Distributed Complexity of P_k-Freeness: Decision and Certification

Masayuki Miyamoto ✉
University of Tsukuba, Japan

Abstract

The class of graphs that do not contain a path on k nodes as an induced subgraph (P_k-free graphs) has rich applications in the theory of graph algorithms. This paper explores the problem of deciding P_k-freeness from the viewpoint of distributed computing.

For specific small values of k, we present the *first* CONGEST algorithms specified for P_k-freeness, utilizing structural properties of P_k-free graphs in a novel way. Specifically, we show that P_k-freeness can be decided in $\tilde{O}(1)$ rounds for $k = 4$ in the broadcast CONGEST model, and in $\tilde{O}(n)$ rounds for $k = 5$ in the CONGEST model, where n is the number of nodes in the network and $\tilde{O}(\cdot)$ hides a polylog(n) factor. The main technical contribution is a novel technique used in our algorithm for P_5-freeness to distinguish induced 5-paths from non-induced ones, which is potentially applicable to other induced subgraphs. This technique also enables the construction of a local certification of P_5-freeness with certificates of size $\tilde{O}(n)$. This improves $\tilde{O}(n^{3/2})$ by Bousquet and Zeitoun (TCS 2025), and is nearly optimal, given our $\Omega(n^{1-o(1)})$ lower bound on certificate size.

For general k, we establish the first CONGEST lower bound, which is of the form $n^{2-1/\Theta(k)}$. The $n^{1/\Theta(k)}$ factor is unavoidable, in view of the $O(n^{2-2/(3k+2)})$ upper bound by Eden et al. (Dist. Comp. 2022). Additionally, our approach yields the *first* superlinear lower bound on certificate size for local certification. This partially answers the conjecture on the optimal certificate size of P_k-freeness, asked by Bousquet et al. (arXiv:2402.12148).

Finally, we propose a novel variant of the problem called ordered P_k detection. We show that in the CONGEST model, the round complexity of ordered P_k detection is $\tilde{\Omega}(n)$ for $k \geq 5$, and in contrast, proving any nontrivial lower bound for ordered P_3 detection implies a strong circuit lower bound. As a byproduct, we establish a circuit-complexity barrier for $\Omega(n^{1/2+\varepsilon})$ *quantum* CONGEST lower bounds for induced 4-cycle detection. This is complemented by our $\tilde{O}(n^{3/4})$ quantum upper bound, which surpasses the classical $\tilde{\Omega}(n)$ lower bound by Le Gall and Miyamoto (ISAAC 2021).

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases subgraph detection, CONGEST model, local certification

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2025.51

Related Version *Full Version*: https://arxiv.org/abs/2410.20353

Acknowledgements The author would like to thank François Le Gall for discussions and for commenting on an earlier version of this paper. The author also acknowledges the anonymous reviewers for their helpful comments.

1 Introduction

1.1 Background

The subgraph detection problems in the distributed setting of limited communication bandwidth, including the CONGEST model, are interesting and well studied problems. For a given specific small pattern graph H (the description of H is known to all nodes of the network), the goal of H detection is to decide if the network G contains H as a subgraph (or an induced subgraph). If the network G contains H, at least one node outputs "yes", otherwise all nodes of the network output "no". (H detection is often referred as H freeness, where at least one node outputs "no" if the network contains H, and otherwise all nodes

of the network output "yes". In this paper we abuse these two problems since they are essentially equivalent.) There are several variants previously considered in the literature: H listing requires to output all copies of H in G (each node outputs some copies of H and the union of all output lists are equivalent to the list of all copies of H in G). In recent years, significant progress in distributed subgraph detection has been made, and we now understand well the round complexities for detection of e.g., cliques and cycles. However, much less is known about induced paths. In the full version, we also summarize the previous results in this context.

Since it is known that in the CONGEST model, detecting non-induced k-paths can be done in $O(1)$ rounds for constant k [27], our focus in this paper is specifically on the induced k-path detection problem. Induced k-paths are particularly more important than non-induced paths because graphs that do not contain an induced k-path (often called P_k-free graphs) have numerous applications in algorithmic graph theory, particularly in the centralized setting. The absence of induced k-node paths imposes structural constraints on graphs, which can be leveraged to develop efficient algorithms for problems like independent set or coloring [5, 20, 22, 31, 23, 36]. Despite its importance, few results are known for P_k-freeness in the distributed setting. The only known result we are aware of is the result of Nikabadi and Korhonen [34], who explored the multicolored variant of the induced k-path detection, where each node is colored by an integer from $\{1, 2, \ldots, k\}$, and the goal is to detect an induced path on k different colors. They demonstrated that multicolored induced P_k detection requires $\Omega(n/\log n)$ rounds for any $k \geq 7$. However, their proof fails for P_k-freeness. To the best of our knowledge, there is no nontrivial result[1] for P_k-freeness in the CONGEST model beyond the general result by [14]: for any subgraph H on k nodes, there is a randomized CONGEST algorithm that solves induced H detection in $\tilde{O}(n^{2-\frac{2}{3k+1}})$ rounds.

