Approximating Pathwidth for Graphs of Small Treewidth

CARLA GROENLAND, Utrecht University, The Netherlands
GWENAËL JORET, Université libre de Bruxelles, Belgium
WOJCIECH NADARA, University of Warsaw, Poland
BARTOSZ WALCZAK, Jagiellonian University, Poland

We describe a polynomial-time algorithm which, given a graph G with treewidth t, approximates the pathwidth of G to within a ratio of $O(t\sqrt{\log t})$. This is the first algorithm to achieve an $f(t)$-approximation for some function f.

Our approach builds on the following key insight: every graph with large pathwidth has large treewidth or contains a subdivision of a large complete binary tree. Specifically, we show that every graph with pathwidth at least $th + 2$ has treewidth at least t or contains a subdivision of a complete binary tree of height $h + 1$. The bound $th + 2$ is best possible up to a multiplicative constant. This result was motivated by, and implies (with $c = 2$), the following conjecture of Kawarabayashi and Rossman (SODA'18): there exists a universal constant c such that every graph with pathwidth $\Omega(k^c)$ has treewidth at least k or contains a subdivision of a complete binary tree of height k.

Our main technical algorithm takes a graph G and some (not necessarily optimal) tree decomposition of G of width t' in the input, and it computes in polynomial time an integer h, a certificate that G has pathwidth at least h, and a path decomposition of G of width at most $(t' + 1)h + 1$. The certificate is closely related to (and implies) the existence of a subdivision of a complete binary tree of height h. The approximation algorithm for pathwidth is then obtained by combining this algorithm with the approximation algorithm of Feige, Hajiaghayi, and Lee (STOC'05) for treewidth.

CCS Concepts: • **Mathematics of computing → Graph algorithms**; **Approximation algorithms;**

Additional Key Words and Phrases: Treewidth, pathwidth

A preliminary version of this paper appeared in *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1965–1976. Society for Industrial and Applied Mathematics, Philadelphia, 2021. That version contains two serious errors—in the proof of Theorem 3.1 and in the complexity analysis of the algorithm.

Carla Groenland is supported by the project CRACKNP (ERC grant agreement No 853234). Gwenaël Joret is supported by an ARC grant from the Wallonia-Brussels Federation of Belgium, by a CDR grant from the National Fund for Scientific Research (FNRS), and by the Wallonia Brussels International (WBI) agency. Wojciech Nadara's research is part of a project that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme Grant Agreement 714704. Bartosz Walczak is partially supported by the Polish National Agency for Academic Exchange (NAWA).

Authors' addresses: C. Groenland, Utrecht University, Department of Information and Computing Sciences, Budapestlaan 6, 3584 CD Utrecht, The Netherlands; email: c.e.groenland@uu.nl; G. Joret, Université libre de Bruxelles, Département d'Informatique, Boulevard du Triomphe CP 212, 1050 Brussels, Belgium; email: gwenael.joret@ulb.be; W. Nadara, University of Warsaw, Institute of Informatics, Faculty of Mathematics, Informatics, and Mechanics, ul. Banacha 2, 02-097 Warsaw, Poland; email: w.nadara@mimuw.edu.pl; B. Walczak, Jagiellonian University, Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Łojasiewicza 6, 30-348 Kraków, Poland; email: bartosz.walczak@uj.edu.pl.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

1549-6325/2023/03-ART16 $15.00
https://doi.org/10.1145/3576044

ACM Reference format:
Carla Groenland, Gwenaël Joret, Wojciech Nadara, and Bartosz Walczak. 2023. Approximating Pathwidth for Graphs of Small Treewidth. *ACM Trans. Algor.* 19, 2, Article 16 (March 2023), 19 pages.
https://doi.org/10.1145/3576044

1 INTRODUCTION

Tree decompositions and path decompositions are fundamental objects in graph theory. For algorithmic purposes, it would be highly useful to be able to compute such decompositions of minimum width, that is, achieving the treewidth and the pathwidth of the graph, respectively. However, both these problems are NP-hard, and remain so even when restricted to very specific graph classes [1, 4, 18, 19, 23, 24, 26].

The best known polynomial-time approximation algorithm for treewidth, due to Feige, Hajiaghayi, and Lee [15], computes a tree decomposition of an input graph G whose width is within a factor of $O(\sqrt{\log \mathrm{tw}(G)})$ of the treewidth $\mathrm{tw}(G)$ of G. A modification of that algorithm produces a path decomposition whose width is within a factor of $O(\log n \cdot \sqrt{\log \mathrm{tw}(G)})$ of the pathwidth $\mathrm{pw}(G)$ of G, where n is the number of vertices of G, using the fact that $\mathrm{tw}(G) \leqslant \mathrm{pw}(G) \leqslant (\mathrm{tw}(G)+1)\log_2 n$ [15]. Combining it with existing FPT algorithms for treewidth and pathwidth [5, 8] leads to a polynomial-time algorithm approximating pathwidth to within a factor of $O(\mathrm{pw}(G)\,\mathrm{tw}(G)\sqrt{\log \mathrm{tw}(G)})$.[1] This is the best known approximation ratio for pathwidth that we are aware of. A simple polynomial-time $f(\mathrm{pw}(G))$-approximation algorithm (with f exponential) follows from the result of [9].

We describe a polynomial-time algorithm that approximates pathwidth to within a factor of $O(\mathrm{tw}(G)\sqrt{\log \mathrm{tw}(G)})$, thus effectively dropping the $\mathrm{pw}(G)$ factor in the above. To our knowledge, this is the first algorithm to achieve an approximation ratio that depends only on treewidth.

Our approach builds on the following key insight: every graph with large pathwidth has large treewidth or contains a subdivision of a large complete binary tree.

Theorem 1.1. *Every graph with treewidth $t-1$ has pathwidth at most $th+1$ or contains a subdivision of a complete binary tree of height $h+1$.*

The bound $th+1$ is best possible up to a multiplicative constant (see Section 5). Our original motivation for Theorem 1.1 was the following result of Kawarabayashi and Rossman [22] about treedepth, which is an upper bound on pathwidth: every graph with treedepth $\Omega(k^5 \log^2 k)$ has treewidth at least k, or contains a subdivision of a complete binary tree of height k, or contains a path of order 2^k. The bound $\Omega(k^5 \log^2 k)$ was recently lowered to $\Omega(k^3)$ by Czerwiński et al. [12], who also devised an $O(\mathrm{tw}(G)\log^{3/2}\mathrm{tw}(G))$-approximation algorithm for treedepth. Kawarabayashi and Rossman [22] conjectured that the third outcome of their theorem, the path of order 2^k, could be avoided if one considered pathwidth instead of treedepth: they conjectured the existence of a universal constant c such that every graph with pathwidth $\Omega(k^c)$ has treewidth at least k or contains a subdivision of a complete binary tree of height k. Theorem 1.1 implies their conjecture with $c=2$, which is best possible (see Section 5). We remark that Wood [29] also conjectured a statement of this type, with a bound of the form $f(t)\cdot h$ on the

[1] Here is a brief sketch. Instances G such that $\mathrm{pw}(G)\,\mathrm{tw}(G) = O(\log n)$ can be solved optimally in polynomial time thanks to the following two algorithms. (1) Bodlaender [5] described an algorithm that, given G and a number t, constructs a tree decomposition of G of width at most t or finds that $\mathrm{tw}(G) > t$ in $2^{O(t^2)}n$ time. (2) Bodlaender and Kloks [8] described an algorithm that, given G, a tree decomposition of G of width t, and a number p, constructs a path decomposition of G of width at most p or finds that $\mathrm{pw}(G) > p$ in $2^{O(pt)}n$ time. When $\mathrm{pw}(G)\,\mathrm{tw}(G) = \Omega(\log n)$, the aforementioned $O(\log n \cdot \sqrt{\log \mathrm{tw}(G)})$-approximation algorithm of Feige et al. [15] achieves the claimed bound.

pathwidth for some function f (see also [25, Lemma 6] and [20, Conjecture 6.4.1]). Both Theorem 1.1 and the treedepth results [12, 22] are a continuation of a line of research on excluded minor characterizations of graphs with small values of their corresponding width parameters (treewidth/pathwidth/treedepth), which was started by the seminal Grid Minor Theorem [27] and its improved polynomial versions [10, 11]. In the last section, we propose yet another problem in a similar vein (see Conjecture 6.2).

Since the complete binary tree of height h has pathwidth $\lceil h/2 \rceil$ [28], any subdivision of it (as a subgraph) can be used to certify that the pathwidth of a given graph is large. The following key concept provides a stronger certificate of large pathwidth, more suitable for our purposes. Let $(\mathcal{T}_h)_{h=0}^{\infty}$ be a sequence of classes of graphs defined inductively as follows: $\mathcal{T}_0$ is the class of all connected graphs, and $\mathcal{T}_{h+1}$ is the class of connected graphs G that contain three pairwise disjoint sets of vertices V_1, V_2, and V_3 such that $G[V_1], G[V_2], G[V_3] \in \mathcal{T}_h$ and any two of V_1, V_2, and V_3 can be connected in G by a path avoiding the third one. Every graph in $\mathcal{T}_h$ has the following properties:

- it has pathwidth at least h (see Lemma 2.1), and
- it contains a subdivision of a complete binary tree of height h (see Lemma 2.2).

Theorem 1.1 has a short and simple proof (see Section 3). It proceeds by showing that every connected graph with treewidth $t-1$ has pathwidth at most $th+1$ or belongs to $\mathcal{T}_{h+1}$. The stronger assertion allows us to apply induction on h. Unfortunately, this proof is not algorithmic.

To obtain the aforementioned approximation algorithm, we prove the following algorithmic version of Theorem 1.1. Its proof is significantly more involved (see Section 4).

Theorem 1.2. *For every connected graph G with treewidth at most $t-1$, there is an integer $h \geqslant 0$ such that $G \in \mathcal{T}_h$ and G has pathwidth at most $th+1$. Moreover, there is a polynomial-time algorithm to compute such an integer h, a path decomposition of G of width at most $th+1$, and a subdivision of a complete binary tree of height h in G given a tree decomposition of G of width at most $t-1$.*

Since every graph in $\mathcal{T}_h$ has pathwidth at least h, combining Theorem 1.2 (applied to every connected component of the input graph) with the aforementioned approximation algorithm for treewidth of Feige et al. [15], we obtain the following approximation algorithm for pathwidth.

Corollary 1.3. *There is a polynomial-time algorithm which, given a graph G of treewidth t and pathwidth p, computes a path decomposition of G of width $O(t\sqrt{\log t} \cdot p)$. Moreover, if a tree decomposition of G of width t' is also given in the input, the resulting path decomposition has width at most $(t'+1)p+1$.*

We remark that if we consider graphs G coming from a fixed class of graphs with bounded treewidth, then we can first use an algorithm of Bodlaender [5] to compute an optimal tree decomposition of G in linear time, and then use the above algorithm to approximate pathwidth to within a ratio of roughly $\mathrm{tw}(G)+1$. We note the following two precursors of this result in the literature (with slightly better approximation ratios): Bodlaender and Fomin [7] gave a 2-approximation algorithm for computing the pathwidth of outerplanar graphs (a subclass of graphs of treewidth at most 2), and Fomin and Thilikos [16] gave a 3-approximation algorithm for computing the pathwidth on Halin graphs (a subclass of graphs of treewidth at most 3).

We conclude this introduction with a remark about parameterized algorithms, even though our focus in this paper is approximation algorithms with running time polynomial in the size of the input graph. Bodlaender [5] (see also [6]) designed a linear-time FPT algorithm for computing pathwidth when parameterized by the pathwidth. That is, for an n-vertex input graph G, his algorithm computes the pathwidth $\mathrm{pw}(G)$ and an optimal path decomposition of G in $f(\mathrm{pw}(G)) \cdot n$ time for some computable function f. Bodlaender and Kloks [8] considered the problem of computing the

pathwidth when the input graph has small treewidth. They devised an XP algorithm for computing pathwidth when parameterized by the treewidth: given an n-vertex graph G, the algorithm computes $\mathrm{pw}(G)$ and an optimal path decomposition of G in $n^{f(\mathrm{tw}(G))}$ time for some computable function f. It is an old open problem whether pathwidth is fixed-parameter tractable when parameterized by the treewidth, that is, whether there exists an algorithm to compute the pathwidth of an n-vertex input graph G in $f(\mathrm{tw}(G)) \cdot n^{O(1)}$ time for some computable function f. This question was first raised by Dean [13]. Fomin and Thilikos [16] pointed out that even obtaining an approximation of pathwidth when parameterized by treewidth is open. Our approximation algorithm is a solution to the latter problem (in a strong sense—with polynomial dependence of the running time in the parameter) and can be seen as a step in the direction of Dean's question.

