Results on H-Freeness Testing in Graphs of Bounded r-Admissibility

Christine Awofeso
Birkbeck, University of London, UK

Patrick Greaves
Birkbeck, University of London, UK

Oded Lachish
Birkbeck, University of London, UK

Felix Reidl
Birkbeck, University of London, UK

Abstract

We study the property of *H-freeness* in graphs with known bounded average degree, i.e. the property of a graph not containing some graph H as a subgraph. H-freeness is one of the fundamental graph properties that has been studied in the property testing framework.

Levi [10] showed that triangle-freeness is testable in graphs of bounded *arboricity*, which is a superset of e.g. planar graphs or graphs of bounded degree. Complementing this result is a recent preprint [7] by Eden *et al.* which shows that, for every $r \geq 4$, C_r-freeness is not testable in graphs of bounded arboricity.

We proceed in this line of research by using the *r-admissibility* measure that originates from the field of structural sparse graph theory. Graphs of bounded 1-admissibility are identical to graphs of bounded arboricity, while graphs of bounded degree, planar graphs, graphs of bounded genus, and even graphs excluding a fixed graph as a (topological) minor have bounded r-admissibility for any value of r [12].

In this work we show that H-freeness is testable in graphs with bounded 2-admissibility for all graphs H of diameter 2. Furthermore, we show the testability of C_4-freeness in bounded 2-admissible graphs directly (with better query complexity) and extend this result to C_5-freeness. Using our techniques it is also possible to show that C_6-freeness and C_7-freeness are testable in graphs with bounded 3-admissibility. The formal proofs will appear in the journal version of this paper.

These positive results are supplemented with a lower bound showing that, for every $r \geq 4$, C_r-freeness is not testable for graphs of bounded $(\lfloor r/2 \rfloor - 1)$-admissibility. This lower bound will appear in the journal version of this paper. This implies that, for every $r > 0$, there exists a graph H of diameter $r+1$, such that H-freeness is not testable on graphs with bounded r-admissibility. These results lead us to the conjecture that, for every $r > 4$, and $t \leq 2r + 1$, C_t-freeness is testable in graphs of bounded r-admissibility, and for every $r > 2$, H-freeness for graphs H of diameter r is testable in graphs with bounded r-admissibility.

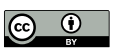
2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Property Testing, Sparse Graphs, Degeneracy, Admissibility

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.12

1 Introduction

A *graph property* is a class of graphs closed under isomorphisms. A property-tester for a property P is a probabilistic algorithm that receives as input the size of a graph G, a distance parameter $\epsilon > 0$ (among potentially other parameters), and oracle access to the graph G. The algorithm accepts with probability at least $2/3$ any input from P and rejects with probability at least $2/3$ an input that it is ϵ-far from the property P. The term "ϵ-far" is a notion of distance that depends on the exact problem setting and discuss it further



42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).
Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thắng;
Article No. 12; pp. 12:1–12:16

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

below. The complexity measure of a property-tester is a function that bounds above the total number of queries to the oracle the algorithm uses, as a function of the input parameters ϵ, size of the graph and any other input parameters provided. A property is *testable* if it has a property-tester whose query complexity is independent of the size of the input graph.

We study here the property of H-*freeness*, where H is a fixed known graph, i.e. the property of graphs that do not have a subgraph isomorphic to H, which is one of the fundamental graph properties that has been studied in the property testing framework. H-freeness was studied in the dense, sparse and general graph models. In the dense model it was shown implicitly in [1] that H-freeness is testable, for a more explicit result see [2]. Goldreich and Ron [8], showed that H-freeness is testable in the bounded degree graph model. In an effort to move towards larger sparse[1] graph classes, Czumaj *et al.* [4] showed that H-freeness is testable in sparse graphs when H is a tree. This result is most likely tight for sparse graphs as Alon *et al.* [3] showed that triangle-freeness is not testable in sparse graphs.

While this settles the question for the most general notion of sparse graphs, the question is still open for a plethora of sparse graph subclasses which are more structured. Possibly the most famous among them is the class of planar graphs. Czumaj *et al.* [5] showed that H-freeness is testable for this class. Proceeding in this line of research, and moving to a much larger class of sparse graphs, Levi [10] showed that triangle-freeness is testable in graphs of bounded *arboricity*, which is a superset of the family of planar graphs. In Eden *et al.* [7] it is shown that, for every $r \geq 4$, C_r-freeness is not testable in graphs of bounded arboricity. Specifically, it is shown that, in graphs of bounded arboricity, the query complexity of C_4-freeness and C_5-freeness is $\tilde{\Theta}(n^{1/4})$, the query complexity of C_6-freeness is $\tilde{O}(n^{1/2})$, and for every $k \geq 6$, the query complexity of C_6-freeness is $O(n^{1-1/\lfloor k/2 \rfloor})$ and $\Omega(n^{1/3})$. These results also leave open the question of which sparse classes – somewhere between bounded arboricity and planar graphs – and which values of r is C_r-freeness testable.

In order to identify a suitable notion of sparseness, we draw inspiration from the field of structural sparse graph theory and propose the r-*admissibility* measure of graphs as a parameter that governs the testability of H-freeness. 2-admissibility was originally defined in [9], where it was simply referred to as admissibility. The more general notion of r-admissibility[2] for natural values of r was first defined in [6]. Strictly speaking, r-admissibility is a family of measures where r governs how "deep" into the graph we look. We remark that, graph classes with bounded 1-admissibility are equivalent to graphs with bounded arboricity (both measures lie within a constant factor of each other). Many well-known sparse classes, like planar graphs, graphs of bounded genus, graphs excluding a fixed (topological) minor and graphs of bounded degree have bounded r-admissibility [12, 14, 6] meaning that the r-admissibility of any member of such a class can be bounded by a function which only depends on r and the class itself.

We show that C_4-freeness is testable in graphs with bounded 2-admissibility, and that H-freeness, for every H of diameter 2, is also testable in graphs with bounded 2-admissibility and in particular C_5-freeness is testable in graphs with bounded 2-admissibility. Using our techniques it is possible to show that C_6 and C_7 are testable in graphs with bounded 3-admissibility. This result will appear in the journal version of this paper that will also contain a lower bound which shows that, for every $r \geq 4$, C_r-freeness is not testable in

[1] Here sparse should be understood as having a linear number of edges or equivalently bounded average degree

[2] If you are interested in a more formal definition in this stage, we suggest that you proceed to Section 3, and read up to Proposition 3 before you proceed here.

graphs of bounded $(\lfloor r/2 \rfloor - 1)$-admissibility. This implies that for every $r > 0$, there exists a graph H of diameter $r + 1$ such that H-freeness is not testable on graphs with bounded r-admissibility. The above leads us to conjecture that, for every $r \in \mathcal{N}$ and $t \leq 2r + 1$, C_t-freeness is testable in graphs with bounded r-admissibility and furthermore for every $r > 2$, H-freeness, for H of diameter r, is testable in graphs with bounded r-admissibility.

Techniques

All the lower bounds use Yao's Minimax Principle [13] and the construction of suitable input instances. The graph constructions used for the lower bounds, unsurprisingly, are very similar to the ones used in [7] to prove a lower bound on testing C_4-freeness in bounded arboricity graphs. We chose to provide our lower bounds here and not refer to [7], to demonstrate how they apply to testing C_r-freeness of graphs of bounded $\lfloor r/2 \rfloor - 1$-admissibility, for natural values of r. This is not covered in [7]. In addition, the graphs provided in the proof also demonstrate that, for every natural value of r and large enough n, there are bounded r-admissibility graphs with n vertices some of which have degree $\Omega(\sqrt{n})$. It should be noted that these graphs are also "far" from being planar.