Local certification of P_k-freeness

Another line of research focuses on local certification of P_k-freeness. In local certification, small labels called certificates are assigned to the nodes of the graph, and each node decides if the graph satisfies some property by using its local view. There are two important parameters, that is, the size of certificates and the locality of verification. In local certification of size s and locality ℓ, each node receives an $O(s)$-bit certificate from the external entity called the *prover*, and each node decides its binary output (*accept* or *reject*) only using the unique identifiers and certificates of its ℓ-hop neighbors. The scheme (with the locality $\ell = 1$) is introduced by [28] under the name of proof-labeling schemes, which is now a popular setting in the community of distributed computing:

▶ **Definition 1** (proof-labeling schemes). *A proof-labeling scheme (PLS_ℓ) of size s is a pair $(c, \mathcal{A})$ of functions c and $\mathcal{A}$ of the input graph $G = (V, E)$ such that*

- *for each node u, the function c assigns a bit string $c(u)$ of length s called the certificate of u;*
- *for each node u, the function $\mathcal{A}$, called the verification algorithm, depends on the certificates of ℓ-hop neighbors of u as well as the identifiers of these nodes, and decides the output of u. More precisely, let $N_\ell(u) = \{v_1, \ldots, v_d\}$ be the ℓ-hop neighbors of u, then*

$$\mathcal{A}(\mathsf{id}(u), c(u), \mathsf{id}(v_1), c(v_1), \ldots, \mathsf{id}(v_d), c(v_d))$$

is the output of u.

[1] For the trivial case of $k = 3$, P_3 detection can be done in $O(1)$ rounds.

Let $\mathcal{P}$ be a graph property. We say that there is a PLS_ℓ *that certifies $\mathcal{P}$ if there exists a verification algorithm $\mathcal{A}$ with locality ℓ (which outputs* Accept *or* Reject*) satisfies the following conditions.*

- *If G satisfies the property $\mathcal{P}$, there exists a certificate function c such that all nodes in G output* Accept*.*
- *If G does not satisfy the property $\mathcal{P}$, for any certificate function c, at least one node in G outputs* Reject*.*

When the locality of the verification algorithm is $\ell \geq \frac{k}{2}$, there is a trivial way to certify P_k-freeness: by assigning the list of IDs of neighbors as a certificate, each node try to find a P_k by looking edges adjacent to its $k/2$-hop neighbors. If there is a P_k, then its central node can now detect it. Thus, our interest lies in cases where the path length is long relative to the locality. Recently, Bousquet, Cook, Feuilloley, Pierron, and Zeitoun [2] studied this topic by analyzing the relationship among path length, certificate size, and locality of verification. They provided various nontrivial upper and lower bounds on certificate size (we discuss details in Section 1.2). While local certification of various subgraph-related problems has been studied (e.g., [3, 11, 18, 19]), the only known result for local certification of P_k-freeness prior to [2] was for $k = 4$: there is a PLS_1 that certifies P_4-freeness with $O(\log n)$ bits [17]. Moreover, [17] proved that every MSO_1 property Π can be certified with $O(\log^2 n)$ bits and locality 1 if all graphs satisfying Π have bounded clique-width. Since P_k-freeness can be expressed by MSO_1, this suggests that the difficulty gap between certifying P_4-freeness and P_5-freeness can be attributed to the fact that P_4-free graphs have bounded clique-width, whereas P_5-free graphs do not. After that, Bousquet and Zeitoun [4] constructed a PLS_1 that certifies P_5-freeness with $O(n^{3/2})$ bits.

1.2 Our results

1.2.1 Result 1: P_k-freeness for $k = 4$ and $k = 5$

We first show the following upper bounds for $k = 4$ and $k = 5$. Throughout the paper, we use "a randomized algorithm" as an algorithm that outputs the correct answer with probability at least $2/3$.

▶ **Theorem 2.** *There exists a randomized algorithm that solves P_4-freeness in the* broadcast CONGEST *model, running in $O\left(\frac{\log n}{\log \log n}\right)$ rounds.*

▶ **Theorem 3.** *There exists a randomized algorithm that solves P_5-freeness in the* CONGEST *model, running in $O\left(n \log^2 n\right)$ rounds.*

Technical challenges

As mentioned above, there were no nontrivial algorithms for P_k-freeness for $k \geq 4$ in the CONGEST model beyond the general $O(n^{2-2/(3k+2)})$ upper bound [14]. Moreover, the previous CONGEST algorithms for subgraph detection have been mostly focused on non-induced cases. Most techniques from these results seem to be useless for induced subgraph detecion. Typical examples are the *expander decomposition* applicable to cliques [8, 9], and the *BFS search with threshold* applicable to (non-induced) cycles [6, 18]. We thus need to exploit structural properties of P_4/P_5-free graphs, resulting our algorithms quite different from the other algorithms for subgraph detection. We believe that our approach hints new algorithms for other induced subgraphs.

We turn our attention to the framework of local certification. We get the following certification scheme for P_5-freeness.