2 PRELIMINARIES AND TOOLS

2.1 Basic Definitions

Graphs considered in this paper are finite, simple, and undirected. We use standard graph-theoretic terminology and notation. We allow a graph to have no vertices; by convention, such a graph *is not connected* and has no connected components. The vertex set of a graph G is denoted by $V(G)$. A vertex v of a graph G is considered a neighbor of a set $X \subseteq V(G)$ if $v \notin X$ and v is connected by an edge to some vertex in X. The neighborhood (thus defined) of a set X in G is denoted by $N_G(X)$.

A *tree decomposition* of a graph G is a pair $(T, \{B_x\}_{x \in V(T)})$ where T is a tree and $\{B_x\}_{x \in V(T)}$ is a family of subsets of $V(G)$ called *bags*, satisfying the following two conditions:

- for each vertex v of G, the set of nodes $\{x \in V(T) : v \in B_x\}$ induces a non-empty subtree of T;
- for each edge uv of G, there is a node x in T such that both u and v belong to B_x.

The *width* of a tree decomposition $(T, \{B_x\}_{x \in V(T)})$ is $\max_{x \in V(T)} |B_x| - 1$. The *treewidth* of a graph G is the minimum width of a tree decomposition of G. A *path decomposition* and *pathwidth* are defined analogously with the extra requirement that the tree T is a path. The treewidth and the pathwidth of a graph G are denoted by $\mathrm{tw}(G)$ and $\mathrm{pw}(G)$, respectively. We refer the reader to [14] for background on tree decompositions.

A *complete binary tree of height h* is a rooted tree in which every non-leaf node has two children and every path from the root to a leaf has h edges. Such a tree has $2^{h+1} - 1$ nodes. A *complete ternary tree of height h* is defined analogously but with the requirement that every non-leaf node has three children. A *subdivision* of a tree T is a tree obtained from T by replacing each edge uv with some path connecting u and v whose internal nodes are new nodes private to that path.

2.2 Witnesses for Large Pathwidth

Recall that $(\mathcal{T}_h)_{h=0}^{\infty}$ is the sequence of classes of graphs defined inductively as follows: $\mathcal{T}_0$ is the class of all connected graphs, and $\mathcal{T}_{h+1}$ is the class of connected graphs G that contain three pairwise disjoint sets of vertices V_1, V_2, and V_3 such that $G[V_1], G[V_2], G[V_3] \in \mathcal{T}_h$ and any two of V_1, V_2, and V_3 can be connected in G by a path avoiding the third one.

A *$\mathcal{T}_h$-witness* for a graph $G \in \mathcal{T}_h$ is a complete ternary tree of height h of subsets of $V(G)$ defined inductively following the definition of $\mathcal{T}_h$. The $\mathcal{T}_0$-witness for a connected graph G is the tree with the single node $V(G)$, denoted by $\langle V(G) \rangle$. A $\mathcal{T}_{h+1}$-witness for a graph $G \in \mathcal{T}_{h+1}$ is a tree with root $V(G)$ and with three subtrees W_1, W_2, W_3 of the root that are $\mathcal{T}_h$-witnesses of $G[V_1], G[V_2], G[V_3]$ for some sets V_1, V_2, V_3 as in the definition of $\mathcal{T}_{h+1}$; it is denoted by $\langle V(G); W_1, W_2, W_3 \rangle$.

It clearly follows from these definitions that every graph in $\mathcal{T}_h$ has at least 3^h vertices and every $\mathcal{T}_h$-witness of an n-vertex graph has $O(n)$ nodes. The next two lemmas explain the connection of $\mathcal{T}_h$ to pathwidth and to subdivisions of complete binary trees.

Lemma 2.1. *If $G \in \mathcal{T}_h$, then $\mathrm{pw}(G) \geqslant h$.*

Proof. The proof goes by induction on h. The case $h = 0$ is trivial. Now, assume that $h \geqslant 1$ and the lemma holds for $h-1$. Since $G \in \mathcal{T}_h$, there are sets $V_1, V_2, V_3 \subseteq V(G)$ interconnected as in the definition of $\mathcal{T}_h$, such that $G[V_i] \in \mathcal{T}_{h-1}$ and thus $\mathrm{pw}(G[V_i]) \geqslant h-1$ for $i = 1, 2, 3$. Let P be a path decomposition of G. With bags restricted to V_i, it becomes a path decomposition of $G[V_i]$. It follows that for $i = 1, 2, 3$, there is a bag B_i in P such that $|B_i \cap V_i| \geqslant h$. Assume without loss of generality that B_1, B_2, B_3 occur in this order in P. Since $G[V_1]$ and $G[V_3]$ are connected, there is a path that connects $B_1 \cap V_1$ and $B_3 \cap V_3$ in G avoiding V_2. This path must have a vertex in B_2, so $|B_2 \setminus V_2| \geqslant 1$ and thus $|B_2| \geqslant h+1$. This proves that $\mathrm{pw}(G) \geqslant h$. □

The proof of Lemma 2.1 generalizes the well-known proof of the fact that (a subdivision of) a complete binary tree of height h has pathwidth at least $\lceil h/2 \rceil$. Actually, it is straightforward to show that such a tree belongs to $\mathcal{T}_{\lceil h/2 \rceil}$.

Lemma 2.2. *If $G \in \mathcal{T}_h$, then G contains a subdivision of a complete binary tree of height h as a subgraph. Moreover, it can be computed in polynomial time from a $\mathcal{T}_h$-witness for G.*

Proof. We prove, by induction on h, that for every graph $G \in \mathcal{T}_h$ and every $v \in V(G)$, the following structure exists in G: a subdivision S of a complete binary tree of height h with some root r and a path P from v to r such that $V(P) \cap V(S) = \{r\}$. This is trivial for $h = 0$. For the induction step, assume that $h \geqslant 1$ and the statement holds for $h-1$. Let $G \in \mathcal{T}_h$ and $v \in V(G)$. Let $V_1, V_2, V_3 \subseteq V(G)$ be as in the definition of $\mathcal{T}_h$. Assume without loss of generality that $v \in V_3$ or v can be connected with V_3 by a path in G avoiding $V_1 \cup V_2$. For $i = 1, 2$, since $G[V_3]$ is connected and G has a path connecting V_i with V_3 and avoiding V_{3-i}, there is also a path in G from v to some vertex $v_i \in V_i$ avoiding $V_1 \cup V_2 \setminus \{v_i\}$. These paths can be chosen so that they first follow a common path P from v to some vertex r in $G - (V_1 \cup V_2)$ and then they split into a path Q_1 from r to v_1 and a path Q_2 from r to v_2 so that r is the only common vertex of any two of P, Q_1, Q_2. For $i = 1, 2$, the induction hypothesis provides an appropriate structure in $G[V_i]$: a subdivision S_i of a complete binary tree of height $h-1$ with root r_i and a path P_i from v_i to r_i such that $V(P_i) \cap V(S_i) = \{r_i\}$. Connecting r with S_1 and S_2 by the combined paths Q_1P_1 and Q_2P_2, respectively, yields a subdivision S of a complete binary tree of height h with root r in G. The construction guarantees that $V(P) \cap V(S) = \{r\}$.

Clearly, given a $\mathcal{T}_h$-witness for G, the induction step described above can be performed in polynomial time, and therefore the full recursive procedure of computing a subdivision of a complete binary tree of height h in G works in polynomial time. □

2.3 Combining Path Decompositions

The following lemma will be used several times in the paper to combine path decompositions.

Lemma 2.3. *Let G be a graph and $(T, \{B_x\}_{x \in V(T)})$ be a tree decomposition of G of width $t-1$.*

(1) *If $q \in V(T)$ and every connected component of $G - B_q$ has pathwidth at most ℓ, then there is a path decomposition of G of width at most $\ell + t$ which contains B_q in every bag.*
(2) *If Q is the path connecting x and y in T and every connected component of $G - \bigcup_{q \in V(Q)} B_q$ has pathwidth at most ℓ, then there is a path decomposition of G of width at most $\ell + t$ which contains B_x in the first bag and B_y in the last bag.*

In either case, there is a polynomial-time algorithm to construct such a path decomposition of G from the path decompositions of the respective components C of width at most ℓ.

Proof. In case (1), the path decomposition of G is obtained by concatenating the path decompositions of the connected components of $G - B_q$ (which have width at most ℓ) and adding B_q to every bag. Now, consider case (2). For every node q of Q, let T_q be the subtree of T induced on the nodes z such that the path from q to z in T contains no other nodes of Q, and let $V_q = \bigcup_{z \in V(T_q)} B_z$.

Apply case (1) to the graph $G[V_q]$, the tree decomposition $(T_q, \{B_z\}_{z \in V(T_q)})$ of $G[V_q]$, and the node $q \in V(T_q)$ to obtain a path decomposition of $G[V_q]$ of width at most $\ell + t$ containing B_q in every bag. Then, concatenate the path decompositions thus obtained for all nodes q of Q (in the order they occur on Q) to obtain a requested path decomposition of G. □

3 PROOF OF THEOREM 1.1

The statement of Theorem 1.1 on a graph G follows from the same statement on every connected component of G. By Lemma 2.2, every graph in $\mathcal{T}_h$ contains a subdivision of a complete binary tree of height h. Thus, Theorem 1.1 is a direct corollary to the following statement.

Theorem 3.1. *For every $h \in \mathbb{N}$, every connected graph with treewidth at most $t-1$ has pathwidth at most $th+1$ or belongs to $\mathcal{T}_{h+1}$.*

A tree decomposition of G is *optimal* if its width is equal to $\mathrm{tw}(G)$. For the proof of Theorem 3.1, we need optimal tree decompositions with an additional property. Namely, consider a connected graph G and a tree decomposition $(T, \{B_x\}_{x \in V(T)})$ of G. Every edge xy of T splits T into two subtrees: $T_{x|y}$ containing x and $T_{y|x}$ containing y. For every oriented edge xy of T, let $G_{x|y}$ denote the subgraph of G induced on the union of the bags of the nodes in $T_{x|y}$. The property we need is that every subgraph of the form $G_{x|y}$ is connected. It is known that such a tree decomposition always exists [17], but for completeness, we present a short proof of this fact in the following lemma.

Lemma 3.2. *Every connected graph G has an optimal tree decomposition $(T, \{B_x\}_{x \in V(T)})$ with the property that $G_{x|y}$ is connected for every oriented edge xy of T.*

Proof. Let $t = \mathrm{tw}(G) + 1$. The *fatness* of an optimal tree decomposition $(T, \{B_x\}_{x \in V(T)})$ of G is the t-tuple $(k_0, \ldots, k_{t-1})$ such that k_i is the number of bags B_x of size $t - i$. Let $(T, \{B_x\}_{x \in V(T)})$ be an optimal tree decomposition of G with lexicographically minimum fatness. (The idea of taking such a tree decomposition comes from the proof of existence of optimal "lean" tree decompositions due to Bellenbaum and Diestel [2, Theorem 3.1].) We show that it has the required property.

Suppose it does not. Let xy be an edge of T such that $G_{x|y}$ is disconnected and the number of nodes in the subtree $T_{x|y}$ of T is minimized. Let $\mathcal{C}$ be the family of connected components of $G_{x|y}$, so that $|\mathcal{C}| \geqslant 2$. Let $Z = N_T(x) \setminus \{y\}$. For every node $z \in Z$, let C_z be the connected component of $G_{x|y}$ that contains $G_{z|x}$, which exists because the choice of xy guarantees that $G_{z|x}$ is connected.

We modify $(T, \{B_x\}_{x \in V(T)})$ into a new tree decomposition of G as follows. We keep all nodes other than x (with their bags B_x) and all edges non-incident to x. We replace the node x by $|\mathcal{C}|$ nodes x_C with bags $B_{x_C} = B_x \cap V(C)$ for each $C \in \mathcal{C}$. We replace the edge xy by $|\mathcal{C}|$ edges $x_C y$ for each $C \in \mathcal{C}$, and we replace the edge xz by an edge $x_{C_z} z$ for each $z \in Z$. It is straightforward to verify that what we obtain is indeed a tree decomposition of G and it has width t.