All upper bounds are based on the same algorithm and can be seen as a variation of the "standard" BFS (breadth first search) based testers for the bounded degree graph model. Many examples of "standard" BFS testers can be found in [8]. In such testers, initially a small subset of the vertices of the graph is selected uniformly at random (u.a.r.) and then a fixed depth BFS is performed (using oracle queries) from every vertex in the selected set. The tester presented here (Algorithm 1) differs from the "standard" testers at the BFS stage as follows: While the "standard" testers queries proceeds with the BFS by querying all neighbours of a vertex v, Algorithm 1 randomly selects size at most $\min\{\alpha, \deg_G(v)\}$ subset of $[\deg_G(v)]$, where α is a parameter provided to the algorithm and $\deg_G v$ is the degree of the vertex v in the input graph G, and queries the identity of the ith neighbour of the vertex, for every i in the selected set. We note Algorithm 1 behaves like the "standard" BFS based tester if the input graph has maximum degree α.

H-subgraph

Algorithm 1 has a one-sided error, i.e. it only rejects if it detects an H-subgraph (a subgraph that is isomorphic to H), therefore, to prove its correctness, it is sufficient to show that, given oracle access to a bounded admissibility graph that is ϵ-far from being H-free, Algorithm 1 rejects with high probability.

To prove the required rejection probability we show the following. Given an input graph G with bounded admissibility, we can remove edges from the graph in a process we refer to as *trimming* to obtain a new auxiliary graph $\tilde{G}$. We relate this graph $\tilde{G}$ to G in two ways: first we show that if G is far from H-freeness, then so is $\tilde{G}$. Then we show that if there exists an H-subgraph in $\tilde{G}$, then this subgraph includes a vertex with low degree, such that if Algorithm 1 starts its search from this vertex in G (not in $\tilde{G}$), then with high probability it will discover an H-subgraph (though not necessarily the one we just referred to). The "high probability" is proved to be sufficiently large by using the properties of the graph $\tilde{G}$. Note that the construction of $\tilde{G}$ is only a tool for this proof, it is not actually constructed by the testing algorithm.

Finally, we show that when $\tilde{G}$ is far from being H-free, then there are many such low degree vertices in G. This implies that with sufficiently high probability Algorithm 1 initially selects such a vertex.

2 Preliminaries

$\mathcal{N}$,$\mathcal{R}$,[k] We use $\mathcal{N}$ for the set of integer numbers, $\mathcal{R}$ for the set of real numbers, $\mathcal{R}^+$ for the set of positive real numbers. For an integer k, we use $[k]$ as a shorthand for the set $\{1, 2, \ldots, k\}$. In this paper, all graphs are simple. For a graph G we use $V(G)$ and $E(G)$ to refer to its vertex- and edge-set, respectively. For a vertex $u \in V(G)$ we use the notation $N_G(u)$ to denote the set of all of u's neighbours in G. We omit the subscript, when clear for context. The diameter of a graph is the maximum of the distance between u and v over all pairs of vertices u and v in $V(G)$.

Heavy We use the notation $\text{Heavy}_\alpha(G)$, where $\alpha \in \mathcal{N}$ to denote the set of vertices in $V(G)$ that have degree larger than α. We write Heavy_α when G is clear from context. In some places we will use the notation without initially stating the value of α, in those cases α is calculated further down when its concrete value is required.

xPy For sequences of vertices $x_1, x_2, \ldots, x_\ell$, in particular paths, we use notations like $x_1 P x_\ell$, x_1 or $x_1 P$ or $P x_\ell$ to indicate subpaths P that are part of the larger path. For example, the notation uPv for a path u, a, b, c, v would mean that P is the subpath a, b, c, whereas uP in the same context would mean that P is the subpaths a, b, c, v. All the paths in this paper are simple. Though paths here are undirected, we often treat them as directed by specifying a start and end vertex.

Property testing

We work only with properties of (p, r)-admissible graphs, which we formally define in the next section. At this stage it is enough to know that both p and r take integer values that are strictly positive and that a (p, r)-admissible graph of n vertices can have at most pn edges.

graph property, far, close A *graph property* (or simply *property* in this paper) is a class of graphs closed under isomorphism and we say that a graph has the property if it is contained in this class. The distance of a graph G from a property of (p, r)-admissible graphs is the minimum number of edges that must be removed/added to G in order to arrive at a (p, r)-admissible graph with the property. We say a graph is ϵ-far from a property of (p, r)-admissible graph, if the graph's distance to the property is at least ϵpn (an ϵ portion of the maximum number of edges possible in (p, r)-admissible graphs) and otherwise we say that it is ϵ-close to the property.

Property tester A property tester is a randomised algorithm that receives oracle access to a graph as part of its input. An oracle can answer the following queries for vertices $u, v \in V(G)$:

- the degree $\deg(v)$ of a vertex v (degree query),
- the ith neighbour of v in G (neighbour query),
- whether $\{u, v\}$ is an edge in G (adjacency query).

By combining these queries it can in particular reveal the whole neighbourhood of a vertex v using $1 + \deg(v)$ queries. The oracle returns the special symbol "$\perp$" when asked out of range neighbour queries, e.g. when asked to return the 10th neighbour of a vertex with less than 10 neighbours.

Formally, a *property tester* for a property P of (p, r)-admissible graphs, is a randomized algorithm $\mathcal{A}$ that receives as input parameters $n \in \mathcal{N}$, $p, r \in [n]$, $\epsilon > 0$ and oracle access to a (p, r)-admissible graph G with n vertices. If the graph G is ϵ-far from P, then $\mathcal{A}$ rejects with probability at least $2/3$. If the graph G is in P, then $\mathcal{A}$ accepts with probability 1 (if the tester is *one-sided*) or with probability at least $2/3$ (if the tester is *two-sided*). The

complexity measure of a property tester is a function depending on n, p, r, and ϵ which bounds the maximum number of queries the tester makes on an input graph with those parameters.

In this paper, we study the property of (p, r)-admissible graphs that are H-free. That is, graphs that are (p, r)-admissible and do not have any H-subgraph (a subgraph isomorphic to H). In some cases, H may be a specific graph, for example, H may be a C_i (cycle of length i), for some $i \geq 3$ and we then refer to the problem as *C_i-freeness*.

H-free, H-subgraph

In further sections, we use the term *knowledge graph* to refer to the graph that includes all the vertices, edges and anti-edges (learned from negative answers to neighbour queries) that the algorithm discovered via queries.

Knowledge graph

3 Graph degeneracy and admissibility, related notations and necessary lemmas

An *ordered graph* is a pair $\mathbb{G} = (G, \leq)$ where G is a graph and $\leq$ is a total order relation on $V(G)$. We write $\leq_{\mathbb{G}}$ to denote the ordering of $\mathbb{G}$ and extend this notation to derive relations $<_{\mathbb{G}}$, $>_{\mathbb{G}}$, $\geq_{\mathbb{G}}$. For simplicity we will call $\mathbb{G}$ an *ordering of* G and we write $\pi(G)$ for the set of all possible orderings of G.

$\mathbb{G}$, ordered graph, $\pi(G)$

We use the same notations for graphs and ordered graphs, additionally we write $N^-_{\mathbb{G}}(u) := \{v \in N(u) \mid v <_{\mathbb{G}} u\}$ for the *before neighbourhood* and $N^+_{\mathbb{G}}(u) := \{v \in N(u) \mid v >_{\mathbb{G}} u\}$ for the *after neighbourhood* of a vertex $u \in \mathbb{G}$. We omit the graphs in the subscripts if clear from the context. We further use $\deg^-_{\mathbb{G}}(u) := |N^-_{\mathbb{G}}(u)|$ and $\deg^+_{\mathbb{G}}(u) := |N^+_{\mathbb{G}}(u)|$, as well as $\Delta^-(\mathbb{G}) := \max_{u \in \mathbb{G}} \deg^-_{\mathbb{G}}(u)$ and $\Delta^+(\mathbb{G}) := \max_{u \in \mathbb{G}} \deg^+_{\mathbb{G}}(u)$. We omit subscripts if clear from the context.

before and after neighbourhood, $N^-_{\mathbb{G}}(u)$, $N^+_{\mathbb{G}}(u)$, $\Delta^+_{\mathbb{G}}(u)$, $\Delta^-(\mathbb{G})$

▶ **Definition 1** (Degeneracy). *A graph G is γ-degenerate if there exists an ordering $\mathbb{G}$ (a* degeneracy ordering*) such that $\Delta^-(\mathbb{G}) \leq \gamma$.*

Degeneracy

In particular, for every vertex v in a γ-degenerate graph G and every degeneracy ordering $\mathbb{G}$ of the graph it holds that $\deg^-_{\mathbb{G}}(v) \leq \gamma$. Consequently, the number of edges in a γ-degenerate graph is bounded by γn. A degeneracy ordering of a graph can be computed in time $O(n+m)$ and $O(\gamma n)$ for γ-degenerate graphs [11].

