Received: 23 December 2021 | Revised: 17 January 2023 | Accepted: 20 March 2023

DOI: 10.1002/jgt.22964

ARTICLE

WILEY

Automated testing and interactive construction of unavoidable sets for graph classes of small path-width

Oliver Bachtler[1] | **Irene Heinrich**[2]

[1]Optimization Research Group, Technische Universität Kaiserslautern, Kaiserslautern, Germany

[2]Graphs and Groups, Technische Universität Darmstadt, Darmstadt, Germany

Correspondence
Irene Heinrich, Graphs and Groups, Technische Universität Darmstadt, Darmstadt, Germany.
Email: heinrich@mathematik.tu-darmstadt.de

Funding information
European Research Council

Abstract

Let $\mathcal{G}$ be a class of graphs with a membership test, $k \in \mathbb{N}$, and let $\mathcal{G}^k$ be the class of graphs in $\mathcal{G}$ of path-width at most k. We present an interactive framework that finds an *unavoidable set* for $\mathcal{G}^k$, which is a set of graphs $\mathcal{U}$ such that any graph in $\mathcal{G}^k$ contains an isomorphic copy of a graph in $\mathcal{U}$. At the core of our framework is an algorithm that verifies whether a set of graphs is, indeed, unavoidable for $\mathcal{G}^k$. While obstruction sets are well-studied, so far there is no general theory or algorithm for finding unavoidable sets. In general, it is undecidable whether a finite set of graphs is unavoidable for a given graph class. However, we give a criterion for termination: our algorithm terminates whenever $\mathcal{G}$ is locally checkable of bounded maximum degree and $\mathcal{U}$ is a finite set of connected graphs. For example, l-regular graphs, l-colourable graphs, and H-free graphs are locally checkable classes. We put special emphasis on the case that $\mathcal{G}$ is the class of cubic graphs and tailor the algorithm to this case. In particular, we introduce the new concept of high-degree-first path-decompositions, which enables highly efficient pruning techniques. We exploit our framework to prove a new lower bound on the path-width of cubic graphs. Moreover, we determine the extremal girth values of cubic graphs of path-width k for all $k \in \{3, ..., 10\}$ and all smallest graphs which take on these extremal girth values. Further, we present a new constructive characterisation of the extremal cubic graphs of path-width 3 and girth 4.

KEYWORDS
cubic graph, girth, path-width, unavoidable structure

1 | INTRODUCTION

A set of graphs is *unavoidable* for a graph class if every graph in the class contains an isomorphic copy of a graph in the set. Unavoidable sets are extensively used in

- ▹ recursive algorithms and inductive proofs, for example, [5, 18],
- ▹ structural graph theory, where unavoidable sets give insights into the possible composition of graphs, see [10] for an example, and
- ▹ interactive proofs, for example, Appel and Haken's proof of the famous 4-colouring conjecture, cf. [1, 2].

While discharging [13] is a tool to find unavoidable structures for colouring problems and Ramsey theory [11] studies unavoidable sets in extremal graph theory, there is no generic approach for finding or checking whether a given set is unavoidable. Frequently, tedious case distinctions are necessary to prove that some set is indeed unavoidable for a considered class, cf. [6, 18]. We contribute a new automated method to this sparse list of techniques to find unavoidable sets.

We present an interactive framework for the automatic construction and testing of unavoidable sets parametrised by path-width. More precisely, let a class $\mathcal{G}$ with a membership test and a number $k \in \mathbb{N}$ be given and let

$$\mathcal{G}^k := \{G \in \mathcal{G} : G \text{ is of path-width at most } k\}.$$

At the core of our framework is an algorithm which investigates the hypothesis that a set of graphs $\mathcal{U}$ is unavoidable for $\mathcal{G}^k$. We prove that the algorithm terminates for a large variety of graph classes, for example, if $\mathcal{G}$ is the class of all ℓ-regular graphs for some $\ell \in \mathbb{N}$ and the graphs in $\mathcal{U}$ are connected.

The algorithm's role in the framework is the following: the framework starts with a (potentially empty) set of graphs and runs the algorithm on it. If the set is not unavoidable, the provided counterexample can be used to extend the set of structures, either by adding the graph itself or a subgraph. This is the interactive part of the framework since the desired or useful unavoidable structures often depend on the application. This process is repeated until the set of structures is unavoidable.

As an application of our framework, we combine it with combinatorial techniques to prove a new lower bound on the path-width of certain cubic graphs. Such lower bounds are invaluable to speed up exact path-width computations. Moreover, we give a complete list of all girth-extremal cubic graphs of small path-width.

1.1 | High-level description of the algorithm and challenges

Roughly speaking, our algorithm searches for a minimum counterexample to the hypothesis that $\mathcal{U}$ is unavoidable for $\mathcal{G}^k$. We faced two major challenges on our way to a practically applicable algorithm: a very large search space and the fact that, in general, it is undecidable whether a finite set of graphs is unavoidable for a given graph class, even if the class is decidable and of path-width 2 (see Lemma 3.5). To reduce the search space we exploit a natural linear vertex-ordering which is given by a path-decomposition and we make strong use of isomorphism rejection (see also the rest of this paragraph as well as Section 3). Concerning the undecidability, we give a general criterion which allows us to guarantee termination whenever $\mathcal{G}$ is locally checkable and of bounded maximum degree and the graphs in $\mathcal{U}$ are connected. Note that, amongst many others, the following graph classes are locally checkable: bipartite graphs, ℓ-colourable graphs, and H-free graphs.

To be a bit more precise, the algorithm runs in phases. After the ith phase, it either

- ▷ returns a counterexample of order $k + i$, or
- ▷ returns `unavoidable`, guaranteeing that there is no counterexample to the hypothesis, or
- ▷ guarantees that there is no counterexample of order $k + i$ and proceeds with phase $i + 1$.

In essence, the algorithm checks graphs $G \in \mathcal{G}^k$ by simulating the traversal of a smooth path-decomposition (see [4] or Section 2 of this paper). It manages a queue that, at the beginning of phase i, contains all pairs of the form (U, H) which satisfy that

- ▷ H is a subgraph of some graph $G \in \mathcal{G}^k$ which has a smooth path-decomposition with U as its ith bag. In particular, $|V(H)| = k + i$,
- ▷ H contains all information provided by the bags preceding U in the path-decomposition, and
- ▷ H is a potential subgraph of a counterexample to $\mathcal{U}$ being unavoidable for $\mathcal{G}^k$.

The algorithm checks for the current pair (U, H) whether adding edges to H results in a graph in $\mathcal{G}^k$ that avoids all graphs in $\mathcal{U}$. If this is the case, then the obtained graph is returned as a certificate that $\mathcal{U}$ is not unavoidable for $\mathcal{G}^k$. Otherwise, (U, H) is replaced by multiple pairs (U', H'), where H' is a supergraph of H of order $|V(H)| + 1$. If no pairs remain in the queue, without a counterexample being found, then the algorithm guarantees that $\mathcal{U}$ is unavoidable for $\mathcal{G}^k$.

To drastically limit the amount of additional pairs created, we heavily rely on isomorphism rejection to prune the resulting search tree. To avoid a combinatorial explosion, we introduce the new concept of *high-degree-first* path-decompositions. These turn out to be invaluable when tailoring the algorithm to cubic graphs since they allow us to impose strong restrictions on the next bag in the path-decomposition (U' in the above notation). The results of this optimisation can be seen in Table 1, where the last two rows contain the amount of pairs regarded by the base algorithm and the one employing both isomorphism rejection and high-degree-first path-decompositions.

As already mentioned, checking a set for being unavoidable is undecidable, see Lemma 3.5. However, we give a criterion ensuring termination.

TABLE 1 Computational results for cubic graphs of path-width k and unavoidable structures $\mathcal{U}$.

k	3	3	4	5	5	6	6	7
$\mathcal{U}$	$\mathcal{U}_3$	$\mathcal{U}_4$	$\mathcal{U}_4$	$\mathcal{U}_4$	$\mathcal{U}_5$	$\mathcal{U}_5$	$\mathcal{U}_6$	$\mathcal{U}_6$
Result	$K_{3,3}$	Unavoidable	Unavoidable	Petersen	Unavoidable	Heawood	Unavoidable	Unavoidable
Pairs base	6	5	81	12,484	3841	–	–	–
Pairs cubic	3	2	3	7	5	15	9	19

Note: The row "Pairs base" contains the amount of pairs regarded by the base version of the algorithm while the row "Pairs cubic" shows how many are left after the use of isomorphism rejection and tailoring the algorithm to cubic graphs. The three dashed cells contain no values as the algorithm did not terminate in a reasonable time on our machines.

Theorem 1.1. *Let $\mathcal{G}$ be a locally checkable graph class of bounded maximum degree, $\mathcal{U}$ a finite set of connected graphs, and $k \in \mathbb{N}$. There is an algorithm which decides whether $\mathcal{U}$ is unavoidable for $\mathcal{G}^k$ or not. In the latter case, the algorithm returns a counterexample of the smallest order.*

The class $\mathcal{G}$ is *locally checkable* if membership can be tested using a global vertex labelling (with a constant amount of bits at each vertex) and a local verifier which sees a ball of a fixed radius around a vertex. A graph is in $\mathcal{G}$ if and only if the verifier accepts at every vertex. This allows us to locally check global properties. For example, for the class of bipartite graphs we assign each vertex a bit whose value determines its side of the partition. The verifier checks whether the neighbours of a vertex are assigned the other value. A precise definition is given in Section 2.

Note that Courcelle's theorem [12] implies Theorem 1.1. However, it does not provide an explicit algorithm since encoding a bound on the path-width in monadic second-order logic uses access to the forbidden minors, which exist by [22], but are unknown for values larger than 2. Moreover, our algorithm has more desirable runtime properties than the one which exists according to [12] (the decision procedure of Courcelle's theorem has nonelementary complexity [17]).

1.2 | Consequences in structural graph theory

Inductive proofs in structural graph theory are our main motivation for the research on unavoidable subsets and, indeed, our new framework allows us to prove various structural results. The girth of a graph is the minimum length amongst its cycles. Consider the following example: let $\mathcal{G}$ be the class of cubic graphs and write $\mathcal{U}_i$ for the set $\{C_3, \ldots, C_i\}$, where C_ℓ denotes a cycle on ℓ vertices. To investigate the maximal girth values of cubic graphs in relation to their path-width, we define $\xi(k)$ as the maximal girth of a cubic graph of path-width k.

For the case $k = 3$ we first check whether all cubic graphs of path-width 3 have a triangle by running the algorithm on $\mathcal{U}_3$ and $k = 3$. It returns the $K_{3,3}$ which has girth 4. We extend the set $\mathcal{U}_3$ to $\mathcal{U}_4$. The algorithm returns `unavoidable`, and $\xi(3) = 4$. With our implementation of this method,[1] we find the value of $\xi(k)$ for $k = 3, \ldots, 7$, as shown in Table 1.

Combining our algorithmic formalisms with combinatorial techniques, we prove a new bound on ξ which is stronger than the bound in [9] (see also the further related work)

[1]GitLab repository containing an implementation of our algorithm for checking whether a set $\mathcal{U}$ is unavoidable for $\mathcal{G}^k$, https://gitlab.rlp.net/obachtle/testing-unavoidable-sets-for-small-path-width, October 2022.

whenever the path-width is less than 26, that is, in all practically relevant cases when it comes to computing the path-width.

Theorem 1.2. *For all $k \in \mathbb{N}_{\geq 3}$ the following inequality is satisfied:*

$$\xi(k) \leq \frac{2}{3}k + \frac{10}{3}.$$

Moreover we prove the upper bounds on the girth in Theorem 1.3, which are an improvement on the bound in Theorem 1.2 for $k \leq 13$.

Theorem 1.3. *The values of ξ for small values of k are shown in the table below:*

k	3	4	5	6	7	8	9	10
$\xi(k)$	4	4	5	6	6	7	8	8

Additionally, $\xi(k) \leq k - 2$ holds for all $k \geq 10$.

The precise values for $k \leq 10$ are obtained by proving upper bounds which are attained by the graphs in Table 2. In fact, we can give a complete list of the minimal graphs of path-width k and girth $\xi(k)$ for all $k \in \{3, \ldots, 10\}$.

Theorem 1.4. *For $k \in \{3, \ldots, 10\} \setminus \{4\}$ there is a unique smallest graph of path-width k and girth $\xi(k)$. There are two smallest graphs of path-width 4 and girth 4. See Table 2.*

A (d, g)-*cage* is a minimal d-regular graph of girth g. The study of cages dates back to [24]. A recent survey on this topic is [15]. If $d = 3$, then there is a unique $(3, g)$-cage for all $g \in \{3, \ldots, 8\}$. Clearly, if a cubic cage of girth $\xi(k)$ has path-width k, then it appears in the table. Several of the graphs in Table 2 are cages: the $K_{3,3}$, the Petersen graph, the Heawood graph, and the Tutte–Coxeter graph.

As a further application of our algorithm we obtain that the graphs in $\{K_{3,3}, G_1, \ldots, G_6\}$ (see Figure 2) are unavoidable for the class of cubic graphs of path-width 3 and girth 4. We exploit this to prove the following classification.

Theorem 1.5.

(i) *Every 3-connected cubic graph of path-width 3 and girth 4 can be obtained from a $K_{3,3}$ by a finite number of the construction steps $\varkappa_1$ and $\varkappa_2$ (see Figure 2).*
(ii) *Every cubic graph of path-width 3 and girth 4 can be constructed from a $K_{3,3}$ by applying a finite number of the construction steps $\varkappa_1, \ldots, \varkappa_6$ (see Figure 2).*

TABLE 2 All smallest graphs of path-width k and girth $\xi(k)$.

3	4	5	6	7	8	9	10
$K_{3,3}$	Cube Twisted cube	Petersen	Heawood	Pappus	McGee	Tutte– Coxeter	$G(10)$

Note: The top row specifies the value of k while the bottom one contains the graphs. Here G_{10} denotes the unique cubic graph of path-width 10 and girth 8. We refer to Figure 1 for drawings of these graphs.