Since G is connected, we have $B_{x_C} = B_x \cap V(C) \neq \emptyset$ for every $C \in \mathcal{C}$. This and the assumption that $|\mathcal{C}| \geqslant 2$ imply that $|B_{x_C}| < |B_x|$ for all $C \in \mathcal{C}$. We conclude that the fatness of the new tree decomposition is lexicographically less than the fatness of $(T, \{B_x\}_{x \in V(T)})$, which contradicts the assumption that the latter is lexicographically minimal. □

Proof of Theorem 3.1. The proof goes by induction on h. The statement is true for $h = 0$: if a connected graph G has a cycle or a vertex of degree at least 3, then $G \in \mathcal{T}_1$, and otherwise G is a path, so $\mathrm{pw}(G) \leqslant 1$. For the rest of the proof, assume that $h \geqslant 1$ and the statement is true for $h-1$.

Let G be a connected graph width treewidth at most $t-1$ and $(T, \{B_x\}_{x \in V(T)})$ be an optimal tree decomposition of G obtained from Lemma 3.2. Thus, $|B_x| \leqslant t$ for every node x of T and $G_{x|y}$ is connected for every oriented edge xy of T. For every oriented edge xy of T, let $F_{x|y}$ be the subgraph of G induced on the vertices not in $G_{y|x}$, that is, on the vertices that belong only to bags in $T_{x|y}$ and

to no other bags. Let E be the set of edges xy of T such that both $F_{x|y}$ and $F_{y|x}$ have a connected component belonging to $\mathcal{T}_h$.

Suppose $E = \emptyset$. It follows that every pair of trees of the form $T_{x|y}$ such that $F_{x|y}$ has a connected component in $\mathcal{T}_h$ has a common node. This implies that all such trees have a common node, say z, by the well-known fact that subtrees of a tree have the Helly property [21, Theorem 4.1]. For every neighbor y of z in T and every connected component C of $F_{y|z}$, since $C \notin \mathcal{T}_h$, the induction hypothesis gives $\mathrm{pw}(C) \leqslant t(h-1)+1$. Lemma 2.3 (1) applied with $q = z$ yields $\mathrm{pw}(G) \leqslant th+1$.

For the rest of the proof, assume $E \neq \emptyset$. Since every connected supergraph of a graph from $\mathcal{T}_h$ belongs to $\mathcal{T}_h$, the set E is the edge set of some subtree K of T. Let Z be the set of leaves of K. Since K has at least one edge, we have $|Z| \geqslant 2$.

Suppose $|Z| \geqslant 3$. Choose any distinct $z_1, z_2, z_3 \in Z$. For each $i \in \{1,2,3\}$, let C_i be a connected component of $F_{z_i|x_i}$ that belongs to $\mathcal{T}_h$, where x_i is the unique neighbor of z_i in K. For each $i \in \{1,2,3\}$, the subgraph $G_{x_i|z_i}$ is connected, is vertex-disjoint from C_i, and contains the other two of C_1, C_2, and C_3. Consequently, any two of the sets $V(C_1)$, $V(C_2)$, and $V(C_3)$ can be connected by a path in G avoiding the third one. This shows that $G \in \mathcal{T}_{h+1}$.

Now, suppose $|Z| = 2$. It follows that K is a path $x_1 \dots x_m$, where $Z = \{x_1, x_m\}$. For every node x_i of K, every neighbor y of x_i in $T - K$, and every connected component C of $F_{y|x_i}$, since $C \notin \mathcal{T}_h$, the induction hypothesis gives $\mathrm{pw}(C) \leqslant t(h-1)+1$. Lemma 2.3 (2) applied with $Q = K$ yields $\mathrm{pw}(G) \leqslant th+1$. □

4 PROOF OF THEOREM 1.2

By Lemma 2.2, every graph in $\mathcal{T}_h$ contains a subdivision of a complete binary tree of height h, which can be computed in polynomial time from a $\mathcal{T}_h$-witness of G. Therefore, Theorem 1.2 is a direct corollary to the following statement, the proof of which is presented in this section.

Theorem 4.1. *There is a polynomial-time algorithm which, given a connected graph G and a tree decomposition of G of width at most $t-1$, computes*

- *a number $h \in \mathbb{N}$ such that $G \in \mathcal{T}_h$ and $\mathrm{pw}(G) \leqslant th+1$,*
- *a $\mathcal{T}_h$-witness for G,*
- *a path decomposition of G of width at most $th+1$.*

4.1 Normalized Tree Decompositions

Let G be a graph with a fixed rooted tree decomposition $(T', \{B'_x\}_{x \in V(T')})$ of width at most $t-1$, which we call the *initial tree decomposition* of G. For a node x of T', let T'_x be the subtree of T' consisting of x and all nodes of T' lying below x, and let V'_x be the set of vertices of G contained only in bags of T'_x (i.e., in no bags of $T' - T'_x$). We show how to turn $(T', \{B'_x\}_{x \in V(T')})$ into a *normalized tree decomposition* of G. Intuitively, the latter will be a cleaned-up version of the initial tree decomposition with some additional properties that will be useful in the algorithm.

For a subset X of the vertex set of a graph G, let $\mathcal{C}_G(X)$ denote the family of connected components of the induced subgraph $G[X]$ (which is empty when $X = \emptyset$). Let

$$\mathrm{Sub}(G) = \bigcup_{x \in V(T')} \mathcal{C}_G\left(V'_x\right).$$

For a connected graph G, the set $\mathrm{Sub}(G)$ will be the node set of the normalized tree decomposition of G. We will use Greek letters α, β, and the like, to denote members of $\mathrm{Sub}(G)$ (nodes of the normalized tree decomposition of G). Here are some easy consequences of these definitions and the assumption that $(T', \{B'_x\}_{x \in V(T')})$ is a tree decomposition of G.

Lemma 4.2. *The following holds for every graph G and for $\mathrm{Sub}(G)$ defined as above.*

(1) *If G is connected, then $G \in \mathrm{Sub}(G)$.*
(2) *If $\alpha, \beta \in \mathrm{Sub}(G)$, then $V(\alpha) \subseteq V(\beta)$, or $V(\beta) \subseteq V(\alpha)$, or $V(\alpha) \cap V(\beta) = \emptyset$.*
(3) *If $\alpha, \beta \in \mathrm{Sub}(G)$ and $V(\alpha) \cap V(\beta) = \emptyset$, then no edge of G connects $V(\alpha)$ and $V(\beta)$.*

Now, assume that G is a connected graph. By Lemma 4.2, the members of $\mathrm{Sub}(G)$ are organized in a rooted tree with root G and with the following properties for any nodes $\alpha, \beta \in \mathrm{Sub}(G)$:

- if β is a descendant of α (i.e., α lies on the path from the root to β), then $V(\beta) \subseteq V(\alpha)$;
- if neither of α and β is a descendant of the other, then $V(\alpha)$ and $V(\beta)$ are disjoint and non-adjacent in G.

Let T denote this rooted tree (not to be confused with T'). The normalized tree decomposition will be built on this tree T. For each $\alpha \in \mathrm{Sub}(G)$, let A_α be the set of vertices $v \in V(G)$ for which α is the smallest subgraph in $\mathrm{Sub}(G)$ containing v (which must be unique), and let $B_\alpha = A_\alpha \cup N_G(V(\alpha))$. (Recall that $N_G(V(\alpha)) \subseteq V(G) \setminus V(\alpha)$ by the definition of neighborhood.)

Lemma 4.3. *The following holds for every connected graph G.*

(1) *$\{A_\alpha\}_{\alpha \in \mathrm{Sub}(G)}$ is a partition of $V(G)$ into non-empty sets.*
(2) *$(T, \{B_\alpha\}_{\alpha \in \mathrm{Sub}(G)})$ is a tree decomposition of G.*
(3) *$|B_\alpha| \leqslant t$ for every $\alpha \in \mathrm{Sub}(G)$.*

Proof. It is clear from the definition that $\bigcup_{\alpha \in \mathrm{Sub}(G)} A_\alpha = V(G)$. Let $\alpha \in \mathrm{Sub}(G)$. Since α is connected and the vertex sets of the children of α in T are pairwise disjoint and non-adjacent, at least one vertex of α is not a vertex of any child of α and thus belongs to A_α. This shows (1).

For the proof of (2), let $\alpha \in \mathrm{Sub}(G)$ and $v \in A_\alpha$. Then $v \in B_\alpha$. If $v \in B_\beta$ for some other $\beta \in \mathrm{Sub}(G)$, then v must be a neighbor of $V(\beta)$ in G, which implies that β is a descendant of α in T. In that case, v is also a neighbor of $V(\gamma)$ (and thus $v \in B_\gamma$) for every internal node γ on the path from α to β in T. This shows that the nodes β of T such that $v \in B_\beta$ form a non-empty connected subtree of T. It remains to show that any two adjacent vertices u and v of G belong to a common bag B_α. Let α be a minimal member of $\mathrm{Sub}(G)$ containing at least one of u and v; say, it contains v. Then $v \in A_\alpha$. If $u \notin A_\alpha$, then $u \notin V(\alpha)$, by minimality of α, so u is a neighbor of $V(\alpha)$ in G. In either case, $u, v \in B_\alpha$.

For the proof of (3), let $\alpha \in \mathrm{Sub}(G)$, and let x be the lowest node of T' such that $\alpha \in C_G(V'_x)$. Every vertex $v \in A_\alpha$ belongs to B'_x; otherwise it would belong to V'_y for some child y of x in T', and the connected component of $G[V'_y]$ containing v would be a proper induced subgraph of α, contradicting $v \in A_\alpha$. Now, let v be a neighbor of $V(\alpha)$ in G. Thus v is a neighbor of some vertex $u \in V(\alpha)$ while $v \notin V(\alpha)$. It follows that $u \in V'_x$ while $v \notin V'_x$, which implies that v belongs to some bag of $T' - T'_x$ as well as some bag of T'_x (as uv is an edge of G), so it belongs to B'_x. This shows that $B_\alpha \subseteq B'_x$, so $|B_\alpha| \leqslant |B'_x| \leqslant t$. □

We call $(T, \{B_\alpha\}_{\alpha \in \mathrm{Sub}(G)})$ the *normalized tree decomposition* of G. By Lemma 4.3, it has at most $|V(G)|$ nodes and its width is at most $t - 1$. The following lemma is a direct consequence of the construction of the normalized tree decomposition and will be used in the next subsection. We emphasize that by a *subtree* of the rooted tree T we simply mean a connected subgraph of T, that is, a subtree does not need to be closed under taking descendants.

Lemma 4.4. *If Q is a subtree of T and γ is a connected component of $G - \bigcup_{\xi \in V(Q)} B_\xi$, then either*

(a) *γ is a node of $T - Q$ that is a child in T of some node of Q, or*
(b) *$V(\gamma) \cap V(\xi) = \emptyset$ where ξ is the root of Q in T, that is, the node of Q closest to the root in T.*

The normalized tree decomposition depends on the choice of the initial tree decomposition for G. In the algorithm, the graphs G considered are induced subgraphs of a common *input graph*

G^{in} given with an *input tree decomposition* $(T^{\text{in}}, \{B^{\text{in}}_x\}_{x\in V(T^{\text{in}})})$, and the initial tree decomposition $(T', \{B'_x\}_{x\in V(T')})$ of G considered above in the definition of $\text{Sub}(G)$ is the input tree decomposition of G^{in} with appropriately restricted bags (some of which may become empty):

$$\left(T', \{B'_x\}_{x\in V(T')}\right) = \left(T^{\text{in}}, \left\{B^{\text{in}}_x \cap V(G)\right\}_{x\in V(T^{\text{in}})}\right).$$

The fact that the normalized tree decompositions for all induced subgraphs G of G^{in} considered in the algorithm come from a common input tree decomposition of G^{in} has the following consequences, which will be used in the complexity analysis of the algorithm in Section 4.3. (We emphasize again that, in the following lemma and later in the paper, $\text{Sub}(G)$ is always computed with respect to the initial tree decomposition that is the restriction of the input tree decomposition to G.)

