ time by Lemma 5.4. If deleting an edge of F decomposes a maximal connected $U \subseteq V(F)$ into U_1 and U_2 with $|U_1| \leq |U_2|$, then it takes $O(|U_1| \log n)$ time to delete the vertices of U_1 from T_U, construct T_{U_1}, and obtain R_{U_1}. The resulting T_U and R_U become T_{U_2} and R_{U_2}. D_2 can be updated in $O(1)$ time. Whenever a vertex y moves to a new connected component, the number of vertices of the connected component containing y is halved. Hence, the T_U for all maximal connected sets $U \subseteq V(F)$ are changed overall $O(n \log n)$ times. Thus, the overall time throughout the algorithm to maintain D_2 and all representative sets R_U is $O(n \log^2 n)$, not affecting the correctness of Lemmas 5.1(1) and 5.1(2) and the first half of Lemma 5.1(3). It remains to



(a) $|\partial C| = 2$ (b) $|\partial C| = 1$ (c) $|\partial C| = 0$

Figure 11: The cases of joining the child clusters A and B with $|\partial A| \ge |\partial B|$ into their parent cluster $C = A \cup B$ on a top tree. The first row shows the three cases with $|\partial A| = |\partial B|$. The second row shows the two cases with $|\partial A| > |\partial B|$. The vertex in $A \cap B$ is in purple. The vertices in ∂C are in black. If $|\partial C| = 2$, then the black line indicates $\Pi(C)$. If $|\partial A| = 2$, then the red line indicates $\Pi(A)$. If $|\partial B| = 2$, then the yellow line indicates $\Pi(B)$.

prove the second half of Lemma 5.1(3) for the case $D_1 = \varnothing$ and $D_2 \neq \varnothing$, i.e., each $N(y, X)$ with $y \in V(G - X)$ is tamed and $\mathbb{H}^\dagger$ is not taming.

A top tree is defined over a dynamic tree T and a dynamic set ∂T of at most two vertices of T. For any subtree C of T, $\partial C = \partial_{(T,\partial T)} C$ consists of the vertices of C belonging to ∂T or adjacent to $V(T) \setminus V(C)$. A *cluster* [3] of $(T, \partial T)$ is a subtree C of T with $|E(C)| \ge 1$ and $|\partial C| \le 2$. If $|\partial C| = 2$, then let $\Pi(C)$ denote the path of T between the vertices of ∂C. If $|E(T)| = 0$, then $(T, \partial T)$ admits no cluster and the *top tree* over $(T, \partial T)$ is empty. If $|E(T)| \ge 1$, then a *top tree* $\mathscr{T}$ over $(T, \partial T)$ is a binary tree on clusters of $(T, \partial T)$ such that (1) the root of $\mathscr{T}$ is the maximal cluster T of $(T, \partial T)$, (2) the leaves of $\mathscr{T}$ are the edges of T, i.e., the minimal clusters of $(T, \partial T)$, and (3) the children A and B of any cluster C of $(T, \partial T)$ on $\mathscr{T}$ are edge disjoint clusters of $(T, \partial T)$ with $C = A \cup B$ and $|V(A) \cap V(B)| = 1$. Figure 11 illustrates all possible cases of joining child clusters A and B into their parent cluster C on $\mathscr{T}$. If $|\partial A| = |\partial C| = 2$, then $\Pi(A) \subseteq \Pi(C)$. Moreover, $\Pi(A) = \Pi(C)$ if and only if $|\partial B| \le 1$. For each vertex $v \in V(T) \setminus \partial T$, let C_v denote the lowest cluster of $(T, \partial T)$ on $\mathscr{T}$ with $v \in V(C_v) \setminus \partial C_v$. If $|\partial C| = 2$, then $v \in V(C)$ is an internal vertex of $\Pi(C)$ if and only if $|\partial A| = 2$ holds for every cluster A on the CC_v-path of $\mathscr{T}$. A *top forest* $\mathscr{F}$ over a forest F consists of top trees, one for each maximal subtree of F. According to Lemma 5.5, each update to F either deletes an edge of F or adds an edge between two maximal subtrees of F. In addition to that, $\mathscr{F}$ also needs be modified if ∂T for a maximal subtree T of F is updated. To accommodate each update to F or ∂T, we modify $\mathscr{F}$ via a sequence of operations such that there can be temporary top trees $\mathscr{T}_C$ rooted at clusters C that are not maximal subtrees of F. Specifically, $\mathscr{F}$ is modified via the following $O(1)$-time *top-tree operations*:

- *Create* or *destroy* a top tree on a single cluster that is an edge.
- *Split* a top tree $\mathscr{T}_C$ into the two immediate subtrees of $\mathscr{T}_C$ by deleting the root C.
- *Merge* top trees $\mathscr{T}_A$ and $\mathscr{T}_B$ with $|V(A) \cap V(B)| = 1$ into a top tree $\mathscr{T}_C$ rooted at $C = A \cup B$.

Lemma 5.6 (Alstrup, Holm, de Lichtenberg, and Thorup [3]). *An n-vertex forest F admits an $O(n)$-space top forest $\mathscr{F}$ consisting of $O(\log n)$-height top trees such that for any maximal subtree T of F,*

1. *it takes $O(1)$ time to obtain on the top tree $\mathscr{T}$ for T (a) the cluster C_v for any $v \in V(T) \setminus \partial T$, (b) the parent of a nonroot cluster, (c) the children of a non-leaf cluster, and (d) ∂C for a cluster C and*
2. *it takes $O(\log n)$ time to identify a sequence of $O(\log n)$ top-tree operations with which $\mathscr{F}$ can be modified in $O(\log n)$ time with respect to (a) updating ∂T, (b) deleting an edge of T, or (c) adding an edge between T and another maximal subtree of F.*

We use Lemma 5.6 to maintain a top forest $\mathscr{F}$ over the spanning forest F of $G-X$ maintained by Lemma 5.5. For each cluster C on each nonempty top tree $\mathscr{T}$ of $\mathscr{F}$, we maintain a representative set R_C of $N(V(C)\setminus\partial C, X)$. We first show that maintaining the representative sets R_C does not affect the complexity of maintaining $\mathscr{F}$ stated in Lemma 5.6 and that of maintaining the colors of the vertices of X stated in Lemmas 5.1(1) and 5.1(2). By Lemma 5.4, the following *bottom-up update* for a cluster B on a top tree $\mathscr{T}$ of $\mathscr{F}$ takes $O(\log n)$ time: For each cluster C on the BT-path of $\mathscr{T}$ from B to T, if C is an edge uv of T, then an R_C can be obtained from $R_u \cup R_v$ in $O(1)$ time; if C is not an edge of T, then an R_C can be obtained from $R_{C_1} \cup R_{C_2} \cup R_c$ in $O(1)$ time, where C_1 and C_2 are the children of C on $\mathscr{T}$ and c is the vertex in $V(C_1)\cap V(C_2)$. Hence, the initial R_C for all clusters C of all top trees $\mathscr{T}$ of $\mathscr{F}$ can be obtained in overall $O(m\log n)$ time by performing a bottom-up update for each leaf cluster of each top tree. With respect to each top-tree operation, the representative sets R_C can be updated in $O(1)$ time: For destroy and split, we simply delete R_C together with the root C of $\mathscr{T}_C$. For create and merge, we just perform a bottom-up update for C in $O(1)$ time. Thus, maintaining the representative sets R_C does not affect the complexity of maintaining $\mathscr{F}$ stated in Lemma 5.6. If a vertex $v \in V(G-X)$ moves to X or a vertex $v \in X$ changes color, we update R_C for all $O(d(v)\log n)$ clusters C with $v \in N(V(C)\setminus\partial C, X)$. Specifically, for each of the $O(d(v))$ vertices $y \in V(G-X)$ with $v \in N(y,X)$, we perform a bottom-up update for C_y in $O(\log n)$ time. Thus, maintaining the representative sets R_C does not affect the correctness of Lemmas 5.1(1) and 5.1(2).

The rest of the subsection proves the second half of Lemma 5.1(3) for the case $D_1 = \varnothing$ and $D_2 \neq \varnothing$ in two steps. Let $T = F[U]$ for an arbitrary U kept in D_2. Step 1 calls TREE-WILD(T) to obtain $\{u,w\}$ for distinct vertices u and w of T such that the vertices of the uw-path of T is a minimal untamed connected vertex set of T. Step 2 calls GRAPH-WILD$(\{u,w\})$ to obtain a minimal untamed set Y such that $G[Y]$ is a uw-path of G.

Step 1: Let $\mathscr{T}$ be the top tree of $\mathscr{F}$ for T. For any cluster C on $\mathscr{T}$, let $R_{\partial C}$ be the union of R_v over the vertices $v \in \partial C$. Let CLOSEST(S,C,c) for

- a tamed set $S \subseteq X$ with $|S| \leq 6$,
- a cluster C on $\mathscr{T}$ with untamed $S \cup R_C \cup R_{\partial C}$, and
- a vertex $c \in \partial C$ such that $S \cup R_c$ is tamed

be the following $O(\log n)$-time recursive algorithm that outputs a $y \in V(C)$ such that

- $S \cup R_y$ is untamed and
- $S \cup R_z$ is tamed for every internal vertex z of the yc-path of T:

If C is an edge bc, then return b. If C is not an edge, then let C_1 and C_2 be the children of C and let b be the vertex in $C_1 \cap C_2$. If there is an $i \in \{1,2\}$ with $c \in \partial C_i$ such that $S \cup R_{C_i} \cup R_{\partial C_i}$ is untamed, then return CLOSEST(S,C_i,c). Otherwise, we have $b \neq c$ and that $S \cup R_{C_i} \cup R_{\partial C_i}$ is untamed for the index $i \in \{1,2\}$ with $c \notin \partial C_i$. Return CLOSEST(S,C_i,b).

Let TREE-WILD(C) for a cluster C on $\mathscr{T}$ with untamed $R_C \cup R_{\partial C}$ be the following recursive subroutine: If C is an edge uw of T, then return $\{u,w\}$. Otherwise, let C_1 and C_2 be the children of C on $\mathscr{T}$. If there is an $i \in \{1,2\}$ with untamed $R_{C_i} \cup R_{\partial C_i}$, then return TREE-WILD$(C_i)$. Otherwise, $R_C \cup R_{\partial C}$ is untamed and $R_{C_1} \cup R_{\partial C_1}$ is tamed. Let c be the vertex in $V(C_1)\cap V(C_2)$. Call CLOSEST$(R_{C_1} \cup R_{\partial C_1}, C_2, c)$ to obtain in $O(\log n)$ time a $w \in V(C_2)$ such that

- $R_{C_1} \cup R_{\partial C_1} \cup R_w$ is untamed and
- $R_{C_1} \cup R_{\partial C_1} \cup R_v$ is tamed for every internal vertex v of the wc-path of T.

Call CLOSEST(R_w, C_1, c) to obtain in $O(\log n)$ time a $u \in V(C_1)$ such that

- $R_w \cup R_u$ is untamed and
- $R_w \cup R_v$ is tamed for every internal vertex v of the uc-path of T.

Let P be the uw-path of T. $V(P)$ is a minimally untamed subset of $V(T)$ that is connected in T: Let u' and w' be distinct vertices of $V(P)$ with $\{u',w'\} \neq \{u,w\}$ such that $R_{u'} \cup R_{w'}$ is untamed and u' is closer to u than w in P. Since $R_{C_1} \cup R_{\partial C_1}$ and $R_{C_2} \cup R_{\partial C_2}$ are both tamed, we have $u' \in V(C_1) \setminus \partial C_1$ and $w' \in V(C_2) \setminus \partial C_2$. Since $R_{C_1} \cup R_{\partial C_1} \cup R_v$ is tamed for every internal vertex v of the wc-path of T and $u' \in V(C_1)$, we have $w' = w$. Since $R_w \cup R_v$ is tamed for every internal vertex v of the uc-path of T, we have $u' = u$.