▶ **Theorem 4.** *There is a* PLS_1 *that certifies* P_5*-freeness with certificates of size* $O(n \log n)$*.*

To obtain this result, we directly use the algorithm of Theorem 3. This is somewhat interesting, as CONGEST algorithms for subgraph freeness cannot be used for local certification of subgraph freeness in general (unless they are broadcast CONGEST algorithms). We also show that the size of our certificates is optimal, up to a subpolynomial factor:

▶ **Theorem 5.** *For any* $k \geq 4\ell + 1$*, any* PLS_ℓ *for* P_k*-freeness requires certificates of size* $\Omega\left(\frac{n}{e^{O(\sqrt{\log n})}}\right)$*.*

This result is proved via a combination of several known reduction techniques: we first reduce the nondeterministic three-party communication complexity of set-disjointness function to triangle freeness, which is then reduced to P_5-freeness.

Independent and concurrent works

After submitting the first draft of this paper, we learned about the following independent and concurrent works: Bousquet and Zeitoun [4] proved that there is a PLS_1 that certifies P_5-freeness with certificates of size $O(n^{3/2})$. Additionally, as noted in [4], Chaniotis, Cook, Hajebi, and Spirkl (independently and concurrently) obtained $\Omega(n^{1-o(1)})$ lower bound for P_5-freeness.

1.2.2 Result 2: P_k-freeness for general k

We provide the following lower bounds for P_k-freeness in the (randomized) CONGEST model using the two-party communication complexity, which is the standard framework for distributed subgraph detection problems.

▶ **Theorem 6.** *Let* d *be any positive integer.*

- *For* $d \leq 2$*,* P_k*-freeness for* $k \geq 11d$ *require* $\tilde{\Omega}(n^{2-1/d})$ *rounds in the randomized* CONGEST *model.*
- *For* $d \geq 3$*,* P_k*-freeness for* $k \geq 8d$ *requires* $\tilde{\Omega}(n^{2-1/d})$ *rounds in the randomized* CONGEST *model.*

The $n^{1/d} = n^{1/\Theta(k)}$ factor is unavoidable, in light of the upper bound of $n^{2-\Theta(1/k)}$ established by [14]. We will show these lower bounds using the standard reduction from two-party communication complexity. Although the framework used here is classic, there are several challenges that makes the proof highly non-trivial, which we discuss in detail later in Section 5.2.

Additionally, our constructions apply to the nondeterministic two-party communication setting, leading to the following.

▶ **Theorem 7.** *Let* ℓ *and* d *be positive integers.*

- *For* $d \leq 2$*, any* PLS_ℓ *for* P_k*-freeness requires certificates of size* $\tilde{\Omega}(n^{2-1/d})$ *for* $k \geq 4d\ell + 7d$*.*
- *For* $d \geq 3$*, any* PLS_ℓ *for* P_k*-freeness requires certificates of size* $\tilde{\Omega}(n^{2-1/d})$ *for* $k \geq 4d\ell + 4d$*.*

Comparison with the recent results by Bousquet et al. [2]

Let us now compare our results with those of Ref. [2], who considered local certification of P_k-freeness and obtained the following:

Table 1 Summary of our results and previous results on local certification of P_k-freeness. Here n denotes the number of nodes in the network.

Path length	Certificate size	Model	Reference
5	$O(n \log n)$	PLS_1	Thm 4
5	$O(n^{3/2})$	PLS_1	[4]
$4\ell + 1$	$\Omega(n^{1-o(1)})$	PLS_ℓ	Thm 5
$8\ell + 14$	$\tilde{\Omega}(n^{3/2})$	PLS_ℓ	Thm 7
$4d\ell + 4d$, $d \geq 3$	$\tilde{\Omega}(n^{2-1/d})$	PLS_ℓ	Thm 7
$3\ell - 1$	$O(n \log^3 n)$	PLS_ℓ	[2]
$4\ell + 3$	$\Omega(n)$	PLS_ℓ	[2]
$\lceil \frac{14\ell}{3} \rceil - 1$	$O(n^{3/2} \log^2 n)$	PLS_ℓ	[2]

▶ **Theorem 8** ([2]). *If the locality is $\ell \geq 1$, then:*

- *P_k-freeness for $k \leq 3\ell - 1$ can be certified with certificates of size $O(n \log^3 n)$;*
- *P_k-freeness for $k \leq \lceil \frac{14\ell}{3} \rceil - 1$ can be certified with certificates of size $O(n^{3/2} \log^2 n)$;*
- *P_k-freeness for $k \geq 4\ell + 3$ requires certificates of size $\Omega(n/\ell)$.*

They also conjectured the following.

▶ **Conjecture 9** ([2]). *For all $\alpha > 0$, the optimal size for local certification of $P_{\alpha\ell}$-free graphs with locality ℓ is of the form $\Theta(n^{2-1/f(\alpha)})$, for some unbounded increasing function f.*

We summarize our results and previous results in Table 1. For $\ell = 1$, the results of Bousquet et al. [2] provide nontrivial lower bound for P_7-freeness, and Bousquet and Zeitoun [4] provide nontrivial upper bound for P_5-freeness. In Theorems 4 and 5, we show the nearly optimal bound of P_5-freeness for $\ell = 1$.