Lemma 4.5. *The following holds for every induced subgraph G of the input graph G^{in}.*

(1) *If $\alpha \in \text{Sub}(G)$ and $V(\alpha) \subseteq X \subseteq V(G)$, then $\alpha \in \text{Sub}(G[X])$.*
(2) *If $\alpha, \beta \in \text{Sub}(G)$, then $\alpha \in \text{Sub}(\beta)$, or $\beta \in \text{Sub}(\alpha)$, or $V(\alpha) \cap V(\beta) = \emptyset$.*
(3) *If $\alpha \in \text{Sub}(G)$, then $\text{Sub}(\alpha) = \{\beta \in \text{Sub}(G) \colon V(\beta) \subseteq V(\alpha)\}$.*
(4) *If $\alpha \in \text{Sub}(G)$, $X \subseteq V(G)$, and $\gamma \in \mathcal{C}_G(V(\alpha) \cap X)$, then $\gamma \in \text{Sub}(G[X])$.*

Proof. Let T^{in}_x and V^{in}_x be defined like T'_x and V'_x but for the tree decomposition $(T^{\text{in}}, \{B^{\text{in}}_x\}_{x\in V(T^{\text{in}})})$ of G^{in}. Let G be an induced subgraph of G^{in}. The fact that $\text{Sub}(G)$ is computed with respect to the initial tree decomposition that is the restriction of the input tree decomposition to G implies

$$\text{Sub}(G) = \bigcup_{x\in V(T^{\text{in}})} \mathcal{C}_{G^{\text{in}}}\left(V^{\text{in}}_x \cap V(G)\right).$$

If $\alpha \in \text{Sub}(G)$ and $V(\alpha) \subseteq X \subseteq V(G)$, then the fact that α is a connected component of $G^{\text{in}}[V^{\text{in}}_x \cap V(G)]$ for some node x of T^{in} implies that it is also a connected component of $G^{\text{in}}[V^{\text{in}}_x \cap X]$, which in turn implies $\alpha \in \text{Sub}(G[X])$. This shows (1).

If $\alpha, \beta \in \text{Sub}(G)$ and $V(\alpha) \subseteq V(\beta)$, then property (1) with $X = V(\beta)$ yields $\alpha \in \text{Sub}(\beta)$. This implies (2) by Lemma 4.2 (2), and it also implies the "$\supseteq$" inclusion in (3) (with α and β interchanged). For the converse inclusion in (3), let $\alpha \in \text{Sub}(G)$ and $\beta \in \text{Sub}(\alpha)$. Thus $V(\beta) \subseteq V(\alpha)$. Let x be a node of T^{in} such that α is a connected component of $G^{\text{in}}[V^{\text{in}}_x \cap V(G)]$. Since $V(\alpha) \subseteq V^{\text{in}}_x$, there is a node y in T^{in}_x (implying $V^{\text{in}}_y \subseteq V^{\text{in}}_x$) such that β is a connected component of $G^{\text{in}}[V^{\text{in}}_y \cap V(\alpha)]$. It follows that β is a connected component of $G^{\text{in}}[V^{\text{in}}_y \cap V(G)]$, so $\beta \in \text{Sub}(G)$.

Finally, if $\alpha \in \text{Sub}(G)$, $X \subseteq V(G)$, and $\gamma \in \mathcal{C}_G(V(\alpha) \cap X)$, then α is a connected component of $G^{\text{in}}[V^{\text{in}}_x \cap V(G)]$ for some node x of T^{in}, whence it follows that γ is a connected component of $G^{\text{in}}[V^{\text{in}}_x \cap X]$ and thus $\gamma \in \text{Sub}(G[X])$. This shows (4). □

4.2 Main Procedure

The core of the algorithm is a recursive procedure $\text{solve}(G, b)$, where G is a connected graph with $\text{tw}(G) \leqslant t - 1$ and $b \in \mathbb{N} \cup \{\infty\}$ is an *upper bound request*. It computes the following data:

- a number $h = h(G, b) \in \mathbb{N}$ such that $h \leqslant b$,
- a $\mathcal{T}_h$-witness $W(G, b)$ of G,
- only when $h < b$: a path decomposition $P(G, b)$ of G of width at most $th + 1$.

The algorithm uses memorization to compute these data only once for each pair (G, b). A run of $\text{solve}(G, \infty)$ produces the outcome requested in Theorem 4.1.

The purpose of the upper bound request is optimization—we tell the procedure that if it can provide a $\mathcal{T}_b$-witness for G, then we no longer need any path decomposition for G. This allows the

procedure to save some computation, perhaps preventing many unnecessary recursive calls. Our complexity analysis of the algorithm will rely on this optimization.

Below, we present the procedure solve(G, b) for a fixed connected graph G and a fixed upper bound request $b \in \mathbb{N} \cup \{\infty\}$. The procedure assumes access to the normalized tree decomposition $(T, \{B_\alpha\}_{\alpha \in \mathrm{Sub}(G)})$ of G of width at most $t-1$ obtained from some initial tree decomposition of G as described in the definition of $\mathrm{Sub}(G)$. In the next subsection, we show that a full run of solve$(G^{\mathrm{in}}, \infty)$ on an input graph G^{in} makes recursive calls to solve(G, b) for only polynomially many distinct pairs (G, b) if the normalized tree decompositions of all induced subgraphs G of G^{in} occurring in these calls are obtained from a common input tree decomposition of G^{in} as described in Section 4.1.

If $b = 0$, then we just set $h(G, 0) = 0$ and $W(G, 0) = \langle V(G) \rangle$, and we terminate the run of solve(G, b), saying that it is *pruned*. Assume henceforth that $b \geqslant 1$.

Suppose T has only one node, that is, $\mathrm{Sub}(G) = \{G\}$. Since $V(G)$ is the bag of that node, $|V(G)| \leqslant t$. If G has a cycle or a vertex of degree at least 3, then it has three vertices v_1, v_2, v_3 such that any two of them can be connected by a path in G avoiding the third one. In that case, we set $h(G, b) = 1$ and $W(G, b) = \langle V(G); \langle\{v_1\}\rangle, \langle\{v_2\}\rangle, \langle\{v_3\}\rangle\rangle$, and (if $b > 1$) we let $P(G, b)$ be the path decomposition of G consisting of the single bag $V(G)$, which has width at most $t-1$. If G has no cycle or vertex of degree at least 3, then it is a path. In that case, we set $h(G, b) = 0$ and $W(G, b) = \langle V(G) \rangle$, and we let $P(G, b)$ be any path decomposition of G of width 1.

Assume henceforth that T has more than one node. For each node $\alpha \in \mathrm{Sub}(G) \setminus \{G\}$, we run solve$(\alpha, b)$ to compute $h(\alpha, b)$, $W(\alpha, b)$, and $P(\alpha, b)$ when $h(\alpha, b) < b$. We call these recursive calls to solve *primary*. If any of them leads to $h(\alpha, b) = b$, we just set $h(G, b) = b$ and $W(G, b) = W(\alpha, b)$, and we terminate the run of solve(G, b), again saying that it is *pruned*.

Assume henceforth that the run of solve(G, b) is not pruned, that is, we have $h(\alpha, b) < b$ for every $\alpha \in \mathrm{Sub}(G) \setminus \{G\}$. Let k be the maximum value of $h(\alpha, b)$ for $\alpha \in \mathrm{Sub}(G) \setminus \{G\}$. Thus, $k < b$. We will consider several further cases, each leading to one of the following two outcomes:

(A) We set $h(G, b) = k$. In that case, we let $W(G, b)$ be $W(\alpha, b)$ for any node $\alpha \in \mathrm{Sub}(G) \setminus \{G\}$ such that $h(\alpha, b) = k$, and we only need to provide an appropriate path decomposition $P(G, b)$.

(B) We set $h(G, b) = k + 1$. In that case, if $k + 1 < b$, we let $P(G, b)$ be the path decomposition of G of width $t(k+1)+1$ obtained by applying Lemma 2.3 (1) with q the root of T, and we only need to provide an appropriate $\mathcal{T}_{k+1}$-witness $W(G, b)$.

Let Z be the set of minimal nodes $\zeta \in \mathrm{Sub}(G) \setminus \{G\}$ such that $h(\zeta, b) = k$. It follows that $Z \neq \emptyset$ (by the definition of k) and the sets $V(\zeta)$ with $\zeta \in Z$ are pairwise disjoint and non-adjacent in G.

Suppose that Z consists of a single node ζ. Let Q be the path from the root to ζ in T. Thus, $h(\gamma, b) < k$ for every node γ in $T - Q$. Every connected component γ of $G - \bigcup_{\xi \in V(Q)} B_\xi$ needs to satisfy Lemma 4.4 (a), so it is a node of $T - Q$; in particular, $\gamma \in \mathrm{Sub}(G) \setminus \{G\}$ and $h(\gamma, b) < k$, so $P(\gamma, b)$ is a path decomposition of γ of width at most $t(k-1)+1$. We set $h(G, b) = k$ and apply Lemma 2.3 (2) to obtain a path decomposition $P(G, b)$ of G of width at most $tk + 1$.

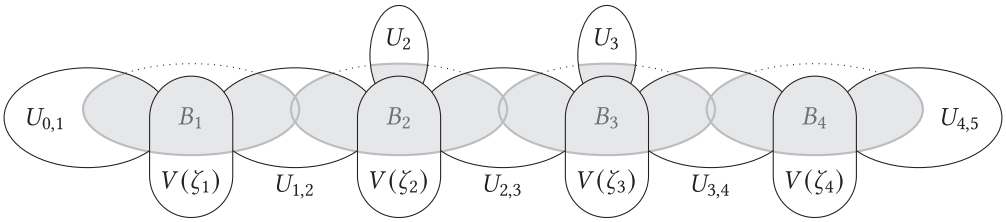
Assume henceforth that $|Z| \geqslant 2$. Let $U = V(G) \setminus \bigcup_{\zeta \in Z} V(\zeta)$. A *$U$-path* is a path in G with all internal vertices in U. Consider an auxiliary graph H with vertex set Z where $\zeta\xi$ is an edge if and only if there is a U-path connecting $V(\zeta)$ and $V(\xi)$. The graph H is connected, because so is G.

Suppose that H has a cycle or a vertex of degree at least 3. Then there are $\zeta_1, \zeta_2, \zeta_3 \in Z$ such that any two of them can be connected by a path in H avoiding the third one. This and connectedness of the induced subgraphs in Z imply that any two of the sets $V(\zeta_1), V(\zeta_2), V(\zeta_3)$ can be connected by a path in G avoiding the third one. We set $h(G, b) = k + 1$ and $W(G, b) = \langle V(G); W(\zeta_1, b), W(\zeta_2, b), W(\zeta_3, b) \rangle$, while $P(G, b)$ is constructed as in the description of outcome (B).

Assume henceforth that H has no cycle or vertex of degree at least 3. The run of solve(G, b) is called a *key run* if this case is reached. It follows that H is a path $\zeta_1 \dots \zeta_m$ with $m = |Z| \geqslant 2$, and every vertex in U is connected by a U-path to only one set or to two consecutive sets among $V(\zeta_1), \dots, V(\zeta_m)$. Define subsets $U_{0,1}, U_{1,2}, \dots, U_{m,m+1}$ of U as follows:

- $U_{0,1}$ is the set of vertices in U connected by a U-path to $V(\zeta_1)$ but not to $V(\zeta_2)$;
- for $1 \leqslant i < m$, $U_{i,i+1}$ is the set of vertices in U connected by a U-path to $V(\zeta_i)$ and to $V(\zeta_{i+1})$;
- $U_{m,m+1}$ is the set of vertices in U connected by a U-path to $V(\zeta_m)$ but not to $V(\zeta_{m-1})$.

For $1 < i < m$, let U_i be the set of vertices in $U \setminus \bigcup_{j=0}^{m} U_{j,j+1}$ connected by a U-path to $V(\zeta_i)$. The sets U_i $(1 < i < m)$ and $U_{i,i+1}$ $(0 \leqslant i \leqslant m)$ are pairwise disjoint, and their union is U. Let $B_i = B_{\zeta_i}$ for $1 \leqslant i \leqslant m$. All possible intersections and adjacencies between the sets $V(\zeta_i)$, $U_{i,i+1}$, U_i, and B_i are illustrated in the following diagram by overlaps and touchings:



Observe that $B_i \subseteq V(\zeta_i) \cup U_{i-1,i} \cup U_i \cup U_{i,i+1}$ for $1 < i < m$ and $B_i \subseteq V(\zeta_i) \cup U_{i-1,i} \cup U_{i,i+1}$ for $i = 1$ and $i = m$. This is because $B_i \subseteq V(\zeta_i) \cup N_G(V(\zeta_i))$, by the definition of $\{B_\alpha\}_{\alpha \in \mathrm{Sub}(G)}$.