Step 2: To obtain in $O(d(Y)\log n)$ time a set Y such that $G[Y]$ is a uw-path of $G - X$, it suffices to show an $O(d(u)\log n)$-time subroutine $\textsc{jump}(u,w)$ returning for any distinct vertices u and w of T the vertex $v \in N_G(u, V(P))$ that is closest to w in the uw-path P of T: With $Y = \{u\}$ initially, we repeatedly add $v = \textsc{jump}(u,w)$ into Y and let $u = v$ until $v = w$. The subroutine $\textsc{jump}(u,w)$ starts with updating $\mathscr{T}$ for setting $\partial T = \{u,w\}$ in $O(\log n)$ time by Lemma 5.6(2). Recall that $U = N_G(u, V(P - w))$ consists of the vertices $v \in N_G(u)$ such that $|\partial B| = 2$ holds for every cluster B on the TC_v-path of $\mathscr{T}$. By Lemma 5.6(1), it takes $O(d(u)\log n)$ time for $\textsc{jump}(u,w)$ to obtain U and the set $\mathscr{C}$ consisting of the clusters on the TC_v-path of $\mathscr{T}$ for all vertices $v \in U$. If $U = \varnothing$, then $\textsc{jump}(u,w)$ returns w, since uw is an edge of T. If $U \neq \varnothing$, then $\textsc{jump}(u,w)$ returns $v = \textsc{next}(T,w)$, where $\textsc{next}(C,w)$ for a cluster $C \in \mathscr{C}$ and a vertex $w \in \partial C$ is the following $O(\log n)$-time recursive subroutine: If $w \in N_G(u)$, then $\textsc{next}(C,w)$ returns w. If $w \notin N_G(u)$, then C is not an edge of T. Let C_1 and C_2 be the children of C on $\mathscr{T}$ with $w \in \partial C_2 \setminus \partial C_1$. Let c be the vertex in $V(C_1) \cap V(C_2)$. If $C_2 \in \mathscr{C}$, then $\textsc{next}(C,w)$ returns $\textsc{next}(C_2,w)$; otherwise, $\textsc{next}(C,w)$ returns $\textsc{next}(C_1,c)$.

6 Improved graph recognition and detection algorithms

Section 6.1 gives our algorithms for detecting thetas, pyramids, and beetles. Section 6.2 gives our algorithms for recognizing perfect graphs and detecting odd holes. Section 6.3 gives our algorithm for detecting even holes.

6.1 Improved theta, pyramid, and beetle detection

Each previous algorithm for detecting a family $\mathbb{F}$ of graphs in G via the three-in-a-tree algorithm identifies a set $\mathbb{G}$ of a polynomial number of subgraphs H of G, each associated with a set $L(H)$ of three terminals, such that G is $\mathbb{F}$-free if and only if each graph H in $\mathbb{G}$ does not admit an induced tree containing $L(H)$. In addition to Theorem 1.1, our improvement are obtained via exploiting that the graphs H in $\mathbb{G}$ need not be subgraphs of G. For instance, if $\mathbb{F}$ are thetas, then Chudnovsky and Seymour [28] obtained a set $\mathbb{G}$ of $O(n^7)$ subgraphs of G. Each $H \in \mathbb{G}$ with $L(H) = \{a_1,a_2,a_3\}$ is uniquely determined from vertices b, b_1, b_2, b_3, a_1, a_2, and a_3 of G such that bb_1, bb_2, bb_3, a_1b_1, a_2b_2, and a_3b_3 are the distinct edges of $G[\{b,b_1,b_2,b_3,a_1,a_2,a_3\}]$. We observe that the requirement that a_1b_1, a_2b_2, and a_3b_3 are the distinct edges of $G[\{a_1,a_2,a_3,b_1,b_2,b_3\}]$ can be achieved by making the neighbors of each b_i with $i \in \{1,2,3\}$ in $V(G) \setminus \{b,b_1,b_2,b_3\}$ a clique. As a result, each $H \in \mathbb{G}$ is determined from four vertices b, b_1, b_2, and b_3 such that bb_1, bb_2, and bb_3 are the distinct edges of $G[\{b,b_1,b_2,b_3\}]$. Thus, there is a set $\mathbb{G}$ of $O(n^4)$ n-vertex graphs H with $L(H) = \{b_1,b_2,b_3\}$ such that G is theta-free if and only if each graph H in $\mathbb{G}$ does not admit an induced tree containing $L(H)$. An n^3-factor is reduced from the number of the three-in-a-tree problems to be solved in order to determine whether G is theta-free. Beetle detection can be improved similarly. Improving the algorithm for pyramid detection needs additional care, since a pyramid has to contain exactly one triangle.

6.1.1 Proving Theorem 1.2

Theorem 1.2 is immediate from Theorem 1.1 and the next lemma.

Lemma 6.1. *Thetas in an n-vertex m-edge graph G can be detected by solving the three-in-a-tree problem on $O(mn^2)$ linear-time-obtainable n-vertex graphs.*

Proof. Observe that H is a theta of G if and only if there are vertices b, b_1, b_2, and b_3 of H such that bb_1, bb_2, and bb_3 are the distinct edges of $G[\{b, b_1, b_2, b_3\}]$ and $H - b$ is an induced subtree of $G - b$ having exactly three leaves b_1, b_2, and b_3. See Figure 2(a). For each of the $O(mn^2)$ choices of vertices b, b_1, b_2, and b_3 such that bb_1, bb_2, and bb_3 are the distinct edges in $G[\{b, b_1, b_2, b_3\}]$, let $G(b, b_1, b_2, b_3)$ denote the graph that is $O(m+n)$-time obtainable from G by (1) deleting $N[b] \setminus \{b_1, b_2, b_3\}$ and (2) adding edges to make the remaining vertices in each $N(b_i)$ with $i \in \{1, 2, 3\}$ a clique. We show that G admits a theta H if and only if one of the $O(mn^2)$ graphs $G^* = G(b, b_1, b_2, b_3)$ admits an induced subtree T^* containing $\{b_1, b_2, b_3\}$.

($\Rightarrow$) $G^* = G(b, b_1, b_2, b_3)$ exists for the vertices b, b_1, b_2, and b_3 of H. The vertices deleted from G in Step (1) are not in $T = H - b$, implying that T is a subtree of G^* containing $\{b_1, b_2, b_3\}$. Since b_1, b_2, and b_3 are the leaves of T, each edge added by Step (2) is incident to at most one vertex of T, implying that T is an induced subtree T^* of G^* containing $\{b_1, b_2, b_3\}$.

($\Leftarrow$) The distinct edges of $G[\{b, b_1, b_2, b_3\}]$ are bb_1, bb_2, and bb_3. By Step (2), b_1, b_2, and b_3 are the leaves of T^*. Since each edge deleted in Step (1) is incident to at most one vertex of T^*, T^* is an induced subtree of $G - b$, implying that $G[T^* \cup \{b\}]$ is a theta H of G. □

6.1.2 Proving Theorem 1.3

A *pyramid* [28] of graph G is the subgraph of G induced by the vertices of an induced subtree T of $G - \{b_1b_2, b_2b_3, b_3b_1\}$ having exactly three leaves b_1, b_2, and b_3 such that $G[\{b_1, b_2, b_3\}]$ is the only triangle of $G[T]$. See Figure 2(b). Theorem 1.3 is immediate from Theorem 1.1 and the next lemma.

Lemma 6.2. *Pyramids in an n-vertex m-edge graph G can be detected by solving the three-in-a-tree problem on $O(mn)$ linear-time-obtainable n-vertex graphs.*

Proof. For each of the $O(mn)$ choices of distinct vertices b_1, b_2, and b_3 such that $G[\{b_1, b_2, b_3\}]$ is a triangle, let $G(b_1, b_2, b_3)$ be the graph obtained from G by (1) adding edges to make each $N(b_i) \setminus \{b_1, b_2, b_3\}$ with $i \in \{1, 2, 3\}$ a clique, (2) deleting edges b_1b_2, b_2b_3, and b_3b_1, and (3) deleting $(N(b_i) \cap N(b_j)) \setminus \{b_1, b_2, b_3\}$ for any distinct indices i and j in $\{1, 2, 3\}$. We show that G admits a pyramid H if and only if one of the $O(mn)$ graphs $G^* = G(b_1, b_2, b_3)$ admits an induced subtree T^* containing $\{b_1, b_2, b_3\}$.

($\Rightarrow$) G^* exists for the vertices b_1, b_2, and b_3 of H. Since $H[\{b_1, b_2, b_3\}]$ is the only triangle of H, H does not intersect any $(N(b_i) \cap N(b_j)) \setminus \{b_1, b_2, b_3\}$ with $1 \le i < j \le 3$. Hence, Steps (2) and (3) do not delete any edge of T, implying that T is a subtree of G^*. Since T is an induced tree of $G - \{b_1b_2, b_2b_3, b_3b_1\}$ having exactly three leaves b_1, b_2, and b_3, each edge added by Step (1) is incident to at most one vertex of T. Thus, T is an induced subtree T^* of G^* containing $\{b_1, b_2, b_3\}$.

($\Leftarrow$) By Step (1), vertices b_1, b_2, and b_3 are the leaves of the subtree T^* of G. Since each edge deleted in Step (3) is incident to at most one vertex of T^*, T^* is an induced subtree of $G - \{b_1b_2, b_2b_3, b_3b_1\}$ by Step (2). By Steps (2) and (3), $G[\{b_1, b_2, b_3\}]$ is the only triangle of $G[T^*]$. Thus, $G[T^*]$ is a pyramid H of G. □

6.1.3 Proving Theorem 1.5

A *beetle* [15] of graph G is an induced subgraph of G consisting of a cycle $b_1b_2b_3b_4b_1$ with a chord b_2b_4 (i.e., a *diamond* [36, 62] of G) and a tree T of $G - b_4$ having exactly three leaves b_1, b_2, and b_3. See Figure 2(c). Theorem 1.5 is immediate from Theorem 1.1 and the next lemma.

Lemma 6.3. *Beetles in an n-vertex m-edge graph G can be detected by solving the three-in-a-tree problem on $O(m^2)$ linear-time-obtainable n-vertex graphs.*

Proof. For each of the $O(m^2)$ choices of vertices b_1, b_2, b_3, and b_4 such that $G[\{b_1, b_2, b_3, b_4\}]$ is a cycle $b_1b_2b_3b_4b_1$ with exactly one chord b_2b_4, let $G(b_1, b_2, b_3, b_4)$ be the $O(m+n)$-time obtainable graph from G by (1) deleting $N[b_4] \setminus \{b_1, b_2, b_3\}$ and (2) adding edges to make the remaining vertices in each $N(b_i) \setminus \{b_1, b_2, b_3\}$ with $i \in \{1, 2, 3\}$ a clique. We show that G admits a beetle H if and only if one of the $O(m^2)$ graphs $G(b_1, b_2, b_3, b_4)$ admits an induced subtree T^* containing $\{b_1, b_2, b_3\}$.

($\Rightarrow$) $G^* = G(b_1, b_2, b_3, b_4)$ exists for the vertices b_1, b_2, b_3, and b_4 of H. The vertices deleted fro G in Step (1) are not in T, implying that T is a subtree of G^* containing $\{b_1, b_2, b_3\}$. Since T intersects each $N(b_i) \setminus \{b_1, b_2, b_3\}$ with $i \in \{1, 2, 3\}$ at exactly one vertex, each edge added by Step (2) is incident to at most one vertex of T. Thus, T is an induced subtree T^* of G^* containing $\{b_1, b_2, b_3\}$.

($\Leftarrow$) $G[\{b_1, b_2, b_3, b_4\}]$ is a cycle $b_1b_2b_3b_4b_1$ with exactly one chord b_2b_4. By Step (2), b_1, b_2, and b_3 are the leaves of T^*. Since each edge deleted in Step (1) is incident to at most one vertex of T^*, we have $G[T^*] = T^* \cup \{b_1b_2, b_2b_3\}$, implying that $G[T^* \cup \{b_4\}]$ is a beetle H of G. $\square$

6.2 Improved perfect-graph recognition and odd-hole detection

As summarized by Maffray and Trotignon [68, §2], the algorithm of Chudnovsky et al. [18] consists of two $O(n^9)$-time phases. The first phase (a) detects pyramids in G in $O(n^9)$ time, (b) detects the so-called $\mathcal{T}_i$ configurations with $i \in \{1, 2, 3\}$ in $O(n^6)$ time,[1] and (c) detects jewels in $\bar{G}$ in $O(n^6)$ time. If any of them is detected, then either G or $\bar{G}$ contains odd holes, implying that G is not perfect. Otherwise, each shortest odd hole C of G is amenable, i.e., any anti-connected component of the C-major vertices is contained by $N_G(u) \cap N_G(v)$ for some edge uv of C. The second phase (a) computes in $O(n^5)$ time a set $\mathbb{X}$ of $O(n^5)$ subsets of $V(G)$ such that if G contains an amenable shortest odd hole, then $\mathbb{X}$ contains a near cleaner of G and (b) spends $O(n^4)$ time on each $X \in \mathbb{X}$ to either obtain an odd hole of G or ensure that X is not a near cleaner of G. Theorem 1.3 reduces the time of detecting pyramids to $O(n^6)$. Lemma 6.5 reduces the time of Phase 2(b) from $O(n^4)$ to the time of performing $O(n)$ multiplications of Boolean $n \times n$ matrices [38, 64, 82]. Therefore, the time of recognizing perfect graphs is already reduced to $O(n^{8.377})$ without resorting to our improved odd-hole detection algorithm.