For general k, Theorem 7 shows the first superlinear lower bounds, improving the previous linear lower bounds. Moreover, Theorem 7 partially answers Conjecture 9: f is at least linearly increasing.

1.2.3 Result 3: Ordered path detection and applications

We then define and study the following problem called ordered P_k detection.

▶ **Definition 10** (ordered P_k detection). *Each node of the graph has a color from $\{1, \ldots, k\}$, and the goal is to detect an induced path that consists of edges $\{(p_i, p_{i+1})\}_{i \in \{1,\ldots,k-1\}}$ on k nodes $\{p_i\}_{i \in \{1,\ldots,k\}}$ where p_i is colored by i.*

Motivation

The definition of ordered path detection may seem somewhat artificial; however, it can be motivated in a manner similar to that of multicolored path detection studied in [34]. Algorithms for the multicolored/ordered variants of these problems with color-coding techniques [1] are often used to address the standard version. For example, the state-of-the-art CONGEST algorithm for detecting $2k$-node cycles [6, 16] actually detects the ordered variant of $2k$-cycles. Consequently, lower bounds for the ordered variant reflect the limitations of algorithms that employ color-coding. Moreover, as demonstrated in Theorem 12, we will show that ordered path detection also provides some nontrivial insight for subgraph detection in the quantum CONGEST model, which is not restricted to algorithms using color-coding.

Contribution

We first prove the following lower bound for $k = 5$.

▶ **Theorem 11.** *Any randomized algorithm that solves ordered P_k detection for $k \geq 5$ requires $\tilde{\Omega}(n)$ rounds in the* CONGEST *model.*

We then focus on ordered P_3 detection. Our next finding is that, similarly to triangle detection, proving non-trivial lower bounds for ordered P_3 detection is difficult. This result also shows a circuit complexity barrier for induced C_4 detection.

▶ **Theorem 12** (Informal)**.** *For any constant $\varepsilon > 0$, proving any lower bound of the form $\Omega(n^{\varepsilon})$ for ordered 3-path detection in the* CONGEST *model or $\Omega(n^{1/2+\varepsilon})$ for induced C_4 detection in the quantum* CONGEST *model implies super-linear lower bounds on circuit complexity for an explicit family of boolean functions.*

Note that our circuit complexity barrier for induced C_4 detection is in the *quantum* CONGEST model [30]. We then complement this result by showing a nontrivial upper bound for induced C_4.

▶ **Theorem 13.** *In the quantum* CONGEST *model, induced C_4 detection can be solved in $O(n^{3/4})$ rounds with high probability.*

Previously, no nontrivial upper bound for induced C_4 detection was known, and our bound for induced C_4 detection beats a classical $\tilde{\Omega}(n)$ lower bound [29].

2 Preliminaries

We write $[n] = \{1, 2, \ldots, n\}$. We let $N(v)$ be the set of neighbors of v. For a graph $G = (V(G), E(G))$, we say that $H = (V(H), E(H))$ is a subgraph of G if there is an injective function $\phi : V(H) \to V(G)$ such that $(u, v) \in E(H) \Rightarrow (\phi(u), \phi(v)) \in E(G)$ for any pair of nodes $u, v \in V(H)$. We say that H is an induced subgraph of G if there is an injective function $\phi : V(H) \to V(G)$ such that $(u, v) \in E(H) \Leftrightarrow (\phi(u), \phi(v)) \in E(G)$ for any pair of nodes $u, v \in V(H)$. Let P_k be a path on k nodes. We say that G is P_k-free if G does not contain P_k as an induced subgraph. Throughout the paper, we assume that the input graph is connected, otherwise we can treat each connected component separately.

The CONGEST model and variants

In the CONGEST model [35], each node of the network $G = (V, E)$ has a distinct $O(\log n)$-bit identifier and can communicate with its neighbors in a synchronized manner. In each round each node (1) does some local computation and (2) sends an $O(\log n)$-bit message to each of its neighbors. In the initial state, each node only knows its own unique identifier (represented by $O(\log n)$ bits) and input, and the unique identifiers of its neighbors. The broadcast CONGEST model is a weaker model in the sense that each node can only broadcast a $O(\log n)$-bit message in each round. In the congested clique model [32], we allow all-to-all communication in each round. That is, the communication topology is always the n-node clique, and the input graph G is a subgraph of the clique. In the quantum CONGEST model [30], each node can send $O(\log n)$-qubit quantum message to each of its neighbors, instead of $O(\log n)$-bit classical message.

3 Algorithm for P_4-freeness

In this section we prove Theorem 2.

It is well known that a graph is P_4-free iff it is a cograph [10]. A cograph is defined as a graph that is constructed using the following rules:

- A single-node graph K_1 is a cograph;
- For cographs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, its disjoint union $(V_G \uplus V_H, E_G \uplus E_H)$ is a cograph;
- For cographs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, its join $(V_G \uplus V_H, E_G \uplus E_H \uplus (V_G \times V_H))$ is a cograph;

Using this characterization, we can construct a low-congestion spanning tree for any cograph as follows. A similar property on the spanning tree of cographs is also used in distributed (interactive) proofs for P_4-freeness [17, 33]. We use the following results on the balls into bins problem.