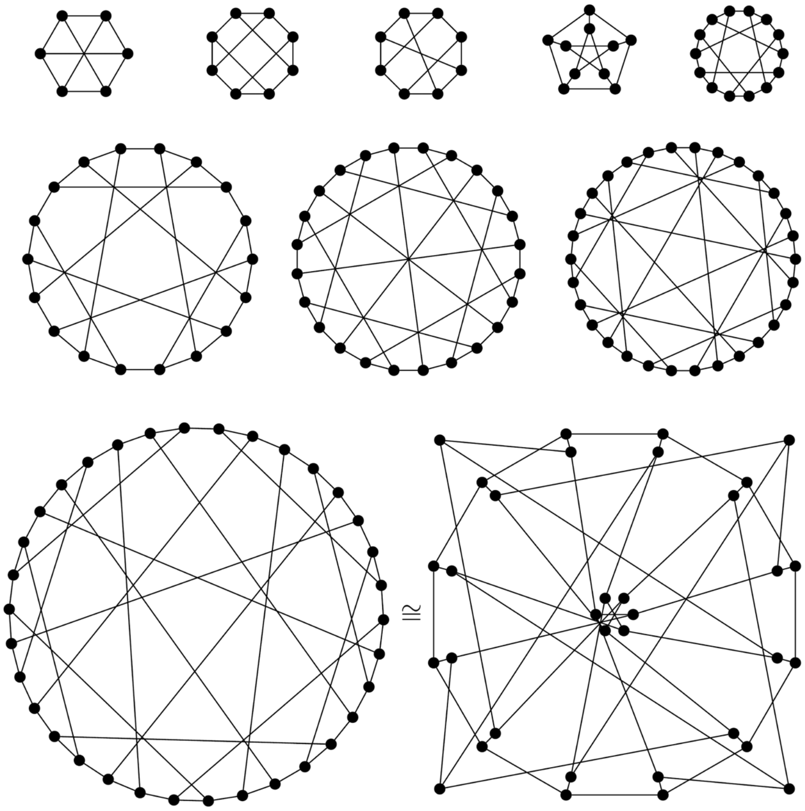

FIGURE 1 All graphs of path-width k and girth $\xi(k)$ for $k \in \{3, 4, 5, 6, 7, 8, 9, 10\}$. First row: $K_{3,3}$, cube, twisted cube, Petersen graph, Heawood graph. Middle row: Pappus graph, McGee graph, Tutte–Coxeter graph. Last row: Two drawings of the unique graph of girth 8 and path-width 10, which we denote by $G(10)$.

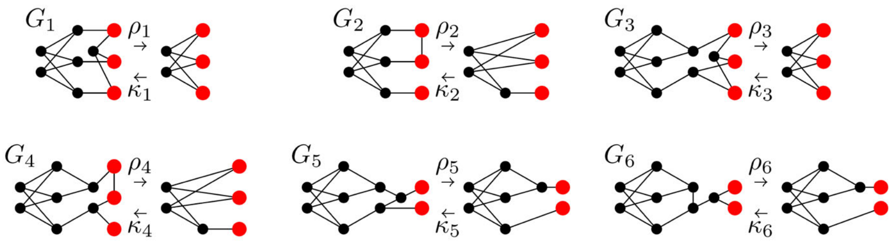
G_1
$\overset{\rho_1}{\rightarrow}$
$\underset{\kappa_1}{\leftarrow}$
G_2
$\overset{\rho_2}{\rightarrow}$
$\underset{\kappa_2}{\leftarrow}$
G_3
$\overset{\rho_3}{\rightarrow}$
$\underset{\kappa_3}{\leftarrow}$
G_4
$\overset{\rho_4}{\rightarrow}$
$\underset{\kappa_4}{\leftarrow}$
G_5
$\overset{\rho_5}{\rightarrow}$
$\underset{\kappa_5}{\leftarrow}$
G_6
$\overset{\rho_6}{\rightarrow}$
$\underset{\kappa_6}{\leftarrow}$

FIGURE 2 Reductions for graphs of path-width 3 and girth 4: if a graph of path-width 3 and girth 4 contains one of the graphs G_i (consisting of the black and red vertices and black edges) as a subgraph, then the reduction ρ_i replaces the all black vertices and edges on the left side by the ones on the right side. The inverse operation is the construction step κ_i.

1.3 | Further related work

A recent introduction to locally checkable proofs is [16]. We refer to [4] for an introduction to path- and tree-width. Bodlaender and Koster [7] survey lower bounds for tree-width, amongst them is the following result of Chandran and Subramanian [9]. If G is a cubic graph of girth at least g, path-width k, and tree-width t, then $k \geq t \geq (e(g+1))^{-1}2^{\lfloor (g-1)/2 \rfloor - 2} - 2$, where e denotes Euler's number.[2] Theorems 1.2 and 1.3 are stronger than this exponential bound whenever the path-width is less than 26. Our bounds provide a proper improvement in the context of path-width computations.

1.4 | Outline

Section 2 contains basic notation and preliminary results. In particular, we introduce the new concept of high-degree-first path-decompositions. In Section 3, we give a detailed discussion of the algorithm that checks whether a set is unavoidable for a path-width bounded class. Section 4 contains the proofs of Theorems 1.2 to 1.5.

2 | PRELIMINARIES

2.1 | Basic notation

The notation for this paper is based on [14], but we briefly summarise what we need here. All graphs are nonempty, finite, and simple. We write uv for an edge with ends u and v and $E(v)$ denotes the set of edges incident to the vertex v. The set of neighbours of v in G is $N_G(v)$ or $N(v)$. A *path* is a graph P with $V(P) = \{v_1, \ldots, v_n\}$ and $E(P) = \{v_1v_2, \ldots, v_{n-1}v_n\}$, for which we write $P = v_1 \ldots v_n$. The *order* $|V(P)|$ of a path P is denoted by $|P|$ and its *length* is $\|P\| = |E(P)| = |P| - 1$. If u and v are distinct vertices of a tree T, then uTv is the unique path in T joining u and v. If $|uTv| \geq 3$, then we set $\mathring{u}T\mathring{v} := uTv - \{u, v\}$. We write $G + H$ for the disjoint union of two graphs G and H. The empty graph on n vertices is denoted by E_n and K_{r_1,r_2} is the complete bipartite graph with parts of size r_1, r_2. The *girth* of G is the minimum length of a cycle in G. We set

$$\begin{aligned} \xi : \mathbb{N}_{\geq 3} &\to \mathbb{N}, \\ k &\mapsto \max\{g : \text{there is a simple cubic graph of girth } g \text{ and path-width } k\}. \end{aligned}$$

Let G and H be two graphs and φ be a bijection with domain $V(G)$. For $U \subseteq V(G)$ and $F \subseteq E(G)$ we define $\varphi(U) := \{\varphi(u) : u \in U\}$ and $\varphi(F) := \{\varphi(u)\varphi(v) : uv \in F\}$. We write $\varphi(G)$ for the graph with vertex set $\varphi(V(G))$ and edge set $\varphi(E(G))$. The map φ is an *isomorphism* from G to H if $H = \varphi(G)$. The graph G is *isomorphic* to H, denoted $G \cong H$, if an isomorphism from G to H exists. If $\varphi(G) = G$, then φ is an *automorphism* of G. The automorphisms of a graph G form a group and any subgroup Γ of this group acts naturally on $V(G)$. Let $v \in V(G)$. The *orbit* of v is the set $v^\Gamma := \{\varphi(v) : \varphi \in \Gamma\}$. The *stabiliser* of v is $\Gamma_v := \{\varphi \in \Gamma : \varphi(v) = v\}$.

2.2 | Path-decompositions and path-width

Let G be a graph, $P = 1 \ldots n'$ a path, and $\mathcal{V} = \{V_i : i \in P\} \subseteq 2^{V(G)}$. The pair $(P, \mathcal{V})$ is a *path-decomposition of* G if:

[2]The bound in [9] is more general since graphs of average degree d are considered. We inserted $d = 3$ and added the inequality $k \geq t$ to emphasise the connection to Theorems 1.3 and 1.2.

(i) every vertex $v \in V(G)$ is contained in some set V_i,
(ii) for each edge $uv \in E(G)$ there exists a set V_i such that $\{u, v\} \subseteq V_i$, and
(iii) the set $\{i : v \in V_i\}$ induces a subpath of P for all $v \in V(G)$.

The sets V_i are *bags* of the decomposition and the *width of* $(P, \mathcal{V})$ is $\max\{|V_i| : i \in P\} - 1$. Furthermore, the *path-width of* G is the minimal width of any of its path-decompositions. A path-decomposition $(P, \mathcal{V})$ of width k is *smooth* if all bags have cardinality $k + 1$ and $|V_i \cap V_{i+1}| = k$ for all $1 \leq i < n'$. Any graph of path-width k allows for a smooth width k path-decomposition [4]. Barring few exceptions, all path-decompositions occurring in the remainder of this paper are smooth and adhere to the naming convention in the definition above, that is, their path is $P = 1 \ldots n'$ and their bags are $\mathcal{V} = \{V_i : i \in P\}$.

For $i \in \{1, \ldots, n' - 1\}$, we say that a vertex v *enters* bag V_{i+1} if it is the unique vertex in $V_{i+1} \setminus V_i$. Analogously, v *leaves* V_i if it is the unique vertex in $V_i \setminus V_{i+1}$. Moreover, we define the *graph associated with* V_i as

$$G_i := G\left[\bigcup_{1 \leq j \leq i} V_j\right] - E(G[V_i]).$$

Intuitively, G_i contains all information provided by the bags preceding V_i as it has (exactly) the edges incident to vertices that have left already. Note that if v leaves V_i, then $G_{i+1} = G_i + E(v) + v_{i+1}$ for some new vertex v_{i+1}. We say u is a *new neighbour* of v if v is the vertex leaving V_i and $u \in N_{G_{i+1}}(v) \setminus N_{G_i}(v)$. Thus, the new neighbours of v are exactly those vertices u in G_{i+1} for which uv is an edge of G that was not already present in G_i.

In this paper, $\mathcal{G}$ is a graph class and $\mathcal{U}$ is a finite set of graphs. We set

$$\mathcal{G}^k := \{G \in \mathcal{G} : \mathrm{pw}(G) \leq k\} \text{ and}$$
$$\mathrm{super}(\mathcal{U}) := \{G : S \cong S' \subseteq G \text{ for some } S \in \mathcal{U}\}.$$

With this, the question whether $\mathcal{U}$ is unavoidable for $\mathcal{G}^k$ translates to $\mathcal{G}^k \subseteq \mathrm{super}(\mathcal{U})$?

2.3 | Locally checkable graphs

The following definition is closely related to the class LCP($\Theta(1)$) in [19]: Let G be a graph. For $v \in V(G)$ and $r \geq 0$ let $B_G^r(v) := B^r(v) := \{u \in V(G) : d(u, v) \leq r\}$.

(i) A *proof* for G is a map $\mathbb{P} : V(G) \to \{0, 1\}^*$ that assigns each vertex a binary string. Its *size* is the maximum amount of bits in any $\mathbb{P}(v)$.
(ii) A *verifier* is a map

$$\mathcal{A} : \{(G, \mathbb{P}, v) : G \text{ is a graph, } \mathbb{P} \text{ is a proof for } G, v \in V(G)\} \to \{0, 1\}.$$

It is *local* if there exists a constant $r \geq 0$ such that

$$\mathcal{A}(G, \mathbb{P}, v) = \mathcal{A}(G[B^r(v)], \mathbb{P}[B^r(v)], v) \text{ for all } G, \mathbb{P}, v.$$

Here $\mathbb{P}[B]$ denotes the restriction of $\mathbb{P}$ to B.

(iii) A graph class $\mathcal{G}$ is *locally checkable* if there is a local verifier $\mathcal{A}$ and some $s \in \mathbb{N}$ such that
- ▷ for all $G \in \mathcal{G}$ there is a proof $\mathbb{P}$ of size at most s such that $\mathcal{A}(G, \mathbb{P}, v) = 1$ for all $v \in V(G)$,
- ▷ for all $G \notin \mathcal{G}$ and all proofs $\mathbb{P}$ for G there exists a $v \in V(G)$ such that $\mathcal{A}(G, \mathbb{P}, v) = 0$, and
- ▷ the verifier is invariant under isomorphism, that is, for a bijection φ we have that $\mathcal{A}(G, \mathbb{P}, v) = 1$ implies that $\mathcal{A}(\varphi(G), \varphi(\mathbb{P}), \varphi(v)) = 1$, where $\varphi(\mathbb{P})$ is the map $\varphi(\mathbb{P}) : \varphi(G) \to \{0, 1\}^*$ with $\varphi(\mathbb{P})(\varphi(v)) = \mathbb{P}(v)$.

We say that $\mathcal{A}$ *accepts* proof $\mathbb{P}$ for G if $\mathcal{A}(G, \mathbb{P}, v) = 1$ for all $v \in V(G)$.

Examples for locally checkable graph classes are l-regular graphs (using $r = 1$, $s = 0$), bipartite and, more generally, l-colourable graphs ($r = 1$, $s = \lceil \log l \rceil$), and graphs without an (induced) subgraph H $\left(r = \left\lceil \log \frac{\operatorname{diam}(H)}{2} \right\rceil, s = 0\right)$.

2.4 | High-degree-first path-decompositions

In the following, we prove that we can make strong assumptions on the path-decompositions of cubic graphs without loss of generality. These serve as a powerful tool for both tailoring our algorithm to cubic graphs and proving results on the girth of cubic graphs by hand.

Observation 2.1. Let $(P, \mathcal{V})$ be a smooth path-decomposition of G and let $i \in \{1, \ldots, n' - 1\}$. If $N_G(v) \subseteq V(G_i)$ for some $v \in V_i$, then there exists a smooth path-decomposition $(P, \mathcal{W})$ of G such that $V_j = W_j$ for $j \leq i$, and v leaves W_i.

Proof. Let $v \in V_i$ be such a vertex and assume that some other vertex $u \neq v$ leaves V_i. Replacing v by u in all bags of index greater than i yields another path-decomposition $(P, \mathcal{W})$ of G as all edges with v as an end are already covered by the bags up to V_i. It is also smooth, satisfies $V_j = W_j$ for $j \leq i$, and v leaves the bag W_i as required. □

Lemma 2.2. *Let $(P, \mathcal{V})$ be a smooth path-decomposition of G, let $i \in \{1, \ldots, n' - 1\}$. If G_i contains a vertex v with $|N_G(v) \setminus N_{G_i}(v)| \leq 1$, then there is a smooth path-decomposition $(P, \mathcal{W})$ of G with $V_j = W_j$ for $j < i$, $V_{i-1} \setminus V_i = W_{i-1} \setminus W_i$ and v leaves W_i.*

Proof. Let v be as in the statement and assume that $u \neq v$ is the vertex leaving V_i. If $N_G(v) \subseteq V(G_i)$, then we apply Observation 2.1 to obtain the claim.