Let G be an n-vertex m-edge graph. A *k-hole* (respectively, *k-cycle* and *k-path*) is a k-vertex hole (respectively, cycle and path). For any odd hole C of G, a vertex $x \in V(G) \setminus V(C)$ is *C-major* [18] if $N_G(x, C)$ is not contained by any 3-path of C. Let $M_G(C)$ consist of the C-major vertices. We have $M_G(C) \cap V(C) = \varnothing$. A shortest odd hole C of G is *clean* if G does not contain any C-major vertex. A set $X \subseteq V(G)$ is a *near cleaner* [18] if there is a shortest odd hole C of G such that (1) $C[X]$ is contained by a 3-path of C and (2) all C-major vertices of G are in X. A jewel of G is an $O(n^6)$-time detectable induced subgraph of G [18]. If G contains jewels or beetles, then G contains odd holes. Let $\bar{G}$ denote the complement of graph G.

[1] in [68] we omit the complicated definitions of $\mathcal{T}_i$ configurations, which are not needed by our improved algorithms.

Lemma 6.4 (Chudnovsky, Cornuéjols, Liu, Seymour, and Vušković [18, 4.1])**.** *Let u and v be distinct vertices of a clean shortest odd hole C of a pyramid-free jewel-free graph G. (1) The shortest uv-path of C is a shortest uv-path of G. (2) The graph obtained from C by replacing the shortest uv-path of C with a shortest uv-path of G remains a clean shortest odd hole of G.*

6.2.1 An improved algorithm for recognizing perfect graphs

Although Theorem 1.4(1) implies Theorem 1.4(2), this subsection shows that we already have an improved algorithm for recognizing perfect graphs without resorting to Theorem 1.4(1). The next lemma reduces the time of Chudnovsky et al.'s algorithms [18, 4.2 and 5.1] from $O(n^4)$ to $O(n^{3.377})$.

Lemma 6.5. *For any given vertex set X of an n-vertex pyramid-free jewel-free graph G, it takes the time of performing $O(n)$ multiplications of $n \times n$ Boolean matrices to either obtain an odd hole of G or ensure that X is not a near cleaner of a shortest odd hole of G.*

Proof. It takes overall $O(n^3)$ time to obtain for any distinct vertices u and v of G that are connected in $G(u,v) = G-(X\setminus\{u,v\})$ (i) the length $d(u,v)$ of a shortest uv-path $P(u,v)$ in $G(u,v)$ and (ii) the neighbor $N(u,v)$ of u in $P(u,v)$. Assume $P(u,v) = P(v,u)$ for all u and v without loss of generality. If u and v are not connected in $G(u,v)$, then let $d(u,v)=\infty$. It takes overall $O(n^3)$ time to compute for any distinct vertices x and y of G the set $Z(x,y)$ represented by an n-bit array, consisting of the vertices z of G with $d(z,x)=1$ and $d(z,y)>d(x,y)$. If

$$\begin{array}{rcl} d(x_1,x_2) & \geq & 2 \\ d(x_1,y_1) & = & d(x_2,y_2) \;=\; d(x_1,y_2)-1 \;=\; d(x_2,y_1)-1 \\ Z(x_1,y_1)\cap Z(x_2,y_2) & \neq & \varnothing \end{array} \tag{1}$$

with $y_1 = N(y_2,x_1)$ hold for any distinct vertices x_1, x_2, and y_2 with minimum $d(x_2,y_2)$, then the $O(n^2)$-time obtainable $C = G[P(x_1,y_1)\cup P(x_2,y_2)\cup\{z\}]$ for any $z\in Z(x_1,y_1)\cap Z(x_2,y_2)$ is an odd hole of G: Paths $P(x_1,y_1)$ and $P(x_2,y_2)$ are chordless. By $z\in Z(x_1,y_1)\cap Z(x_2,y_2)$, the only neighbors of z in C are x_1 and x_2. By $d(x_1,x_2)\geq 2$, $d(x_i,y_i) = d(x_i,y_{3-i})-1$ for each $i\in\{1,2\}$, and the minimality of $d(x_2,y_2)$, the only edge between $P(x_1,y_1)$ and $P(x_2,y_2)$ is y_1y_2. Thus, C is an odd hole of G. For each y_2, we construct a directed acyclic tripartite graph $G(y_2)$ on three n-vertex sets X_1, Z, X_2 such that (1) x_1z with $x_1\in X_1$ and $z\in Z$ is a directed edge of $G(y_2)$ if and only if $z\in Z(x_1,N(y_2,x_1))$ and (2) zx_2 with $z\in Z$ and $x_2\in X_2$ is a directed edge of $G(y_2)$ if and only if $z\in Z(x_2,y_2)$. It takes the time of multiplying two $n\times n$ Boolean matrices to obtain the $O(n^2)$ pairs of reachability in $G(y_2)$ from X_1 to X_2. Thus, the time required to determine whether there is a choice of x_1, x_2, and y_2 satisfying Equation (1) is that of performing $O(n)$ multiplications for $n\times n$ Boolean matrices.

It remains to show that such a choice of x_1, x_2, and y_2 exists for the case that X is a near cleaner of a shortest odd hole C of G. Let P be a 3-path of C such that $C-V(P)$ does not intersect the C-major vertices of G, implying that C is a clean shortest odd hole of $H = G-(X\setminus V(P))$. Let x_1 and x_2 be the end-vertices of P. Let y_2 be the vertex of C such that the shortest x_1y_2-path of C is one edge longer than the shortest x_2y_2-path of C. By Lemma 6.4, each shortest x_iy_2-path P_i of C with $i\in\{1,2\}$ is a shortest x_iy_2-path of H. Since X does not intersect the interior of P_1 and P_2, each $P(x_i,y_2)$ with $i\in\{1,2\}$ is a shortest x_iy_2-path of H. Applying Lemma 6.4(2) on C to replace P_i with $P(x_i,y_2)$ for each $i\in\{1,2\}$, we obtain a clean shortest odd hole C^* of H, via which one can verify Equation (1) for the chosen x_1, x_2, and y_2: Let $y_1 = N(y_2,x_1)$. Since C^* is chordless in G, $d(x_1,x_2)\geq 2$. Since X does not intersect the vertices of C^* other than x_1, x_2, and the internal vertex z of the shortest x_1x_2-path of C^*, we have $d(x_1,y_1) = d(x_2,y_2) = d(x_1,y_2)-1 = d(x_2,y_1)-1$ by Lemma 6.4(1). We have $d(z,x_1) = d(z,x_2) = 1$. By Lemma 6.4(1), $d(z,y_i) > d(x_i,y_i)$ for both

$i \in \{1,2\}$ or else the shortest zy_i-path of C^* for an $i \in \{1,2\}$ would not be a shortest zy_i-path of H. Thus, $z \in Z(x_1, y_1) \cap Z(x_2, y_2)$. □

Lemma 6.6 (Chudnovsky, Cornuéjols, Liu, Seymour, and Vušković [18]). *Let G be an n-vertex graph such that G and $\bar{G}$ are pyramid-and-jewel-free. It takes $O(n^6)$ time to (1) ensure that G contains odd holes or (2) obtain a set $\mathbb{X}$ of $O(n^5)$ vertex subsets of G such that if G contains odd holes, then $\mathbb{X}$ contains a near cleaner of G.*

By Theorem 1.3, it takes $O(n^6)$ time to detect pyramids or jewels in G and $\bar{G}$. If G or $\bar{G}$ contains pyramids or jewels, then G is not perfect. By Lemma 6.6, it suffices to consider the case that we are given a set $\mathbb{X}$ of $O(n^5)$ vertex subsets such that if G or $\bar{G}$ is not odd-hole-free, then $\mathbb{X}$ contains a near cleaner of G or $\bar{G}$. By Lemma 6.5, it takes overall $O(n^{8.377})$ time [38, 64, 82] to either obtain an odd hole of G or $\bar{G}$ or ensure that both G and $\bar{G}$ are odd-hole-free.

6.2.2 Proving Theorem 1.4

The recent odd-hole detection algorithm of Chudnovsky, Scott, Seymour, and Spirkl has seven $O(n^9)$-time bottleneck subroutines. One is for pyramid detection, which is eliminated by Theorem 1.3. The remaining six are in two groups [26, §4]. The first (respectively, second) group handles the case that the longest x-gap (i.e., a path D of C such that $G[D \cup \{x\}]$ is a hole of G) over all C-major vertices x for a shortest odd hole C is shorter (respectively, longer) than one half of C. We give a two-phase algorithm to handle both cases in $O(n^8)$ time. For the first case, Phase 1 tries all $O(n^5)$ choices of five vertices to obtain an approximate cleaner for C, with which a shortest odd hole can be identified in $O(n^3)$ time via Lemmas 6.5 and 6.8. For the second case, Phase 2 tries all $O(n^6)$ choices of six vertices to obtain an approximate cleaner for C, with which a shortest odd hole can be identified in $O(n^2)$ time via Lemma 6.9.

Lemma 6.7 (Chudnovsky, Scott, Seymour, and Spirkl [26, Theorem 3.4]). *Let G be a jewel-free, pyramid-free, and 5-hole-free graph. Let C be a shortest odd hole in G. If $x \in M_G(C)$, then there is an edge of C adjacent to each vertex of $M_G(C) \setminus N_G(x)$ in G.*

A vertex set $X \subseteq V(G)$ is an *approximate cleaner* of C if X contains all C-major vertices and $X \cap V(C) \subseteq \{c_1, c_2\}$ holds for two vertices c_1 and c_2 with $d_C(c_1, c_2) = 3$. The second statement of the next lemma reduces the running time of an $O(n^8)$-time subroutine of Chudnovsky et al. [26, Theorem 2.4] to $O(n^5)$.

Lemma 6.8. *For any given vertex set X of an n-vertex m-edge pyramid-free jewel-free 5-hole-free graph G, (1) it takes $O(n^3)$ time to obtain an odd hole of G or ensure that X is not an approximate cleaner of any shortest odd hole of G and (2) it takes $O(mn^3)$ time to either obtain an odd hole of G or ensure that there is no shortest odd hole C of G such that an edge of C is adjacent to all C-major vertices of G.*

Proof. We first show that Statement 1 implies Statement 2: For each edge $b_1 b_2$ of G, we apply Statement 1 with $X = (N_G(b_1) \cup N_G(b_2)) \setminus \{b_1, b_2\}$ in overall $O(mn^3)$ time. If no odd hole is detected, then report that there is no shortest odd hole C of G such that an edge of C is adjacent to all C-major vertices of G. To see the correctness, observe that if C is a shortest odd hole of G such that an edge $b_1 b_2$ is adjacent to all C-major vertices of G, then $(N_G(b_1) \cup N_G(b_2)) \setminus \{b_1, b_2\}$ is an approximate cleaner of C. Thus, Statement 2 holds.