Admissibility

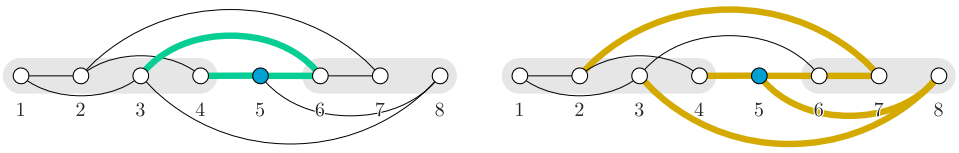


Figure 1 On the left, the green highlighted edges form an example of a maximal 2-admissible path packing of size 2 that is rooted at the blue vertex in position 5. On the right, the yellow highlighted edges form an example of a maximal 3-admissible path packing of size 3 that is rooted at the blue vertex in position 5. In both packings the only vertex common to all the paths is their root. Also in both packing every path's end vertex is smaller than the root and all the path's internal vertices are all larger than the root.

Let $\mathbb{G} = (G, \leq)$ and $v \in V(G)$. A path vPx is *r-admissible* in $\mathbb{G}$ if its length $\|vPx\| \leq r$, $x <_{\mathbb{G}} v$ and $\min P >_{\mathbb{G}} v$. That is, the path goes from v to x using only vertices w such that $w >_{\mathbb{G}} v$ and x satisfies $v >_{\mathbb{G}} x$.

r-admissible path

Target

For every integer $i > 0$ we let $\text{Target}^i_{\mathbb{G}}(v)$ be the set of all vertices in $u \in V(G)$ such that $u <_{\mathbb{G}} v$ and u is reachable from v via an *r-admissible* path vPu of length exactly i. We omit the subscript $\mathbb{G}$ when it is clear from context.

$(r, \mathbb{G})$-*admissible path packing*

An *r-admissible path packing* is a collection of paths $\{vP_ix_i\}_i$ with joint *root* v and the additional properties that every path vP_ix_i is r-admissible and the subpaths P_ix_i are all pairwise vertex-disjoint (*cf.* Figure 1). In particular, all endpoints $\{x_i\}_i$ are distinct. We write $\text{pp}^r_{\mathbb{G}}(v)$ to denote maximum size of any r-admissible path packing rooted at v.

$\text{pp}^r_{\mathbb{G}}$, $\text{adm}_r(\mathbb{G})$, $\text{adm}_r(G)$

▶ **Definition 2** (Admissibility). *The* r-admissibility *of an ordered graph* $\mathbb{G}$*, denoted* $\text{adm}_r(\mathbb{G})$ *and the admissibility of an unordered graph* G*, denoted* $\text{adm}_r(G)$ *are*[3]

$$\text{adm}_r(\mathbb{G}) \coloneqq \max_{v \in \mathbb{G}} \text{pp}^r_{\mathbb{G}}(v) \quad \textit{and} \quad \text{adm}_r(G) \coloneqq \min_{\mathbb{G} \in \pi(G)} \text{adm}_r(\mathbb{G}).$$

Admissibility ordering

If $\mathbb{G}$ is an ordering of G such that $\text{adm}_r(\mathbb{G}) = \text{adm}_r(G)$, then we call $\mathbb{G}$ an *admissibility ordering* of G. The 1-admissibility of a graph coincides with its degeneracy and therefore such orderings are easily computable in linear time. For $r \geq 2$ an optimal ordering can also be computed in linear time in n, albeit the machinery required is much more complicated, see [6].

(p, r)-*admissible*, adm_r-*bounded*

We say that a graph G is (p, r)-admissible if $\text{adm}_r(G) = p$. Note that, by definition, if a graph G is (p, r)-admissible it is also (p, r')-admissible for all $r' \leq r$. We call a graph class adm_r-*bounded* if all of its members are (p, r)-admissible for some finite value p.

The following is a well-known result in the field of sparse graphs, we replicate it here using our notation for completeness:

▶ **Proposition 3.** *Let* r *and* p *be natural numbers, and* $\mathbb{G} = (G, \leq)$ *such that* $\text{adm}_r(G) = p$*, then for every* $v \in V(G)$*, and* $h \in [r]$ *it holds that* $|\text{Target}^h_{\mathbb{G}}(v)| \leq p(p-1)^{h-1}$ *and in particular* $|N^-_{\mathbb{G}}(v)| \leq p$*.*

Proof. Let v be an arbitrary vertex in $V(G)$, and $h \in [r]$. Note first that, by construction, $N^-_{\mathbb{G}}(v) = \text{Target}^1_{\mathbb{G}}(v)$, and hence we only need to prove the bound on $|\text{Target}^h_{\mathbb{G}}(v)|$.

For every $u \in \text{Target}^h_{\mathbb{G}}(v)$, let vP_uu be an r-admissible path of length h; such a path exists by the definition of $\text{Target}^h_{\mathbb{G}}(v)$. Let W be a subgraph of G defined as follows: $V(W)$ includes exactly the vertex v, all the vertices in $\text{Target}^h_{\mathbb{G}}(v)$ and all the vertices in P_u, for every $u \in \text{Target}^h_{\mathbb{G}}(v)$; and $E(W)$ includes every edge of G participating in a path vP_uu for some $u \in \text{Target}^h_{\mathbb{G}}(v)$.

By construction, the set of vertices $\text{Target}^h_{\mathbb{G}}(v)$ is independent in W and for every $w \in \text{Target}^h_{\mathbb{G}}(v)$, $\deg_W(w) = 1$. Also by construction, the distance in W of v from any vertex in $\text{Target}^h_{\mathbb{G}}(v)$ is at most h. Hence, we can find a rooted tree T in W with the following properties: v is the root of T, the set $\text{Target}^h_{\mathbb{G}}(v)$ is the set of leaves of T and the depth of T is at most h.

We next show that the degree of every vertex in the tree is at most p. This implies that indeed $|\text{Target}^h_{\mathbb{G}}(v)| \leq p(p-1)^{h-1}$ and in particular $|N^-_{\mathbb{G}}(v)| \leq p$.

Let p_v be the degree of v in W. Since T is a tree, there are at most p_v edge disjoint paths from v to the leaves of T (the vertices in $\text{Target}^h_{\mathbb{G}}(v)$). These paths have length at most h, they share only the vertex v, and they correspond to r-admissible paths of $\mathbb{G}$. The last part holds since $u >_{\mathbb{G}} v$, for every internal vertex u (non-leaf or root vertex) of T and,

[3] Note that some authors choose to define the admissibility as $1 + \max_{v \in \mathbb{G}} \text{pp}^r_{\mathbb{G}}(v)$ as this matches with some other, related measures.

for every $w \in \text{Target}^h_{\mathbb{G}}(v)$ (the leaves of T), it holds that $w <_{\mathbb{G}} v$. Therefore, these paths are an r-admissible packing of $\mathbb{G}$, and hence their number $p_v \leq p$, and in particular the degree of v in T is at most p.

We now show that this applies also to every internal vertex y of T. We notice that it also holds that $w <_{\mathbb{G}} y$, for every $w \in \text{Target}^h_{\mathbb{G}}(v) \cup \{v\}$. However, it is not necessarily the case that $u <_{\mathbb{G}} y$ when u is an internal vertex of T. This is resolved by noticing that for any $x \in \text{Target}^h_{\mathbb{G}}(v)$ and path yPx in T, if P has a vertex that is smaller than y, then instead of taking the path yPx, we take its shortest subpath $yP'x'$ such that $x' <_{\mathbb{G}} y$. Now, the same reasoning we used for v implies that the degree of y in T is at most p. ◀

4 Upper bounds strategy and the testing algorithm

In this section we present Algorithm 1 which is used for all upper bounds presented. Algorithm 1 receives as an input a graph H, a set of parameters including the parameter p and oracle access to a graph G. We note that for each upper bound shown in this paper, the parameters provided to Algorithm 1, in addition to p, are dependent on the graph H.