Hence, we assume there exists a vertex $w \in N_G(v) \setminus V(G_i)$ and let V_j be the bag of the lowest index containing w. By assumption, we have $j > i$ and Observation 2.1 lets us assume that v leaves V_j or V_j is the last bag. We describe how to obtain the desired path-decomposition $(P, \mathcal{W})$ in case the bag V_{j+1} exists, making note of what would change if it does not in parentheses. The process is illustrated in Figure 3.

First, we delete the bag V_j, connecting V_{j-1} to V_{j+1} (if it exists). Note that if a bag uniquely covers an edge of G, then the vertex entering and the one leaving it are an end of this edge, otherwise the bag before or after would have done this as well. The result is a path-decomposition of $G - vw$ ($G - w$). This step is marked in red in the figure.

Next, we replace all occurrences of v in the bags $V_i, \ldots, V_{j-1}$ by w. As all these bags only contain v, they continue to have $k + 1$ elements. Also the now neighbouring bags V_{j-1}

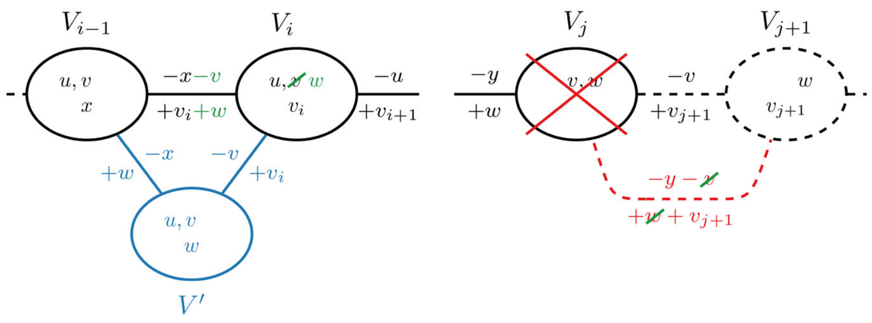

FIGURE 3 The path-decomposition constructed in the proof of Lemma 2.2.

and V_{j+1} have k vertices in common. Since we only removed v from bags and all its neighbours but w already shared a bag before V_i, this is a path-decomposition of $G - vw$ (now also in the case that V_j was the last bag, where V_{j-1} contains all neighbours of w).

Finally, we insert the bag $V' = V_{i-1} \cup \{w\} \backslash \{x\}$, where x is the vertex in $V_{i-1} \backslash V_i$, between V_{i-1} and V_i to make the decomposition smooth and turn it into one of G as this bag contains both v and w. This completes the construction and the proof. □

By iteratively applying this lemma to a smooth path-decomposition of a cubic graph G, we may assume that the vertex leaving bag V_i has degree at least 2 in G_i, whenever such a vertex exists. If, additionally, a degree 3 vertex is chosen to leave whenever present, we obtain a decomposition as defined below.

Definition 2.3. Let $(P, \mathcal{V})$ be a path-decomposition of a cubic graph G and, for $i \in \{1, ..., n'\}$, let

$$d_i := \max\{d_{G_i}(v): v \in V_i\}.$$

Then $(P, \mathcal{V})$ is called *high-degree-first (hdf)* or an *hdf-decomposition* if the vertex v_i leaving the ith bag satisfies

$$d_{G_i}(v_i) = 3 \text{ if } d_i = 3 \text{ and } d_{G_i}(v_i) = 2 \text{ if } d_i = 2.$$

3 | DECIDING UNAVOIDABILITY FOR GRAPHS OF BOUNDED PATH-WIDTH ALGORITHMICALLY

We start this section with a toy example that illustrates the by-hand method we automate and is illustrated in Figure 4. Let $\mathcal{G}$ be the class of all cubic graphs. We prove that the set $\mathcal{U} = \{C_3, C_4\}$ is unavoidable for the class $\mathcal{G}^3$ of cubic graphs of path-width at most 3. To this end, let $(P, \mathcal{V})$ be a smooth width-3 path-decomposition of a graph $G \in \mathcal{G}^3$. Let v and v' be the vertices leaving the bags V_1 and V_2, respectively. Since v leaves V_1 (and has degree 3), we obtain $N_G(v) = V_1 \backslash \{v\}$. If v' is a neighbour of v, then it has two other neighbours in $V_2 \backslash \{v'\}$. Consequently, v and v' have a common neighbour and G contains a C_3. Otherwise, v' is the vertex entering V_2. In this case, $N_G(v') = V_2 \backslash \{v'\} = V_1 \backslash \{v\}$ and we obtain a C_4. Altogether, it follows that $\xi(3) = 4$.

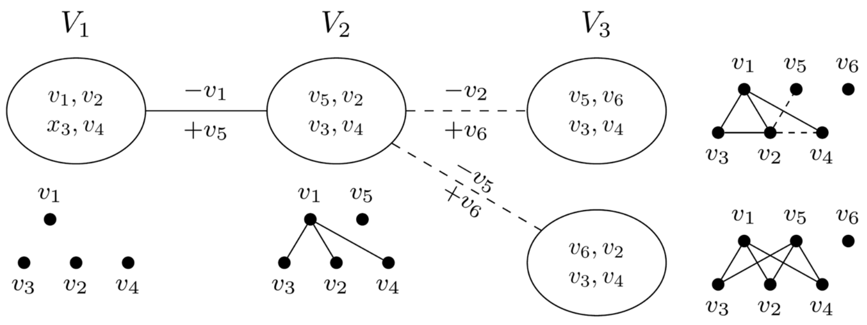

FIGURE 4 An illustration of the by-hand method for the example of cubic graphs of path-width at most 3. The vertex $v = v_1$ leaves the first bag, so it is adjacent to v_2, v_3, v_4. For the vertex v' leaving the second bag, we can, by symmetry, assume that $v' = v_2$ or $v' = v_5$. The first case yields a C_3, the second a C_4.

Outline of this chapter: We formally introduce the basic version of our algorithm in Section 3.1. To make the algorithm practically applicable, we describe how isomorphism rejection can be exploited to speed up the run time in Section 3.2. We show that the general problem of deciding whether a finite set of graphs is unavoidable for a graph class is undecidable and we prove our termination result (Theorem 1.1) in Section 3.3. In Section 3.4 we employ our concept of high-degree-first decompositions to obtain a further speed-up for our algorithm if it is applied to a subclass of cubic graphs. Finally, in Section 3.5 we discuss how our algorithm can be adapted to check if a set of graphs is an unavoidable set of minors (respectively, induced subgraphs).

3.1 | The base algorithm

We describe an algorithm that answers the question whether $\mathcal{G}^k \subseteq \text{super}(\mathcal{U})$, that is, whether $\mathcal{U}$ is unavoidable for $\mathcal{G}^k$, by implementing the proof technique illustrated in the toy example above. The only requirement on $\mathcal{G}$ is that it is given together with a membership test. By checking all graphs of order at most k explicitly, we may assume that $\mathcal{G}$ only contains graphs with at least $k + 1$ vertices. This ensures that all graphs in $\mathcal{G}$ of path-width at most k have a smooth path-decomposition of width k.

Recall the naming conventions for path-decompositions introduced in Section 2. For each smooth path-decomposition $(P, \mathcal{V})$ of a graph G and each $i \in \{1, ..., n'\}$ we say that G *contains* (V_i, G_i). A pair (U, H) is *good* if every $G \in \mathcal{G}^k$ containing it satisfies $G \in \text{super}(\mathcal{U})$. We remark that any pair (U, H) with $H \in \text{super}(\mathcal{U})$ is good. This holds as any graph G containing (U, H) has H as a subgraph. As a preparation for the formal description and the correctness proof of our algorithm, we show the following lemma.

Lemma 3.1. *Let G be a graph with path-decomposition $(P, \mathcal{V})$ containing the pair (U, H). Let φ be a bijection with domain $V(G)$.*

(i) *Renaming the vertices of G according to φ, both in G and in its path-decomposition, yields an isomorphic graph $\varphi(G) \cong G$ with path-decomposition $(P, \{\varphi(V_i) : i \in P\})$.*
(ii) *The graph $\varphi(G)$ contains the pair $(\varphi(U), \varphi(H))$.*
(iii) *If $(V_1, (V_1, \varnothing))$ is a good pair, then $\mathcal{G}^k \subseteq \text{super}(\mathcal{U})$.*

Proof. Statements (i) and (ii) are immediate from the respective definitions. For (iii), consider a graph $G \in \mathcal{G}^k$ with a smooth path-decomposition of width k. In particular, the graph G_1 associated with V_1 is $(V_1, \emptyset)$. The graph G contains $(V_1, (V_1, \emptyset))$ and, hence, $G \in \text{super}(\mathcal{U})$. □

We are now ready to describe the algorithm, whose pseudocode can be found in Algorithm 3.1. Given a graph class $\mathcal{G}$ with a membership test, $k \in \mathbb{N}$, and a finite set of graphs $\mathcal{U}$, the algorithm manages a queue Q of pairs (U, H). By Lemma 3.1(i) we may assume that $V(G) = V := \{u_1, ..., u_k, v_1, ...v_{n'}\}$ and the first bag of the path-decomposition is $V_1 = \{u_1, ..., u_k, v_1\}$. Therefore, the queue is initialised with $(V_1, (V_1, \emptyset))$. We maintain the following invariant:

$$\text{If all pairs in the queue are good, then } \mathcal{G}^k \subseteq \text{super}(\mathcal{U}). \tag{1}$$

This holds initially by Lemma 3.1(iii) and, once the queue is empty, we may return that $\mathcal{U}$ is `unavoidable`. The algorithm iteratively removes a pair (U, H) from the queue. To maintain (1), the algorithm needs to ensure that any graph $G \in \mathcal{G}$ containing (U, H) is in super$(\mathcal{U})$. Such a graph has a smooth path-decomposition with U as a bag with associated graph H. If this is the last bag in the path-decomposition, then G is obtained from H by adding edges between vertices of U and the algorithm checks all such augmentations to see if one of them avoids $\mathcal{U}$. Should this occur, it returns it as a counterexample. Otherwise, the decomposition does not end with the bag U and the algorithm checks all options for the next bag and its corresponding graph, adding these new pairs to the queue. If any option for a subsequent pair is good, then so is the original.

Algorithm 3.1: Base algorithm for checking whether $\mathcal{G}^k \subseteq \text{super}(\mathcal{U})$.

```
Input: A class of graphs $\mathcal{G}$ with a membership test, a finite set of graphs
       $\mathcal{U}$, and a path-width value $k$.
Output: An element of $\mathcal{G}^k \setminus \text{super}(\mathcal{U})$ or unavoidable if no such graph
        exists.
Procedure: Test-Unavoidable-Structures($\mathcal{G}, \mathcal{U}, k$)
1  $V_1 := \{u_1, \ldots, u_k, v_1\}$;
2  $Q := [(V_1, (V_1, \emptyset))]$;
3  if $(V_1, \emptyset)$ is good then $Q$.pop();
4  while $Q$ is not empty do
5      $(U, H) := Q$.pop();
6      foreach $E' \subseteq \{xy : x, y \in U, x \neq y\}$ do
7          if $H + E' \in \mathcal{G} \setminus \text{super}(\mathcal{U})$ then
8              return $H + E'$;
9      Let $i = |H| - k$;
10     foreach $u \in U$ do
11         $U' := U \setminus \{u\} \cup \{v_{i+1}\}$;
12         foreach $Y \subseteq U \setminus \{u\}$ do
13             $H' := H + v_{i+1} + \{uy : y \in Y\}$;
14             if $H' \notin \text{super}(\mathcal{U})$ and there exists a $G \in \mathcal{G}$ containing $(U', H')$
               then
15                 $Q$.append($(U', H')$);
16 return unavoidable;
```

Theorem 3.2. *The answer returned by Algorithm* 3.1 *is correct.*

Proof. We first prove that $V(H) = \{u_1, ..., u_k, v_1, ..., v_{|H|-k}\}$ for every pair (U, H) in the queue. Furthermore, we define the *path-decomposition of* (U, H) for such pairs, which is a smooth path-decomposition $(P, \mathcal{V})$ of H of width k with last bag U. For $(V_1, (V_1, \emptyset))$ the claim on the vertex set holds and we use a path of length 0 with bag V_1 as our decomposition. Now let (U', H') be a pair added in the iteration in which (U, H) was removed. As $V(H') = V(H) \cup \{v_{i+1}\}$ for $i = |H| - k$ and $v_{i+1} \notin V(H)$, we get $|H'| = |H| + 1$ and $V(H')$ satisfies the claim. To obtain the decomposition of (U', H'), we extend the one of (U, H) by adding an additional vertex to the end of the path-width bag U'. This decomposition has width k and is smooth as $|U'| = k + 1$ and U, U' differ in exactly one vertex.

Next, we prove that the results returned are correct. If the algorithm returns a graph $H + E'$ in Line 8, then $H + E' \in \mathcal{G} \setminus \text{super}(\mathcal{U})$. It has path-width at most k, since the path-decomposition of (U, H) has width k with last bag U, which is also a path-decomposition for $H + E'$. Therefore, it suffices to check that $\mathcal{G}^k \subseteq \text{super}(\mathcal{U})$ in case the algorithm returns `unavoidable`.

We verify this by inductively proving the following invariant: before any iteration of the while loop (1) holds. We have already argued that (1) holds for the initialisation of Q, by renaming the vertices and using Lemma 3.1(iii). If, before the first iteration, the single pair in Q is removed, then this pair is good and (1) holds. So it is true before the first iteration of the while loop.

Now assume that the invariant holds up to iteration l and regard the queue before iteration $l + 1$. In iteration l only a single element (U, H) was removed from Q. Consequently, it suffices to prove that (U, H) is good if all newly added pairs are. To verify this, let $G \in \mathcal{G}^k$ be a graph containing (U, H) with corresponding decomposition $(P, \mathcal{V})$. First assume that U is the last bag of this decomposition, that is, $G = H + E'$ for some $E' \subseteq \{xy : x, y \in U, x \neq y\}$. The algorithm regards this graph in some iteration of the for loop in Line 6. Since it does not terminate in this iteration and $G \in \mathcal{G}$, we have $G \in \text{super}(\mathcal{U})$.