It remains to prove Statement 1. It takes overall $O(n^3)$ time to obtain for any distinct vertices u and v of G that are connected in $G(u, v) = G - (X \setminus \{u, v\})$ (i) the length $d(u, v)$ of a shortest uv-path $P(u, v)$ in $G(u, v)$ and (ii) the neighbor $N(u, v)$ of u in $P(u, v)$. Assume $P(u, v) = P(v, u)$ for all u

and v without loss of generality. If u and v are not connected in $G(u,v)$, then let $d(u,v)=\infty$. It takes overall $O(n^3)$ time to determine whether $C=G[P(c_1,c_2)\cup P(c_1,b)\cup P(c_2,b)]$ is a 7-hole or the following equation holds for any distinct vertices b, c_1, and c_2 of G:

$$\begin{array}{rcl} d(c_1,c_2) & = & 3 \\ d(c_1,N(c_2,b)) & > & 3 \\ d(c_2,N(c_1,b)) & > & 3 \\ d(c_1,b) & = & d(c_2,b) \;=\; d(c_1,N(b,c_2))-1 \;=\; d(c_2,N(b,c_1))-1. \end{array} \tag{2}$$

If Equation (2) holds for distinct vertices b, c_1, and c_2 with minimum $d(c_1,b)$, then C is an odd hole of G: Both $P(b,c_1)$ and $P(b,c_2)$ are chordless. By $d(c_1,b)=d(c_2,b)=d(c_1,N(b,c_2))-1=d(c_2,N(b,c_1))-1$ and the minimality of $d(c_1,b)$, paths $P(b,c_1)-b$ and $P(b,c_2)-b$ are disjoint and nonadjacent. The interior of $P(c_1,c_2)$ is disjoint from and nonadjacent to $P((c_1,b)-c_1)\cup(P(c_2,b)-c_2)$, since otherwise $d(c_i,N(c_{3-i},b))\le 3$ or $d(c_i,b)\ge d(c_i,N(b,c_{3-i}))$ would hold for an $i\in\{1,2\}$. Thus, C is an odd hole of G. It remains to show that if X is an approximate cleaner for a shortest odd hole C of G, then there is a choice of b, c_1, and c_2 such that Equation (2) holds or $C^*=G[P(c_1,c_2)\cup P(c_1,b)\cup P(c_2,b)]$ is a 7-hole. Let c_1 and c_2 be two vertices of C with $X\cap V(C)\subseteq\{c_1,c_2\})$. Thus, C is a clean shortest odd hole of $H=G-(X\setminus\{c_1,c_2\})$. By $d_C(c_1,c_2)=3$, $|V(C)|\ge 7$, and Lemma 6.4, we have $d(c_1,c_2)=3$. Let b be the vertex of C with $d_C(b,c_1)=d_C(b,c_2)$. Apply Lemma 6.4 on C to replace the shortest bc_1-path of C with $P(b,c_1)$, replace the shortest bc_2-path of C with $P(b,c_2)$, and replace the shortest c_1c_2-path of C with $P(c_1,c_2)$. We obtain the clean shortest odd hole C^* of H. Suppose $|V(C^*)|\ge 9$. By $X\cap V(C)\subseteq\{c_1,c_2\}$, $|V(C^*)|\ge 9$, and Lemma 6.4, we have $d(c_1,b)=d(c_2,b)=d(c_1,N(b,c_2))-1=d(c_2,N(b,c_2))-1$. By Lemma 6.4 and $|V(C^*)|\ge 9$, we have $d(c_i,N(c_{3-i},b))>3$ for both $i\in\{1,2\}$. Thus, Equation (2) holds. □

Lemma 6.9. *Let d, b_1, and b_2 be distinct vertices of an n-vertex graph G. Let each T_i with $i\in\{1,2\}$ be a subtree of $G-\{b_1,b_2\}$ containing d. It takes $O(n^2)$ time to determine whether there is a leaf c_i of T_i for each $i\in\{1,2\}$ such that if each P_i with $i\in\{1,2\}$ is the dc_i-path of T_i, then $G[P_1\cup\{b_1,b_2\}\cup P_2]$ is an odd hole of G.*

Proof. For each $i\in\{1,2\}$, let T_i' (respectively, T_i'') be the union of all d-to-leaf paths of T_i with odd (respectively, even) lengths. In order for $G[P_1\cup\{b_1,b_2\}\cup P_2]$ to be an odd hole, if P_1 is path of T_1' (respectively, T_1''), then P_2 is a path of T_2' (respectively, T_2''). Therefore, it suffices to work on the case that if each c_i with $i\in\{1,2\}$ is a leaf of T_i, then (1) the union of path $c_1b_1b_2c_2$ and the dc_1-path P_1 of T_1 is an induced path of G, (2) the union of path $c_1b_1b_2c_2$ and the dc_2-path P_2 of T_2 is an induced path of G, and (3) $|E(P_1)|+|E(P_2)|$ is even. It remains to show how to determine in $O(n^2)$ time whether there is an induced c_1c_2-path $P_1\cup P_2$. For each vertex v of T_2-d, let set $S(v)$, implemented by an n-bit array associated with a counter for $|S(v)|$, be initially empty. Perform a depth-first traversal of T_1. When a vertex u of T_1-d is reached from its parent in T_1, insert u into $S(v)$ for each vertex v of T_2-d with $u=v$ or $uv\in E(G)$ in overall $O(n)$ time. When the traversal is about to leave a vertex u of T_1-d for its parent in T_1, run the following $O(n)$-time steps: If u is a leaf c_1 of T_1, then check whether there is a dc_2-path P_2 of T_2 for some leaf c_2 of T_2 such that $S(v)=\varnothing$ holds for all vertices v of P_2-d. If there is such a P_2, then quit the traversal and report an odd hole $G[P_1\cup\{b_1,b_2\}\cup P_2]$. If u is not a leaf of T_1 or there is no such a P_2, then delete u from $S(v)$ for each vertex v of T_2-d with $u\in S(v)$. If the traversal ends normally, then report negatively. The overall running time is $O(n^2)$. To see the correctness, let c_1 be a traversed leaf of T_1. Let c_2 be an arbitrary leaf of T_2. Let each P_i with $i\in\{1,2\}$ be the dc_i-path of T_i. Consider the moment when the traversal is about to leave c_1 for its parent in T_1. By the depth-first nature of the traversal, $S(v)\subseteq V(P_1)$ holds for each vertex v of T_2-d. Therefore, $P_1\cup P_2$ is an induced c_1c_2-path if and only if $S(v)=\varnothing$ holds for each vertex v of P_2-d. □

Proof of Theorem 1.4. It suffices to prove Statement 1. By Theorem 1.3 and Lemma 6.8(1), and the fact that jewels and 5-holes are $O(n^6)$-time detectable, we may assume that G does not contain pyramids, jewels, 5-holes, and clean shortest odd holes. By Lemma 6.8(2), we may further assume that G does not contain any shortest odd hole C such that an edge of C is adjacent to all C-major vertices. The algorithm consists of two $O(m^2n^4)$-time phases. If none of them identifies an odd hole of G, then report that G is odd-hole-free. Let x, d, d_1, d_2, c_1, b_1, and b_2 be vertices of G that are not necessarily distinct. Let

$$\begin{array}{rcl} X_1 & = & (N_G(b_1) \cup N_G(b_2)) \setminus \{b_1, b_2\} \\ X_2 & = & N_G(d_1) \cap N_G(d_2) \\ S_0 & = & \{d_1, d_2\} \\ S_1 & = & \{d_1, d_2, c_1\} \\ S_2 & = & \{d_1, d_2, c_1, b_1\}. \end{array}$$

For each $k \in \{0,1,2\}$, let

$$H_k = G - ((X_1 \cup N_G(x)) \setminus S_k),$$

let I_k consist of the internal vertices of all shortest d_1d_2-paths of H_k, let J_k consist of vertex d and the internal vertices of all shortest dd_1-paths and dd_2-paths of H_k, let $Y_k = N_G(x) \cap N_G(I_k)$, and let $Z_k = N_G(x) \cap N_G(J_k)$. If no odd hole of G is identified via the following two phases, then report that G is odd-hole-free.

Phase 1:

- For each of the $O(m^2n)$ choices of vertices x, d_1, d_2, b_1, b_2 with $x \in N_G(d_1) \cap N_G(d_2)$ and $b_1b_2 \in E(G)$, apply Lemma 6.8(1) with $X = (X_1 \cup X_2 \cup Y_0) \setminus S_0$ in $O(n^3)$ time.
- For each of the $O(m^2n)$ choices of vertices $x, c_1, b_1 = d_1, b_2, d_2$ with $x \in N_G(d_1) \cap N_G(d_2)$ and $b_1b_2 \in E(G)$, apply Lemma 6.8(1) on $X = (X_1 \cup X_2 \cup Y_1) \setminus S_1$ in $O(n^3)$ time.
- For each of the $O(m^2n)$ choices of vertices $x, c_1, b_1, b_2 = d_1, d_2$ with $x \in N_G(d_1) \cap N_G(d_2)$ and $b_1b_2 \in E(G)$, apply Lemma 6.8(1) on $X = (X_1 \cup X_2 \cup Y_2) \setminus S_2$ in $O(n^3)$ time.

Phase 2:

- For each of the $O(m^2n^2)$ choices of vertices x, d, d_1, d_2, b_1, b_2 with $x \in N_G(d_1) \cap N_G(d_2)$ and $b_1b_2 \in E(G)$, apply the following procedure with $X = (X_1 \cup X_2 \cup Z_0) \setminus S_0$ in $O(n^2)$ time.
- For each of the $O(m^2n^2)$ choices of vertices $x, d, c_1, b_1 = d_1, b_2, d_2$ with $x \in N_G(d_1) \cap N_G(d_2)$ and $b_1b_2 \in E(G)$, apply the following procedure on $X = (X_1 \cup X_2 \cup Z_1) \setminus S_1$ in $O(n^2)$ time.
- For each of the $O(m^2n^2)$ choices of vertices $x, d, c_1, b_1, b_2 = d_1, d_2$ with $x \in N_G(d_1) \cap N_G(d_2)$ and $b_1b_2 \in E(G)$, apply the following procedure on $X = (X_1 \cup X_2 \cup Z_2) \setminus S_2$ in $O(n^2)$ time.

Let C_1 (respectively, C_2) consist of the vertices c such that cb_1b_2 (respectively, b_1b_2c) is an induced path of G. Let T_1^* be a tree that is the union of a shortest dc-path in $G - (X \setminus \{c, d\})$ over all vertices $c \in C_1$. Let each T_i with $i \in \{1,2\}$ be a tree that is the union of a shortest dd_i-path and a shortest d_ic-path in $G - (X \setminus \{c, d\})$ over all vertices $c \in C_i$. Apply Lemma 6.9 on d, b_1, b_2, T_1 (respectively, T_1^*), and T_2 to identify an odd hole of G in $O(n^2)$ time.

The rest of the proof assumes that C is a shortest odd hole of G and shows that the above $O(m^2n^4)$-time algorithm outputs an odd hole of G. Since G does not contain any clean shortest odd hole, $M_G(C) \neq \varnothing$. For any $x \in M_G(C)$, a path D of C is an *x-gap* [26] if $G[D \cup \{x\}]$ is a hole of G. There is an $x \in M_G(C)$ with an x-gap or else each edge of C would be adjacent to all vertices of $M_G(C)$. Let $x \in M_G(C)$ maximize the length of a longest x-gap D. Let b_1b_2 be an edge of C adjacent to each vertex of $M_G(C) \setminus N_G(x)$ as ensured by Lemma 6.7, implying $M_G(C) \setminus X_1 \subseteq N_G(x)$. Let d_1 and d_2 be the end-vertices of D. By the maximality of D, each vertex of $M_G(C) \setminus X_2$ is adjacent to the interior of D. Thus, each vertex of $M_G(C) \setminus (X_1 \cup X_2)$ is adjacent to x and the interior of D. Let c_1 and c_2 be the vertices such that $c_1 b_1 b_2 c_2$ is a path of C. We have $k = |V(D) \cap \{b_1, b_2\}| \in \{0, 1, 2\}$. If $k = 0$, then $S_k = \{d_1, d_2\}$ and the interior of D is disjoint from $c_1 b_1 b_2 c_2$. If $k = 1$, then assume without loss of generality $d_1 = b_1$ and that c_1 is the neighbor of d_1 in D, implying $S_k = \{c_1, b_1 = d_1, d_2\}$. If $k = 2$, then assume without loss of generality $d_1 = b_2$, by $x \in N_G(b_1) \cup N_G(b_2)$ and that b_1 is the neighbor of d_1 in D, implying $S_k = \{c_1, b_1, b_2 = d_1, d_2\}$.

For each $k \in \{0, 1, 2\}$, D is a path of H_k: We have $N_G(x) \cap V(D) = \{d_1, d_2\} \subseteq S_k$. By $X_1 \cap V(D) = \{c_1, c_2\} \cap V(D) \subseteq S_k$, we have $D \subseteq H_k$. By $M_G(C) \cap S_k = \varnothing$ and $M_G(C) \setminus X_1 \subseteq N_G(x)$, we have $M_G(C) \subseteq (X_1 \cup N_G(x)) \setminus S_k$, implying $H_k \subseteq G - M_G(C)$.