Algorithm 1 The PBFS.

Input: $n \in \mathcal{N}$, $p \in [n]$, $\alpha, \tau \in \mathsf{R}$, fixed graph H and oracle access to a graph G

Set $S_0 = \emptyset$
Repeat $\lceil 4\alpha/(\epsilon p) \rceil$ **times**
  Add to S_0 an independently and u.a.r selected vertex from $V(G)$
for $i = 1, 2, \ldots, \tau$ **do**
  Set $S_i = \emptyset$
  for $v \in S_{i-1} \setminus \bigcup_{j=0}^{i-2} S_j$ **do**
    Query the degree of v
    Set $X = \emptyset$
    Repeat $\lceil 2\alpha \rceil$ **times**
      Add to X an independently and u.a.r selected integer k from $[\deg(v)]$
    if $\deg(v) \leq \lceil \alpha \rceil$ **then**
      Set $X = [\deg(v)]$
    for $k \in X$ **do**
      query the identity of the k'th neighbour of v and add the answer to S_i
if *the knowledge graph contains a H-subgraph* **then**
  Reject
else
  Accept

Algorithm 1 can be seen as a variation of the "standard" BFS (breadth first search) based testers for properties of bounded degree graphs. In such testers, initially a small subset of the graph's vertices is randomly selected and then a fixed depth BFS is performed (using oracle queries) from every vertex in the selected set.

Algorithm 1 differs from the "standard" testers, at the BFS stage: In the "standard" tester the search is expanded to all neighbours of a discovered vertex (until the fixed depth is reached). In contrast, Algorithm 1 only queries a subset of neighbours, specifically for

a vertex v it randomly selects a size at most $\min\{\deg(v), \lceil\alpha\rceil\}$ subset $X \subseteq [\deg_G(v)]$ and then it queries the identity of every ith neighbour for $i \in X$. We call this type of search *pseudo*-BFS and refer to it with the acronym *PBFS*.

PBFS

▶ **Lemma 4.** *On input $n \in \mathcal{N}$, $p \in [n]$, $\alpha, \tau \in \mathsf{R}$, $\epsilon > 0$, fixed graph H, the query complexity of Algorithm 1 is $O((\alpha/(\epsilon p))\alpha^\tau)$.*

Proof. The proof follows directly from the algorithm. ◀

Upper bound strategy

Algorithm 1 has a one-sided error, i.e. it only rejects if it discovers an H-subgraph. That is, once Algorithm 1 finished its last query, its knowledge graph has an H-subgraph). Therefore, to prove its correctness, it is sufficient to show that, with high constant probability, it rejects a bounded admissibility graph that is ϵ-far from being H-free.

To prove the required rejection probability we proceed as follows: given an input graph G with bounded admissibility, a new graph $\tilde{G}$ is constructed by initially setting $\tilde{G} = G$, and then removing edges from $\tilde{G}$, in a process we refer to as *trimming*. We call $\tilde{G}$ the auxiliary graph.

We relate th auxiliary graph $\tilde{G}$ to G in two ways: first we show that if G is far from H-freeness, then so is $\tilde{G}$. Then we show that if there exists an H-subgraph in $\tilde{G}$, then this subgraph includes a vertex with low degree, such that assuming Algorithm 1 starts its search from this vertex in G (not in $\tilde{G}$), then with high probability it will discover an H-subgraph of G (though not necessarily the one we just referred to). The "high probability" is proved to be sufficiently large by using the properties of the graph $\tilde{G}$. Finally, we show that when $\tilde{G}$ is far from being H-free, then there are many such low degree vertices in G. This implies that with sufficiently high probability Algorithm 1 initially selects such a vertex in the sample set S_0.

The "trimming" process is problem dependent. The simplest case is for C_4-freeness in adm_2-bounded graphs and we provide some intuition here before the full formal treatment in the next section. Suppose that $v \in \text{Heavy}_\alpha$ and $u \in \text{Target}^2_{\mathbb{G}}(v)$. Suppose also that the set X of their common neighbours, not including those in Heavy_α, is small relative to the degree of v in G. In the trimming process we then remove all edges that are incident to v as well as some vertex in X. We can then show that (i) we did not remove too many edges and (ii) if u and v participate in a C_4-subgraph (a subgraph isomorphic to a C_4) in $\tilde{G}$, then a large enough portion of the neighbours of v, are also neighbours of u and are not in Heavy_α. Therefore, if v is encountered in the first iteration of the external loop of Algorithm 1 (guaranteed to happen if S_0 has a neighbour of v that is not in Heavy_α), then with high probability the knowledge graph will include two edges vw_1 and vw_2, where w_1 and w_2 are not in Heavy_α and are common neighbours of both u and v. Since the PBFS continues with these common neighbour and they are both not in Heavy_α, the edges uw_1 and uw_2 will also appear in the knowledge graph. Thus, the knowledge graph contains a C_4-subgraph. Similar – but more involved – ideas work for H-freeness, when H has diameter 2 and for C_6-freeness and C_7-freeness.

It is important to note that proving the above properties (i) and (ii) hold relies on the graph G being adm_r-bounded (for some problem-dependent value of r), and we make extensive use of Proposition 3.

5 Testing C_4-freeness in adm_2-bounded graphs

In this section we fix some $\epsilon > 0$, an integer $p > 0$, $\alpha = 32p^2/\epsilon$. The input graph G is $(p, 2)$-admissible and we let $\mathbb{G}$ be an ordering of G with $\mathrm{adm}_2(\mathbb{G}) = p$.

Trimming. In the trimming procedure we construct $\tilde{G}$ from G. We begin with $\tilde{G} = G$ and then remove edges from $E(\tilde{G})$ as follows:

1. For every $v \in \mathrm{Heavy}_\alpha(G)$, and $u \in N^-(v)$, the edge uv is removed from $E(\tilde{G})$.
2. For every $v \in V(G)$ and $u \in \mathrm{Target}^2_{\mathbb{G}}(v)$, if $|N_{\tilde{G}}(u) \cap N_{\tilde{G}}(v)| \leq \deg_G(v)/(\alpha/2)$, then for every $w \in N_{\tilde{G}}(u) \cap N_{\tilde{G}}(v)$, the edge vw is removed from $E(\tilde{G})$.
3. The previous step is repeated until it does not result in the removal of any edges from $E(\tilde{G})$.

We first show that this trimming procedure preserves farness:

▶ **Lemma 5.** *If G is ϵ-far from being C_4-free, then $\tilde{G}$ is $\epsilon/2$-far from being C_4-free.*

Proof. Initially $E(\tilde{G}) = E(G)$ and then edges are removed from $E(\tilde{G})$ in Steps 1 and 2 of the trimming. We next show that, in Step 1 of the trimming, at most $\epsilon|E(G)|/4$ edges are removed from $E(\tilde{G})$, and that the same applies to Step 2 of the trimming. This implies the lemma.

In Step 1 of the trimming, for every $v \in \mathrm{Heavy}_\alpha(G)$ we remove $|N^-_G(v)|$ edges. Thus, the total number of edges removed in this step is at most $\sum_{v \in \mathrm{Heavy}_\alpha(G)} |N^-_G(u)|$. By Proposition 3, for every $u \in V(G)$, it holds that $|N^-_G(u)| \leq p$. Hence, the preceding sum is at most $\sum_{v \in \mathrm{Heavy}_\alpha(G)} p = p \cdot |\mathrm{Heavy}_\alpha(G)|$ and

$$p \cdot |\mathrm{Heavy}_\alpha(G)| = \frac{p \cdot |\mathrm{Heavy}_\alpha(G)|}{|E(G)|}|E(G)| \leq \frac{p \cdot |\mathrm{Heavy}_\alpha(G)|}{\alpha|\mathrm{Heavy}_\alpha(G)|/2}|E(G)| = \frac{2p}{\alpha}|E(G)| < \frac{\epsilon pn}{4},$$

where the first inequality follows since $|E(G)| \geq \alpha|\mathrm{Heavy}_\alpha(G)|/2$ (the sum of degrees is twice the number of edges), and the last equality holds because $\alpha = 32p^2/\epsilon$.