▶ **Lemma 14** (Balls into bins problem [21]). *Assume that there are n bins and n balls. Each ball uniform randomly selects one bin which it places into. Then, with probability at least $1 - 1/n$, all bins have at most $O\left(\frac{\log n}{\log \log n}\right)$ balls.*

▶ **Lemma 15.** *There exists a* broadcast CONGEST *algorithm that runs in $O(1)$ rounds and performs the following:*

- *If some node rejects, then G is not a connected cograph.*
- *Otherwise, with high probability, it outputs a rooted spanning tree of depth 2, where each node, except the root, has at most $O\left(\frac{\log n}{\log \log n}\right)$ children.*

Proof. We first reject if the diameter of the graph is not $O(1)$, since every connected cograph has diameter at most 2. This can be accomplished in $O(1)$ rounds by running a BFS search from the node with minimum identifier as the depth of the BFS tree provides a 2-approximation of the diameter. From the definition of cographs, a connected cograph $G = (V, E)$ is constructed from the join operation on two distinct node sets V_1 and V_2 where $V = V_1 \uplus V_2$ and $|V_1| \geq |V_2|$. Let r be the node with the maximum degree in G. Note that r can be found in $O(1)$ rounds since the diameter of the graph is constant. If $\deg(r) < n/2$, the algorithm rejects (if G is a connected cograph, then the maximum degree is at least $n/2$ as each node in V_2 has degree at least $|V_1| \geq n/2$). Consider the tree rooted at r where all neighbors of r are at depth 1. As below, we can construct the desired spanning tree with high probability if the graph is a connected cograph.

- Case 1: $r \in V_2$. Define $V_2' = V_2 \backslash (N(r) \cup \{r\})$. Observe that $V_1 \times V_2' \subseteq E$ and $|V_2'| \leq |V_1|$. Each $v \in V_2'$ uniform randomly selects one node from its neighbors as its parent in the tree. Now, each depth-1 node u (bin) is selected as the parent of v (ball) with probability at most $1/|V_2'|$ as $\deg(v) \geq |V_1|$. Then from Lemma 14, all depth-1 nodes have at most $O\left(\frac{\log n}{\log \log n}\right)$ children with high probability.
- Case 2: $r \in V_1$. Define $V_1' = V_1 \backslash (N(r) \cup \{r\})$. Since $\deg(r) \geq |V_1|$, we have

  $$|V_1'| = |V_1| - (\deg(r) - |V_2|) - 1 \leq |V_2|.$$

  Each node v in V_1' selects one of its neighbors as its parent uniform randomly. Now, each depth-1 node u (bin) is selected as the parent of v (ball) with probability at most $1/|V_2|$ as $\deg(v) \geq |V_2|$. Then from Lemma 14, all depth-1 nodes have at most $O\left(\frac{\log n}{\log \log n}\right)$ children with high probability. ◀

Proof of Theorem 2. Ref. [25] demonstrated a protocol in the randomized multiparty simultaneous messages model (also known as the distributed sketching model), where each node sends a message of size $O(\log n)$ to a referee. The referee can then determine if G is a cograph with high probability. We can simulate this protocol in the constructed depth-2 spanning tree, with the root r acting as the referee. Since each depth-1 node has at most $O\left(\frac{\log n}{\log\log n}\right)$ children, sending all messages to r is completed in $O\left(\frac{\log n}{\log\log n}\right)$ rounds. ◀

4 Algorithm and certification for P_5-freeness

In this section we give an overview of the proof strategy for Theorem 3 and Theorem 4. Since these theorems share the same proof strategy, we focus on our CONGEST algorithm for P_5-freeness (Theorem 3). Full proof of Theorem 4 will be found in the full version.

Our algorithm aims to efficiently detect whether a given graph is P_5-free, focusing on cases where the graph has a diameter of 2 or 3 (observe that if the diameter is at least 4, then it must not be P_5-free). We treat these cases separately due to their distinct properties.

4.1 Subgraph freeness in graphs with diameter two

Here we assume that the graph $G = (V, E)$ with n nodes and m edges has diameter 2. Let $r \in V$ be the node with maximum degree. Then $\deg(r) = \Delta = \Omega(m/n)$. Divide the node set $V = \{r\} \cup V_1 \cup V_2$ where V_i is the set of nodes whose distance to r is i. We first consider that every node u broadcasts the list of its neighbors in $O(n)$ rounds. After that r learns all the edges connected to $\{r\} \cup V_1$. In the remaining part we show that, with high probability, r can learn the edge set $E \cap (V_2 \times V_2)$ in another $O(n)$ rounds. Therefore, r can locally decide if the graph is P_5 free.

Assume that we divide V_2 into $\sqrt{m/n}$ subsets V_2^i for $i \in [\sqrt{m/n}]$ as follows (assuming $\sqrt{m/n}$ is an integer): each node $v \in V_2$ chooses an integer $i \in [\sqrt{m/n}]$ uniformly at random and joins V_2^i. Next, we label the set V_1 as $\{v_k | k \in [\Delta]\}$ where $\mathsf{ID}(v_j)$ is the k-th smallest among IDs of nodes in V_1. The partition of $V_2 = \{V_2^i\}$ and the label of V_1 nodes with respect to the order of IDs are informed to all the nodes in the network in $O(n)$ rounds. We use the following lemma.