Phase 1 handles the case $|E(D)| < 0.5 \cdot |E(C)|$: By Lemma 6.4(1), D is a shortest $d_1 d_2$-path of $G - M_G(C)$, implying that D is a shortest $d_1 d_2$-path of H_k. Since no edge of C is adjacent to all C-major nodes of G, we have $|E(D)| \geq 3$ by the maximality of D. Thus, all internal vertices of D are contained by I_k, implying $M_G(C) \setminus (X_1 \cup X_2) \subseteq Y_k$ by the maximality of D. Let D^* be an arbitrary shortest $d_1 d_2$-path of H_k. By $|E(D^*)| = |E(D)|$ and $H_k \subseteq G - M_G(C)$, D^* is a shortest $d_1 d_2$-path of $G - M_G(C)$. By Lemma 6.4(2), the graph C^* obtained from C by replacing D with D^* is a clean shortest odd hole of $G - M_G(C)$. Therefore, the interior of D^* is disjoint from and nonadjacent to $C - V(D)$, implying that I_k is disjoint from and nonadjacent to $C - V(D)$. One can verify that $X = (X_1 \cup X_2 \cup Y_k) \setminus S_k$ is either an approximate cleaner for C with $X \cap V(C) = \{c_1, c_2\}$ or $X \cap V(C) = \{c_2\}$. Thus, Phase 1 outputs an odd hole of G.

Phase 2 handles the case $|E(D)| > 0.5 \cdot |E(C)|$: Let d be a middle vertex of D. For each index $i \in \{1, 2\}$, the dd_i-path D_i of C is a shortest dd_i-path of $G - M_G(C)$ by Lemma 6.4(1), implying that D_i is a shortest dd_i-path of H_k. Thus, all internal vertices of D are contained by J_k, implying $M_G(C) \setminus (X_1 \cup X_2) \subseteq Z_k$. Let each D_i^* with $i \in \{1, 2\}$ be an arbitrary shortest dd_i-path of H_k. By $|E(D_i^*)| = |E(D_i)|$ and $H_k \subseteq G - M_G(C)$, D_i^* is a shortest dd_i-path of $G - M_G(C)$. By Lemma 6.4(2), the graph C^* obtained from C by replacing D with $D_1^* \cup D_2^*$ is a clean shortest odd hole of $G - M_G(C)$. Therefore, the interior of the $d_1 d_2$-path $D_1^* \cup D_2^*$ is disjoint from and nonadjacent to $C - V(D)$, implying that J_k is disjoint from and nonadjacent to $C - V(D)$. One can verify that $X = (X_1 \cup X_2 \cup Z_k) \setminus S_k$ is an approximate cleaner for C with $X \cap V(C) = \{c_1, c_2\}$ or $X \cap V(C) = \{c_2\}$. We have $c_1 \in C_1$ and $c_2 \in C_2$.

- If $k = 0$, then the dc_1-path P_1 of T_1 is the union of a shortest dd_1-path P_1' and a shortest $d_1 c_1$-path P_1'' of $G - (X \setminus \{c_1, d\})$ even if $c_1 = d_1$. By $M_G(C) \subseteq X$, $X \cap V(C) \subseteq \{c_1, c_2\}$, and the fact that the shortest dd_1-path and $d_1 c_1$-path of C are in $G - (X \setminus \{c_1, d\})$, Lemma 6.4(1) implies that P_1' (respectively, P_1'') is a shortest dd_1-path (respectively, $d_1 c_1$-path) of $G - M_G(C)$.
- If $k \in \{1, 2\}$, then c_1 is an internal vertex of D. The dc_1-path P_1 of T_1^* is a shortest dc_1-path of $G - (X \setminus \{c_1, d\})$. By $M_G(C) \subseteq X$ and $X \cap V(C) = \{c_2\}$, Lemma 6.4(1) implies that P_1 is a shortest dc_1-path of $G - M_G(C)$.

The dc_2-path P_2 of T_2 is the union of a shortest dd_2-path P_2' and a shortest $d_2 c_2$-path P_2'' of $G - (X \setminus \{c_2, d\})$ even if $k = 0$ and $c_2 = d_2$. By $M_G(C) \subseteq X$, $X \cap V(C) \subseteq \{c_1, c_2\}$, and the fact that the shortest dd_2-path and $d_2 c_2$-path of C are in $G - (X \setminus \{c_2, d\})$, Lemma 6.4(1) implies that P_2' (respectively, P_2'') is a shortest dd_2-path (respectively, $d_2 c_2$-path) of $G - M_G(C)$. By applying Lemma 6.4(2) at most four times on C, $G[P_1 \cup \{b_1, b_2\} \cup P_2]$ is a clean shortest odd hole of $G - M_G(C)$. Thus, Phase 2 outputs an odd hole of G. □

6.3 Improved even-hole detection

Chang and Lu's algorithm consists of two $O(n^{11})$-time phases. The first phase detects beetles in $O(n^{11})$ time, which is now reduced to $O(n^7)$ time by Theorem 1.5. The second phase maintains a set $\mathbb{T}$ of induced subgraphs of G with the property that if G is even-hole-free, then so is each graph in $\mathbb{T}$ until either $\mathbb{T}$ becomes empty or an $H \in \mathbb{T}$ is found to contain even holes. The initial $\mathbb{T}$ consists of $O(n^5)$ graphs obtained from guesses of (1) a 3-path P on a shortest even hole C of G, (2) an $X \subseteq V(G)$ that contains the major vertices of C without intersecting C, and (3) a $Y \subseteq V(G)$ that contains $N_G^{2,2}(C)$ (see §6.3.2 for definition) without intersecting C. Each iteration of Phase 2 takes $O(n^4)$ time to either ensure that an $H \in \mathbb{T}$ is an extended clique tree that contains even holes or replaces H with 0 (respectively, 1 and 2) smaller graphs via ensuring that H is an even-hole-free extended clique tree (respectively, decomposing H by a star-cutset and decomposing H by a 2-join). The guessed P and Y are crucial in arguing that H can be decomposed by a star-cutset without increasing $|\mathbb{T}|$, implying that each initial $H \in \mathbb{T}$ incurs $O(n)$ decompositions by star-cutsets. Therefore, the overall time for decompositions by star-cutsets is $O(n^{10})$, i.e., $O(n^5)$ times the initial $|\mathbb{T}|$. Each initial $H \in \mathbb{T}$ incurs $O(n^2)$ decompositions by 2-joins, implying that the overall time for detecting even holes in extended clique trees and decompositions by 2-joins is $O(n^{11})$, i.e., $O(n^6)$ times the initial $|\mathbb{T}|$. We reduce the time of Phase 2 from $O(n^{11})$ to $O(n^9)$. As in the proof of Lemma 6.10, a factor of n is removed by reducing the initial $|\mathbb{T}|$ from $O(n^5)$ to $O(n^4)$ via ignoring Y and the internal vertex of P. Guessing only X and the end-vertices of P does complicate the task of decomposing H by a star-cutset, but we manage to handle each decomposition by a star-cutset in the same time bound (see the proof of Lemma 6.11). Another factor of n is removed by reducing the number of decompositions by 2-joins incurred by each initial $H \in \mathbb{T}$ from $O(n^2)$ to $O(n)$ via carefully handling the boundary cases (see the proof of Lemma 6.12).

Let G be an n-vertex m-edge graph. A *major vertex* [20] of an even hole C is a $v \in V(G) \setminus V(C)$ with at least three distinct vertices in $N_G(v) \cap V(C)$ that are pairwise nonadjacent in G. Let $M_G(C)$ consist of the major vertices of an even hole C. A hole without major vertices is *clear*. A *v_1v_2-hole* of G is a clear shortest even hole C of G such that v_1 and v_2 are the end-vertices of a 3-path of C. A *tracer* of G is a triple $\langle H, v_1, v_2\rangle$ such that v_1 and v_2 are vertices of an induced subgraph H of G. A tracer $\langle H, v_1, v_2\rangle$ of G is *lucky* if H contains a v_1v_2-hole. A set $\mathbb{T}$ of tracers of G is *reliable* if $\mathbb{T}$ satisfies the condition that if G contains even holes, then $\mathbb{T}$ contains lucky tracers.

Lemma 6.10. *If G is beetle-free, then it takes $O(m^2n^2)$ time to either ensure that G contains even holes or obtain a reliable set of $O(mn^2)$ tracers of G.*

Subset S of $V(H)$ is a *star-cutset* [31] of a graph H if $S \subseteq N_H[s]$ holds for an $s \in S$ and the number of connected components of $H - S$ is more than that of H.

Lemma 6.11. *For any tracer T of a beetle-free graph G, it takes $O(mn^3)$ time to complete one of the following tasks. Task 1: ensure that G contains even holes. Task 2: ensure that T is not lucky. Task 3: obtain a star-cutset-free induced subgraph H of G such that if T is lucky, then H contains even holes.*

The next lemma improves upon the $O(mn^4)$-time algorithm of Chang and Lu. [15, Lemma 4.2].

Lemma 6.12. *It takes $O(mn^3)$ time to detect even holes in an n-vertex m-edge star-cutset-free graph.*

We first reduce Theorem 1.6 via Theorem 1.5 to Lemmas 6.10, 6.11, and 6.12.

Proof of Theorem 1.6. By Theorem 1.5, it takes $O(m^2n^3)$ time to detect beetles in G. If G contains beetles, then G contains even holes. Otherwise, we apply Lemma 6.10 on the beetle-free G in $O(m^2n^2)$ time. If G is ensured to contain even holes, then the theorem is proved. Otherwise, we have a reliable set $\mathbb{T}$ of $O(mn^2)$ tracers of G. It takes overall $O(m^2n^5)$ time to apply Lemma 6.11 on all $T \in \mathbb{T}$. If Task 1 is completed for any $T \in \mathbb{T}$, then G contains even holes. If Task 2 is completed for all $T \in \mathbb{T}$, then G is even-hole-free. Otherwise, we apply Lemma 6.12 in overall $O(m^2n^5)$ time on each of the $O(mn^2)$ star-cutset-free induced subgraphs H of G corresponding to the tracers $T \in \mathbb{T}$ for which Task 3 is completed. If an H contains even holes, then so does G. Otherwise, G is even-hole-free. □

Lemmas 6.10, 6.11, and 6.12 are proved in §6.3.1, §6.3.2, and §6.3.3, respectively.

6.3.1 Proving Lemma 6.10

Lemma 6.13 (da Silva and Vušković [40]). *Let G be an n-vertex m-edge graph. It takes $O(mn^2)$ time to either ensure that G contains even holes or obtain all $O(m)$ maximal cliques of G.*

Lemma 6.14 (Chang and Lu [15, Lemma 3.4]). *If C is a shortest even hole of a 4-hole-free graph G, then either $M_G(C) \subseteq N_G(v)$ holds for a vertex v of C or $G[M_G(C)]$ is a clique.*

Proof of Lemma 6.10. It takes $O(m^2)$ time to detect 4-holes in G, so we assume that G is 4-hole-free. By Lemma 6.13, it suffices to consider that the set $\mathbb{K}$ of $O(m)$ maximal cliques of G is available. It takes $O(m^2n^2)$ time to obtain the set $\mathbb{T}$ of $O(mn^2)$ tracers of G in the form of (1) $\langle G-(N_G(v)\setminus\{v_1,v_2\}),v_1,v_2\rangle$ with $\{v_1,v,v_2\}\subseteq V(G)$ or (2) $\langle G-V(K),v_1,v_2\rangle$ with $K\in\mathbb{K}$ and $\{v_1,v_2\}\subseteq V(G)$. To see that $\mathbb{T}$ is reliable, let C be a shortest even hole of G. Case 1: $M_G(C)\subseteq N_G(v)$ holds for a vertex v of C. Let v_1 and v_2 be the neighbors of v in C. By $M_G(C)\subseteq N_G(v)\setminus\{v_1,v_2\}$ and $(N_G(v)\setminus\{v_1,v_2\})\cap C=\varnothing$, C is a v_1v_2-hole of $G-(N_G(v)\setminus\{v_1,v_2\})$. Case 2: $M_G(C)\nsubseteq N_G(v)$ holds for all vertices v of C. By Lemma 6.14, $G[M_G(C)]$ is a clique. Let K be a maximal clique with $M_G(C)\subseteq V(K)$. We have $V(K)\cap C=\varnothing$ or else $M_G(C)\cap C=\varnothing$ would imply $M_G(C)\subseteq V(K)\setminus\{v\}\subseteq N_G(v)$ for any $v\in V(K)\cap C$, contradiction. Thus, C is a v_1v_2-hole of $G-V(K)$ for any v_1v_2-path of C with 3 vertices. □

6.3.2 Proving Lemma 6.11

Vertex x *dominates* vertex y in graph H if $x\neq y$ and $N_H[y]\subseteq N_H[x]$. Vertex y is *dominated* in H if some vertex of H dominates y in H. A star-cutset S of graph H is *full* if $S=N_H[s]$ holds for some vertex s of S.