In Step 2 of the trimming, for every $v \in V(G)$ and $u \in \mathrm{Target}^2_{\mathbb{G}}(u)$ at most $\deg_G(v)/(\alpha/2)$ edges are removed. Thus, in this step, at most $\sum_{v \in V(G)} \sum_{u \in \mathrm{Target}^2_{\mathbb{G}}(v)} \deg_G(v)/(\alpha/2)$ edges are removed. By Proposition 3, for every $u \in V(G)$, it holds that $|\mathrm{Target}^2_{\mathbb{G}}(u)| < p^2$. Hence, the preceding sum is strictly less than $\sum_{v \in V(G)} p^2 \deg_G(v)/(\alpha/2) = (\epsilon/8) \sum_{v \in V(G)} \deg_G(v) \leq \epsilon pn/4$, where the first equality follows because $\alpha = 32p^2/\epsilon$. ◀

▶ **Lemma 6.** *Every C_4-subgraph C of $\tilde{G}$ that has more than one vertex from $\mathrm{Heavy}_\alpha(G)$, has exactly two vertices w_1 and w_2 from $\mathrm{Heavy}_\alpha(G)$, all the other two vertices of C are neighbours of both w_1 and w_2 and, there exists $i \in \{1, 2\}$, such that $|N_G(w_1) \cap N_G(w_2)| \geq \deg_G(w_i)/(\alpha/2)$.*

Proof. Let C be a C_4-subgraph of $\tilde{G}$ such that $|V(C) \cap \mathrm{Heavy}_\alpha(G)| > 1$ and w_1 and w_2 be two vertices in $V(C) \cap \mathrm{Heavy}_\alpha(G)$. Assume without loss of generality that $w_1 >_{\mathbb{G}} w_2$, and otherwise rename the vertices accordingly.

According to Step 1 of the trimming, $w_1w_2 \notin E(\tilde{G})$ and hence w_1 and w_2 are not adjacent in C. By the same reasoning w_1 and w_2 are the only vertices of $\mathrm{Heavy}_\alpha(G)$ in $V(C)$. We conclude that C consists of the two vertices w_1 and w_2 in $\mathrm{Heavy}_\alpha(G)$ and two vertices that are not in $\mathrm{Heavy}_\alpha(G)$ and are neighbours of both w_1 and w_2.

Let $v \in V(C) \setminus \mathrm{Heavy}_\alpha(G)$ and $Y = N_{\tilde{G}}(w_1) \cap N_{\tilde{G}}(w_2)$. By the same reasoning as before we may conclude that $v >_{\mathbb{G}} w_1$. Thus, as $w_1 >_{\mathbb{G}} w_2$, $w_2 \in \mathrm{Target}^2_{\mathbb{G}}(w_1)$. Consequently, by Step 1 of the trimming, $|Y| > \deg_G(w_1)/(\alpha/2)$, since otherwise $Y = \emptyset$ in contradiction to $v \in Y$

(because v is adjacent to both w_1 and w_2). As $\tilde{G}$ is a subgraph of G and $Y \cap \text{Heavy}_\alpha(G) = \emptyset$ (because of Step 1 of the trimming), we can conclude that also in the graph G it holds that $|(N_G(w_1) \cap N_G(w_2)) \setminus \text{Heavy}_\alpha(G)| > \deg_G(w_1)/(\alpha/2)$. ◀

The next two lemmas show that if the set S_0 selected by Algorithm 1 contains specific types of vertices from $V(G) \setminus \text{Heavy}_\alpha(G)$, then it rejects with sufficient probability to prove the main Theorem of this section.

▶ **Lemma 7.** *Suppose that Algorithm 1 is executed with oracle access to G and parameters p, $H = C_4$, $\alpha = 32p^2/\epsilon$, $\tau = 3$ and let S_0 be the set selected by the algorithm in the first step. Assume there exists a vertex $v \in S_0 \setminus \text{Heavy}_\alpha(G)$ such that v is in a C_4-subgraph C of $\tilde{G}$ and C satisfies $|V(C) \cap \text{Heavy}_\alpha(G)| \leq 1$. Then Algorithm 1 rejects with probability* 1.

Proof. Let v be as in the statement of the lemma. Since C is a C_4-subgraph that includes, together with v, three vertices from $V(C) \setminus \text{Heavy}_\alpha(G)$ in G, it follows that C includes a path P of length 2 that consists of v and two other vertices from $V(C) \setminus \text{Heavy}_\alpha(G)$.

Now, as Algorithm 1 will execute a PBFS of depth 3 (we set $\tau = 3$) it is guaranteed that the knowledge graph constructed by Algorithm 1 will include *all* vertices of P and *all* edges incident to these vertices. In particular, the knowledge graph eventually has C as a subgraph with probability 1 and the claim follows. ◀

▶ **Lemma 8.** *Suppose that Algorithm 1 is executed with oracle access to G and parameters, p, $H = C_4$, $\alpha = 32p^2/\epsilon$, $\tau = 3$. Let S_0 be the set selected by the algorithm in the first step. Assume there exists a vertex $v \in S_0 \setminus \text{Heavy}_\alpha(G)$ such that v is in a C_4-subgraph C of $\tilde{G}$ and C satisfies $|V(C) \cap \text{Heavy}_\alpha(G)| \geq 2$. Then Algorithm 1 rejects with probability at least* 5/6.

Proof. Let v be as in the statement of the lemma. By Lemma 6, C has exactly two vertices w_1 and w_2 from $\text{Heavy}_\alpha(G)$, the other two vertices of C are neighbours of both w_1 and w_2 and are not in $\text{Heavy}_\alpha(G)$. Since $\deg_G(v) < \alpha$, Algorithm 1 queries all of v's edges, in the first iteration of the PBFS. Thus after the first iteration of the PBFS the knowledge graph already contains the edges vw_1 and vw_2.

We show next that with with probability at least 5/6 Algorithm 1 queries a neighbour $u \neq v$ of w_1 that is not in $\text{Heavy}_\alpha(G)$ and is also a neighbour of w_2, so after the second iteration of the PBFS the knowledge graph also contains the edge uw_1. We note that this implies the lemma, because in the third iteration of the PBFS (this is the last iteration, since $\tau = 3$) Algorithm 1 will discover all the edges incident to u, since $\deg_G(u) \leq \alpha$, and in particular it will discover the edge uw_2, which implies that after the end of the PBFS from v the knowledge graph will contain a C_4-subgraph and hence Algorithm 1 will reject.

Let $Y = (N_G(w_1) \cap N_G(w_2)) \setminus \text{Heavy}_\alpha(G)$. According to Lemma 6, there exists $i \in 1, 2$ such that $|Y| \geq \deg_G(w_i)/(\alpha/2)$. Without loss of generality assume that $i = 1$ and otherwise rename w_1 and w_2 accordingly. Since $\deg_G(w_1) \geq \alpha$, we can conclude that $|Y| \geq 2$ and hence we can assume that at least 1/2 of the vertices in Y are not v. Thus, $|Y \setminus \{v\}| \geq \deg_G(w_i)/(2\alpha/2))$. Algorithm 1 selects a vertex from $N(w_1)$ independently and u.a.r. 2α times and hence the probability that none of them are from $|Y \setminus \{v\}|$ is at most $(1 - 1/\alpha)^{2\alpha} < e^{-2} < 1/6$. So with probability at least 5/6 Algorithm 1 finds a vertex u in $Y \setminus \{v\})$ and the claim follows. ◀

▶ **Theorem 9.** *Suppose that Algorithm 1 is executed with oracle access to G and parameters, p, $H = C_4$, $\alpha = 32p^2/\epsilon$, $\tau = 3$, then (i) if G is C_4-free, then Algorithm 1 accepts with probability* 1*; and (ii) if G is ϵ-far from being C_4-free, then Algorithm 1 rejects with probability at least* 2/3*. Algorithm 1 uses at most $O(p^7/\epsilon^5)$ queries.*

Proof. The query complexity of the algorithm follows from Lemma 4 and the values of α and τ.