▶ **Lemma 16** ([8]). *Consider a graph with $\bar{m}$ edges and $\bar{n}$ vertices. We generate a subset S by letting each vertex join S independently with probability p. Suppose that the maximum degree is $\Delta \leq \bar{m}p/20\log\bar{n}$ and $p^2\bar{m} \geq 400\log^2\bar{n}$. Then, with probability $1 - 1/n^c$ for sufficiently large c, the number of edges in the subgraph induced by S is at most $6p^2\bar{m}$.*

Here we set $\bar{n} = n$, $\bar{m} = m$, $p = \sqrt{n/m}$. Clearly, $p^2\bar{m} \geq 400\log^2\bar{n}$ holds. As $\Delta < n$, one can verify that $\Delta \leq \bar{m}p/20\log\bar{n}$ holds if $\bar{m} = m \geq 400n\log^2 n$. We assume that $m \geq 400n\log^2 n$ holds since otherwise a simple $O(m) = \tilde{O}(n)$-round algorithm suffices. Then, with high probability, $|E \cap V_2^i \times V_2^j| = O(p^2 m) = O(n)$ for all $i, j \in [\sqrt{m/n}]$. Consider the node $v_k \in V_1$ collects the edge set $E \cap V_2^i \times V_2^j$ for $k = i + (j-1)\sqrt{m/n}$. To do this, assume that $(u, v) \in E \cap V_2^i \times V_2^j$. Then there exists a node w such that $(u, w), (w, v_k) \in E$ since the diameter of the graph is 2. w knows the edge (u, v) because of the previous $O(n)$ round communication and thus w can send it to v_j. Since $|E \cap V_2^i \times V_2^j| = O(n)$, this is done in $O(n)$ rounds. Finally, v_k sends $E \cap V_2^i \times V_2^j$ to r in $O(n)$ rounds. Now we have established the following theorem.

▶ **Theorem 17.** *In graphs with diameter 2, for any subgraph H, (non-induced and induced) H-freeness can be solved in $O(n \log^2 n)$ rounds in the* CONGEST *model with high probability.*

It is noteworthy that this result does not hold when the diameter is greater than 2, since for any constant $\varepsilon > 0$, there is a pattern graph H such that H-freeness in graphs with diameter 3 requires $\Omega(n^{2-\varepsilon})$ rounds [15, 29].

4.2 P_5-freeness in graphs with diameter three

A High-Level Overview of Our Approach

In the case of a graph with diameter 3, the larger diameter necessitates more tricky definitions and case analysis, resulting in a more involved proof. We outline the main ideas of our proof here before giving details.

Let V_i be the set of nodes at distance i from r, for $i \in \{1, 2, 3\}$. The edges (except the ones incident to r) are partitioned as $E_{i,j} = E \cap (V_i \times V_j)$ for $i \leq j$. We employ a procedure $\mathcal{P}_{\mathsf{collect}}$ (Algorithm 1) that runs in $O(n)$ rounds, enabling r to gather $E_{1,1}$ and $E_{1,2}$.

Algorithm 1 $\mathcal{P}_{\mathsf{collect}}$.

r: the node with maximum degree.
T: the rooted spanning tree created by the BFS traversal from r.
Step 1 Each node broadcasts the list of its neighbors in $O(n)$ rounds.
Step 2 Consider the partition of V into $\sqrt{m/n}$ subsets V^i for $i \in [\sqrt{m/n}]$ as follows: Each node $v \in V$ chooses an integer $i \in [\sqrt{m/n}]$ uniformly at random, joins V^i and tells it to all other nodes so that all nodes in the graph learn the partition $\mathcal{V} = \{V^i\}_{i \in [\sqrt{m/n}]}$. The node r broadcasts the set of nodes that contains all nodes in V_1 in ascending order with respect to their IDs so that all nodes in the graph agree with $v_k \in V_1$ such that $\mathsf{ID}(v_k)$ is the k-th smallest ID in V_1.
Step 3 Define $\tilde{E}_{2,2}$ by the set of edges $(u, v) \in E_{2,2}$ satisfying at least one of the following:
1. $N(v_k) \cap (N(u) \cup N(v)) \neq \emptyset$ where $u \in V^i$ and $v \in V^j$ for the partition $\mathcal{V}$, and v_k is a neighbor of r such that the integer k is assigned in Step 2 for $k = i + (j-1)\sqrt{m/n}$,
2. $(N(u) \cup N(v)) \cap V_3 \neq \emptyset$.