Now assume that the path-decomposition of G does not end with the bag U. Let U' be the subsequent bag. We know that, for $i = |H| - k$, $V(H) = \{u_1, ..., u_k, v_1, ..., v_i\}$. Thus v_{i+1} is not in $V(H)$ and we can assume that the element that enters U' is v_{i+1} by Lemma 3.1(ii). (Simply choose a mapping φ that is the identity restricted to H and maps the vertex that enters U' to v_{i+1}.) Denote the vertex leaving U by u, then the graph H' associated with the bag U' has the form $H + E' + v_{i+1}$ where $E' \subseteq \{uy : y \in U \setminus \{u\}\}$ is the set of edges between u and its new neighbours. The algorithm regards the set $U' = U \setminus \{u\} \cup \{v_{i+1}\}$ in the for loop in Line 10 and also looks at the graph H' in Line 12. Since $G \in \mathcal{G}^k$, it is an element of $\mathcal{G}$ that contains (U', H'). If $H' \in \text{super}(\mathcal{U})$, then $G \in \text{super}(\mathcal{U})$ and if it is not, then (U', H') is added to the queue. In this case, we have assumed it is a good pair and G contains it, meaning that G is in $\text{super}(\mathcal{U})$. □

The algorithm remains correct if we remove the check whether there exists a graph $G \in \mathcal{G}$ containing (U', H') in Line 14. It is used to reduce the amount of pairs added to the queue. Should this condition be hard to check, it can be omitted. Heuristics may be used instead as long as they are correct in case they return a negative answer. For example, if $\mathcal{G}$ is the class of cubic graphs, we can check whether H' is subcubic.

3.2 | Isomorphism rejection

Now that we have proven the base algorithm to be correct, we demonstrate how to drastically improve the running time by exploiting isomorphism rejection, which reduces the amount of pairs added in each iteration (see also Table 1).

Lemma 3.3. *For a bijection φ with domain $V(H)$, the pair (U, H) is good if and only if the pair $(\varphi(U), \varphi(H))$ is good.*

Proof. Let $(\varphi(U), \varphi(H))$ be a good pair and G be a graph containing (U, H). Let $\overline{\varphi}$ be the extension of φ to $V(G)$, where $\overline{\varphi}(v) = v$ for all $v \in V(G)\setminus V(H)$. By Lemma 3.1(ii) the graph $\overline{\varphi}(G)$ contains the pair $(\varphi(U), \varphi(H))$ which is good. Hence $\overline{\varphi}(G) \in$ super$(\mathcal{U})$ and $\overline{\varphi}(G) \cong G$, which implies $G \in$ super$(\mathcal{U})$. This shows that (U, H) is good. The missing direction follows by regarding φ^{-1}. □

As a consequence of Lemma 3.3 we can improve our base algorithm: we only need to add pairs (U', H') to Q for which the queue does not already have an element of the form $(\varphi(U'), \varphi(H'))$ for some $\varphi \in \mathrm{Aut}(U, H)$.

What we describe now are special cases of Lemma 3.3 that can be checked before Line 14. Assume we are in the iteration in which the pair (U, H) is removed from the queue. We denote the set of automorphisms of H that stabilise U as a set by $\mathrm{Aut}(U, H)$. For convenience, the automorphisms φ in $\mathrm{Aut}(U, H)$ are called (U, H)-*maps* and we note that $\mathrm{Aut}(U, H)$ is a subgroup of the automorphism group of G. We use these maps to eliminate certain pairs without needing to regard them. To facilitate this, the set $\mathrm{Aut}(U, H)$ is computed directly after the pair (U, H) is removed.

Our goal is to optimise all three for loops in Lines 6, 10, and 12 by reducing the amount of potential counterexamples, candidates for subsequent bags, and associated graphs regarded. See Algorithm 3.2 for the pseudocode of the algorithm with these additions.

First, we improve the for loop in Line 6 in which the algorithm looks for counterexamples. If we have checked an edge set $E' \subseteq \{xy\colon x, y \in U, x \neq y\}$, then we do not need to check the sets $\varphi(E')$ for $\varphi \in \mathrm{Aut}(U, H)$. This holds as $H + E' \cong \varphi(H + E') = H + \varphi(E')$.

Second, we reduce the amount of candidates for the next bag in Line 10. By using (U, H)-maps, we can reduce the amount of pairs that are regarded. Let $\varphi \in \mathrm{Aut}(U, H)$ with $\varphi(v) = u$ for some u and v in U and let $\overline{\varphi}$ be the extension of φ to $V(H) \cup \{v_{i+1}\}$ where $\overline{\varphi}(v_{i+1}) = v_{i+1}$. We set $U' = U\setminus\{u\} \cup \{v_{i+1}\}$ and $U'' = U\setminus\{v\} \cup \{v_{i+1}\}$. By Lemma 3.3 we know that if the pair (U'', H'') is good if and only if $(\overline{\varphi}(U''), \overline{\varphi}(H''))$ is, where $\overline{\varphi}(U'') = U'$. Hence, if for all $Y' \subseteq U\setminus\{u\}$ the pair (U', H') with $H' = H + v_{i+1} + \{uy : y \in Y'\}$ is good, then the same holds for the pairs (U'', H'') where $H'' = H + v_{i+1} + \{vy : y \in Y''\}$ with $Y'' \subseteq U\setminus\{v\}$. To see this, let H'' be of the form above, then $\overline{\varphi}(H'') = \overline{\varphi}(H) + \overline{\varphi}(v_{i+1}) + \overline{\varphi}(\{vy : y \in Y''\}) = H + v_{i+1} + \{uy : y \in \overline{\varphi}(Y'')\}$. But as $\overline{\varphi}(Y'') \subseteq \overline{\varphi}(U\setminus\{v\}) = U\setminus\{u\}$, the graph $\overline{\varphi}(H'')$ is a candidate for H' and this pair is good by assumption.

This means that we only need to regard the case where u leaves, so the pairs (U', H') above, and can disregard the pairs with U'' altogether. In other words, it suffices to regard one vertex from each orbit (where the group $\mathrm{Aut}(U, H)$ acts on $V(H)$). We may thus replace the for loop iterating over all $u \in U$ by one that iterates over the set $\overline{U} \coloneqq \{u^{\mathrm{Aut}(U,H)} : u \in U\}$. (Notice that elements of $\mathrm{Aut}(U, H)$ stabilise U as a set, so any representative of this orbit can be chosen.)

Thirdly, we reduce the amount of vertex sets that are checked in Line 12. Assume the next bag is $U' = U \backslash \{u\} \cup \{v_{i+1}\}$. We do not need to add a pair (U', H') for a set Y' if we have already added the pair (U', H'') for a set Y'' and there exists an (H, U)-map φ with $\varphi(Y') = Y''$ and $\varphi(u) = u$. To see this, observe that the pair (U', H') is good if and only if the pair $(\overline{\varphi}(U'), \overline{\varphi}(H'))$ is, where $\overline{\varphi}$ is, again, the extension of φ to v_{i+1} with $\overline{\varphi}(v_{i+1}) = v_{i+1}$. By assumption we have that $\overline{\varphi}(U') = U'$. Denote the set $\{uy' : y' \in Y'\}$ by E' and $\{uy'' : y'' \in Y''\}$ by E''. But, since $\varphi(H' - E') = \varphi(H + v_{i+1}) = H + v_{i+1} = H'' - E''$ and $\varphi(E') = E''$, we get that $\varphi(H') = H''$. Consequently, we may ignore all sets $\varphi(Y')$, for any map φ in the stabiliser $\mathrm{Aut}(U, H)_u = \{\varphi \in \mathrm{Aut}(U, H) : \varphi(u) = u\}$, upon adding the pair (U', H') in Line 12.

Remark 3.4. Since $|\mathrm{Aut}(U, H)| = |\mathrm{Aut}(U, H)_u| \cdot |u^{\mathrm{Aut}(U,H)}|$ holds by the orbit-stabiliser theorem we know that, assuming a reasonable size of $\mathrm{Aut}(U, H)$, either the orbit or the amount of graphs that can be eliminated in Line 12 is large.

Algorithm 3.2: An optimised version of Algorithm 3.1 using isomorphism rejection.

```
Input: A class of graphs $\mathcal{G}$ with a membership test, a finite set of graphs
       $\mathcal{U}$, and a path-width value $k$.
Output: An element of $\mathcal{G}^k \setminus \mathrm{super}(\mathcal{U})$ or unavoidable if no such graph
        exists.
Procedure: Test-Unavoidable-Structures($\mathcal{G}, \mathcal{U}, k$)
1  $V_1 := \{u_1, \ldots, u_k, v_1\}$;
2  $Q := [(V_1, (V_1, \emptyset))]$;
3  if $(V_1, \emptyset)$ is good then $Q$.pop();
4  while $Q$ is not empty do
5      $(U, H) := Q$.pop();
6      Determine $\mathrm{Aut}(U, H)$;
7      $F := \{xy\colon x, y \in U, x \neq y\}$ and $\mathcal{F} := 2^F$;
8      while $\mathcal{F}$ is not empty do
9          Remove a set $E'$ from $\mathcal{F}$;
10         if $H + E' \in \mathcal{G} \setminus \mathrm{super}(\mathcal{U})$ then
11             return $H + E'$;
12         else
13             $\mathcal{F} := \mathcal{F} \setminus \{\varphi(E')\colon \varphi \in \mathrm{Aut}(U, H)\}$;
14     $i = |H| - l$;
15     Let $\overline{U} \subseteq U$ contain a vertex from every orbit;
16     foreach $u \in \overline{U}$ do
17         $U' := U \setminus \{u\} \cup \{v_{i+1}\}$;
18         $\mathcal{Y} := 2^{U \setminus \{u\}}$;
19         while $\mathcal{Y}$ is not empty do
20             Remove a set $Y$ from $\mathcal{Y}$;
21             $H' := H + v_{i+1} + \{uy\colon y \in Y\}$;
22             if $H' \notin \mathrm{super}(\mathcal{U})$ and there exists a $G \in \mathcal{G}$ containing $(U', H')$
               and
23             there is no bijection $\varphi$ with $(\varphi(U'), \varphi(H')) \in Q$ then
24                 $Q$.append($(U', H')$);
25             $\mathcal{Y} := \mathcal{Y} \setminus \{\varphi(Y)\colon \varphi \in \mathrm{Aut}(U, H), \varphi(u) = u\}$;
26 return unavoidable;
```

3.3 | Achieving termination

This section is dedicated to the proof of Theorem 1.1.

3.3.1 | Undecidability of the general problem

Before we begin with the termination proof, we show that the problem of deciding whether a finite set of graphs is unavoidable for a graph class is undecidable.

Lemma 3.5. *It is undecidable to determine whether a set of graphs $\mathcal{U}$ is unavoidable for a graph class $\mathcal{G}$ even if $\mathcal{U}$ is a finite set of connected graphs and $\mathcal{G}$ is decidable and contains graphs of path-width at most* 2.

Proof. We describe a reduction from the Post Correspondence Problem which is known to be undecidable [21]. Let $x_1, ..., x_N$ and $y_1, ..., y_N$ be strings over $\{a, b\}$. Let I be the set of finite sequences in $\{1, ..., N\}$ and $f : \mathbb{N} \to I$ be an enumeration of I. A solution is an element $(i_1, ..., i_k)$ of I such that $x_{i_1} ... x_{i_k} = y_{i_1} ... y_{i_k}$.

For $n \in \mathbb{N}$ let $P_n = v_1 ... v_n$ be the path of length n and set $G_n = P_n$ if $f(n)$ is a solution. Otherwise, set $G_n = P_{n+2} + v_{n+2}v_n$. Hence, all G_n have path-width at most 2 and they contain a C_3 exactly in the latter case. We define $\mathcal{G} = \{G_n : n \in \mathbb{N}\}$, which is decidable. Consequently, $\mathcal{U} = \{C_3\}$ is unavoidable for $\mathcal{G}$ if and only if no path is in $\mathcal{G}$, which is equivalent to there being no solution. □

3.3.2 | Modified algorithm for locally checkable classes

Assume that $\mathcal{G}$ is a locally checkable class with verifier $\mathcal{A}$ and size s, all graphs $G \in \mathcal{G}$ have a maximum degree at most Δ, and $\mathcal{U}$ contains only connected graphs.

Before we modify our algorithm, let us observe a few basic properties. For a pair (U, H), where $U \subseteq V(H)$, we set $L^d_{(U,H)} := \{v \in H : d_H(v, U) > d\}$. First, we show that if we have not found an unavoidable structure in a pair, then vertices at large distance to the last bag are irrelevant for $\mathcal{U}$ being unavoidable.

Observation 3.6. Let G be a graph containing a pair (U, H). If $H \notin \text{super}(\mathcal{U})$ and $D := \max\{\text{diam}(S) : S \in \mathcal{U}\}$, then for any $S' \subseteq G$ with $S' \cong S \in \mathcal{U}$ we have that $S' \subseteq G - L^D_{(U,H)}$.

Proof. Let $S' \subseteq G$ with $S' \cong S \in \mathcal{U}$. If $V(S') \cap L^D_{(U,H)} \neq \emptyset$, then there exists a vertex $v \in S'$ such that $d(v, U) > D$. Hence, any vertex $u \in S'$ satisfies $d(u, U) \geq d(v, U) - d(u, v) > 0$ and $V(S') \cap U = \emptyset$. Since U separates $H - U$ and $G - V(H)$ in G and S' is connected, we get that $S' \subseteq H - U$, contradicting that $H \notin \text{super}(\mathcal{U})$. □

Next we prove a similar result for vertices that are needed to check containment in the class $\mathcal{G}$.

Observation 3.7. Let (U, H) be a pair, $G \in \mathcal{G}$ be a graph containing (U, H), and $\mathbb{P}$ be a proof for G. We have that

- ▷ for all $v \in L^r_{(U,H)}$ the verifier satisfies $\mathcal{A}(G, \mathbb{P}, v) = \mathcal{A}(H, \mathbb{P}[V(H)], v)$ and
- ▷ for all $v \in V(G) \backslash L^r_{(U,H)}$ we obtain $\mathcal{A}(G, \mathbb{P}, v) = \mathcal{A}(G - L, \mathbb{P}[V(G) \backslash L], v)$, where $L \subseteq L^{2r}_{(U,H)}$.