It may seem peculiar that we define sets $U_{0,1}$ and $U_{m,m+1}$ rather than U_1 and U_m, so here is some intuition. The sequence of bags $B_1, \dots, B_m$ with appropriate connections inside the sets $U_{1,2}, \dots, U_{m-1,m}$ provides a skeleton along which a path decomposition of G can be laid out. The sets $U_{0,1}$ and $U_{m,m+1}$ attach to the ends of this skeleton, so we can as well extend it inside these sets (and we do—towards the root of T), making them behave similarly to $U_{1,2}, \dots, U_{m-1,m}$. Furthermore, a path decomposition of G needs to incorporate a path decomposition of each subgraph $G[U_i]$ $(1 < i < m)$ of appropriately smaller width. The argument guarantees that if we cannot provide the latter, then $G[U_i]$ contains an appropriate witness which we can combine with witnesses for ζ_{i-1} and ζ_{i+1} into a larger witness for G. We would lack one of the witnesses if we tried the same for $G[U_1]$ or $G[U_m]$ had they been defined. Having explained the intuition, we proceed with the argument.

Let $V_1 = V(\zeta_1) \cup B_1$, $V_i = V(\zeta_i) \cup B_i \cup U_i$ for $1 < i < m$, and $V_m = V(\zeta_m) \cup B_m$. Let $Q_{0,1}$ be the path in T from the root G to ζ_1, let $Q_{i,i+1}$ be the path in T from ζ_i to ζ_{i+1} for $1 \leqslant i < m$, and let $Q_{m,m+1}$ be the path in T from ζ_m to the root G. Let $B_{Q_{i,i+1}} = \bigcup_{\xi \in V(Q_{i,i+1})} B_\xi$ for $0 \leqslant i \leqslant m$.

Let Γ be the family of connected components of $G[V_i \setminus B_i]$ for $1 \leqslant i \leqslant m$ and of connected components of $G[U_{i,i+1} \setminus B_{Q_{i,i+1}}]$ for $0 \leqslant i \leqslant m$. Suppose that for each $\gamma \in \Gamma$, we have a path decomposition P_γ of γ of width at most $t(k-1)+1$. Then we apply

- Lemma 2.3 (1) to $G[V_i]$, the tree decomposition $(T, \{V_i \cap B_\alpha\}_{\alpha \in \mathrm{Sub}(G)})$ of $G[V_i]$ and the node ζ_i to obtain a path decomposition P_i of $G[V_i]$ of width at most $tk+1$ containing the set B_i in every bag, for $1 \leqslant i \leqslant m$;
- Lemma 2.3 (2) to $G[U_{i,i+1}]$, the tree decomposition $(T, \{U_{i,i+1} \cap B_\alpha\}_{\alpha \in \mathrm{Sub}(G)})$ of $G[U_{i,i+1}]$ and the path $Q_{i,i+1}$ to obtain a path decomposition $P_{i,i+1}$ of $G[U_{i,i+1}]$ of width at most $tk+1$, for $0 \leqslant i \leqslant m$; moreover, Lemma 2.3 (2) guarantees that the path decomposition $P_{i,i+1}$ contains $U_{i,i+1} \cap B_i$ in the first bag if $i \geqslant 1$ and $U_{i,i+1} \cap B_{i+1}$ in the last bag if $i < m$.

We set $h(G, b) = k$, and we concatenate these path decompositions $P_{0,1}, P_1, P_{1,2}, \dots, P_m, P_{m,m+1}$ to obtain a path decomposition $P(G, b)$ of G of width at most $tk+1$, as shown by the following claim.

Claim 4.6. *The concatenation of $P_{0,1}, P_1, P_{1,2}, \ldots, P_m, P_{m,m+1}$ is a path decomposition of G.*

Proof. If $v \in V(\zeta_i)$ with $1 \leqslant i \leqslant m$ or $v \in U_i$ with $1 < i < m$, then v belongs to bags in P_i, where the corresponding nodes form a non-empty subpath, and it belongs to no bags in the other path decompositions among $P_{0,1}, P_1, P_{1,2}, \ldots, P_m, P_{m,m+1}$. If $v \in U_{i,i+1}$ with $0 \leqslant i \leqslant m$, then v belongs to bags in $P_{i,i+1}$, where the corresponding nodes form a non-empty subpath, and moreover,

- if $v \in B_i \cap U_{i,i+1}$ with $1 \leqslant i \leqslant m$, then v belongs to all bags of P_i and to the first bag of $P_{i,i+1}$;
- if $v \in U_{i,i+1} \cap B_{i+1}$ with $0 \leqslant i < m$, then v belongs to the last bag of $P_{i,i+1}$ and to all bags of P_{i+1};
- v belongs to no bags in the other path decompositions among $P_{0,1}, P_1, P_{1,2}, \ldots, P_m, P_{m,m+1}$.

In all cases, the nodes whose bags contain v form a non-empty subpath in the concatenation.

Now, consider an edge uv of G. If $u, v \in V_i$ ($1 \leqslant i \leqslant m$) or $u, v \in U_{i,i+1}$ ($0 \leqslant i \leqslant m$), then both u and v belong to a common bag of P_i or $P_{i,i+1}$ (respectively). Since the sets of the form U_i and $U_{i,i+1}$ are pairwise non-adjacent, if the edge uv connects two different sets of the form $U_{i,i+1}$, U_i, or $V(\zeta_i)$, then it connects $V(\zeta_i)$ with $N_G(V(\zeta_i))$ ($1 \leqslant i \leqslant m$), where the latter set is contained in B_i (by definition), so $u, v \in V_i$. Therefore, the above exhausts all the edges of G, showing that every edge is realized in some bag of the concatenation. □

It remains to provide the path decompositions P_γ for all $\gamma \in \Gamma$ or to deal with the cases where we cannot provide them. Let K be the unique subtree of T that has G as the root and the nodes in Z as the leaves. Thus, $h(\gamma, b) < k$ for every node γ in $T - K$, by the definition of Z. For $0 \leqslant i \leqslant m$, let $\zeta_{i,i+1}$ be the root of $Q_{i,i+1}$ in T, that is, the lowest common ancestor of ζ_i and ζ_{i+1} in T.

Claim 4.7. *If γ is a connected component of $G[V_i \setminus B_i]$ where $1 \leqslant i \leqslant m$, then either*

(a) *γ is a node in $T - K$ that is a child of ζ_i in T, or*
(b) *$V(\gamma) \cap V(\zeta_i) = \emptyset$ and $\gamma \in C_G(U_i \setminus B_i)$, which is possible only when $1 < i < m$.*

If γ is a connected component of $G[U_{i,i+1} \setminus B_{Q_{i,i+1}}]$ where $0 \leqslant i \leqslant m$, then either

(c) *γ is a node in $T - K$ that is a child in T of a node from $Q_{i,i+1}$, or*
(d) *$V(\gamma) \cap V(\zeta_{i,i+1}) = \emptyset$, which is possible only when $1 \leqslant i < m$.*

Proof. For the proof of the first statement, let γ be a connected component of $G[V_i \setminus B_i]$ where $1 \leqslant i \leqslant m$. Since $N_G(V(\zeta_i)) \subseteq B_i$ and $N_G(U_i) \subseteq V(\zeta_i)$ (if $1 < i < m$), we have $N_G(V(\zeta_i) \setminus B_i) \subseteq B_i$ and $N_G(U_i \setminus B_i) \subseteq B_i$ (if $1 < i < m$). Thus, $N_G(V_i \setminus B_i) \subseteq B_i$, which implies that γ is a connected component of $G - B_i$. By Lemma 4.4 applied with Q consisting of the single node ζ_i, either

(a) γ is a child of ζ_i in T, so it is a node in $T - K$, as ζ_i is a leaf of K, or
(b) $V(\gamma) \cap V(\zeta_i) = \emptyset$, which is possible only when $1 < i < m$, because $V_1 = V(\zeta_1) \cup B_1$ and $V_m = V(\zeta_m) \cup B_m$; in that case, $V(\gamma) \subseteq U_i \setminus B_i$, so $\gamma \in C_G(U_i \setminus B_i)$, as $N_G(U_i \setminus B_i) \subseteq B_i$.

For the proof of the second statement, let γ be a connected component of $G[U_{i,i+1} \setminus B_{Q_{i,i+1}}]$ where $0 \leqslant i \leqslant m$. We have $N_G(U_{i,i+1}) \subseteq V(\zeta_i) \cup V(\zeta_{i+1})$, $N_G(V(\zeta_i)) \subseteq B_i$, and $N_G(V(\zeta_{i+1})) \subseteq B_{i+1}$, which imply $N_G(U_{i,i+1} \setminus B_{Q_{i,i+1}}) \subseteq B_{Q_{i,i+1}}$, as $\zeta_i, \zeta_{i+1} \in V(Q_{i,i+1})$. Therefore, γ is a connected component of $G - B_{Q_{i,i+1}}$. By Lemma 4.4 applied with $Q = Q_{i,i+1}$, either

(c) γ is a node in $T - Q_{i,i+1}$ that is a child in T of some node ξ of $Q_{i,i+1}$, so γ is a node in $T - K$, as $V(\gamma)$ is disjoint from $V(\zeta_j)$ for every leaf ζ_j of K, or
(d) $V(\gamma) \cap V(\zeta_{i,i+1}) = \emptyset$, which is possible only when $1 \leqslant i < m$ (otherwise $\zeta_{i,i+1} = G$). □

Let a component $\gamma \in \Gamma$ be called a *child component* when case (a) or (c) of Claim 4.7 holds for γ and a *parent component* when case (b) or (d) of Claim 4.7 holds for γ. Case (a) or (c) of Claim 4.7 implies that for every child component γ, there has been a primary recursive call to solve(γ, b) and $h(\gamma, b) < k$, so $P_\gamma = P(\gamma, b)$ is a path decomposition of γ of width at most $t(k-1) - 1$.

An attempt to deal with parent components γ would be to run solve(γ, k) for each of them to compute $h(\gamma, k)$, $W(\gamma, k)$, and $P(\gamma, k)$ when $h(\gamma, k) < k$. If every such recursive call led to $h(\gamma, k) < k$, then $P(\gamma, k)$ would be a requested path decomposition P_γ of γ of width at most $t(k-1)+1$ for every parent component γ. If some of these recursive calls led to $h(\gamma, k) = k$, then we would set $h(G, b) = k+1$ and use $W(\gamma, k)$ to construct a requested $\mathcal{T}_{k+1}$-witness $W(G, b)$, while $P(G, b)$ would be constructed as described in outcome (B) with no need for explicit path decompositions of the parent components. However, this approach fails in that we are unable to provide a polynomial upper bound on the number of distinct pairs (G, b) for which recursive calls to solve(G, b) would be made.

We overcome this difficulty as follows. For each parent component γ, instead of running solve(γ, k), we run solve$(\hat{\gamma}, k)$ for an appropriately defined connected induced subgraph $\hat{\gamma}$ of $G[U]$ such that $V(\gamma) \subseteq V(\hat{\gamma})$, to compute $h(\hat{\gamma}, k)$, $W(\hat{\gamma}, k)$, and $P(\hat{\gamma}, k)$ when $h(\hat{\gamma}, k) < k$. We call these recursive calls *secondary*. Their role is analogous to the role of the calls to solve(γ, k) in the attempt above. If every secondary call leads to $h(\hat{\gamma}, k) < k$, then $P(\hat{\gamma}, k)$ is a path decomposition of $\hat{\gamma}$ of width at most $t(k-1)+1$, and the requested path decomposition P_γ of every parent component γ is obtained from the respective $P(\hat{\gamma}, k)$ by restricting the bags to $V(\gamma)$. If some secondary call leads to $h(\hat{\gamma}, k) = k$, then we set $h(G, b) = k+1$ and use $W(\hat{\gamma}, k)$ to construct a requested $\mathcal{T}_{k+1}$-witness $W(G, b)$ (as described in Claim 4.8 below), while $P(G, b)$ is constructed as described in outcome (B) with no need for explicit path decompositions of the parent components. The induced subgraphs $\hat{\gamma}$ are defined in a way (described below) that behaves nicely in the recursion and will allow us (in Section 4.3) to provide a polynomial upper bound on the number of distinct pairs (G, b) for which recursive calls to solve(G, b) are made.