We also define F_{bad} by the set of edges $(u, v) \in E_{2,2} \backslash \tilde{E}_{2,2}$ satisfying $N(u) \cap V_1 = N(v) \cap V_1$. Each node u tells the neighbors the list of its incident edges in $\tilde{E}_{2,2}$ and F_{bad}.
Step 4 For each edge (v_k, w), w sends the edge (u, v) between V^i and V^j to v_k if w received (u, v) in Step 1. After that, each $v_k \in V_1$ sends these edges between V^i and V^j to r. This takes $O(n)$ rounds since the number of edges between V^i and V^j is $O(n)$ w.h.p., due to Lemma 16.
Step 5 r computes $|E_{2,3}|, |\tilde{E}_{2,2}|, |F_{bad}|$. This is done by using the tree T. r then rejects if the following holds:
Condition 1 There is an edge in $E_{2,3} \cup \tilde{E}_{2,2}$ that is not sent to r.
For each edge $(u, v) \in E_{2,2} \backslash \tilde{E}_{2,2}$, u rejects if the following holds:
Condition 2 $N(u) \cap V_1 \neq N(v) \cap V_1$.
For each edge $(u, v) \in F_{bad}$, u rejects if the following holds:
Condition 3 There is a node $w \in N(u) \cap V_2$ such that $N(u) \cap V_1 \nsubseteq N(w)$.
For each edge $(u, v) \in E_{3,3}$, u rejects if the following holds:
Condition 4 $N(u) \cap V_2 \neq N(v) \cap V_2$.

For this procedure, we have the following.

▶ **Lemma 18** (Restated in Lemma 22). *If any of the conditions in $\mathcal{P}_{\mathsf{collect}}$ (Conditions 1-4) are met, then G contains an induced P_5.*

After the execusion of $\mathcal{P}_{\mathsf{collect}}$, if r detects any missing edge in $(E_{2,2}\backslash F_{bad}) \cup E_{2,3}$ – where F_{bad} is a carefully defined subset of $E_{2,2}$ (defined in Step 3 of $\mathcal{P}_{\mathsf{collect}}$) – it can safely conclude that the graph is not P_5-free. This is because (i) the invalidity of Condition 2 of $\mathcal{P}_{\mathsf{collect}}$ ensures that $E_{2,2}\backslash F_{bad} = \tilde{E}_{2,2}$, and (ii) the invalidity of Condition 1 of $\mathcal{P}_{\mathsf{collect}}$ ensures that all edges in $\tilde{E}_{2,2} \cup E_{2,3} = (E_{2,2}\backslash F_{bad}) \cup E_{2,3}$ have been sent to r. Therefore, if no node rejects in $\mathcal{P}_{\mathsf{collect}}$, r learns $E\backslash(F_{bad} \cup E_{3,3})$.

The following lemmas shows that, for any P_5 containing an edge from $F_{bad} \cup E_{3,3}$, detection is performed by nodes connected to the endpoints of these edges.

▶ **Lemma 19** (Restated in Lemma 23). *Assume that no node rejects in $\mathcal{P}_{\mathsf{collect}}$. If G contains an induced P_5 with at least one edge from F_{bad}, then there is a node that detects an induced P_5.*

▶ **Lemma 20** (Restated in Lemma 24). *Assume that no node rejects in $\mathcal{P}_{\mathsf{collect}}$. If G contains an induced P_5 with at least one edge from $E_{3,3}$, then there is a node that detects an induced P_5.*

For any P_5 composed solely of edges from $E\backslash(F_{bad} \cup E_{3,3})$, detection is performed by node r, which now has access to $E\backslash(F_{bad} \cup E_{3,3})$. The key challenge here is distinguishing between proper P_5s (induced by the entire edge set E) and improper P_5s (induced by $E\backslash(F_{bad} \cup E_{3,3})$ but not by E). We address this by having r count the improper patterns and subtract this count from the total number of P_5s it finds. As illustrated in Figure 1, there are 17 non-isomorphic patterns for possible improper P_5s. We introduce the concept of a "dangerous" induced subgraph. An induced copy of $H \in \mathcal{H}$ in the input network $G = (V, E)$ is *dangerous* iff it induces a P_5 in $E\backslash(F_{bad} \cup E_{3,3})$. The following lemma shows that the number of such dangerous patterns can be counted efficiently.

▶ **Lemma 21** (Restated in Lemma 25). *Let $\mathcal{H} = \{H_i\}_{i\in[17]}$ be a set of five-node graphs that contain P_5 as a (non-necessarily induced) subgraph as in Figure 1. Then, for any $H \in \mathcal{H}$, the number of induced copies of H in the graph that are dangerous can be counted (by the node r) in $O(n)$ rounds.*

This yields the correct count of proper P_5s, since the number of P_5s in G is now obtained by subtracting the number of dangerous patterns in G from the number of P_5s in $(V, E\backslash(F_{bad} \cup E_{3,3}))$.

4.2.1 Algorithm for P_5-freeness in graphs with diameter three

Assume that the diameter of the input graph (with n nodes and m edges) is 3 and $m \geq 400n\log^2 n$. Similarly to the previous case, we divide the node set into $V = \{r\} \cup V_1 \cup V_2 \cup V_3$ where V_i is the set of nodes whose distance to r is i. For all $i, j \in \{1, 2, 3\}$, $i \leq j$, we also define $E_{i,j}$ by the edge set between V_i and V_j. We use a subprotocol $\mathcal{P}_{\mathsf{collect}}$ described in Algorithm 1, and analyze it here.