Proof. Let $v \in H$ and $B := B^r_G(v)$. If $d(v, U) > r$, then $B \subseteq H - U = G[V(H) \backslash U]$ since U separates $H - U$ and $G - V(H)$. Consequently, $\mathcal{A}(G, \mathbb{P}, v) = \mathcal{A}(G[B], \mathbb{P}[B], v) = \mathcal{A}(H[B], \mathbb{P}[B], v) = \mathcal{A}(H, \mathbb{P}[V(H)], v)$. Otherwise, if $v \in H$ and $d(v, U) \leq r$ or $v \notin H$, then $B \subseteq G - L$ and we get that $\mathcal{A}(G, \mathbb{P}, v) = \mathcal{A}(G[B], \mathbb{P}[B], v) = \mathcal{A}((G - L)[B], \mathbb{P}[B], v) = \mathcal{A}(G - L, \mathbb{P}[V(G) \backslash L], v)$. □

Now we are ready to describe the adjustments for Algorithm 3.2. We use the verifier to check membership in $\mathcal{G}$ (in Line 10) and extendability (in Line 22). To do this, we associate each pair of (U, H) with a set of proofs $\mathbb{P}(U, H)$ that result in all vertices in $L^r_{(U,H)}$ being accepted by $\mathcal{A}$. If no such proof exists, then, by Observation 3.7, no graph in $\mathcal{G}$ contains (U, H). When checking for counterexamples, we check $H + E' \in \mathcal{G}$ by determining whether $\mathcal{A}$ accepts some proof in $\mathbb{P}(U, H)$ for $H + E'$.

Intuitively, this causes the algorithm to terminate since it allows us to exclude pairs (U', H') (with corresponding proofs) that coincide close to the vertices in U'. Due to the bound on the maximum degree, the amount of such vertices is finite and so is the amount of graphs. We now make this argument precise.

We remark that any pair (U, H) added to the queue now satisfies that

- ▷ $H' \notin \text{super}(\mathcal{U})$,
- ▷ there exists a smooth path-decomposition $(Q, \mathcal{W})$ of H with last bag U, and
- ▷ $\mathbb{P}(U, H) \neq \emptyset$ and $\mathcal{A}(H, \mathbb{P}, v) = 1$ for all $v \in L^r_{(U,H)}$ and $\mathbb{P} \in \mathbb{P}(U, H)$.

The first two parts were true before and the last one holds due to our implementation of the extendability test.

We extend our terminology for pairs to triples $(U, H, \mathbb{P})$, where $\mathbb{P} \in \mathbb{P}(U, H)$: a graph G contains $(U, H, \mathbb{P})$ if it contains (U, H) and there exists a proof $\overline{\mathbb{P}}$ that $\mathcal{A}$ accepts for G which satisfies $\overline{\mathbb{P}}[V(H)] = \mathbb{P}$. Such a triple is *good* if any graph containing it is in $\text{super}(\mathcal{U})$. We observe that (U, H) is good if and only if $(U, H, \mathbb{P})$ is good for all $\mathbb{P} \in \mathbb{P}(U, H)$. Lemma 3.3 generalises directly:

Corollary 3.8. *For a bijection φ with domain $V(H)$, the triple $(U, H, \mathbb{P})$ is good if and only if the triple $(\varphi(U), \varphi(H), \varphi(\mathbb{P}))$ is good.*

We can now formulate the main result of this section, which allows us to exclude pairs from being added to the queue and provides us with a proof for Theorem 1.1 as a corollary.

Theorem 3.9. *Let $(U, H, \mathbb{P})$ and $(U', H', \mathbb{P}')$ be two triples. Set*

$$D := \max\{\text{diam}(S) : S \in \mathcal{U}\}, d := \max\{D, 2r\}, \ L := L^d_{(U,H)}, \text{ and } L' := L^d_{(U',H')}.$$

Assume that

- ⊳ *(U, H) was added to the queue by the algorithm and*
- ⊳ *there exists a bijection $\varphi : V(H)\setminus L \to V(H')\setminus L'$ satisfying $\varphi(U) = U'$, $\varphi(H - L) = H' - L'$, and $\varphi(\mathbb{P})[V(H')\setminus L'] = \mathbb{P}'[V(H')\setminus L']$.*

If $(U, H, \mathbb{P})$ is good, then so is $(U', H', \mathbb{P}')$.

Proof. Let G' be a graph containing $(U', H', \mathbb{P}')$. By definition, we obtain a path-decomposition $(P', \mathcal{V}')$ of G' with $P' = 1 \dots n'$ such that $V'_j = U'$ and $G'_j = H'$ for some j. Additionally, we know that there is a proof $\overline{\mathbb{P}'}$ such that $\mathcal{A}(G', \overline{\mathbb{P}'}, v) = 1$ for all $v \in G'$ and $\overline{\mathbb{P}'}[V(H')] = \mathbb{P}'$.

We extend the bijection φ to H by choosing the images of L such that $\varphi(L) \cap V(G') = \emptyset$. Regard the graph

$$G := \varphi(H) \cup (G' - L')$$

We see that $\varphi(V(H)) \cap V(G' - L') = V(H' - L')$, $V(G)$ is the disjoint union of $\varphi(V(H))$ and $V(G' - V(H'))$, and $G - \varphi(L) = G' - L'$. In the following, we show that G is in $\mathcal{G}^k$ and contains the triple $(\varphi(U), \varphi(H), \varphi(\mathbb{P}))$.

We construct a smooth path-decomposition of G of width k that contains $\varphi(U)$ as a bag with associated graph $\varphi(H)$. To this end, let $(Q, \mathcal{W})$ be a smooth path-decomposition of H with the last bag $W_l = U$. Define $P := 1 \dots n' + l - j$ and let $\mathcal{V}$ contain the sets

$$V_i := \begin{cases} \varphi(W_i) & \text{for } i \in \{1, \dots, l\}, \\ V'_{i+j-l} & \text{for } i \in \{l + 1, \dots, n' - j + l\}. \end{cases}$$

Notice that $V_l = \varphi(U) = U' = V_j$, giving us a smooth decomposition of width k if it is a path-decomposition. Moreover, $G_l = \varphi(H)$, so G contains the pair $(\varphi(U), \varphi(H))$. We now verify the properties of a path-decomposition to get that G has path-width at most k and contains $(\varphi(U), \varphi(H))$.

Let $v \in G$. If $v \in \varphi(H)$ then $v \in \varphi(W_i) = V_i$ for some $i \le l$ and, otherwise, $v \in G' - V(H')$ and v enters a bag V'_i for $i > j$, placing it in V_{i+l-j}. For an edge $uv \in E(G)$ we get that uv is covered by a bag $V_i = \varphi(W_i)$ with $i < l$ if u is in $\varphi(H - U)$, as it leaves one of the first $l - 1$ bags. If this is not the case, we may assume that both u and v are either in $G' - V(H')$ or in U'. This means that uv is an edge of G' and covered by a bag V'_i with $i \ge j$. Hence uv is covered by the bag V_{i+l-j}. Finally, note that the set $\{i : v \in V_i\}$ can be written as $\{i \le l : v \in \varphi(W_i)\} \cup \{i \ge l : v \in V'_{i+j-l}\}$, making it the union of two paths. If both sets are nonempty, then $v \in \varphi(V(H)) \cap (V(G')\setminus V(H') \cup U') = U'$ and l is in both sets. Thus, $(P, \mathcal{V})$ is a path-decomposition.

As a next step, we verify that $G \in \mathcal{G}$ by exhibiting a proof $\overline{\mathbb{P}}$ for G that $\mathcal{A}$ accepts and which satisfies $\overline{\mathbb{P}}[\varphi(H)] = \varphi(\mathbb{P})$. This shows that $G \in \mathcal{G}$ and that G contains the triple $(\varphi(U), \varphi(H), \varphi(\mathbb{P}))$. We define

$$\overline{\mathbb{P}}(v) := \begin{cases} \varphi(\mathbb{P})(v) & \text{if } v \in \varphi(H), \\ \overline{\mathbb{P}'}(v) & \text{otherwise.} \end{cases}$$

Note that $\overline{\mathbb{P}}[V(H')\backslash L'] = \varphi(\mathbb{P})[V(H')\backslash L'] = \mathbb{P}'[V(H')\backslash L']$, yielding $\overline{\mathbb{P}}[V(G)\backslash\varphi(L)] = \overline{\mathbb{P}}[V(G')\backslash L'] = \overline{\mathbb{P}'}[V(G')\backslash L']$. Using Observation 3.7 we conclude that

$$\mathcal{A}(G, \overline{\mathbb{P}}, v) \\ = \begin{cases} \mathcal{A}(\varphi(H), \varphi(\mathbb{P}), v) = \mathcal{A}(H, \mathbb{P}, \varphi^{-1}(v)) & \text{if } v \in L^{r}_{(\varphi(U),\varphi(H))}, \\ \mathcal{A}(G - \varphi(L), \overline{\mathbb{P}}[V(G)\backslash\varphi(L)], v) = \mathcal{A}(G', \overline{\mathbb{P}'}, v) & \text{otherwise} \end{cases}$$

and in both cases $\mathcal{A}(G, \overline{\mathbb{P}}, v) = 1$.

In summary, G contains $(\varphi(U), \varphi(H), \varphi(P))$, which is a good triple. Hence, we can find a subgraph $S' \subseteq G$ with $S' \cong S \in \mathcal{U}$. By Observation 3.6 we know that $S' \subseteq G - L^{D}_{(\varphi(U),\varphi(H))} \subseteq G - L = G' - L' \subseteq G'$ and the triple $(U', H', \mathbb{P}')$ is good. □

Proof of Theorem 1.1. Theorem 3.9 shows that we can eliminate certain triples: if we added a pair (U, H) to the queue and we are currently regarding a pair (U', H') for which there exists a bijection φ with $\varphi(U) = U'$ and $\varphi(H - L) = H' - L'$, then we can modify $\mathbb{P}(U', H')$ by removing all elements $\varphi(\mathbb{P})$ for $\mathbb{P} \in \mathbb{P}(U, H)$. If the set $\mathbb{P}(U', H')$ is empty after these removals, then we need not add it to the queue again, since all triples $(U', H', \mathbb{P}')$ for $\mathbb{P}' \in \mathbb{P}(U', H')$ are good or in the queue to be checked as such.

This limits the pairs that are added to the queue to a finite amount. The bound Δ on the maximum degree on graphs in $\mathcal{G}$ gives us a bound on the vertices in $H - L$, namely, $(k + 1)\Delta(\Delta - 1)^{d-1} =: v$. Hence, the amount of pairs (U, H) with distinct graphs $H - L$ that we add to the queue is finite as well. Additionally, a pair is only added again if it has a proof that was not checked previously. Since there are at most 2^{sv} proofs for a graph H, we only add a finite amount of pairs to the queue and the algorithm terminates. □

3.3.3 | Runtime analysis

In this subsection we analyse the runtime of our algorithm to show that it is elementary and, in particular, it is better than the runtime obtained by Courcelle's theorem.

The iterations of the outer loop are bounded by the amount of pairs (U, H) we add to the queue. These are limited to one per potential graph $H - L$ (of which there are at most $x := v2^{v^2}$ many) and each of these can be added up to 2^{sv} times, once for every proof.

Both determining automorphisms as well as checking whether H has a subgraph in $\mathcal{U}$ can be bounded by $\mathcal{O}(|\mathcal{U}|v!v^2)$ by checking all permutations of the vertices of $H - L$. As a result, the runtime is dominated by that of the inner while loops, which contain these expensive operations. The while loop in Line 8 is iterated at most $2^{(k+1)^2}$ many times, the one in Line 19 is iterated at most $(k + 1)2^k$ times. (We reference Algorithm 3.2 despite the fact that we are analysing the modified version from the previous paragraph because their basic structure is the same.)

We bound the runtime of these two loops, starting with the former, for which we get $\mathcal{O}(2^{sv}vT(\mathcal{A}) + |\mathcal{U}|v!v^2)$, where $T(\mathcal{A})$ be a bound on the runtime of $\mathcal{A}$ when faced with a neighbourhood of size at most v. To see this, note that the first addend originates from the check whether a graph is in $\mathcal{G}$, by seeing whether one of the proofs associated with the pair is accepted by the vertices of $H - L$. The second addend covers both the check whether a subgraph in $\mathcal{U}$ exists and the removal of further elements from $\mathcal{F}$.

The latter loop's runtime is in $\mathcal{O}(2^{s(v+1)}vT(\mathcal{A}) + |\mathcal{U}|v!v^2 + v^2x)$. Here we note that the creation of H' also requires us to determine the new potential proofs for it. This means we need to regard at most $2^{s(v+1)}$ new proofs, each of which can each be checked in $\mathcal{O}(vT(\mathcal{A}))$ time, giving us the first term. The second covers the check $H' \notin \text{super}(\mathcal{U})$ as well as the adjustment of the set $\mathcal{Y}$. Together with the last term, it also accounts for the comparison with previously seen pairs (canonise and compare).

3.3.4 | Necessity of the adjustments to the algorithm

To wrap up, we show that our adjustments to the algorithm were necessary to achieve termination and that omitting the requirement on the degree yields an example for which the algorithm no longer terminates. Since local checkability was essential for the correctness of the adjustments we made, dropping this assumption forces us to revert to the original version, for which termination is not guaranteed. We note that in case a counterexample exists the algorithm always terminates, returning one of the smallest sizes. However, if the graphs in $\mathcal{U}$ are, indeed, unavoidable, then the algorithm, in its general form, need not terminate. As an example consider the class $\mathcal{G}$ in which every graph consists of one or more components, each of which is the union of a path P with a triangle K where $V(K) \cap V(P) = \{v\}$ and v is an end of P. We set $\mathcal{U} = \{K_3\}$ and $k = 2$. The reason the algorithm does not terminate for this set is that the triangle can appear arbitrarily late in a path-decomposition. This is an example of a class with a bounded maximum degree and it is also locally checkable, a proof of which can be obtained by making use of [20, Lemma 2.1] which allows us to orient the path towards the end with the triangle in every component.