Lemma 6.15 (Chvátal [31, Theorem 1]). *A graph without dominated vertices and full star-cutsets is star-cutset-free.*

Lemma 6.16 (Chudnovsky, Kawarabayashi, and Seymour [20, Lemma 2.2]). *If x is a major vertex of a shortest even hole C of graph G, then $|N_G(x,C)|$ is even.*

Let $N_G^i(C)$ consist of the vertices $x\in N_G(C)\setminus M_G(C)$ such that $|N_G(x,C)|=i$ and $C[N_G(x,C)]$ is connected. Let $N_G^{i,i}(C)$ consist of the vertices $x\in N_G(C)\setminus M_G(C)$ such that $C[N_G(x,C)]$ has two connected components, each of which has i vertices.

Lemma 6.17 (Chang and Lu [15, Lemma 2.2]). *For any clear shortest even hole C of a beetle-free graph G, we have*

$$N_G(C)\subseteq N_G^1(C)\cup N_G^2(C)\cup N_G^3(C)\cup N_G^{1,1}(C)\cup N_G^{2,2}(C).$$

Proof of Lemma 6.11. We first prove the lemma using the following two claims for any tracer $T=\langle H,v_1,v_2\rangle$ of an n-vertex m-edge beetle-free connected graph G:

Claim 1: It takes $O(mn^2)$ time to obtain a tracer $T' = \langle H', v_1', v_2' \rangle$ of G, where H' is an induced subgraph of H having no dominated vertices, such that if T is lucky, then so is T'.

Claim 2: It takes $O(mn^2)$ time to (1) ensure that H is full-star-cutset-free, (2) obtain an even hole of G, or (3) obtain a proper induced subgraph H' of H such that if T is lucky, then so is $\langle H', v_1, v_2 \rangle$.

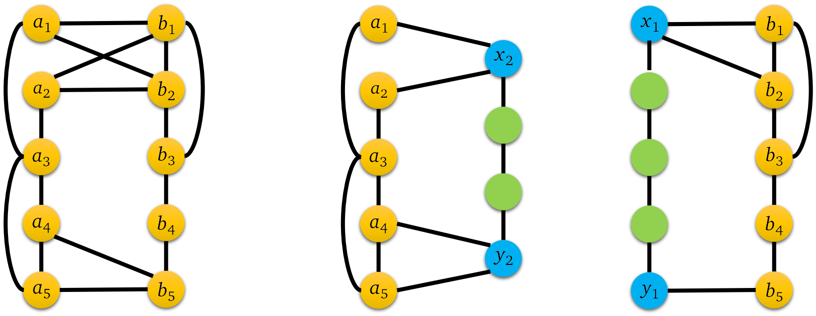
The algorithm proceeds in $O(n)$ iterations to update $T = \langle H, v_1, v_2 \rangle$. Each iteration starts with applying Claim 1 to update T without destroying its luckiness by replacing $\langle H, v_1, v_2 \rangle$ with the ensured $\langle H', v_1', v_2' \rangle$ such that H' is an induced subgraph of H that does not contain any dominated vertex. It then applies Claim 2 on the resulting $T = \langle H, v_1, v_2 \rangle$. If H is ensured to be full-star-cutset-free, then Task 3 is completed by Lemma 6.15. If we obtain an even hole of G, then Task 2 is completed. Otherwise, it updates T without destroying its luckiness by replacing H with the obtained proper induced subgraph H' of H and proceed to the next iteration. The overall running time is $O(mn^3)$.

To prove Claim 1, the $O(mn^2)$-time algorithm outputs the resulting T after iteratively updating the initial $T = \langle H, v_1, v_2 \rangle$ by the following procedure until H contains no dominated vertices: (1) spend $O(mn)$ time to detect vertices x and y of H such that x dominates y in H, (2) let $H = H - \{y\}$, and (3) if $y = v_i$ with $i \in \{1, 2\}$, then let $v_i = x$. The resulting H is an induced subgraph of the initial H. For the correctness, it suffices to prove that if a tracker T is lucky, then so is the resulting T after an iteration of the loop. Suppose that a v_1v_2-hole C of H contains y or else C remains a v_1v_2-hole of $H' = H - \{y\}$. Since C is an even hole, we have $x \notin V(C)$ and $|N_C[y]| = 3$, implying a connected component of $C[N_G(x, C)]$ with at least 3 vertices. By Lemma 6.17, we have $x \in N_H^3(C)$, implying that $N_G(x, C)$ consists of y and the two neighbors of y in C. Thus, $C' = H[C \cup \{x\} \setminus \{y\}]$ remains a shortest even hole of H'. Let v_0 be a vertex of C such that $v_1v_0v_2$ is a 3-path of C. For each $i \in \{0, 1, 2\}$, if $y = v_i$, then let $u_i = x$; otherwise, let $u_i = v_i$. Clearly, $u_1u_0u_2$ is a 3-path of C'. It remains to show that C' is clear. Assume for contradiction $z \in M_{H'}(C')$, implying $y \neq z$ and $z \in M_H(C')$. By Lemma 6.16, $|N_{C'}(z)| \geq 4$ and $|N_{C'}(z)| \neq 5$. By Lemma 6.17, $M_H(C) = \varnothing$ implies $|N_C(z)| \leq 4$. By $C - \{y\} = C' - \{x\}$, exactly one of x and y is adjacent to z in H or else $z \in M_H(C')$ would imply $z \in M_H(C)$. Thus, $z \in N_H(x) \setminus N_H(y)$, implying $|N_C(z)| = |N_{C'}(z)| - 1 = 3$. Lemma 6.17 implies $z \in N_H^3(C)$. Since $C[N_G(z, C)]$ is a 3-path, $H[C' \cup \{z\}]$ is a beetle B of H in which $B[N_B[z] \setminus \{x\}]$ is a diamond, contradiction.

To prove Claim 2, it takes $O(mn)$ time to detect full star-cutsets in H. It suffices to focus on the case that H contains a full star-cutset $S = N_H[s]$. Let $\mathbb{B}$ consist of the connected components of $H - S$. It takes $O(n^3)$ time to obtain, for every two nonadjacent vertices s_1 and s_2 of S, the list $L(s_1, s_2)$ of elements in $\mathbb{B}$ that are adjacent to both s_1 and s_2. It takes $O(m^2)$ time to check whether the following conditions hold:

1. There are distinct $B_i \in L(s_1, s_2)$ with $\{s_1, s_2\} \subseteq S$ for $i \in \{1, 2\}$.
2. There are disjoint edges s_is_{i+2} of $H[S]$ with distinct $B_i \in L(s_{2i-1}, s_{2i})$ for $i \in \{1, 2\}$.

If Condition 1 holds, then $H[P_1 \cup P_2 \cup s]$ (is a theta and thus) contains even holes for any shortest s_1s_2-path P_i in $H[B_i \cup \{s_1, s_2\}]$. If Condition 2 holds, then $H[P_1 \cup P_2 \cup s]$ contains even holes for any shortest $s_{2i-1}s_{2i}$-path P_i in $H[B_i \cup \{s_{2i-1}, s_{2i}\}]$. The rest of the proof assumes that neither condition holds. If there were a v_1v_2-hole C of H intersecting distinct B_1 and B_2 of $\mathbb{B}$, then $s \notin C$, implying that $C[N_G(s, C)]$ is not connected. By Lemma 6.17, either $s \in N_H^{1,1}(C)$, implying Condition 1, or $s \in N_H^{2,2}(C)$, implying Condition 2. Hence, each v_1v_2-hole C of H intersects at most one element of $\mathbb{B}$. If a $B \in \mathbb{B}$ contains one or both of v_1 and v_2, then the claim is proved with $H' = H[B \cup S]$. It remains to consider the case $\{v_1, v_2\} \subseteq S$. Let C be a v_1v_2-hole intersecting exactly one $B \in \mathbb{B}$. If $s \in C$, then $V(C) \cap S = \{v_1, s, v_2\}$, implying $B \in L(v_1, v_2)$. If $s \notin C$, then $s \in N_H^3(C) \cup N_H^{1,1}(C) \cup N_H^{2,2}(C)$ by Lemma 6.17, also implying $B \in L(v_1, v_2)$. Since Condition 1 does not hold, $|L(v_1, v_2)| \leq 1$.



(a) H

(b) H_1 and H_2

Figure 12: A 2-join $J = (V_1, V_2, X_1, X_2, Y_1, Y_2)$ of H with $V_1 = \{a_1, \ldots, a_6\}$, $V_2 = \{b_1, \ldots, b_6\}$, $X_1 = \{a_1, a_2\}$, $X_2 = \{b_1, b_2\}$ $Y_1 = \{a_5, a_6\}$, and $Y_2 = \{b_6\}$ and and the parity-preserving blocks of decomposition H_1 and H_2 for J.

Therefore, if $|L(v_1, v_2)| = 1$, then the claim is proved with $H' = H[B \cup S]$, where B is the only element in $L(v_1, v_2)$. If $|L(v_1, v_2)| = 0$, then the claim is proved with $H' = H[S]$. □

6.3.3 Proving Lemma 6.12

$(V_1, V_2, X_1, X_2, Y_1, Y_2)$ is a 2-*join* [41, §1.3] (which is called a non-path 2-join in, e.g., [15, 77, 79]) of a connected graph H if

1. V_1 and V_2 form a disjoint partition of $V(H)$ with $|V_1| \geq 3$ and $|V_2| \geq 3$,
2. X_i and Y_i are disjoint nonempty subsets of V_i for each i,
3. $H[V_i]$ is not a minimal X_iY_i-path for each i, and
4. if $v_i \in V_i$ for each i, then $v_1v_2 \in E(H)$ if and only if $v_i \in X_i$ for each i or $v_i \in Y_i$ for each i.

See Figure 12(a) for an example.

Lemma 6.18 (Trotignon and Vušković [79, Lemma 3.2]). *If $(V_1, V_2, X_1, Y_1, X_2, Y_2)$ is a 2-join of a star-cutset-free connected graph H, then the following statements hold for each $i \in \{1, 2\}$:*

1. *Each connected component of $H[V_i]$ intersects both X_i and Y_i.*
2. *Each vertex of X_i (respectively, Y_i) has a non-neighbor of H in Y_i (respectively, X_i).*

Lemma 6.19 (Charbit, Habib, Trotignon, and Vušković [17, Theorem 4.1]). *Given an n-vertex m-edge connected graph H, it takes $O(mn^2)$ time to either obtain a 2-join of H or ensure that H is 2-join-free.*

Lemma 6.20 (da Silva and Vušković [41, Corollary 1.3]). *A connected even-hole-free star-cutset-free 2-join-free graph is an extended clique tree.*

Let $J = (V_1, V_2, X_1, X_2, Y_1, Y_2)$ be a 2-join of a star-cutset-free connected graph H. Let P_i with $i \in \{1, 2\}$ be a shortest induced X_iY_i-path P_i of $H[V_i]$ as ensured by Lemma 6.18(1). If $|V(P_i)|$ is even (respectively, odd), then let $p_i = 4$ (respectively, $p_i = 5$). The *parity-preserving blocks of decomposition* [79] for J are the graphs H_i with $i \in \{1, 2\}$ consisting of $H[V_i]$, a p_j-vertex x_jy_j-path with $j = 3 - i$, edges xx_j for all vertices x of X_i, and edges yy_j for all vertices y of Y_i. See Figure 12(b) for an example.

Lemma 6.21 (Trotignon and Vušković [79, Lemma 3.8]). *Let H_1 and H_2 be the parity-preserving blocks of decomposition for a 2-join of an m-edge star-cutset-free connected graph H.*

1. *Both H_1 and H_2 are star-cutset-free.*
2. *Both H_1 and H_2 are even-hole-free if and only if H is even-hole-free.*

Lemma 6.22 (Chang and Lu [15, Lemma 4.12]). *Each of the parity-preserving blocks of decomposition for a 2-join for an n-vertex m-edge star-cutset-free connected graph has at most n vertices and m edges.*

Graph H is an *extended clique tree* [41] if there is a set S of two or fewer vertices of H such that each biconnected component of $H-S$ is a clique. It takes $O(mn^2)$ time to determine whether an n-vertex m-edge graph is an extended clique tree.