If G is C_4-free, then the knowledge graph constructed by Algorithm 1 will not have a C_4-subgraph and hence Algorithm 1 accepts with probability 1. So assume in the sequel that G is ϵ-far from C_4-freeness.

Let U be the set of all vertices in $V(G) \setminus \text{Heavy}_\alpha(G)$ that participate in a C_4-subgraph of $\tilde{G}$. Since the degree of all vertices in $V(G) \setminus \text{Heavy}_\alpha(G)$ is less than α, if we remove every edge that is incident to a vertex from U, then the total number of edges we removed is bounded above by $\alpha|U|$ and the resulting graph does not have a C_4-subgraph. Now, by Lemma 5, $\tilde{G}$ is $\epsilon/2$-far from C_4-freeness it must be the case that $\alpha|U| \geq \epsilon np/2$. Consequently, $|U| \geq \epsilon np/(2\alpha)$.

Algorithm 1 selects $\lceil 4\alpha/(\epsilon p)\rceil$ vertices u.a.r. for the set S_0. The probability that none of them are from U is at most $(1-\epsilon p/(2\alpha))^{4\alpha/(\epsilon p)} < e^{-2} < 1/6$. So with probability at least $5/6$, the set S_0 includes a vertex $v \in V(G) \setminus \text{Heavy}_\alpha(G)$ that participates in a C_4-subgraph of $\tilde{G}$.

Let C be a C_4-subgraph of $\tilde{G}$ that includes a vertex $v \in V(G) \setminus \text{Heavy}_\alpha(G)$. One of two cases applies to C: Either (a) C includes at most one vertex from $\text{Heavy}_\alpha(G)$ or (b) C includes more than one vertex from $\text{Heavy}_\alpha(G)$. We now show that given $v \in S_0$, with probability at least $5/6$, the knowledge graph of Algorithm 1 eventually has a C_4-subgraph. By using the union bound we can conclude that Algorithm 1 rejects with probability at least $2/3$.

According to Lemma 7, if case (a) occurs, then Algorithm 1 rejects with probability 1. According to Lemma 8, if case (b) occurs, then Algorithm 1 rejects with probability at least $5/6$. We conclude that Algorithm 1 rejects with probability at least $(5/6)^2 > 2/3$, as claimed. ◀

6 Testing H-freeness in adm_2-bounded graphs when H has diameter 2

In this section we fix $\epsilon > 0$, $p \in \mathcal{N}$, $\alpha = 3|V(H)|\lceil \epsilon^{-1}2^{2|V(H)|+4p^2\log p}\rceil$. As before, G is an arbitrary $(p,2)$-admissible graph and $\mathbb{G}$ is an ordering of G with $\text{adm}_2(\mathbb{G}) = p$. Finally, H is an arbitrary diameter 2 graph.

The trimming process in this section depends on the structure of H and requires an extra construct (**H**). Let us first provide some intuition why this is required.

Suppose that $\tilde{H}$ is an H-subgraph of $\tilde{G}$ and h is the largest vertex in $V(\tilde{H}) \setminus \text{Heavy}_\alpha(G)$. We show below that the number of vertices like h in $\tilde{G}$ is large enough to ensure that with high enough probability one of them will be in the set S_0 selected by Algorithm 1.

Ideally, in $\tilde{H}$, every vertex $u \in V(\tilde{H})$, is reachable from h via vertices not from $\text{Heavy}_\alpha(G)$. This is an ideal case, because of the following. The depth of the PBFSs used is $|V(H)|$. Thus, as the PBFS queries all the neighbours of vertices $V(\tilde{H}) \setminus \text{Heavy}_\alpha(G)$, all these vertices will be discovered and also all edges incident to them. This means that the algorithm discovers all the edges of $\tilde{H}$ except those that are incident on two vertices from $V(\tilde{H}) \cap \text{Heavy}_\alpha(G)$. The trimming we use ensures that there are no edges between heavy vertices, so in particular, this latter case cannot occur and Algorithm 1 will discover $\tilde{H}$.

If we do not find ourselves in the ideal case, $\tilde{H}$ has a separator $\tilde{K}$ that consists only of vertices from $\text{Heavy}_\alpha(G)$. By similar reasoning to the ideal case, if h is included in the set S_0 that Algorithm 1 selects, it discovers all the vertices in $\tilde{H}$ that can be reached from h via vertices in $V(\tilde{H}) \setminus \text{Heavy}_\alpha(G)$, this includes all the vertices of $\tilde{K}$. Let us denote the subgraph found so far by $\tilde{H}_1$. The algorithm now still needs to find the remaining vertices of $\tilde{H}$ that are "behind" the separator, let us call denote this subgraph by $\tilde{H}_2$.

The problem here is that the vertices $\tilde{K}$ of on the boundary between $\tilde{H}_1$ and $\tilde{H}_2$ have a high degree, therefore the probability to discover $\tilde{H}_2$ may be arbitrarily small. We therefore design a trimming process that will ensure that there are many "useful" $\tilde{H}_2$-subgraphs $\tilde{H}'_2$ isomorphic to $\tilde{H}_2$ in $\tilde{G}$ that can serves to complete $\tilde{H}_1$ into a graph isomorphic to H. Specifically, a subgraph $\tilde{H}'_2$ is useful if $\tilde{K} \subset V(\tilde{H}'_2)$ and there exists an isomorphism from $\tilde{H}'_2$ to $\tilde{H}_2$ which maps every vertex of $\tilde{K}$ to itself.

Let Q' be a maximum family of vertex disjoint useful $\tilde{H}_2$-subgraphs with respect to the fixed graph $\tilde{H}_1$. If Q' is small then we can remove them all simply by removing all edges between members of Q' and $\tilde{K}$. If this is not possible, Q' must be sufficiently large and the PBFS can discover one of its members with sufficiently high probability.

The above is a simplification, since $\tilde{H}$ might disconnect into more than two components when removing the separator $\tilde{K}$. In this case we also need to ensure that the members of Q' are sufficiently disjoint.

Finally, in the preceding discussion we fixed a single $\tilde{H}$ with separator $\tilde{K}$, however $\tilde{H}$ is of course not known in advance. We therefore enumerate all possible sets $K \subset H$ that can take the role of $\tilde{K}$ (not that, by Proposition 3, we only need to look at sets of size $< p^2$). Since K consists only of heavy vertices and our trimming removes edges between heavy vertices, we can further assume K to be independent.

Kernel, $\mathbf{H}$

▶ **Definition 10** (Kernel and $\mathbf{H}$). *A* kernel *K of the graph H is an independent subset of $V(H)$ of size less than p^2 for which H has two subgraphs H_1 and H_2 such that*

1. *both H_1 and H_2 are induced and connected,*
2. *$V(H_1) \cap V(H_2) = K$,*
3. *$V(H_1) \cup V(H_2) = V(H)$,*
4. *every edge of H that is incident to a vertex from $V(H_1)$ and $V(H_2)$, is incident to an edge from K (K is a vertex separator in H),*
5. *the induced subgraph of H_1 on the vertices $V(H_1) \setminus K$ is connected.*

We define $\mathbf{H}$ to be the family of ordered pairs (H_2, K) over all kernels K of H.

▶ **Definition 11** (Sibling subgraphs). *Let L be a a subset of $V(\tilde{G})$, and R_1 and R_2 be two subgraphs of $\tilde{G}$, both including the set of vertices L. We say that R_1 and R_2 are* siblings *if $V(R_1) \cap V(R_2) = L$ and there exists an isomorphism ϕ from R_1 to R_2, such that, for every $\ell \in L$, $\phi(\ell) = \ell$.*

We say a set of graphs is a set of sibling graphs by L, if every pair of graphs in the set are siblings by L.