The definition that follows is technical, but it is exactly what we need to be able to prove Lemma 4.9. For $\sigma \in \mathrm{Sub}(G)$, let A_σ be as in the definition of normalized tree decomposition, so that $A_\sigma \subseteq V(\sigma)$ and $B_\sigma = A_\sigma \cup N_G(V(\sigma))$. Observe that if $v \in B_\xi$ where $\xi \in \mathrm{Sub}(G)$, then all nodes in $\mathrm{Sub}(G)$ containing v, in particular the node σ such that $v \in A_\sigma$, lie on the path from ξ to the root G in T. For $1 \leqslant i < m$, let $R_{i,i+1}$ be the set of vertices $v \in U \cap B_{\zeta_{i,i+1}}$ with the following property: if σ is the node in $\mathrm{Sub}(G)$ such that $v \in A_\sigma$ (which therefore lies on the path from $\zeta_{i,i+1}$ to the root G in T), then v is connected by a $(U \cap V(\sigma))$-path to $V(\zeta_i)$ and to $V(\zeta_{i+1})$; thus $R_{i,i+1} \subseteq U_{i,i+1}$. For $1 \leqslant i \leqslant m$, let $R_i = U \cap B_i = B_i \setminus V(\zeta_i)$; thus $R_1 \subseteq U_{0,1} \cup U_{1,2}$, $R_i \subseteq U_{i-1,i} \cup U_i \cup U_{i,i+1}$ if $1 < i < m$, and $R_m \subseteq U_{m-1,m} \cup U_{m,m+1}$. Let $R = R_1 \cup R_{1,2} \cup R_2 \cup \cdots \cup R_{m-1,m} \cup R_m$.

For each parent component γ, the induced subgraph $\hat{\gamma}$ used in the description above is defined as follows. If $\gamma \in C_G(U_i \setminus B_i)$ with $1 < i < m$, as in case (b) of Claim 4.7, then $\hat{\gamma} = \gamma$, which is a connected component of $G[U_i \setminus R]$, because $U_i \cap R = U_i \cap R_i = U_i \cap B_i$. If $\gamma \in C_G(U_{i,i+1} \setminus B_{Q_{i,i+1}})$ with $1 \leqslant i < m$, as in case (d) of Claim 4.7, then $\hat{\gamma}$ is the connected component of $G[U_{i,i+1} \setminus R]$ such that $V(\gamma) \subseteq V(\hat{\gamma})$, which exists because $U_{i,i+1} \cap R = U_{i,i+1} \cap (R_i \cup R_{i,i+1} \cup R_{i+1}) \subseteq B_{Q_{i,i+1}}$. Let each $\hat{\gamma}$ obtained this way from a parent component γ be called a *secondary component*. Every secondary component is a connected component of $G[U \setminus R]$ and thus of $G - R$, as $N_G(\bigcup_{i=1}^{m} V(\zeta_i)) \subseteq R$.

Claim 4.8. *If $h(\hat{\gamma}, k) = k$ for a secondary component $\hat{\gamma}$, then the following is a $\mathcal{T}_{k+1}$-witness for G:*

- $\langle V(G); W(\zeta_{i-1}, b), W(\hat{\gamma}, k), W(\zeta_{i+1}, b)\rangle$ *when* $\hat{\gamma} \in C_G(U_i \setminus R)$ *with* $1 < i < m$;
- $\langle V(G); W(\zeta_i, b), W(\hat{\gamma}, k), W(\zeta_{i+1}, b)\rangle$ *when* $\hat{\gamma} \in C_G(U_{i,i+1} \setminus R)$ *with* $1 \leqslant i < m$.

Proof. If $\hat{\gamma} \in C_G(U_i \setminus R)$ with $1 < i < m$, then there is a path in G connecting

- $V(\zeta_{i-1})$ with $V(\hat{\gamma})$ via $U_{i-1,i} \cup V_i$, thus avoiding $V(\zeta_{i+1})$,
- $V(\hat{\gamma})$ with $V(\zeta_{i+1})$ via $V_i \cup U_{i,i+1}$, thus avoiding $V(\zeta_{i-1})$,
- $V(\zeta_{i-1})$ with $V(\zeta_{i+1})$ via $U_{i-1,i} \cup V(\zeta_i) \cup U_{i,i+1}$, thus avoiding $V(\hat{\gamma})$.

This shows that $\langle V(G); W(\zeta_{i-1}, b), W(\hat{\gamma}, k), W(\zeta_{i+1}, b)\rangle$ is a $\mathcal{T}_{k+1}$-witness for G.

Now, suppose $\hat{\gamma} \in C_G(U_{i,i+1} \setminus R)$ with $1 \leqslant i < m$. There is a path in G connecting

- $V(\zeta_i)$ with $V(\hat{\gamma})$ via $U_{i,i+1}$, thus avoiding $V(\zeta_{i+1})$,
- $V(\hat{\gamma})$ with $V(\zeta_{i+1})$ via $U_{i,i+1}$, thus avoiding $V(\zeta_i)$.

To show that $\langle V(G); W(\zeta_i, b), W(\hat{\gamma}, k), W(\zeta_{i+1}, b)\rangle$ is a $\mathcal{T}_{k+1}$-witness for G, it remains to provide a path in G connecting $V(\zeta_i)$ with $V(\zeta_{i+1})$ that avoids $V(\hat{\gamma})$.

Let X be the set of vertices in $U \cap V(\zeta_{i,i+1})$ that are connected by a $(U \cap V(\zeta_{i,i+1}))$-path to $V(\zeta_i)$ and to $V(\zeta_{i+1})$. Since $\zeta_{i,i+1}$ is connected and $V(\zeta_i) \cup V(\zeta_{i+1}) \subseteq V(\zeta_{i,i+1})$, there is a $V(\zeta_{i,i+1})$-path connecting $V(\zeta_i)$ and $V(\zeta_{i+1})$. A minimal path with that property has all internal vertices in U and therefore in X. To complete the proof, we show that $V(\hat{\gamma}) \cap X = \emptyset$.

Suppose for the sake of contradiction that $V(\hat{\gamma}) \cap X \neq \emptyset$. Since $\hat{\gamma}$ is a secondary component, there is a parent component $\gamma \in C_G(U_{i,i+1} \setminus B_{Q_{i,i+1}})$ such that $V(\gamma) \cap V(\zeta_{i,i+1}) = \emptyset$ and $V(\gamma) \subseteq V(\hat{\gamma})$, so $V(\hat{\gamma}) \not\subseteq X$. Therefore, since $\hat{\gamma}$ is connected, there is a vertex $v \in V(\hat{\gamma}) \setminus X$ with a neighbor in X. It follows that v is connected by a $(U \cap V(\zeta_{i,i+1}))$-path to $V(\zeta_i)$ and to $V(\zeta_{i+1})$. Since $v \in U \setminus X$, we have $v \notin V(\zeta_{i,i+1})$ (by the definition of X) and thus $v \in N_G(V(\zeta_{i,i+1})) \subseteq B_{\zeta_{i,i+1}}$. Let σ be the node on the path from $\zeta_{i,i+1}$ to the root in T such that $v \in A_\sigma$. Thus $V(\zeta_{i,i+1}) \subseteq V(\sigma)$. We conclude that $v \in U \cap B_{\zeta_{i,i+1}}$ and v is connected by a $(U \cap V(\sigma))$-path to $V(\zeta_i)$ and to $V(\zeta_{i+1})$, so $v \in R_{i,i+1}$, which contradicts the fact that $V(\hat{\gamma}) \cap R = \emptyset$. □

This completes the description of the procedure solve(G, b). Since all recursive calls of the form solve(γ, c) that it makes are for proper connected induced subgraphs γ of G (and for $c \leqslant b$), the procedure terminates and correctly computes the requested outcome.

4.3 Complexity Analysis

The algorithm consists in running solve(G^{in}, ∞) on the input graph G^{in}. It makes further recursive calls to solve(β, b) for various connected induced subgraphs β of G^{in} and upper bound requests b. Let every pair (β, b) such that solve(β, b) is run by the algorithm (somewhere in the recursion tree) be called a *subproblem*. We show that if G^{in} has n vertices, then there are only $O(n \log n)$ subproblems.

A *pruned subproblem* is a subproblem (β, b) for which the run of solve(β, b) is pruned, which implies $h(\beta, b) = b$. Observe that unless (β, b) is pruned, the run of solve(β, b) performs operations that do not depend on the value of b (including the operations performed in all recursive calls). Indeed, unless (β, b) is pruned, no primary recursive call made by solve(β, b) is pruned, so (by induction) these calls perform operations and lead to results that do not depend on b, and every secondary recursive call made by solve(β, b) is of the form solve$(\hat{\gamma}, k)$ where k does not depend on b.

A *key subproblem* is a subproblem (β, b) for which the run of solve(β, b) is a key run, that is, $|Z| \geqslant 2$ and H is a path. In particular, a key subproblem is not pruned. A *key subgraph* is a connected induced subgraph β of G^{in} such that (β, b) is a key subproblem for some $b \in \mathbb{N} \cup \{\infty\}$.

For every key subgraph β, let level(β) denote the value of k defined in a key run of solve(β, b), that is, the maximum of $h(\gamma, b)$ over all $\gamma \in \text{Sub}(\beta) \setminus \{\beta\}$. (As we observed above, these values do not depend on b, so neither does level(β).) For every key subgraph β, let $R(\beta)$ be the set $R = R_1 \cup R_{1,2} \cup R_2 \cup \cdots \cup R_{m-1,m} \cup R_m$ defined in a key run of solve(β, b). (As before, it does not depend on b.) The following lemma exhibits the crucial property of the sets $R(\beta)$.

Lemma 4.9. *The following holds for any key subgraphs α and β such that $\alpha \in \text{Sub}(\beta)$.*

(1) level$(\alpha) \leqslant$ level(β).
(2) *If* level$(\alpha) <$ level(β), *then* $R(\beta) \cap V(\alpha) = \emptyset$.
(3) *If* level$(\alpha) =$ level(β), *then* $R(\beta) \cap V(\alpha) = R(\alpha)$.

Proof. Since $\alpha \in \mathrm{Sub}(\beta)$, it follows from Lemma 4.5 (3) that $\mathrm{Sub}(\alpha) \subseteq \mathrm{Sub}(\beta)$. This directly shows (1), by the definition of level.

Let $(T, \{B_\sigma\}_{\sigma \in \mathrm{Sub}(\beta)})$ be the normalized tree decomposition of β used in a key run of solve on β, and let $\{A_\sigma\}_{\sigma \in \mathrm{Sub}(\beta)}$ be the corresponding sets from the definition of normalized tree decomposition. (Their construction from the input tree decomposition of G^{in} depends only on β.) Lemma 4.5 (3) implies that $(T^\alpha, \{B_\sigma \cap V(\alpha)\}_{\sigma \in \mathrm{Sub}(\alpha)})$, where T^α is the subtree of T comprising α and the nodes below α in T, is the normalized tree decomposition of α used in any key run of solve on α. Let Z, m, H, U, ζ_i, $\zeta_{i,i+1}$, B_i, R_i, and $R_{i,i+1}$ be defined as in a key run of solve on β.

For the proof of (2), assume $\mathrm{level}(\alpha) < \mathrm{level}(\beta)$. The definition of level implies that none of the nodes in Z lie below α in T. Recall that if $v \in B_\xi$ where $\xi \in \mathrm{Sub}(\beta)$, then all nodes in $\mathrm{Sub}(\beta)$ containing v lie on the path from ξ to the root β in T. Therefore, for $1 \leqslant i \leqslant m$, we have $R_i \cap V(\alpha) = \emptyset$, as $R_i = B_i \setminus V(\zeta_i)$. Similarly, for $1 \leqslant i < m$, we have $R_{i,i+1} \cap V(\alpha) = \emptyset$, as $R_{i,i+1} \subseteq B_{\zeta_{i,i+1}}$, where the node $\zeta_{i,i+1}$ lies above ζ_i and ζ_{i+1} in T. This yields $R(\beta) \cap V(\alpha) = \emptyset$.