Lemma 6.23 (Chang and Lu [15, Lemma 4.6]). *It takes $O(n^4)$ time to detect even holes in an n-vertex connected extended clique tree.*

Proof of Lemma 6.12. Let $W(H)$ consist of the $v \in V(H)$ with $|N_H(v)| \geq 3$. Let $h(H) = |V(H)| + |W(H)|$. We first prove the claim that if H_1 and H_2 are the parity-preserving blocks of decomposition for a 2-join $(V_1, V_2, X_1, X_2, Y_1, Y_2)$ of a star-cutset-free connected graph H, then (a) $X_i \cup V_j$, $Y_i \cup V_j$, or $X_i \cup Y_i \cup V_j$ with $\{i, j\} = \{1, 2\}$ induces a 6-hole of H or (b) we have

$$\begin{aligned} h(H_1) + h(H_2) &\leq h(H) + 14 && (3) \\ \max\{h(H_1), h(H_2)\} &\leq h(H) - 1. && (4) \end{aligned}$$

By definition of H_i and H_j with $\{i, j\} = \{1, 2\}$, (i) if $v \in V_i$, then $|N_{H_i}(v)| \leq |N_H(v)|$, (ii) if $x_j \in W(H_i)$, then $X_j \subseteq W(H)$, and (iii) if $y_j \in W(H_i)$, then $Y_j \subseteq W(H)$. Thus, $|W(H_i)| \leq |W(H)|$. By Lemma 6.22, $h(H_i) \leq h(H)$. By $|V(H_i)| = |V_i| + p_j \leq |V_i| + 5$ and $W(H_i) \setminus W(H) \subseteq \{x_j, y_j\}$, Equation (3) holds. To see Equation (4), assume $h(H_i) = h(H)$, implying

$$\begin{aligned} |V(H_i)| &= |V(H)| && (5) \\ |W(H_i)| &= |W(H)|. && (6) \end{aligned}$$

By $|V(H_i)| = |V(H)| - |V_j| + p_j$ and Equation (5), $|V_j| = p_j$. If $|V(P_j)| \in \{4, 5\}$, then $|V_j| = p_j = |V(P_j)|$ contradicts $H[V_j] \neq P_j$. By $p_j \in \{4, 5\}$, we have $|V(P_j)| \in \{2, 3\}$.

Case 1: $|V(P_j)| = 2$. $|V_j| = p_j = 4$. By Lemma 6.18(2), $|X_j| = |Y_j| = 2$. Thus, $|X_i| = |Y_i| = 1$ or else $X_j \subseteq W(H)$ or $Y_j \subseteq W(H)$, contradicting Equation (6). Hence, $|N_{H_i}(x_j)| = |N_{H_i}(y_j)| = 2$. By Equation (6), $X_j \cap W(H) = Y_j \cap W(H) = \varnothing$. By Lemma 6.18(1), $H[X_i \cup Y_i \cup V_j]$ is a 6-hole.

Case 2: $|V(P_j)| = 3$. $|V_j| = p_j = 5$. Let $Z = V_j \setminus V(P_j)$. Thus, $Z \cap (X_j \cup Y_j) \neq \varnothing$ or else $V(P_j)$ is a star-cutset of H. Let $z \in Z \cap X_j$ without loss of generality. $|X_i| = 1$ or else $X_j \subseteq W(H)$ with $|X_j| \geq 2$ contradicts Equation (6). Hence, $|N_{H_i}(x_j)| = 2$, implying $X_j \cap W(H) = \varnothing$ by Equation (6). By Lemma 6.18(1), $|N_H(z)| = 2$. Let z' be the neighbor of z in V_j. We know $z' \notin Y_j$ or else zz' is shorter than P_j. By Equation (6), the internal vertex of P_j has degree 2 in H. Thus, $Z = \{z, z'\}$ and $z'y_j \in E(H)$ by Lemma 6.18(1). $H[X_i \cup V_j]$ is a 6-hole.

It suffices to prove the lemma for any given n-vertex m-edge star-cutset-free connected graph H_0. Let $\mathbb{H}$ initially consist of H_0. Repeat the following loop until $\mathbb{H} = \varnothing$ or the current H is ensured to contain an even hole: Each iteration starts with getting a current $H \in \mathbb{H}$ and deleting H from $\mathbb{H}$. If $w(H) \leq 15$, then detect even holes in H in $O(1)$ time. If H is even-hole-free, then proceed to the next iteration; otherwise, exit the loop. If $w(H) \geq 16$, then apply Lemma 6.19 on H in $O(mn^2)$ time.

- Case 1: H is 2-join-free. Determine whether H is an extended clique tree in $O(mn^2)$ time. If H is an extended clique tree, then apply Lemma 6.23 to detect even holes in H in $O(n^4)$ time; otherwise, H contains an even hole by Lemma 6.20. If H contains an even hole, then exit the loop; otherwise, proceed to the next iteration.

- Case 2: H admits a 2-join $J = (V_1, V_2, X_1, X_2, Y_1, Y_2)$ of H. Spend $O(1)$ time to detect 6-holes in H from $H[X_i \cup V_j]$, $H[Y_i \cup V_j]$, or $H[X_i \cup Y_i \cup V_j]$ with $\{i, j\} = \{1, 2\}$. If H contains a 6-hole, then exit the loop. Otherwise, add to $\mathbb{H}$ the $O(m)$-time obtainable parity-preserving blocks of decomposition for J, each of which has at most n vertices and m edges according to Lemma 6.22, and proceed to the next iteration.

By Lemma 6.21, if the loop stops with an empty $\mathbb{H}$, then H_0 is even-hole-free; otherwise, H_0 contains an even hole. We bound the number of iterations by $O(n)$ as follows. Let Case 2 occur $f(h)$ times with $h = h(H_0)$. By Equations (3) and (4), if $h \leq 15$, then $f(h) = 0$; otherwise,

$$f(h) \leq \max\{1 + f(h_1) + f(h_2) : h_1, h_2 \leq h - 1, h_1 + h_2 \leq h + 14\}.$$

By induction on h, we prove $f(h) \leq \max(h - 15, 0)$, which holds for $h \leq 15$. For $h \geq 16$,

$$\begin{aligned} f(h) &\leq \max\{1 + \max(h_1 - 15, 0) + \max(h_2 - 15, 0) : h_1, h_2 \leq h - 1, h_1 + h_2 \leq h + 14\} \\ &\leq \max\{\max(h_1 + h_2 - 29, h_1 - 14, h_2 - 14, 1) : h_1, h_2 \leq h - 1, h_1 + h_2 \leq h + 14\} \\ &\leq \max(h - 15, h - 15, h - 15, 1) \\ &= \max(h - 15, 0). \end{aligned}$$

Since the number of iterations is $O(h) = O(n)$, the overall running time is $O(mn^3)$ except for that of applying Lemma 6.23. Since each iteration increases the overall number of vertices of graphs in $\mathbb{H}$ by $O(1)$, the overall number of vertices of the graphs in $\mathbb{H}$ remains $O(n)$ throughout. Thus, all $O(n)$ iterations of applying Lemma 6.23 take overall $O(n^4) = O(mn^3)$ time. □

7 Concluding remarks

We solve the three-in-a-tree problem on an n-vertex m-edge undirected graph in $O(m \log^2 n)$ time, leading to improved algorithms for recognizing perfect graphs and detecting thetas, pyramids, beetles, and odd and even holes. It would be interesting to see if the complexity of the three-in-a-tree problem can be further reduced. The amortized cost of maintaining the connectivity information for the dynamic graph $G - X$ can be improved to $O(\log^2 n / \log \log n)$ using [84] or even to $O(\log n \log \log^{O(1)} n)$ using [76]. Since $G - X$ is purely decremental, we can use the randomized algorithm in [75] for further speedup. However, this is not our only $O(\log^2 n)$ bottleneck: At the moment we pay $O(\log n)$ time for each neighbor of a vertex in X when it changes color, so if it changes color $O(\log n)$ times, then it will be hard to beat the $O(\log^2 n)$ factor.

Acknowledgments

We thank the anonymous reviewers of STOC 2020 and Evangelos Kipouridis for helpful comments. We thank Ho-Lin Chen and Meng-Tsung Tsai for commenting on a preliminary version [63] of the paper.

References

[1] P. Aboulker, M. Radovanović, N. Trotignon, and K. Vušković. Graphs that do not contain a cycle with a node that has at least two neighbors on it. *SIAM Journal on Discrete Mathematics*, 26(4):1510–1531, 2012. doi:10.1137/11084933X.

[2] L. Addario-Berry, M. Chudnovsky, F. Havet, B. Reed, and P. Seymour. Bisimplicial vertices in even-hole-free graphs. *Journal of Combinatorial Theory, Series B*, 98(6):1119–1164, 2008. doi:10.1016/j.jctb.2007.12.006.

[3] S. Alstrup, J. Holm, K. D. Lichtenberg, and M. Thorup. Maintaining information in fully dynamic trees with top trees. *ACM Transactions on Algorithms*, 1(2):243–264, 2005. doi:10.1145/1103963.1103966.

[4] J. Bang-Jensen, F. Havet, and A. K. Maia. Finding a subdivision of a digraph. *Theoretical Computer Science*, 562:283–303, 2015. doi:10.1016/j.tcs.2014.10.004.

[5] J. Bang-Jensen, F. Havet, and N. Trotignon. Finding an induced subdivision of a digraph. *Theoretical Computer Science*, 443:10–24, 2012. doi:10.1016/j.tcs.2012.03.017.

[6] C. Berge. Les problèmes de coloration en théorie des graphes. *Publications de l'Institut de statistique de l'Université de Paris*, 9:123–160, 1960.

[7] C. Berge. Färbung von Graphen deren sämtliche bzw. deren ungerade Kreise starr sind (Zusammenfassung). *Wissenschaftliche Zeitschrift, Martin Luther Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe*, 10:114–115, 1961.

[8] C. Berge. *Graphs*. North-Holland, Amsterdam, New York, 1985.

[9] D. Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85–92, 1991. See [10] for corrigendum, doi:10.1016/0012-365X(91)90098-M.

[10] D. Bienstock. Corrigendum to: D. Bienstock, "On the complexity of testing for odd holes and induced odd paths" Discrete Mathematics 90 (1991) 85–92. *Discrete Mathematics*, 102(1):109, 1992. doi:10.1016/0012-365X(92)90357-L.

[11] V. Boncompagni, M. Radovanović, and K. Vušković. The structure of (theta, pyramid, 1-wheel, 3-wheel)-free graphs. *Journal of Graph Theory*, 90(4):591–628, 2019. doi:10.1002/jgt.22415.

[12] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*. SIAM, 1999.

[13] H. Bruhn and A. Saito. Clique or hole in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 102(1):1–13, 2012. doi:10.1016/j.jctb.2011.02.004.

[14] K. Cameron, S. Chaplick, and C. T. Hoàng. On the structure of (pan, even hole)-free graphs. *Journal of Graph Theory*, 87(1):108–129, 2018. doi:10.1002/jgt.22146.

[15] H.-C. Chang and H.-I. Lu. A faster algorithm to recognize even-hole-free graphs. *Journal of Combinatorial Theory, Series B*, 113:141–161, 2015. doi:10.1016/j.jctb.2015.02.001.

[16] M.-S. Chang, M.-T. Ko, and H.-I. Lu. Linear-time algorithms for tree root problems. *Algorithmica*, 71(2):471–495, 2015. doi:10.1007/s00453-013-9815-y.

[17] P. Charbit, M. Habib, N. Trotignon, and K. Vušković. Detecting 2-joins faster. *Journal of Discrete Algorithms*, 17:60–66, 2012. doi:10.1016/j.jda.2012.11.003.

[18] M. Chudnovsky, G. Cornuéjols, X. Liu, P. D. Seymour, and K. Vušković. Recognizing Berge graphs. *Combinatorica*, 25(2):143–186, 2005. doi:10.1007/s00493-005-0012-8.

[19] M. Chudnovsky and R. Kapadia. Detecting a theta or a prism. *SIAM Journal on Discrete Mathematics*, 22(3):1164–1186, 2008. doi:10.1137/060672613.

[20] M. Chudnovsky, K.-i. Kawarabayashi, and P. Seymour. Detecting even holes. *Journal of Graph Theory*, 48(2):85–111, 2005. doi:10.1002/jgt.20040.