If we remove the requirement on bounded maximum degree, we can still use our modified algorithm, since we only needed the bound on the degree to estimate the amount of pairs that are added to the queue. But there can now be infinitely many such pairs, as illustrated by the following example. Regard the class $\mathcal{G}$ which contains all graphs that consist of two stars connected by a single edge (and disjoint unions of these). Every component of a graph in this class has diameter 3, making it locally checkable. By choosing $k = 1$ and $\mathcal{U}$ to contain a path of length 3, we obtain an example for which the algorithm does not terminate. Any path-decomposition of a graph $G \in \mathcal{G}$ starts by covering the edges in one of the two stars, followed by the edge connecting it to the second, and ending with the edges of this star. Consequently, the path of length 3 can appear arbitrarily late in the path-decomposition again.

3.4 | Tailoring the algorithm to cubic graphs

Aside from making use of general properties of cubic graphs, such as that $|V(G)|$ is even and $|E(G)| = \frac{3}{2}|V(G)|$, hdf decompositions are immensely helpful in speeding up the algorithm. More precisely, when regarding a pair (U, H), Lemma 2.2 lets us choose the vertex to leave U if H has a vertex of degree at least 2, eliminating the need to iterate over $\overline{U}$ entirely.

Theorem 3.10. *Instead of iterating over the set* $\overline{U} := \{u^{\text{Aut}(U,H)} : u \in U\}$, *it suffices to choose a single vertex of degree at least* 2, *if such a vertex is present, in the case that* $\mathcal{G}$ *contains only cubic graphs.*

Proof. To see that this holds, recall the proof of Theorem 3.2. There we proved the invariant that $\mathcal{G}^k \subseteq \text{super}(\mathcal{U})$ if all pairs in the queue are good. More precisely, we showed that in the iteration where we remove the pair (U, H), this pair is good if all the newly added ones are. Consequently, we need to show that a pair (U, H) is already good if we only add pairs of form (U', H') for which $U' = U \backslash \{u\} \cup \{v_{i+1}\}$ for some vertex u of degree at least 2 in H. (If H has no such vertex, the algorithm remains unchanged.)

In this situation, where u and U' are defined as above, let $G \in \mathcal{G}^k$ be a graph containing (U, H) with corresponding path-decomposition $(P, \mathcal{V})$ and $U = V_i$. Assume, without loss of generality, that v_j is the vertex entering bag j for $j \in \{2, \dots, i + 1\}$. We know that $G \in \text{super}(\mathcal{U})$ if U is the last bag in the decomposition, so we may assume there is a subsequent bag V_{i+1}. By applying Lemma 2.2, we get a smooth path-decomposition $(P, \mathcal{W})$ of G such that $V_j = W_j$ for all $j < i$, $V_{i-1} \backslash V_i = W_{i-1} \backslash W_i$, and u leaves W_i. Let φ be a bijection on $V(G)$ with $\varphi_{|V(G_{i-1})} = \text{id}_{V(G_{i-1})}$ that maps the unique vertices entering W_i and W_{i+1} to v_i and v_{i+1}. This gives us that $\varphi(G)$ contains the pair $(\varphi(W_{i+1}), \varphi(H_{i+1}))$ where H_{i+1} is the graph associated with W_{i+1}. Since the vertex leaving V_{i-1} and W_{i-1} coincide and $\varphi(u) = u$, we get that $\varphi(W_i) = U$ and $\varphi(W_{i+1}) = U'$. (Note that $\varphi(u) = u$ follows from the fact that $d_H(u) > 0$, which implies that it is not the vertex entering V_i.)

The graph $H' = \varphi(H_{i+1})$ is of the form $H + v_{i+1} + \{uy : y \in Y\}$, where $Y \subseteq U \backslash \{u\}$. If $H' \in \text{super}(\mathcal{U})$, then $G \in \text{super}(\mathcal{U})$ and we may assume this is not the case. As the set Y is in $\mathcal{Y}$, it is regarded by the algorithm and added to the queue unless there is already an element $(\psi(U'), \psi(H'))$ contained in it. In either case, the pair (U', H') is good by assumption and thus both $\varphi(G)$ and G are in $\text{super}(\mathcal{U})$. □

3.5 | Unavoidable minors and induced subgraphs

Our algorithm can easily be extended to a test for unavoidable minors or induced subgraphs. We only need the following adaption of the definition of super($\mathcal{U}$) to the set that contains all graphs with a minor or an induced subgraph in $\mathcal{U}$. In these cases a pair (U, H) is good if H has a minor in $\mathcal{U}$, respectively, if $H - U$ has a subgraph in $\mathcal{U}$ since this subgraph is inevitably induced. Our termination proof carries over to the induced subgraph case.

4 | PATH-WIDTH AND GIRTH OF CUBIC GRAPHS

In this section G is a cubic graph of path-width k and girth g. We show that large girth necessitates a large path-width. We determined all smallest graphs of path-width k and girth $\xi(k)$. These can be found in Table 2, where we used the complete list of small cubic graphs [8] and checked the path-width using SageMath [23]. We prove that the graphs in Table 2 maximise the possible girth amongst all cubic graphs of path-width k by providing sharp upper bounds in these cases. Moreover, we prove that a cubic graph of path-width k is of girth at most $\frac{2}{3}k + \frac{10}{3}$.

Before we start formally proving Theorems 1.2 and 1.3, let us briefly sketch how these proofs work. First, we gather information about the graphs G_i for the initial bags, by counting both the connected components of these graphs as well as their amount of vertices of degree 0 to 3. To limit the possible cases and to obtain particularly simple ones with few degrees 2 and 3 vertices, we use hdf path-decompositions. From this we can deduce that a cycle is present by the kth bag and at least two exist by bag $k+2$. Theorem 1.2 is obtained by regarding the last bag l for which G_l contains at most one cycle and uses the small amount of vertices that can lie on cycles at all to deduce that G_{l+1} has a short cycle. For Theorem 1.3 we show that G_i is a forest for all $i \leq g-2$ and then combine this knowledge with our degree and component counts to go through the graphs G_{g-1} to G_{g+3} to find a short cycle.

We now begin with the formal proof. Recall that hdf path-decompositions allow us to assume that degree 2 or 3 vertices, if present, leave a bag. These are useful when determining the structure of the graphs G_i for the initial bags of the decomposition. The *extended graph* of G is defined as

$$G^{\text{ext}} := G + E_{k+1}.$$

This lets us extend the path-decomposition of G by letting the vertices in the last bag leave while adding the new degree 0 ones. We call such a path-decomposition an *extended decomposition of* G. By starting with a hdf path-decomposition of G, this can easily be made hdf as well. This extension allows us to ensure that the decomposition has enough bags so that sufficiently many associated graphs exist. For the remainder of this section, we denote G^{ext} by H and assume that the path-decomposition $(P, \mathcal{V})$ is an extended decomposition of G that is also hdf. We begin by describing the degree distribution in the associated graphs. Set

$$d_j^i := |\{v \in H^i : d_{H_i}(v) = j\} \quad \text{for} \quad j \in \{0,1,2,3\} \text{ and } i \in \{1, ..., n'\}.$$

We write t_i for the amount of nontrivial components of H_i, where nontrivial means not of order 1. To determine the degrees, we proceed by induction on i. We regard H_{i+1}, assuming that the claim holds for H_i and H_{i+1} is of the form $H_i + E(v) + v_{i+1}$.

Lemma 4.1. *Let $i \in \{1, ..., n'\}$. If H_i is a forest, then:*

(i) $d_0^i \geq 1$.
(ii) $d_1^i = i - 1 + 2t_i$.
(iii) *at most one component of H_i contains vertices of degree 2, no subpath of H_i contains more than two degree 2 vertices, and $d_2^i \leq 3$.*
(iv) $d_3^i = i - 1$.

Proof. For $i = 1$ the claim holds as $H_1 = G_1 \cong E_{k+1}$. Assume that the claim holds up to some index $i \geq 1$ and let $H_{i+1} = H_i + E(v) + v_{i+1}$ be acyclic. We note that $d_{H_{i+1}}(v_{i+1}) = 0$ and Property (i) follows. Furthermore, the vertices of degree 3 in H_i are exactly those that have left prior bags, so $d_{H_i}(v) < 3$. Since H_{i+1} is acyclic, v and all new neighbours of v are in different components of H_i. Thus no new neighbour of v has degree 2 in H_i, as otherwise v does too, by hdf, and they lie in the same component of H_i by Property (iii). So $d_3^{i+1} = d_3^i + 1 = i$ and (iv) holds.

Next, we remark that the handshaking lemma yields the following equation, where $\varkappa(H_{i+1})$ denotes the number of components of H_{i+1}:

$$d_1^{i+1} + 2d_2^{i+1} + 3d_3^{i+1} = 2(|V(H_{i+1})| - \varkappa(H_{i+1})) = 2d_1^{i+1} + 2d_2^{i+1} + 2d_3^{i+1} - 2t_i.$$

From this we deduce that $d_1^{i+1} = d_3^{i+1} + 2t_i$ and Property (ii) is satisfied.

This only leaves (iii). First assume $d_2^i > 0$, in which case $d_{H_i}(v) = 2$ by hdf and it has a unique new neighbour u. If $d_{H_i}(u) = 1$, then $d_{H_{i+1}}(u) = 2$ but u is part of the same component as the remaining degree 2 vertices of H_i. The condition that at most two degree 2 vertices lie on any path in H_i is ensured by the existence of a vertex whose removal separates all of them. Such a vertex also separates the vertices of degree 2 in H_{i+1} as u is a neighbour of the prior degree 2 vertex v. If u does not have degree 1, the claim follows directly.

If $d_2^i = 0$, then all degree 2 vertices of H_{i+1} are new neighbours of v that had degree 1 in H_i. Consequently, there are at most three such vertices, they all lie in the same component of H_{i+1}, and they are separated by v. □

Lemma 4.2. *Let $i \in \{1, ..., n'\}$. If H_i contains a unique cycle, then there exists a $j \in \{i, i+1\}$ such that H_j contains a unique cycle and satisfies the following properties:*

(i) $d_0^j \geq 1$.
(ii) $d_1^j = j - 3 + 2t_j$.
(iii) *at most one component of H_j contains vertices of degree* 2 *and* $d_2^j \leq 3$.
(iv) $d_3^j = j - 1$.

Proof. Property (i) is satisfied because v_j has degree 0 in H_j, for all j. The claim holds for $i = 1$ since H_1 is always acyclic. Assume that it holds up to some $i \geq 1$ and let $H_{i+1} = H_i + E(v) + v_{i+1}$ contain a unique cycle. The graph H_i thus contains at most one cycle and we first assume it is acyclic, letting us apply Lemma 4.1.

If $d_{H_i}(v) = 2$, then the new neighbour u of v is in the same component as v in H_i and $t_{i+1} = t_i$. The claim holds for $j = i + 1$ when $d_{H_i}(u) = 1$: here $d_3^j = j - 1$, $d_1^j = j - 3 + 2t_j$, $d_2^j = d_2^i \leq 3$, and all degree 2 vertices are in the same component. This leaves the case that $d_{H_i}(u) = 2$, where H_{i+1} does not satisfy the properties as it has $i + 1$ vertices of degree 3. However, by hdf, u is the next vertex to leave and $H_{i+2} = H_{i+1} + v_{i+2}$. Here, H_{i+2} satisfies $d_3^{i+2} = i + 1$, $d_1^{i+2} = i - 1 + 2t_i$, and $d_2^{i+2} = d_2^i - 2 \leq 1$, completing this case.

We may now assume that $d_{H_i}(v) \leq 1$ and $d_2^i = 0$. As H_{i+1} has a unique cycle, either one new neighbour is in the same component as v in H_i and the remaining ones are in different components or no new neighbour is in the same component as v and exactly two of them share a component. In either case, $d_3^{i+1} = i$ and at most three degree 2 vertices are present in H_{i+1}, which share a component. For the degree 1 vertices, note that any new neighbour of v has degree 0, resulting in a new vertex of degree 1, or it has degree 1, reducing their amount by one. However, every neighbour of degree 1 except one also decreases the amount of nontrivial trees, as there is a unique cycle. This shows Property (ii).

We are left with the case that H_i is not acyclic, which means it contains a unique cycle and the induction hypothesis applies. This gives us that either H_{i+1} satisfies the degree properties, and we are done, or H_i does. We may assume the latter. The only new degree 3 vertex is v since connecting two vertices of degree 2 would result in a new cycle. If $d_{H_i}(v) = 2$, its neighbour has degree 0 or it has degree 1 and is in a different component. This increases the amount of degree 1 vertices by one or decreases their amount and the amount of nontrivial trees by one each. Otherwise, $d_{H_i}(v) \leq 1$ and H_i has no degree 2 vertices. The new neighbours now have degree 1 and are in different components of H_i or degree 0 and the properties hold once more. □

Observation 4.3. The graph H_k is not a forest and H_{k+2} has more than one cycle.

Proof. Suppose H_k is acyclic. Lemma 4.1 states that $2k = |H_k| \geq d_3^k + d_1^k + d_0^k \geq 2k - 1 + 2t_k$, which is a contradiction since $k \geq 3$. Similarly, if H_{k+2} has at most one cycle, then it has exactly one and either H_{k+2} or H_{k+3} satisfy the degree properties of Lemma 4.2. Using these we obtain $2k + 2 = |H_{k+2}| \geq d_3^k + d_1^k + d_0^k \geq 2k + 2t_k + 1$ or $2k + 3 = |H_{k+3}| \geq 2k + 2t_k + 3$, which is a contradiction in both cases. □

We are now ready to prove our first bound: Theorem 1.2.