For the proof of (3), assume $\mathrm{level}(\alpha) = \mathrm{level}(\beta)$. The definition of level implies that $\alpha \notin Z$ and none of the nodes in Z lies above α in T. Let Z^α, H^α, and U^α be the respective Z, H, and U defined in any key run of solve on α. (We keep using Z, H, U, and other notations with no superscript to denote what is defined in a key run of solve on β.) It follows that Z^α is the set of nodes in Z that lie below α in T. Lemma 4.2 (2) implies $V(\zeta_i) \cap V(\alpha) = \emptyset$ for every $\zeta_i \in Z \setminus Z^\alpha$. It follows that $U^\alpha = U \cap V(\alpha)$. Consequently, the path H^α is a subpath of H, and $Z^\alpha = \{\zeta_r, \ldots, \zeta_s\}$ where $1 \leqslant r < s \leqslant m$. Let $R^\alpha_r, R^\alpha_{r,r+1}, R^\alpha_{r+1}, \ldots, R^\alpha_{s-1,s}, R^\alpha_s$ denote the sets $R_1, R_{1,2}, R_2, \ldots, R_{m-1,m}, R_m$ defined in a key run of solve on α in this or the reverse order matching the order of $\zeta_r, \ldots, \zeta_s$.

Let $1 \leqslant i \leqslant m$. If $\zeta_i \in Z^\alpha$, then $R^\alpha_i = U^\alpha \cap B_i \cap V(\alpha) = U \cap B_i \cap V(\alpha) = R_i \cap V(\alpha)$. If $\zeta_i \notin Z^\alpha$, then ζ_i does not lie below α in T, so $R_i \cap V(\alpha) = (B_i \setminus V(\zeta_i)) \cap V(\alpha) = \emptyset$. Now, let $1 \leqslant i < m$. Suppose $\zeta_i, \zeta_{i+1} \in Z^\alpha$. It follows that $\zeta_{i,i+1}$ is a node in T^α. For each node σ on the path from $\zeta_{i,i+1}$ to α in T, since $A_\sigma \subseteq V(\sigma) \subseteq V(\alpha)$, the set A_σ contributes the same vertices to both $R^\alpha_{i,i+1}$ and $R_{i,i+1}$, namely, the vertices in $U \cap B_{\zeta_{i,i+1}} \cap A_\sigma$ that are connected by a $(U \cap V(\sigma))$-path to $V(\zeta_i)$ and to $V(\zeta_{i+1})$. For each node σ above α in T, the set A_σ contributes no vertices to $R^\alpha_{i,i+1}$, and the vertices it contributes to $R_{i,i+1}$ are not in $V(\alpha)$, as $A_\sigma \cap V(\alpha) = \emptyset$. Thus $R^\alpha_{i,i+1} = R_{i,i+1} \cap V(\alpha)$. On the other hand, if $\zeta_i \notin Z^\alpha$ or $\zeta_{i+1} \notin Z^\alpha$, then $\zeta_{i,i+1}$ does not lie in T_α, so $B_{\zeta_{i,i+1}} \cap V(\alpha) = \emptyset$ and thus $R_{i,i+1} \cap V(\alpha) = \emptyset$. We conclude that

$$\begin{aligned} R(\alpha) &= R^\alpha_r \cup R^\alpha_{r,r+1} \cup R^\alpha_{r+1} \cup \cdots \cup R^\alpha_{s-1,s} \cup R^\alpha_s \\ &= (R_1 \cup R_{1,2} \cup R_2 \cup \cdots \cup R_{m-1,m} \cup R_m) \cap V(\alpha) \\ &= R(\beta) \cap V(\alpha). \end{aligned}$$

□

Let ℓ be the maximum of $\mathrm{level}(\beta)$ over all key subgraphs β. It follows that every key subproblem is of the form (β, k) with $k \in \{0, \ldots, \ell, \infty\}$. For $k \in \{0, \ldots, \ell\}$, let R^k be the union of the sets $R(\beta)$ over all key subgraphs β such that $\mathrm{level}(\beta) = k$. (We have $R^k = \emptyset$ if there are no such key subgraphs.) Let $R^{\geqslant k}$ be $R^k \cup \cdots \cup R^\ell$ for $k \in \{0, \ldots, \ell\}$ and empty for $k = \infty$. The following lemma will finally allow us to bound the total number of subproblems.

Lemma 4.10. *For every $k \in \{0, \ldots, \ell, \infty\}$, every subproblem (γ, k) satisfies $\gamma \in \mathrm{Sub}(G^{\mathrm{in}} - R^{\geqslant k})$.*

Proof. We aim to prove that the following two statements, the first of which is the statement of the lemma, hold for all $k \in \{0, \ldots, \ell, \infty\}$.

(i) Every subproblem (γ, k) satisfies $\gamma \in \mathrm{Sub}(G^{\mathrm{in}} - R^{\geqslant k})$.

(ii) Every key subproblem (α, a) with $a \geqslant k > \mathrm{level}(\alpha)$ satisfies $\alpha \in \mathrm{Sub}(G^{\mathrm{in}} - R^{\geqslant k})$.

Suppose not, and let $k \in \{0, \ldots, \ell, \infty\}$ be maximum for which (i) or (ii) fails. Let $R^{>k}$ be $R^{\geqslant k+1}$ for $k \in \{0, \ldots, \ell - 1\}$ and empty for $k \in \{\ell, \infty\}$. By maximality of k, the following statement

holds, which is equivalent to (ii) for $k+1$ if $k < \ell$, equivalent to (ii) for ∞ if $k = \ell$, and vacuous if $k = \infty$.

($*$) Every key subproblem (α, a) with $a > k \geqslant \text{level}(\alpha)$ satisfies $\alpha \in \text{Sub}(G^{\text{in}} - R^{>k})$.

First, suppose that (i) fails for k. Consider a subproblem (γ, k) with γ maximal for which (i) fails, so that (i) holds for all subproblems (β, k) with $V(\gamma) \subset V(\beta)$. Since $G^{\text{in}} \in \text{Sub}(G^{\text{in}})$, (i) holds when $(\gamma, k) = (G^{\text{in}}, \infty)$. Thus, assume $(\gamma, k) \neq (G^{\text{in}}, \infty)$, so that a primary or secondary recursive call to solve(γ, k) occurs in the algorithm. If there is a primary call to solve(γ, k) from solve(β, k) where $\gamma \in \text{Sub}(\beta) \setminus \{\beta\}$, then $\beta \in \text{Sub}(G^{\text{in}} - R^{\geqslant k})$ implies $\gamma \in \text{Sub}(G^{\text{in}} - R^{\geqslant k})$ by Lemma 4.5 (3), which contradicts the assumption that (i) fails for (γ, k). Thus, assume the other case, namely, that $k \leqslant \ell$ and the algorithm makes a secondary call to solve(γ, k) from solve(α, a) for some key subproblem (α, a) with $a > k = \text{level}(\alpha)$. By ($*$), we have $\alpha \in \text{Sub}(G^{\text{in}} - R^{>k})$.

We claim that $R^k \cap V(\alpha) = R(\alpha)$. It is clear that $R^k \cap V(\alpha) \supseteq R(\alpha)$. Now, let $v \in R^k \cap V(\alpha)$. Let (β, b) be a key subproblem with $b > k = \text{level}(\beta)$ such that $v \in R(\beta)$ (which exists by the definition of R^k). By ($*$), we have $\beta \in \text{Sub}(G^{\text{in}} - R^{>k})$. The fact that $v \in V(\alpha) \cap V(\beta)$ and Lemma 4.5 (2) yield $\alpha \in \text{Sub}(\beta)$ or $\beta \in \text{Sub}(\alpha)$. If $\alpha \in \text{Sub}(\beta)$, then Lemma 4.9 (3) yields $v \in R(\beta) \cap V(\alpha) = R(\alpha)$. If $\beta \in \text{Sub}(\alpha)$, then Lemma 4.9 (3) (with α and β interchanged) yields $v \in R(\beta) \subseteq R(\alpha)$. This completes the proof of the claim.

Being a secondary component in solve(α, a), γ is a connected component of $\alpha - R(\alpha)$ and thus of $\alpha - R^k$, as $R^k \cap V(\alpha) = R(\alpha)$. Since $\alpha \in \text{Sub}(G^{\text{in}} - R^{>k})$, Lemma 4.5 (4) applied with $G = G^{\text{in}} - R^{>k}$ and $X = V(G^{\text{in}}) \setminus R^{\geqslant k}$ yields $\gamma \in \text{Sub}(G^{\text{in}} - R^{\geqslant k})$, which is again a contradiction.

Having assumed that k is maximal for which (i) or (ii) fails, we have proved that (i) actually holds for k, so it must be (ii) that fails for k. So let (α, a) be a key subproblem with $a \geqslant k > \text{level}(\alpha)$. If $a = k$, then $\alpha \in \text{Sub}(G^{\text{in}} - R^{\geqslant k})$ by the fact that (i) holds for k that we have proved above. So assume $a > k$. By ($*$), we have $\alpha \in \text{Sub}(G^{\text{in}} - R^{>k})$. Suppose there is a vertex $v \in R^k \cap V(\alpha)$. As before, let (β, b) be a key subproblem with $b > k = \text{level}(\beta)$ such that $v \in R(\beta)$. By ($*$), we have $\beta \in \text{Sub}(G^{\text{in}} - R^{>k})$. The fact that $v \in V(\alpha) \cap V(\beta)$ and Lemma 4.5 (2) yield $\alpha \in \text{Sub}(\beta)$ or $\beta \in \text{Sub}(\alpha)$. If $\alpha \in \text{Sub}(\beta)$, then Lemma 4.9 (2) yields $R(\beta) \cap V(\alpha) = \emptyset$, which is a contradiction. If $\beta \in \text{Sub}(\alpha)$, then $\text{level}(\beta) \leqslant \text{level}(\alpha)$ by Lemma 4.9 (1) (with α and β interchanged), which is again a contradiction. Thus, $R^k \cap V(\alpha) = \emptyset$, which implies $\alpha \in \text{Sub}(G^{\text{in}} - R^{\geqslant k})$ by Lemma 4.5 (1). We conclude that (ii) holds for k, which is a final contradiction. □

Let $n = |V(G^{\text{in}})|$. Lemma 4.3 (1) implies that $|\text{Sub}(G^{\text{in}}[X])| \leqslant |X| \leqslant n$ for every $X \subseteq V(G^{\text{in}})$. This and Lemma 4.10 imply that for every $k \in \{0, \ldots, \ell, \infty\}$, there are at most n subproblems of the form (γ, k). A key subgraph β such that $\ell = \text{level}(\beta)$ has a $\mathcal{T}_\ell$-witness, so $n \geqslant |V(\beta)| \geqslant 3^\ell$. Therefore, the total number of subproblems is $O(n \log n)$. Since the operations performed in a single pass of solve (excluding the recursive calls) clearly take polynomial time, we conclude that the full run of solve(G^{in}, ∞) takes polynomial time. This completes the proof of Theorem 4.1.

5 TIGHTNESS

Theorem 1.1 asserts that every graph with pathwidth at least $th + 2$ has treewidth at least t or contains a subdivision of a complete binary tree of height $h + 1$. While this statement is true for all positive integers t and h, we remark that the interesting case is when $h > \log_2 t - 2$. Indeed, if $h \leqslant \log_2 t - 2$, then the second outcome is known to hold for every graph with pathwidth at least t; this follows from a result of Bienstock et al. [3].[2] We now show that Theorem 1.1 is tight up to a multiplicative factor when $h > \log_2 t - 2$.

[2]In [3], it is proved that for every forest F, graphs with no F minors have pathwidth at most $|V(F)| - 2$. In particular, if G contains no subdivision of a complete binary tree of height $h + 1$, then $\text{pw}(G) < 2^{h+2} \leqslant t$.

Theorem 5.1. *For any positive integers t and h, there is a graph with treewidth t and pathwidth at least $t(h+1)-1$ that contains no subdivision of a complete binary tree of height $3\max(h+1, \lceil \log_2 t \rceil)$.*

Fix a positive integer t. For a tree T, let $T^{(t)}$ be a graph obtained from T by replacing every node of T with a clique on t vertices and replacing every edge of T with an arbitrary perfect matching between the corresponding cliques. For $h \in \mathbb{N}$, let T_h be a complete ternary tree of height h. The following three claims show that the graph $T_h^{(t)}$ satisfies the three conditions requested in Theorem 5.1, thus proving that Theorem 5.1 holds for $T_h^{(t)}$.