[21] M. Chudnovsky and I. Lo. Decomposing and clique-coloring (diamond, odd-hole)-free graphs. *Journal of Graph Theory*, 86(1):5–41, 2017. doi:10.1002/jgt.22110.

[22] M. Chudnovsky, I. Lo, F. Maffray, N. Trotignon, and K. Vušković. Coloring square-free Berge graphs. *Journal of Combinatorial Theory, Series B*, 135:96–128, 2019. doi:10.1016/j.jctb.2018.07.010.

[23] M. Chudnovsky, F. Maffray, P. D. Seymour, and S. Spirkl. Corrigendum to "Even pairs and prism corners in square-free Berge graphs" [J. Combin. Theory, Ser. B 131 (2018) 12–39]. *Journal of Combinatorial Theory, Series B*, 133:259–260, 2018. doi:10.1016/j.jctb.2018.07.004.

[24] M. Chudnovsky, F. Maffray, P. D. Seymour, and S. Spirkl. Even pairs and prism corners in square-free Berge graphs. *Journal of Combinatorial Theory, Series B*, 131:12–39, 2018. See [23] for corrigendum, doi:10.1016/j.jctb.2018.01.003.

[25] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164(1):51–229, 2006. doi:10.4007/annals.2006.164.51.

[26] M. Chudnovsky, A. Scott, P. Seymour, and S. Spirkl. Detecting an odd hole. *Journal of the ACM*, 67(1):5:1–5:12, 2020. doi:10.1145/3375720.

[27] M. Chudnovsky, A. Scott, P. D. Seymour, and S. Spirkl. Induced subgraphs of graphs with large chromatic number. VIII. Long odd holes. *Journal of Combinatorial Theory, Series B*, 140:84–97, 2020. doi:10.1016/j.jctb.2019.05.001.

[28] M. Chudnovsky and P. Seymour. The three-in-a-tree problem. *Combinatorica*, 30(4):387–417, 2010. doi:10.1007/s00493-010-2334-4.

[29] M. Chudnovsky, P. D. Seymour, and N. Trotignon. Detecting an induced net subdivision. *Journal of Combinatorial Theory, Series B*, 103(5):630–641, 2013. doi:10.1016/j.jctb.2013.07.005.

[30] M. Chudnovsky and V. Sivaraman. Odd holes in bull-free graphs. *SIAM Journal on Discrete Mathematics*, 32(2):951–955, 2018. doi:10.1137/17M1131301.

[31] V. Chvátal. Star-cutsets and perfect graphs. *Journal of Combinatorial Theory, Series B*, 39(3):189–199, 1985. doi:10.1016/0095-8956(85)90049-8.

[32] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vušković. Finding an even hole in a graph. In *Proceedings of the 38th Symposium on Foundations of Computer Science*, pages 480–485, 1997. doi:10.1109/SFCS.1997.646136.

[33] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vušković. Even and odd holes in cap-free graphs. *Journal of Graph Theory*, 30(4):289–308, 1999. doi:10.1002/(SICI)1097-0118(199904)30:4<289::AID-JGT4>3.0.CO;2-3.

[34] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vušković. Triangle-free graphs that are signable without even holes. *Journal of Graph Theory*, 34(3):204–220, 2000. doi:10.1002/1097-0118(200007)34:3<204::AID-JGT2>3.0.CO;2-P.

[35] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vušković. Even-hole-free graphs Part I: Decomposition theorem. *Journal of Graph Theory*, 39(1):6–49, 2002. doi:10.1002/jgt.10006.

[36] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vušković. Even-hole-free graphs Part II: Recognition algorithm. *Journal of Graph Theory*, 40(4):238–266, 2002. doi:10.1002/jgt.10045.

[37] M. Conforti, G. Cornuéjols, X. Liu, K. Vušković, and G. Zambelli. Odd hole recognition in graphs of bounded clique size. *SIAM Journal on Discrete Mathematics*, 20(1):42–48, 2006. doi:10.1137/S089548010444540X.

[38] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. doi:10.1016/S0747-7171(08)80013-2.

[39] G. Cornuéjols, X. Liu, and K. Vušković. A polynomial algorithm for recognizing perfect graphs. In *Proceedings of the 44th Symposium on Foundations of Computer Science*, pages 20–27, 2003. doi:10.1109/SFCS.2003.1238177.

[40] M. V. G. da Silva and K. Vušković. Triangulated neighborhoods in even-hole-free graphs. *Discrete Mathematics*, 307(9-10):1065–1073, 2007. doi:10.1016/j.disc.2006.07.027.

[41] M. V. G. da Silva and K. Vušković. Decomposition of even-hole-free graphs with star cutsets and 2-joins. *Journal of Combinatorial Theory, Series B*, 103(1):144–183, 2013. doi:10.1016/j.jctb.2012.10.001.

[42] M. Dalirrooyfard, T. D. Vuong, and V. Vassilevska Williams. Graph pattern detection: hardness for all induced patterns and faster non-induced cycles. In *Proceedings of the 51st Symposium on Theory of Computing*, pages 1167–1178, 2019. doi:10.1145/3313276.3316329.

[43] N. Derhy and C. Picouleau. Finding induced trees. *Discrete Applied Mathematics*, 157(17):3552–3557, 2009. doi:10.1016/j.dam.2009.02.009.

[44] N. Derhy, C. Picouleau, and N. Trotignon. The four-in-a-tree problem in triangle-free graphs. *Graphs and Combinatorics*, 25(4):489–502, 2009. doi:10.1007/s00373-009-0867-3.

[45] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/BF01961541.

[46] E. Diot, M. Radovanović, N. Trotignon, and K. Vušković. The (theta, wheel)-free graphs Part I: Only-prism and only-pyramid graphs. *Journal of Combinatorial Theory, Series B*, 143:123–147, 2020. doi:10.1016/j.jctb.2017.12.004.

[47] E. Diot, S. Tavenas, and N. Trotignon. Detecting wheels. *Applicable Analysis and Discrete Mathematics*, 8(1):111–122, 2014. doi:10.2298/AADM131128023D.

[48] V. F. dos Santos, V. G. da Silva, and J. L. Szwarcfiter. The k-in-a-tree problem for chordal graphs. *Matemática Contemporânea*, 44:1–10, 2015. https://mc.sbm.org.br/volumes/volume-44/.

[49] P. Erdös, M. E. Saks, and V. T. Sós. Maximum induced trees in graphs. *Journal of Combinatorial Theory, Series B*, 41(1):61–79, 1986. doi:10.1016/0095-8956(86)90028-6.

[50] J. Fiala, M. Kaminski, B. Lidický, and D. Paulusma. The k-in-a-path problem for claw-free graphs. *Algorithmica*, 62(1–2):499–519, 2012. doi:10.1007/s00453-010-9468-z.

[51] F. V. Fomin, I. Todinca, and Y. Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM Journal on Computing*, 44(1):54–87, 2015. doi:10.1137/140964801.

[52] D. J. Fraser, A. M. Hamel, and C. T. Hoàng. On the structure of (even hole, kite)-free graphs. *Graphs and Combinatorics*, 34(5):989–999, 2018. doi:10.1007/s00373-018-1925-5.

[53] I. Gitler, E. Reyes, and J. A. Vega. CIO and ring graphs: Deficiency and testing. *Journal of Symbolic Computation*, 79(2):249–268, 2017. doi:10.1016/j.jsc.2016.02.007.

[54] P. A. Golovach, D. Paulusma, and E. J. van Leeuwen. Induced disjoint paths in AT-free graphs. In F. V. Fomin and P. Kaski, editors, *Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory*, Lecture Notes in Computer Science 7357, pages 153–164, 2012. doi:10.1007/978-3-642-31155-0_14.

[55] P. A. Golovach, D. Paulusma, and E. J. van Leeuwen. Induced disjoint paths in claw-free graphs. *SIAM Journal on Discrete Mathematics*, 29(1):348–375, 2015. doi:10.1137/140963200.

[56] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. *Proceedings of the 8th International Symposium on Graph Drawing*, pages 77–90, 2000. doi:10.1007/3-540-44541-2_8.

[57] C. T. Hoàng. On the structure of (banner, odd hole)-free graphs. *Journal of Graph Theory*, 89(4):395–412, 2018. doi:10.1002/jgt.22258.

[58] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM*, 48(4):723–760, 2001. doi:10.1145/502090.502095.

[59] W.-L. Hsu. Recognizing planar perfect graphs. *Journal of the ACM*, 34(2):255–288, 1987. doi:10.1145/23005.31330.

[60] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 52(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

[61] K. Kawarabayashi, Y. Kobayashi, and B. A. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.

[62] T. Kloks, H. Müller, and K. Vušković. Even-hole-free graphs that do not contain diamonds: A structure theorem and its consequences. *Journal of Combinatorial Theory, Series B*, 99(5):733–800, 2009. doi:10.1016/j.jctb.2008.12.005.

[63] K.-Y. Lai. Sapling detection. Master's thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2018.

[64] F. Le Gall. Powers of tensors and fast matrix multiplication. In K. Nabeshima, K. Nagasaka, F. Winkler, and Á. Szántó, editors, *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

[65] C. G. Lekkerkerker and J. C. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962. doi:10.4064/fm-51-1-45-64.

[66] B. Lévêque, D. Y. Lin, F. Maffray, and N. Trotignon. Detecting induced subgraphs. *Discrete Applied Mathematics*, 157(17):3540–3551, 2009. doi:10.1016/j.dam.2009.02.015.

[67] W. Liu and N. Trotignon. The k-in-a-tree problem for graphs of girth at least k. *Discrete Applied Mathematics*, 158(15):1644–1649, 2010. doi:10.1016/j.dam.2010.06.005.

[68] F. Maffray and N. Trotignon. Algorithms for perfectly contractile graphs. *SIAM Journal on Discrete Mathematics*, 19(3):553–574, 2005. doi:10.1137/S0895480104442522.

[69] J. Nesetril and S. Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985. http://eudml.org/doc/17394.

[70] M. Radovanović, N. Trotignon, and K. Vušković. The (theta, wheel)-free graphs Part II: Structure theorem. *Journal of Combinatorial Theory, Series B*, 143:148–184, 2020. doi:10.1016/j.jctb.2019.07.004.

[71] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

[72] D. Rose, R. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. doi:10.1137/0205021.

[73] A. Scott and P. D. Seymour. Induced subgraphs of graphs with large chromatic number. I. Odd holes. *Journal of Combinatorial Theory, Series B*, 121:68–84, 2016. doi:10.1016/j.jctb.2015.10.002.

[74] A. Silva, A. A. da Silva, and C. L. Sales. A bound on the treewidth of planar even-hole-free graphs. *Discrete Applied Mathematics*, 158(12):1229–1239, 2010. doi:10.1016/j.dam.2009.07.010.

[75] M. Thorup. Decremental dynamic connectivity. *Journal of Algorithms*, 33(2):229–243, 1999. doi:10.1006/jagm.1999.1033.

[76] M. Thorup. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the 32nd Symposium on Theory of Computing*, pages 343–350, 2000. doi:10.1145/335305.335345.

[77] N. Trotignon. Decomposing Berge graphs and detecting balanced skew partitions. *Journal of Combinatorial Theory, Series B*, 98(1):173–225, 2008. doi:10.1016/j.jctb.2007.07.004.

[78] N. Trotignon and K. Vušković. A structure theorem for graphs with no cycle with a unique chord and its consequences. *Journal of Graph Theory*, 63(1):31–67, 2010. doi:10.1002/jgt.20405.

[79] N. Trotignon and K. Vušković. Combinatorial optimization with 2-joins. *Journal of Combinatorial Theory, Series B*, 102(1):153–185, 2012. doi:10.1016/j.jctb.2011.06.002.

[80] W. T. Tutte. *Connectivity in graphs*. University of Toronto Press, 1966.

[81] P. van 't Hof, M. Kaminski, and D. Paulusma. Finding induced paths of given parity in claw-free graphs. *Algorithmica*, 62(1-2):537–563, 2012. doi:10.1007/s00453-010-9470-5.

[82] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith–Winograd. In *Proceedings of the 44th Symposium on Theory of Computing*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

[83] K. Vušković. Even-hole-free graphs: A survey. *Applicable Analysis and Discrete Mathematics*, 4(2):219–240, 2010. doi:10.2298/AADM100812027V.

[84] C. Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In *Proceedings of the 24th Symposium on Discrete Algorithms*, pages 1757–1769, 2013. doi:10.1137/1.9781611973105.126.