Proof of Theorem 1.2. To prove this result, we need to show that $g \leq \frac{2}{3}k + \frac{10}{3}$ for the graph G of path-width k and girth g. Regard the extended graph H of G. Let l be the maximal index such that H_l contains at most one cycle. By Observation 4.3 we have that $l \leq k + 1$. Since H_l contains either no cycle, and Lemma 4.1 applies, or it contains exactly one, and Lemma 4.2 can be used, we get that $d_3^l = l - 1$ and $d_2^l \leq 3$. (Note that H_{l+1} contains multiple cycles, so Lemma 4.2 actually applies to H_l.) This lets us estimate the amount of vertices of degree at least 2 in H_{l+1}: if $d_2^l > 0$, then at most one edge is added in the transition to H_{l+1} by hdf. Since at least one end of such an edge has degree 2, we obtain $d_3^{l+1} + d_2^{l+1} \leq d_3^l + d_2^l + 1$. On the other hand, if $d_2^l = 0$, then at most one new degree 3 and three new degree 2 vertices are created. This yields $d_3^{l+1} + d_2^{l+1} \leq d_3^l + 4$. Combined these give us that

$$d_3^{l+1} + d_2^{l+1} \leq d_3^l + 4 = l + 3.$$

Let C and C' be two distinct cycles in H_{l+1}. If they are disjoint, then $|C| + |C'| \leq l + 3$, yielding

$$g \leq \frac{l+3}{2} \leq \frac{k}{2} + 2.$$

Otherwise, C' contains a path Q between two nonadjacent vertices of C. Using this path and C, we obtain two cycles of which one has length at most $\frac{|C|}{2} + ||Q||$. We estimate $|Q| \leq d_3^{l+1} + d_2^{l+1} - |C| + 2 \leq l + 5 - |C|$ to get $g \leq |C|$ and $g \leq \frac{|C|}{2} + l + 4 - |C| = l + 4 - \frac{|C|}{2}$. Consequently,

$$g \leq \frac{2}{3}k + \frac{10}{3}$$

since $|C| = l + 4 - \frac{|C|}{2}$ holds for $|C| = \frac{2}{3}l + \frac{8}{3}$. The bound from the disjoint cycle case is strictly better than this one and, hence, the result follows. □

In the following, we demonstrate that our bounds are tight for small values of k. In preparation, we make the following observations.

Observation 4.4. The neighbours of a vertex v of degree at most 2 in H_i are of degree 3.

Proof. Note that H_i only contains edges incident to vertices that have left one of the first $i-1$ bags. Therefore, at least one end of any edge is of degree 3. □

Observation 4.5. If H_i is a forest and Q is a path in H_i, then

$$|Q| \leq i + 2 + \min\left\{d_2^i, 2\right\} - t_i.$$

If an end of Q has degree 2, then this bound improves by 2 to

$$|Q| \leq i + \min\left\{d_2^i, 2\right\} - t_i.$$

Proof. The path Q has at most 2 vertices of degree 1, the remaining ones are of degree 2 or 3. By Lemma 4.1, H_i has $d_3^i = i - 1$ and at most two of degree 2 lie on Q. Also, any nontrivial component contains at least one degree 3 vertex by Observation 4.4. This gives us that $|Q| \leq 2 + \min\{d_2^i, 2\} + [d_3^i - (t_i - 1)]$, which is the first inequality.

If one end, say v has degree 2, then at most one vertex of degree 1 is in Q. Additionally, both neighbours of v have degree 3 by Observation 4.4 and at most one of them is on Q. Consequently $|Q| \leq 1 + \min\{d_2^i, 2\} + [d_3^i - 1 - (t_i - 1)]$, completing the proof. □

Observation 4.6. If $H_{i+1} = H_i + E(v) + v_{i+1}$, H_i is a forest, and $i \leq g - 2$, then no new neighbour of v is in the same component as v in H_i.
Proof. Suppose that u is a new neighbour of v and in the same component of H_i as v. Let Q be the path joining v and u in H_i. If $d_2^i = 0$, then $|Q| \leq i + 2 - t_i \leq g - 1$ by Observation 4.5. If $d_2^i \geq 1$, then $\deg_{H_i}(v) = 2$ by hdf. Again Observation 4.5 yields $|Q| \leq i + \min\{d_2^i, 2\} - t_i \leq g - 1$. In both cases, the edge vu causes a cycle of length $g - 1 < g$, which is a contradiction. □

Next, we show that high girth necessitates many acyclic-associated graphs.

Lemma 4.7. *For $i \leq g - 2$, H_i is a forest.*

Proof. For $i = 1$ the claim holds since $H_1 \cong E_{k+1}$. Assume the claim holds for some $i \leq g - 3$. If $i = 1$, then $|V(H)| > |V(G)| \geq k + 1 = |V(H_1)|$ and, hence, H_2 exists. If $i \geq 2$, then $d_1^i \geq 1$ by Lemma 4.1 and since any vertex of H is isolated or of degree 3, the graph H_{i+1} exists. Let v be the vertex that leaves V_i. It holds that $H_{i+1} = H_i + E(v) + v_{i+1}$.

First note that Observation 4.6 shows that no new neighbour of v is part of the same component as v in H_i. In particular, if $d_{H_i}(v) = 2$, then H_{i+1} is acyclic. This leaves the case that $d_{H_i}(v) < 2$. Since the path-decomposition is hdf, we know that $d_2^i = 0$ and there are multiple new neighbours of v. In case two of them are in the same component of

H_i, we get a short cycle by noticing that the path between these neighbours has order at most $i + 2 - t_i \leq g - 2$ by Observation 4.5, which is a contradiction. Thus H_{i+1} is a forest. □

With these tools at hand, we can now prove Theorem 1.3.

Proof of Theorem 1.3. The graphs found in Table 2 show that $\xi(k)$ takes at least the value specified above and it suffices to prove that it is not larger. To this end we show that $g \leq k + 1$ in general, $g \leq k$ if $k \geq 4$, $g \leq k - 1$ if $k \geq 7$, and $g \leq k - 2$ if $k \geq 10$. This proves the equalities for the values specified above and shows that $\xi(k) \leq k - 2$ for $k \geq 10$.

We know that $H_1, \ldots, H_{g-2}$ are acyclic by Lemma 4.7. In this proof, we look at the possibilities that arise for the subsequent associated graphs, starting with H_{g-1}. Since H_{g-2} is a forest, Lemma 4.1 implies $d_3^{g-2} = g - 3$, $d_1^{g-2} = g - 3 + 2t_{g-2}$, and $d_0^{g-2} \geq 1$. Therefore

$$1 \leq d_0^{g-2} + d_2^{g-2} = k + g - 2 - (g - 3) - (g - 3 + 2t_{g-2}) = k + 4 - g - 2t_{g-2}.$$

Rearranging yields $g \leq k + 3 - 2t_{g-2} \leq k + 1$ proving the first of the four inequalities.

For the remainder of this proof, we may assume $g \geq k - 1$ and $k \geq 4, g \geq 5$. Using the former, we get $t_{g-2} \leq 2$ and $t_{g-2} = 1$ if $g \geq k$.

Let $H_{g-1} = H_{g-2} + E(v) + v_{g-1}$. Since H_{g-2} has $g - 3$ vertices of degree 3, v is not one of them and it has new neighbours. By Observation 4.6, any new neighbour is in a component different from v. In particular, if $d_{H_{g-2}}(v) = 2$, then H_{g-1} is acyclic. Assume that v has degree at most 1 in H_{g-2} which implies $d_2^{g-2} = 0$ by hdf. If all new neighbours of v are in different components of H_{g-2}, then H_{g-1} is acyclic. In analogy to the proof of Lemma 4.7, if two new neighbours are in the same component, then the path between them has order at most $g - t_{g-2}$ by Observation 4.5. This implies $t_{g-2} = 1$ and the path has length exactly $g - 1$ and contains all degree 3 vertices. We conclude that $d_{H_{g-2}}(v) = 0$ in this case and H_{g-1} is acyclic if $d_{H_{g-2}}(v) = 1$.

If v is isolated, then there either is a unique cycle of length g in H_{g-1} or the third new neighbour of v is also in the same component of H_{g-2} as the other two. In the first case, H_{g-1} has one nontrivial component and satisfies $d_3^{g-1} = g - 2$, $d_2^{g-1} = 2$, and $d_1^{g-1} = g - 2$. Otherwise, if v has all three neighbours u_1, u_2, u_3 in the same component T, the paths u_iTu_j are of order at least $g - 1$ for $1 \leq i < j \leq 3$. Consequently, all three of them contain all $g - 3$ vertices of degree 3 in H_{g-2} and differ only in their ends. Hence, H_{g-1} contains a cycle of length at most 4 and $g \leq 4$, contradicting our assumption.

This completes the inequality $g \leq k$ for $k \geq 4$: If $g = k + 1$, then $t_{g-2} = 1$ and $d_2^{g-2} = 0, d_0^{g-2} = 1$. By the above we get that $d_{H_{g-2}}(v) = 1$ results in a small cycle as v has a new neighbour in its component. If, otherwise, $d_{H_{g-2}}(v) = 0$, then all three neighbours of v in H_{g-1} lie in the same component of H_{g-1}, which is a contradiction.

The first part of this proof showed the bounds for $k \in \{3, 4, 5, 6\}$ and described H_{g-1}, which we recall again below. From now on we may assume that $k, g \geq 7$ and need to verify the last two inequalities. Our next goal is to describe $H_g = H_{g-1} + E(w) + v_g$. We saw that there are two options for H_{g-1}:

$$H_{g-1} \quad \text{is acyclic with} \quad t_{g-1} = 1, \quad d_3^{g-1} = g-2, \quad \text{and} \quad d_1^{g-1} = g \text{ or} \tag{2}$$

$$H_{g-1} \quad \text{has a unique cycle}, \quad t_{g-1} = 1, \quad d_3^{g-1} = d_1^{g-1} = g-2, \quad d_2^{g-1} = 2. \tag{3}$$

Note that we used Lemma 4.1 to obtain $t_{g-1} = 1$ in Option (2), which holds since H_{g-1} has $k + g - 1 \leq 2g$ vertices in total and $d_0^{g-1} \geq 1$. The last inequality is also true for Option (3) and all associated graphs in general, which is why we omit writing it each time. Furthermore, $d_0^{g-1} \leq 2$ in both cases, using the above cardinality argument.

Let us start with Option (3). We know here that $d_{H_{g-1}}(w) = 2$ and we denote its unique new neighbour by u. If u has degree 0, then

$$\begin{gathered} H_g \quad \text{has a unique cycle}, \quad t_g = 1, \quad d_3^g = d_1^g = g-1, \quad d_2^g = 1, \quad \text{and} \\ 1 \leq d_0^g \leq 2. \end{gathered} \tag{4}$$

Otherwise, u lies in the same component as w and we obtain a path of length at most 2 between two vertices of C. This yields a cycle of length at most $\frac{|C|}{2} + 2$ and we get $g \leq \frac{g}{2} + 2$ or $g \leq 4$, a contradiction.

If H_{g-1} is acyclic, there are more options. For ease of notation, let T be the nontrivial component of H_{g-1}. First assume that $d_{H_{g-1}}(w) = 2$ and let u be its unique new neighbour. If u is in T, then $|wTu| \leq g - 1$ by Lemma 4.1 since $d_2^{g-1} \leq 1$, a contradiction. Thus, u has degree 0 and

$$\begin{gathered} H_g \quad \text{is acyclic with} \quad t_g = 1, \quad d_3^g = g-1, \quad d_2^g = 0, \quad d_1^g = g+1, \quad \text{and} \\ d_0^g = 1 \end{gathered} \tag{5}$$

by Lemma 4.1. Here we have used that $g \geq k - 1$ to obtain that $d_2^g = 0$, in particular, this case does not occur if $g = k$.

This just leaves the case that $d_{H_{g-1}}(w) \leq 1$, giving us $d_2^{g-1} = 0$. Assume first that $d_{H_{g-1}}(w) = 1$. Should both new neighbours of w have degree 0, then H_g is acyclic, and we are in Option (5). Otherwise, let $u \in T$ be a new neighbour of w. Again we use Observation 4.5 to see that $|uTw| \leq g$ and obtain a cycle C of length exactly g in $H_{g-1} + uw$. Should the second new neighbour of w also be in T, we obtain a path of length at most 2 between two vertices of C, yielding girth at most 4, just as above. This does not occur, so H_g contains a unique cycle of length g and checking the values d_j^g shows that we are in Option (4).

We are left with the situation where $d_{H_{g-2}}(w) = 0$. Because $d_0^{g-1} \leq 2$ at least two of its new neighbours, say u_1, u_2 are in T. We first argue that the third neighbour, u_3, is not. If this were the case, then each of the three paths $u_i T u_j$ for $1 \leq i < j \leq 3$ would yield a cycle when combined with $u_i w u_j$. Thus each of these paths contains at least $g - 1$ vertices, at least $g - 3$ of which have degree 3. Since H_{g-1} only has $g - 2$ such vertices and $g \geq 7$, we get the existence of a vertex $t \in T$ that occurs on all three paths. We assume that the order $|u_i T t|$ is maximised for u_3, then $|\mathring{u}_1 T \mathring{t}| + |\mathring{u}_2 T \mathring{t}| \leq \frac{2}{3}(g-3)$ since there are only $g - 3$ degree 3 vertices other than t in H_{g-1}. But this means that $|u_1Tu_2| = |\mathring{u}_1T\mathring{t}| + |\mathring{u}_2T\mathring{t}| + 3 \le \frac{2}{3}g + 1$. Hence $g \le \frac{2}{3}g + 2$ or $g \le 6$, contradicting our assumption.

Consequently u_3 has degree 0 and H_g has a unique cycle $u_1Tu_2wu_1$ of length at most $g + 1$, giving us the final option in which

$$H_g \quad \text{has a unique cycle,} \quad t_g = 1, \quad d_3^g = g - 1, \quad d_2^g = 2, \quad d_1^g = g - 1, \quad d_0^g = 1. \tag{6}$$

The last equality follows once more from the fact that H_g has $k + g$ vertices, so $g = k - 1$ in this case.

We now have several possible options for H_g and we begin by taking a look at Option (4), which is the only one relevant for the case that $k = g$. Here $d_{H_g}(x) = 2$ and the identical argumentation to before shows that the new neighbour of x has degree 0 and H_{g+1} still has a unique cycle of length g and $d_3^{g+1} = g$, $d_2^{g+1} = 0$, $d_1^{g+1} = g$, and $1 \le d_0^{g+1} \le 2$.

We claim that this suffices to prove the third inequality. Suppose $g = k$, then we already know we are in Option (4). Hence, H_{g+1} is of the form described above and $d_0^g = 1$. We regard H_{g+2}. If the degree 0 vertex leaves, all its neighbours, u_1, u_2, u_3 say, are in the same component as the cycle C. Suppose u_1 and u_2 have minimal distance in C, then there exists a path of length at most $\frac{g}{3}$ between them and we obtain a cycle of length $\frac{g}{3} + 4$ by extending it to use u_1yu_2. (Note that all degree 1 vertices of H_{g+1} are adjacent to a degree 3 one by Observation 4.4 and all degree 3 vertices are on C.) This yields $g \le 6$, a contradiction. Hence, $d_{H_{g+1}}(y) = 1$ and it has at least one neighbour u in the same component as the cycle. But now we get a cycle of length at most $\frac{g}{2} + 3$ which again yields $g \le 6$, proving the $g \le k - 1$ for $k \ge 7$.