Claim 5.2. *If T is a tree on at least two vertices, then $\mathrm{tw}(T^{(t)}) = t$.*

Proof. For each node x of T, let B_x be the clique of t vertices in $T^{(t)}$ corresponding to x. A tree decomposition of $T^{(t)}$ of width t is obtained from T by taking B_x as the bag of every node x of T and by subdividing every edge xy of T into a path of length $t+1$ with the following sequence of bags, assuming that the vertices $u_1, \ldots, u_t$ in B_x are matched to $v_1, \ldots, v_t$ in B_y, respectively:

$$\{u_1, \ldots, u_t\},\ \{u_1, \ldots, u_t\} \cup \{v_1\},\ \{u_2, \ldots, u_t\} \cup \{v_1, v_2\},\ \ldots,\ \{u_t\} \cup \{v_1, \ldots, v_t\},\ \{v_1, \ldots, v_t\}.$$

This way, for every matching edge $u_i v_i$ with $1 \leqslant i \leqslant t$, there is a bag containing its two endpoints. Consequently, this is a valid tree decomposition of $T^{(t)}$ with bags of size at most $t+1$.

For the proof that $\mathrm{tw}(T^{(t)}) \geqslant t$, let xy be an edge of T, and assume that the vertices $u_1, \ldots, u_t$ of B_x are matched to $v_1, \ldots, v_t$ in B_y, respectively, as before. In any tree decomposition of $T^{(t)}$, there is a node x' whose bag contains the clique B_x and a node y' whose bag contains the clique B_y. Walk on the path from x' to y' and stop at the first node whose bag contains some vertex in B_y. This bag must also contain all of B_x, so it has size at least $t+1$. □

Claim 5.3. *For every $h \in \mathbb{N}$, we have $\mathrm{pw}(T_h^{(t)}) \geqslant t(h+1) - 1$.*

Proof. We define the *root clique* of $T_h^{(t)}$ as the clique in $T_h^{(t)}$ corresponding to the root of T_h. We prove the following slightly stronger statement, by induction on h: in every path decomposition of $T_h^{(t)}$, there are a bag B of size at least $t(h+1)$ and t vertex-disjoint paths in $T_h^{(t)}$ each having one endpoint in the root clique and the other endpoint in B.

For the base case $h = 0$, the graph $T_0^{(t)}$ is simply a complete graph on t vertices, and the statement is trivial. For the induction step, assume that $h \geqslant 1$ and the statement is true for $h-1$. Let R be the root clique of $T_h^{(t)}$. Let $(P, \{B_x\}_{x \in V(P)})$ be a path decomposition of $T_h^{(t)}$ of minimum width. The graph $T_h^{(t)} - R$ has three connected components C_1, C_2, and C_3 that are copies of $T_{h-1}^{(t)}$ with root cliques R_1, R_2, and R_3, respectively. For each $i \in \{1, 2, 3\}$, the induction hypothesis applied to the path decomposition $(P, \{B_x \cap V(C_i)\}_{x \in V(P)})$ of C_i provides a node x_i of P such that

- $|B_{x_i} \cap V(C_i)| \geqslant th$, and
- there are t vertex-disjoint paths in C_i between $B_{x_i} \cap V(C_i)$ and the root clique R_i of C_i.

Assume without a loss of generality that the node x_2 occurs between x_1 and x_3 on P. We prove the induction statement for $B = B_{x_2}$.

For each $i \in \{1, 2, 3\}$, we take the t vertex-disjoint paths from B_{x_i} to R_i in C_i and extend them by the matching between R_i and R to obtain t vertex-disjoint paths from B_{x_i} to R in $T_h^{(t)}[R \cup V(C_i)]$. In particular, there are t vertex-disjoint paths from B_{x_2} to R in $T_h^{(t)}$, as required in the induction statement. Since $|R| = t$, the t paths from B_{x_1} to R and the t paths from B_{x_3} to R together form t vertex-disjoint paths from B_{x_1} to B_{x_3} in $T_h^{(t)}[V(C_1) \cup R \cup V(C_3)]$, which therefore avoid $V(C_2)$. Since x_2 lies between x_1 and x_3 on P, the set $B_{x_2} \setminus V(C_2)$ must contain at least one vertex from each of these t paths. Thus, $|B_{x_2} \setminus V(C_2)| \geqslant t$. Since $|B_{x_2} \cap V(C_2)| \geqslant th$, we conclude that $|B_{x_2}| \geqslant t(h+1)$, as required in the induction statement. □

Claim 5.4. *For any $h \in \mathbb{N}$, the graph $T_h^{(t)}$ contains no subdivision of a complete binary tree of height $3\max(h+1, \lceil \log_2 t \rceil)$.*

Proof. A simple calculation shows that T_h has $\frac{3^{h+1}-1}{2}$ nodes. Thus, $|V(T_h^{(t)})| + 1 \leqslant 3^{h+1}t$ and so

$$\log_2\left(|V\left(T_h^{(t)}\right)| + 1\right) \leqslant \log_2(3^{h+1}t) \leqslant 3\max(h+1, \lceil \log_2 t \rceil).$$

If a graph G contains a subdivision of a complete binary tree of height c, then $|V(G)| \geqslant 2^{c+1} - 1$ and so $\log_2(|V(G)| + 1) \geqslant c + 1$. Therefore, $T_h^{(t)}$ cannot contain a subdivision of a complete binary tree of height $3\max(h+1, \lceil \log_2 t \rceil)$. □

6 OPEN PROBLEM

In Theorem 1.1, we bound pathwidth by a function of treewidth in the absence of a subdivision of a large complete binary tree. In [12, 22], the authors bound treedepth by a function of treewidth in the absence of a subdivision of a large complete binary tree and of a long path. Specifically, Czerwiński et al. [12] proved the following bound.[3]

Theorem 6.1. *Every graph with treewidth t that contains no subdivision of a complete binary tree of height h and no path of order 2^ℓ has treedepth $O(th\ell)$.*

It is natural to ask how large treedepth can be as a function of pathwidth when there is no long path. We offer the following conjecture.

Conjecture 6.2. *Every graph with pathwidth p that contains no path of order 2^ℓ has treedepth $O(p\ell)$.*

This conjecture and Theorem 1.1 would directly imply Theorem 6.1. We note that an $O(p^2\ell)$ bound on the treedepth follows from Theorem 6.1, as every graph with pathwidth p has treewidth at most p and contains no subdivision of a complete binary tree of height $2p+1$. We also note that an $O(p\ell)$ bound on the treedepth would be best possible; see [12, Section 7].

ACKNOWLEDGMENTS

This research was started at the Structural Graph Theory workshop in Gułtowy, Poland, in June 2019. We thank the organizers and the other workshop participants for creating a productive working atmosphere. We also thank the anonymous referees for their helpful comments. We are particularly grateful to one referee for pointing out a serious error in an earlier version of the paper.

REFERENCES

[1] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. 1987. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods* 8, 2 (1987), 277–284.
[2] Patrick Bellenbaum and Reinhard Diestel. 2002. Two short proofs concerning tree-decompositions. *Combinatorics, Probability and Computing* 11, 6 (2002), 541–547.
[3] Dan Bienstock, Neil Robertson, Paul Seymour, and Robin Thomas. 1991. Quickly excluding a forest. *Journal of Combinatorial Theory, Series B* 52, 2 (1991), 274–283.
[4] Hans L. Bodlaender. 1992. A tourist guide through treewidth. *Acta Cybernetica* 11, 1–2 (1992), 1–21.
[5] Hans L. Bodlaender. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25, 6 (1996), 1305–1317.
[6] Hans L. Bodlaender. 2012. Fixed-parameter tractability of treewidth and pathwidth. In *The Multivariate Algorithmic Revolution and Beyond: Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*. Springer, Berlin, 196–227.
[7] Hans L. Bodlaender and Fedor V. Fomin. 2002. Approximation of pathwidth of outerplanar graphs. *Journal of Algorithms* 43, 2 (2002), 190–200.

[3] This bound is stated in [12] for the case $t = h = \ell$, but the proof works as well for the general case.

[8] Hans L. Bodlaender and Ton Kloks. 1996. Efficient and constructive algorithms for the pathwidth and treewidth of Graphs. *Journal of Algorithms* 21, 2 (1996), 358–402.
[9] Kevin Cattell, Michael J. Dinneen, and Michael R. Fellows. 1996. A simple linear-time algorithm for finding path-decompositions of small width. *Information Processing Letters* 57, 4 (1996), 197–203.
[10] Chandra Chekuri and Julia Chuzhoy. 2016. Polynomial bounds for the grid-minor theorem. *J. ACM* 63, 5 (2016), Article No. 40.
[11] Julia Chuzhoy and Zihan Tan. 2021. Towards tight(er) bounds for the Excluded Grid Theorem. *Journal of Combinatorial Theory, Series B* 146 (2021), 219–265.
[12] Wojciech Czerwiński, Wojciech Nadara, and Marcin Pilipczuk. 2021. Improved bounds for the excluded-minor approximation of treedepth. *SIAM Journal on Discrete Mathematics* 35, 2 (2021), 934–947.
[13] Nathaniel Dean. 1993. Open problems. In *Graph Structure Theory: Proceedings of a Joint Summer Research Conference on Graph Minors*, Neil Robertson and Paul Seymour (Eds.). Contemporary Mathematics, Vol. 147. American Mathematical Society, Providence, 677–688.
[14] Reinhard Diestel. 2010. *Graph Theory* (4th ed.). Graduate Texts in Mathematics, Vol. 173. Springer, Berlin.
[15] Uriel Feige, Mohammad Taghi Hajiaghayi, and James R. Lee. 2008. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing* 38, 2 (2008), 629–657.
[16] Fedor V. Fomin and Dimitrios M. Thilikos. 2006. A 3-approximation for the pathwidth of Halin graphs. *Journal of Discrete Algorithms* 4, 4 (2006), 499–510.
[17] Pierre Fraigniaud and Nicolas Nisse. 2006. Connected treewidth and connected graph searching. In *LATIN 2006: Theoretical Informatics*, José R. Correa, Alejandro Hevia, and Marcos Kiwi (Eds.). Lecture Notes in Computer Science, Vol. 3887. Springer, Berlin, 479–490.
[18] Jens Gusted. 1993. On the pathwidth of chordal graphs. *Discrete Applied Mathematics* 45, 3 (1993), 233–248.
[19] Michel Habib and Rolf H. Möhring. 1994. Treewidth of cocomparability graphs and a new order-theoretic parameter. *Order* 11, 1 (1994), 47–60.
[20] Robert Hickingbotham. 2019. *Graph minors and tree decompositions*. Bachelor's thesis. School of Mathematics, Monash University, Melbourne.
[21] William A. Horn. 1972. Three results for trees, using mathematical induction. *Journal of Research of the National Bureau of Standards, Series B: Mathematical Sciences* 76B, 1–2 (1972), 39–43.
[22] Ken-ichi Kawarabayashi and Benjamin Rossman. 2022. A polynomial excluded-minor approximation of treedepth. *Journal of the European Mathematical Society* 24, 4 (2022), 1449–1470.
[23] Ton Kloks, Hans L. Bodlaender, Haiko Müller, and Dieter Kratsch. 1993. Computing treewidth and minimum fill-in: All you need are the minimal separators. In *Algorithms—ESA'93*, Thomas Lengauer (Ed.). Lecture Notes in Computer Science, Vol. 726. Springer, Berlin, 260–271.
[24] Ton Kloks, Dieter Kratsch, and Haiko Müller. 1995. Dominoes. In *Graph-Theoretic Concepts in Computer Science*, Ernst W. Mayr, Gunther Schmidt, and Gottfried Tinhofer (Eds.). Lecture Notes in Computer Science, Vol. 903. Springer, Berlin, 106–120.
[25] Emily A. Marshall and David R. Wood. 2015. Circumference and pathwidth of highly connected graphs. *Journal of Graph Theory* 79, 3 (2015), 222–232.
[26] Burkhard Monien and Ivan Hal Sudborough. 1988. Min Cut is NP-complete for edge weighted trees. *Theoretical Computer Science* 58, 1 (1988), 209–229.
[27] Neil Robertson and Paul Seymour. 1986. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B* 41, 1 (1986), 92–114.
[28] Petra Scheffler. 1989. *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme*. Ph.D. Dissertation. Akademie der Wissenschaften der DDR, Berlin.
[29] David R. Wood. 2013. Personal communication. (2013).

Received 29 March 2021; revised 24 November 2022; accepted 30 November 2022