Trimming

Trimming. In the trimming procedure we construct $\tilde{G}$ from G. We begin with $G = \tilde{G}$ and then remove edges from $E(\tilde{G})$ as follows:

1. For every $v \in \text{Heavy}_\alpha(G)$, and $u \in N^-(v)$, the edge uv is removed from $E(\tilde{G})$.
2. For every $v \in \text{Heavy}_\alpha(G)$, $(H_2, K) \in \mathbf{H}$ and size $|K|$ subset $M \subseteq \text{Target}^2_{\mathbb{G}}(v)$, if a maximum set of vertex disjoint H_2-subgraphs of $\tilde{G}$ that are siblings by M, and are isomorphic to H_2 so that the vertices of M are mapped to the vertices of K, has size at most $2|V(H)| \deg_G(v)/\alpha$, then, for every vertex w in a subgraph of this set, we remove every edge incident to w and a vertex in $\text{Target}^2_{\mathbb{G}}(v)$.
3. The previous steps are repeated until it does not result in the removal of any edges from $E(\tilde{G})$.

▶ **Proposition 12.** *For every $v \in \text{Heavy}_\alpha(G)$, $(H_2, K) \in \mathbf{H}$ and size $|K|$ subset $M \subseteq \text{Target}^2_{\mathbb{G}}(v)$, if in some execution of Step 2, for a maximum family of vertex disjoint H_2-subgraphs that are siblings by M, and are isomorphic to H_2 so that the vertices of M are*

mapped to the vertices of K, the following holds: for every vertex w in a subgraph of this set, we remove every edge incident to w and a vertex in $\text{Target}^2_{\mathbb{G}}(v)$*, then after the specific execution of Step 2, there does not exist any H_2-subgraph that is a sibling by M of a subgraph in the maximum family.*

Proof. Recall that H has diameter 2, so if it has a separator K, then all the vertices of H that are not in the separator must be adjacent to some vertex in the separator. Therefore in Step 2, when for every w in a subgraph of the set all edges between w and $\text{Target}^2_{\mathbb{G}}(v)$ are removed, we have that every such vertex w cannot participate in any subgraph like the one in the set. Hence, as the family is of maximum size the proposition follows. ◀

▶ **Lemma 13.** *If G is ϵ-far from being H-free, then $\tilde{G}$ is $\epsilon/2$-far from being H-free.*

Proof. Note that initially $E(\tilde{G}) = E(G)$ and then edges are removed from $E(\tilde{G})$ in Steps (1) and (2) of the trimming. Given that the value of α here is larger than the value used in Section 5, and that Step 1 of the trimming here is the same as Step 1 in Section 5, by the same considerations as in Lemma 5, at most $\epsilon pn/4$ edges are removed from $E(\tilde{G})$, at Step 1 of the trimming. So, to complete the proof, we only need to show that in Step 1 of the trimming also at most $\epsilon pn/4$ edges are removed from $E(\tilde{G})$.

According to Step 2 of the trimming, the total number of edges removed from $E(\tilde{G})$, for every vertex in $\text{Heavy}_\alpha(G)$, is bounded above by the product of the following values:

- $|\mathbf{H}|$, and
- $(p^2)!$, which is an upper bound on the number of option to choose (while considering order) a size $|K|$ subset of $\text{Target}^2_{\mathbb{G}}(v)$ (where, for every $v \in V(G)$, by Proposition 3, $|\text{Target}^2_{\mathbb{G}}(v)| < p^2$), and
- $2|V(H)|\deg_G(v)/\alpha$, the threshold for edge removal for number of subgraphs in a maximum set of sibling subgraphs, and
- $p^2|V(H)|$, the number of edges incident to a vertex from $\text{Target}^2_{\mathbb{G}}(v)$ and vertices in a subgraph of a maximum set of sibling subgraphs.

The size of $\mathbf{H}$ is the number of subsets $V(H)$ with size less than p^2. Hence, $|\mathbf{H}| < 2^{|V(H)|}$. So, the total number of edges removed from $E(\tilde{G})$ is at most

$$\sum_{v \in \text{Heavy}_\alpha(G)} 2^{|V(H)|} \cdot (p^2)! \cdot (2|V(H)|\deg_G(v)/\alpha) \cdot p^2|V(H)| < \frac{\epsilon}{8} \sum_{v \in \text{Heavy}_\alpha(G)} \deg_G(v) \leq \frac{\epsilon pn}{4},$$

where the first equality follows because $\alpha = 3|V(H)|\lceil \epsilon^{-1} 2^{2|V(H)|+4p^2 \log p} \rceil$. ◀

▶ **Proposition 14.** *Let $\tilde{H}$ be a H-subgraph of $\tilde{G}$.* $\text{Heavy}_\alpha(G)$ *is an independent set in $\tilde{G}$, the largest vertex in $\tilde{H}$ is not in* $\text{Heavy}_\alpha(G)$ *and if $\tilde{H}$ has a separator $U \subseteq \text{Heavy}_\alpha(G)$, then $U = \text{Heavy}_\alpha(G) \cap V(\tilde{H})$ and U is the only separator in $\tilde{H}$ consisting of only vertices from* $\text{Heavy}_\alpha(G)$.

Proof. According to Step 1 of the trimming, for every $v \in \text{Heavy}_\alpha(G)$, if $vu \in E(\tilde{G})$, then $u >_{\mathbb{G}} v$. Thus, as one vertex is greater than the other for every pair of vertices in $\text{Heavy}_\alpha(G)$ there cannot be an edge in $\tilde{G}$ that is incident to both. Therefore, $\text{Heavy}_\alpha(G)$ is an independent set. By the same reasoning, the largest vertex in an H-subgraph of $\tilde{G}$ is not in $\text{Heavy}_\alpha(G)$, since it is adjacent to vertices smaller than it (H is connected, because it has diameter 2).

Finally, for the last part the claim, in the proof of Proposition 12, we noticed that for every separator of H, every vertex of H that is not in the separator must be adjacent to some vertex in the separator. The same applies to $\tilde{H}$. So, if a separator of $\tilde{H}$ is a subset

of $\text{Heavy}_\alpha(G)$, then every vertex that is adjacent to some vertex in the separator is not in $\text{Heavy}_\alpha(G)$ and in particular all the vertices of $\tilde{H}$ that are not in the separator. This also implies that U is the only separator in $\tilde{H}$ consisting of only vertices from $\text{Heavy}_\alpha(G)$. ◀

▶ **Lemma 15.** *Let $\tilde{H}$ be an H-subgraph of $\tilde{G}$ and $U = V(\tilde{H}) \cap \text{Heavy}_\alpha(G)$ and suppose that U is a separator of $\tilde{H}$, then $|U| < p^2$.*

Proof. Let v be the largest vertex in U. Since H is of diameter 2, it has a common neighbour with every vertex in U. By Step 1, v does not have any neighbours smaller than it and hence together with the previous we can conclude that $U \setminus \{v\} \subseteq \text{Target}^2_{\mathbb{G}}(u)$. By Proposition 3, $\text{Target}^2_{\mathbb{G}}(u) \leq p(p-1)$ and the lemma follows. ◀

▶ **Theorem 16.** *If Algorithm 1 is executed with oracle access to G and parameters, p, H, $\alpha = 3|V(H)|\lceil \epsilon^{-1} 2^{2|V(H)|+4p^2 \log p} \rceil$, $\tau = |H|$, then (i) if G is H-free, then Algorithm 1 accepts with probability 1; and (ii) if G is ϵ-far from being H-free, then Algorithm 1 rejects with probability at least $2/3$. Algorithm 1 uses at most $O(2^{(|V(H)|+1)(2|V(H)|+4p^2 \log p)})$ queries.*

Proof. The query complexity of the algorithm follows from Lemma 4 and the values of α and τ.