For the final inequality ($g \le k - 2$ if $k \ge 10$), we can update our assumptions to $g = k - 1$, and $k \ge 10$, so $g \ge 9$. With this we finish the description of the options for H_{g+1}. The option described above is

$$H_{g+1} \quad \text{has a unique cycle,} \quad t_{g+1} = 1, \quad d_3^{g+1} = d_1^{g+1} = g, \quad d_2^{g+1} = 0, \quad d_0^{g+1} = 2. \tag{7}$$

For Option (5), we denote the nontrivial tree of H_g by T. If the vertex of degree 0 leaves, then it has all three neighbours, u_1, u_2, u_3 say, in T. As each path u_iTu_j contains at least $g - 1$ vertices, it has $g - 3$ degree 3 vertices. As before, this yields a vertex t common to all paths as $g \ge 9$ since H_g only has $g - 1$ many in total. The analogous argumentation yields a path, say u_1Tu_2 of order at most $\frac{2}{3}(g - 2) + 3$ and thus a cycle of length $\frac{2}{3}g + \frac{8}{3}$. Hence $g \le 8$, contradicting our assumption.

As a result, $d_{H_g}(x) = 1$. If x has two neighbours in T, then the three paths between x and these neighbours would share a common vertex and the short path from above yields a contradiction again. So x has one new neighbour of degree 0 and the other one in T. Let u be this neighbour, then $|uTx| \le g + 1$ by Observation 4.5 and we obtain

$$H_{g+1} \quad \text{has a unique cycle and} \quad t_{g+1} = d_2^{g+1} = d_0^{g+1} = 1, \quad d_3^{g+1} = d_1^{g+1} = g. \tag{8}$$

Finally, we regard Option (6), which again contains a unique cycle C. Here, $d_{H_g}(x) = 2$ and x has a unique new neighbour u. If $u \notin T$, then this coincides with Option (8). Otherwise, if $u \in T$, we obtain a path between two vertices of C. If $|C| = g + 1$, then C contains all vertices of degrees 3 and 2 in H_g, giving this path length 3. Otherwise, if $|C| = g$, this path potentially has length 4. In either case, we obtain a cycle of length at most $\frac{g+1}{2} + 3 \leq \frac{g}{2} + 4$ and this yields $g \leq 8$, a contradiction.

The next graph is $H_{g+2} = H_{g+1} + E(z) + v_{g+2}$. For H_{g+1} only the two Options (7) and (8) remain, and we start with the latter. Here we have $d_{H_{g+1}}(z) = 2$ by hdf and z is on the unique cycle C. Thus, if $|C| = g$, then we either end up with a unique cycle of length g or G contains a cycle of length at most $\frac{g}{2} + 3$, contradicting $g \geq 9$. If $|C| = g + 1$, a unique cycle of length $g + 1$ remains or a cycle of length at most $\frac{g+1}{2} + 2$ is present, which is a contradiction. Consequently,

$$H_{g+2} \quad \text{has a unique cycle and} \quad d_3^{g+1} = d_1^{g+1} = g + 1, \quad d_2^{g+1} = 0, \quad d_0^{g+1} = 1. \tag{9}$$

If H_{g+1} is as specified in (7), then $d_{H_{g+1}}(z)$ results in a path of length 4 between two vertices of C, which we have already seen to cause $g \leq 8$. So z has degree 1 and its new neighbours have degree 0 as otherwise we obtain a path of length 3 between two vertices of C. This leaves the unique cycle of length g intact and results in $g + 1, 0, g + 1, 1$ vertices of degree $3, 2, 1, 0$, letting us include it in Option (9).

Now we can wrap up the proof by regarding H_{g+3}. Since only Option (9) remains for H_{g+2}, it has a unique cycle C. If a vertex of degree 1 leaves, then it is in the same component as C in H_{g-2} and at least one of its new neighbours is in this component as well. This yields a cycle of length at most $\frac{g}{2} + 4$ or $\frac{g+1}{2} + 3$ in H_{g+3}, which is a contradiction to $g \geq 9$ in either case. Should the degree 0 vertex leave, we get that all three of its neighbours are in the component of H_{g-2} containing C and we find a new cycle of length at most $\frac{g}{3} + 5$ or $\frac{g+1}{3} + 4$. Again this contradicts $g \geq 9$ and the proof is complete. □

We obtain Theorem 1.4 as a direct consequence of Theorem 1.3 and the values of Table 2.

As a further example of an application of our algorithm, we prove Theorem 1.5, which gives a constructive characterisation of the class of all cubic graphs of path-width 3 which are extremal with respect to ξ.

Proof of Theorem 1.5. Let G be a cubic graph of path-width 3 and girth 4. Running Algorithm 3.1 for $k = 3$, $\mathcal{G}$ the class of all cubic girth-4-graphs, and $\mathcal{U} = \{K_{3,3}, G_1, G_2, ..., G_6\}$ confirms that $\mathcal{U}$ is unavoidable for $\mathcal{G}^3$. If $G \cong K_{3,3}$, then the empty sequence yields the desired statement. Therefore, we may assume that G contains one of the graphs $G_1, ..., G_6$ as a subgraph.

Fix $i \in \{1, ..., 6\}$ and assume that G' is obtained from G by the reduction ρ_i. We leave it to the reader to check that G' is a simple cubic graph of girth 4. If $i \in \{1, 2, 3, 5\}$, then G' is a minor of G and, hence, G' is of path-width 3. For $i \in \{4, 6\}$ the path-width remains 3 as well, though checking this is technical and does not yield a lot of insight. Roughly speaking, one contracts the graph G_i to a K_4, which we call M, and takes a path-decomposition for this resulting minor, which still has width 3. This decomposition contains a bag containing M

and since this is not a separator of the graph, it must be at an end of the path-decomposition. Finally, one argues about the structure of path-decomposition of $G - M$, which is simplified by using the hdf concept. This decomposition is then extended to one of G'.

As a result, G' is a smaller graph that satisfies all the necessary properties to use induction, proving Theorem 1.5(ii). For Part (i), we note that the graphs $G_3, \ldots, G_6$ exhibit a 2-edge separator and cannot be a subgraph of G if G is 3-connected. Thus, we only need to verify that the reductions ρ_1 and ρ_2 preserve 3-connectivity.

To see that this is the case, regard $R \coloneqq G' - \{uv, u'v'\}$ for two edges uv and $u'v'$ of G'. If both edges are in G_i, then R remains connected. Let H be the unchanged part of G, that is, $H \coloneqq G - V(G_i)$. We may thus assume that at least one edge we delete is in H. But H is 2-edge-connected since a bridge of H extends to a 2-edge-separator of G since there are only three edges between H and G_i. This means that R is connected unless both deleted edges are in H. In this final case, H is either connected, and we are done, or it decomposes into two components K_1 and K_2. But since $G - \{uv, u'v'\}$ is connected, these components have neighbours in G_i, so they also remain connected in G'. □

5 | CONCLUSION AND FUTURE RESEARCH

We successfully automated the approach of working one's way through a path-decomposition to show that a set of graphs is unavoidable. The algorithm proved its use in practice. For example, it verified that $\{C_3, C_4, \ldots, C_7\}$ is unavoidable for cubic graphs of path-width at most 7. These computational results led us to new insights on $\xi(k)$. A further use of the algorithm can be found in [3, Lemma 29], where it is used to show that any cubic graph of path-width at most 4 contains one of the five specific graphs as a subgraph. This is used to prove the 3-decomposition conjecture for 3-connected cubic graphs of path-width at most 4.

The most obvious question is whether our framework can be generalised to provide an algorithm which verifies unavoidable sets for graph classes of bounded tree-width. A starting point would be to order the bags of a rooted smooth tree-decomposition such that bags of the highest distance to the root are considered first. Observe that also the highest-degree-first-concept transfers to tree-width. The expected problem with this approach is that the search space becomes too large.

Furthermore, it would be desirable to extend the list of girth-extremal graphs of path-width k.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (EngageS: grant agreement No. 820148). Open Access funding enabled and organized by Projekt DEAL.

REFERENCES

1. K. Appel and W. Haken, *Every planar map is four colorable. Part I: Discharging*, Illinois J. Math. **21** (1977), no. 3, 429–490. https://doi.org/10.1215/ijm/1256049011
2. K. Appel, W. Haken, and J. Koch, *Every planar map is four colorable. Part II: Reducibility*, Illinois J. Math. **21** (1977), no. 3, 491–567. https://doi.org/10.1215/ijm/1256049012
3. O. Bachtler and I. Heinrich, *Reductions for the 3-decomposition conjecture*. Version 2. 2022. arXiv: 2104.15113 [math.CO]. https://doi.org/10.48550/arXiv.2104.15113
4. H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoret. Comput. Sci. **209** (1998), no. 1, 1–45. https://doi.org/10.1016/S0304-3975(97)00228-4

5. H. L. Bodlaender, B. A. Burton, F. V. Fomin, and A. Grigoriev, *Knot diagrams of treewidth two*, Version 2. 2019. arXiv: 1904.03117 [cs.DS]. https://doi.org/10.48550/arXiv.1904.03117
6. H. L. Bodlaender, B. Burton, F. V. Fomin, and A. Grigoriev, *Knot diagrams of treewidth two*, Graph-Theoretic Concepts in Computer Science, 46th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2020, Lecture Notes in Computer Science (I. Adler, and H. Müller, eds.), **12301**, Springer, Cham, Switzerland, 2020, pp. 80–91. https://doi.org/10.1007/978-3-030-60440-0_7
7. H. L. Bodlaender and A. M. C. A. Koster, *Treewidth computations II. Lower bounds*, Inform. and Comput. **209** (2011), no. 7, 1103–1119. https://doi.org/10.1016/j.ic.2011.04.003
8. G. Brinkmann, K. Coolsaet, J. Goedgebeur, and H. Mélot, *House of graphs: A database of interesting graphs*, Discrete Appl. Math. **161** (2013), no. 1, 311–314. https://doi.org/10.1016/j.dam.2012.07.018
9. L. S. Chandran and C. R. Subramanian, *Girth and treewidth*, J. Combin. Theory Ser. B. **93** (2005), no. 1, 23–32. https://doi.org/10.1016/j.jctb.2004.05.004
10. M. Chudnovsky, R. Kim, S.-I. Oum, and P. Seymour, *Unavoidable induced subgraphs in large graphs with no homogeneous sets*, J. Combin. Theory Ser. B. **118** (2016), 1–12. https://doi.org/10.1016/j.jctb.2016.01.008
11. D. Conlon, J. Fox, and B. Sudakov, *Recent developments in graph Ramsey theory*, Surveys in Combinatorics, London Mathematical Society Lecture Note Series (A. Czumaj, et al, ed.), vol. **424**, Cambridge University Press, Cambridge, UK, 2015, pp. 49–118. https://doi.org/10.1017/CBO9781316106853.003
12. B. Courcelle, *The monadic second-order logic of graphs. I. Recognizable sets of finite graphs*, Inform. and Comput. **85** (1990), no. 1, 12–75. https://doi.org/10.1016/0890-5401(90)90043-H
13. D. W. Cranston and D. B. West, *An introduction to the discharging method via graph coloring*, Discrete Math. **340** (2017), no. 4, 766–793. https://doi.org/10.1016/j.disc.2016.11.022
14. R. Diestel, *Graph theory*, Fifth. Graduate Texts in Mathematics (S. Axler, and K. Ribet, eds.), vol. **173**, Springer, Heidelberg, 2016. https://doi.org/10.1007/978-3-662-53622-3
15. G. Exoo and R. Jajcay, *Dynamic cage survey*, Electron. J. Combin. (2012), Dynamic Survey no. **16**, 1–55. https://doi.org/10.37236/37
16. L. Feuilloley, *Introduction to local certification*, Discrete Math. Theoret. Comput. Sci. **23** (2021), no. 3. https://doi.org/10.46298/dmtcs.6280
17. M. Frick and M. Grohe, *The complexity of first-order and monadic second-order logic revisited*, Annals of Pure and Applied Logic (I. Moerdijk, ed.), 130.1. Papers presented at the 2002 IEEE Symposium on Logic in Computer Science (LICS), Elsevier, Amsterdam, Netherlands, 2004, pp. 3–31.https://doi.org/10.1016/j.apal.2004.01.007
18. E. Fuchs, L. Gellert, and I. Heinrich, *Cycle decompositions of pathwidth-6 graphs*, J. Graph Theory. **94** (2020), no. 2, 224–251. https://doi.org/10.1002/jgt.v94.2
19. M. Göös and J. Suomela, *Locally checkable proofs in distributed computing*, Theory Comput. **12** (2016), no. 19, 1–33. https://doi.org/10.4086/toc.2016.v012a019
20. A. Korman, S. Kutten, and D. Peleg, *Proof labeling schemes*, Distrib. Comput. **22** (2010), no. 4, 215–233. https://doi.org/10.1007/s00446-010-0095-3
21. E. L. Post, *A variant of a recursively unsolvable problem*, Bull. Amer. Math. Soc. **52** (1946), no. 4, 264–268. https://doi.org/10.1090/S0002-9904-1946-08555-9
22. N. Robertson and P. D. Seymour, *Graph minors. X.X. Wagner's conjecture*, J. Combin. Theory Ser. B. **92.2** (2004), 325–357. Special Issue Dedicated to Professor W.T. Tutte. https://doi.org/10.1016/j.jctb.2004.08.001
23. The Sage Developers, *SageMath, the Sage mathematics software system*, Version 9.0. 2020 https://www.sagemath.org
24. W. T. Tutte, *A family of cubical graphs*, Math. Proc. Cambridge Philos. Soc. **43** (1947), no. 4, 459–474. https://doi.org/10.1017/S0305004100023720

How to cite this article: O. Bachtler and I. Heinrich, *Automated testing and interactive construction of unavoidable sets for graph classes of small path-width*, J. Graph Theory. 2023;104:289–319. https://doi.org/10.1002/jgt.22964