If G is H-free, then knowledge graph of Algorithm 1, will never have an H-subgraph and hence, Algorithm 1 accepts with probability 1. So, from here on in this proof, assume that G, is ϵ-far from H-freeness.

Let U be the set of all vertices in $V(G)$ that are the largest vertices in a H-subgraph of $\tilde{G}$. By Proposition 14, $U \cap \text{Heavy}_\alpha(G) = \emptyset$ and therefore, by the same reasoning as in Theorem 9, it holds that $|U| > \epsilon np/(2\alpha)$ and, with probability at least $5/6$ that $S_0 \cap U \neq \emptyset$. So, assume that v is a vertex in S_0 that is the largest vertex in a H-subgraph $\tilde{H}$ of $\tilde{G}$.

One of two cases applies to $\tilde{H}$: (a) every vertex $u \in V(\tilde{H})$, is reachable from v via the vertices in $V(\tilde{H}) \setminus \text{Heavy}_\alpha(G)$, and (b) there exists a set M such that $M = V(\tilde{H}) \cap \text{Heavy}_\alpha(G)$ that is a separator of $\tilde{H}$.

If case (a) applies, then as we already described previously in this section, with probability 1, the knowledge graph of Algorithm 1 will eventually have an H-subgraph. Thus, Algorithm 1 will reject with probability 1. So, from here on we assume that case (b) applies.

So, assume that $\tilde{H}$ has a separator M consisting only of vertices from $\text{Heavy}_\alpha(G)$ (by Proposition 14, this separator includes all the vertices of $\text{Heavy}_\alpha(G) \cap V(\tilde{H})$). Since $v \notin \text{Heavy}_\alpha(G)$, the diameter of H is 2 and $\text{Heavy}_\alpha(G)$ is an independent set, for every vertex in M, v is either its neighbour or shares a neighbour x with it, where $x \notin \text{Heavy}_\alpha(G)$. This ensures that, with probability 1, after two steps of the PBFS all the vertices in M are discovered. By similar consideration to the previous case, with probability 1, all vertices reachable from v via vertices not in $\text{Heavy}_\alpha(G)$ are added to the knowledge graph. For every one these vertices that is not in $\text{Heavy}_\alpha(G)$ also all the edges incident on them are also in the knowledge graph. This implies that all the edges between the vertices of M and the vertices added to the knowledge graph as described are also in the knowledge graph.

It remains to show that with sufficiently high probability, the edges required to complete the above to H-subgraph are included in the knowledge graph. Let ℓ be the largest vertex in M. Let $\tilde{H}'$ be an induced subgraph of $\tilde{H}$ that includes all the vertices of M and all other vertices of $\tilde{H}$ that are separated from v by M. By Step 2, of the trimming we are guaranteed that there exists a family $\mathcal{Q}$, of size greater than $2|V(H)| \deg_G(v)/\alpha$, that consists of vertex disjoint subgraphs of $\tilde{G}$, where every graph in the family is either $\tilde{H}'$ or its sibling by M.

Let be the graph $\tilde{H}^*$ induced by $\tilde{H}'$ on $V(\tilde{H}') \setminus M$ If we were guaranteed that $\tilde{H}^*$ is connected, then we would only need to show that with high probability Algorithm 1 will discover an edge incident to ℓ and a vertex x of this subgraph (or similar subgraph of one of

its siblings in M, that does not share any vertices with the part of the H-subgraph already discovered). This holds because, every vertex in $\tilde{H}^*$ is not in $\text{Heavy}_\alpha(G)$, and there are enough steps of the PBFS so that the PBFS reaches all the vertices in $\tilde{H}^*$ and discovers all the edges adjacent to them (the PBFS has $|V(H)|$ steps, and the vertices on a shortest path from ℓ to x are not in $\tilde{H}^*$), thus the PBFS will discover all the vertices of $\tilde{H}^*$ and the edges incident on them.

However, the above guarantee does not hold. So $\tilde{H}^*$ may contain almost $|V(H)|$ connected components. Regardless, if for every one of these connected components, the knowledge graph has it as a subgraph or has its isomorphic equivalent in one of the subgraphs in $\mathcal{Q}$, and if none of the isomorphic equivalent shares a vertex with other vertices of $\tilde{H}'$ in the knowledge graph, the knowledge graph has an H-subgraph.

For each connected component, there are at least $3|V(H)|\deg_G(\ell)/\alpha \geq 3|V(H)|\alpha/\alpha = 3|V(H)|$ vertex disjoint copies (where the inequality follows because $\ell \in \text{Heavy}_\alpha(G)$). So, at least two thirds of the copies of such connected component do not include any other vertices from $\tilde{H}$.

The probability of not discovering one such component is $(1 - 4|V(H)|/(3\alpha))^{2\alpha} \leq e^{-|V(H)|}/6 < (6|V(H)|)^{-1}$. By the union bound, with probability at least $5/6$, the knowledge graph has all the subgraphs required so that in has a H-subgraph. Thus, the proof is complete. ◀

7 Testing C_6 and C_7-freeness in adm_3-bounded graphs

The proof of the following theorem will appear in the journal version of this paper.

▶ **Theorem 17** (⋆). *If Algorithm 1 is executed with oracle access to G and parameters, $H = C_7$, $\alpha = \lceil 2^{12}p^4/\epsilon \rceil$, $\tau = 7$, then (i) if G is C_7-free, then Algorithm 1 accepts with probability 1; and (ii) if G is ϵ-far from being C_7-free, then Algorithm 1 rejects with probability at least $2/3$. Algorithm 1 uses at most $O(p^{27}/\epsilon^8)$ queries.*

8 Lower bounds for testing C_r-freeness for $r \geq 4$

The proof of the following theorem will appear in the journal version of this paper.

▶ **Theorem 18** (⋆). *For every integer $r \geq 4$ and sufficiently large integer n, every two-sided Property-Tester for the C_r-freeness, has query complexity $\Omega(n^{1/4})$, on $(2\lfloor r/2 \rfloor - 1)$ input graphs of size n.*

References

1 Noga Alon, Richard A. Duke, Hanno Lefmann, Vojtech Rödl, and Raphael Yuster. The algorithmic aspects of the regularity lemma. *J. Algorithms*, 16(1):80–109, 1994. doi:10.1006/JAGM.1994.1005.

2 Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Comb.*, 20(4):451–476, 2000. doi:10.1007/S004930070001.

3 Noga Alon, Tali Kaufman, Michael Krivelevich, and Dana Ron. Testing triangle-freeness in general graphs. *SIAM Journal on Discrete Mathematics*, 22(2):786–819, 2008. doi:10.1137/07067917X.

4 Artur Czumaj, Oded Goldreich, Dana Ron, C Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures & Algorithms*, 45(2):139–184, 2014. doi:10.1002/RSA.20462.

5 Artur Czumaj and Christian Sohler. A characterization of graph properties testable for general planar graphs with one-sided error (it's all about forbidden subgraphs). In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1525–1548. IEEE, 2019. doi:10.1109/FOCS.2019.00089.

6 Zdeněk Dvořák. Constant-factor approximation of the domination number in sparse graphs. *European Journal of Combinatorics*, 34(5):833–840, July 2013. doi:10.1016/j.ejc.2012.12.004.

7 Talya Eden, Reut Levi, and Dana Ron. Testing c_k-freeness in bounded-arboricity graphs. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 60:1–60:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.60.

8 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 406–415, 1997. doi:10.1145/258533.258627.

9 H. A. Kierstead and W. T. Trotter. Planar graph coloring with an uncooperative partner. *Journal of Graph Theory*, 18(6):569–584, October 1994. doi:10.1002/jgt.3190180605.

10 Reut Levi. Testing triangle freeness in the general model in graphs with arboricity $O(\sqrt{n})$. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 93:1–93:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.93.

11 David W Matula and Leland L Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, 1983. doi:10.1145/2402.322385.

12 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity: Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

13 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227. IEEE Computer Society, 1977.

14 Xuding Zhu. Colouring graphs with bounded generalized colouring number. *Discrete Mathematics*, 309(18):5562–5568, 2009. doi:10.1016/J.DISC.2008.03.024.