▶ **Lemma 22.** *If any of the conditions in $\mathcal{P}_{\mathsf{collect}}$ (Conditions 1-4) are met, then G contains an induced P_5.*

Proof.

Condition 1 Assume that there is an edge in $E_{2,3}$ that is not sent to r in $\mathcal{P}_{\mathsf{collect}}$. Then for some i, j, k satisfying $k = i + (j-1)\sqrt{m/n}$, there exists an edge (u, v) between V^i and V^j such that no node in $N(u) \cup N(v)$ is connected to v_k. W.l.o.g., we assume that $u \in V_2$ and $v \in V_3$. Then there exists a node $w \in N(u) \cap V_1$ that is not connected to $v_k \in V_1$. $\{v_k, r, w, u, v\}$ induces a P_5.

Assume that there is an edge in $\tilde{E}_{2,2}$ that is not sent to r in $\mathcal{P}_{\mathsf{collect}}$. First, observe that if an edge (u,v) in $\tilde{E}_{2,2}$ satisfies the first condition of $\tilde{E}_{2,2}$, it is sent to r in $\mathcal{P}_{\mathsf{collect}}$. Assume that (u,v) satisfies the second condition of $\tilde{E}_{2,2}$ and is not sent to r in $\mathcal{P}_{\mathsf{collect}}$. Then w.l.o.g., there are two nodes $w \in N(u) \cap V_3$ and $x \in N(u) \cap V_1$. Furthermore, assume that $u \in V^i$, $v \in V^j$, and $k = i + (j-1)\sqrt{m/n}$. $\{v_k, r, x, u, w\}$ induces a P_5.

Condition 2 Suppose that, for some i, j, k satisfying $k = i + (j-1)\sqrt{m/n}$, there exists an edge $(u,v) \in E_{2,2} \backslash \tilde{E}_{2,2}$ between V^i and V^j satisfying $N(u) \cap V_1 \backslash N(v) \cap V_1 \neq \emptyset$. Then there exist a node $w \in V_1 \cap (N(u) \backslash N(v))$ that is not connected to $v_k \in V_1$. We can conclude that $\{v_k, r, w, u, v\}$ is an induced P_5.

Condition 3 Let $(u,v) \in F_{bad} \cap (V^i \times V^j)$. For a node $w \in (N(u) \cup N(v)) \cap V_2$ such that $N(u) \cap V_1 \nsubseteq N(w)$, if there is $y \in (N(u) \cap V_1) \backslash N(w)$, then $\{v_k, r, y, u, w\}$ is an induced P_5 for $k = i + (j-1)\sqrt{m/n}$.

Condition 4 Let $w \in (N(u) \backslash N(v)) \cap V_2$. Then there exists a node $x \in N(w) \cup V_1$. $\{r, x, w, u, v\}$ is a P_5. ◀

Now, assuming that $\mathcal{P}_{\mathsf{collect}}$ does not reject, the following properties hold for each edge $(u,v) \in F_{bad}$ between V^i and V^j.

Property 1 $N(u) \cap V_1 = N(v) \cap V_1$,

Property 2 $N(v_k) \cap (N(u) \cup N(v)) = \emptyset$ for $k = i + (j-1)\sqrt{m/n}$,

Property 3 $(N(u) \cup N(v)) \cap V_3 = \emptyset$,

Property 4 for each $w \in (N(u) \cup N(v)) \cap V_2$, $N(u) \cap V_1 = N(v) \cap V_1 \subseteq N(w)$.

▶ **Lemma 23.** *Assume that no node rejects in $\mathcal{P}_{\mathsf{collect}}$. If G contains an induced P_5 with at least one edge from F_{bad}, then there is a node that detects an induced P_5.*

Proof. We write $\{(p_i, p_{i+1})\}_{i \in \{1,2,3,4\}}$ be the four edges constructing some induced P_5. Here we assume that exactly one edge of them is bad. First, consider that $(p_2, p_3) \in F_{bad}$. Fix one node $v \in N(p_2) \cap N(p_3) \cap V_1$. Then from Property 1 and Property 3, p_4 is in V_2. Moreover, $p_4 \in N(v)$ from Property 4. Similarly, $p_1 \in V_2 \cap N(v)$. Now observe that v knows $\{(p_i, p_{i+1})\}_{i \in \{1,2,3,4\}}$ and v can detect the path.

We then consider that $(p_1, p_2) \in F_{bad}$. Similarly to the above case, we can assume that $p_3 \in V_2$. We distinguish the following cases.

Case 1: $p_4 \in V_2$. If $p_5 \in V_1$, p_3 can detect the path since p_3 knows $p_5 \notin N(p_2)$ and thus $p_5 \notin N(p_1)$ due to Property 1. If $p_5 \in V_3$, p_3 can detect the path due to Property 3. If $p_5 \in V_2$, for each node $v \in N(p_1) \cap V_1$, $v \in N(p_2) \cap N(p_3)$. If $(p_4, v) \in E$ or $(p_5, v) \in E$, v can detect the path. Otherwise $\{p_1, v, p_3, p_4, p_5\}$ is an induced P_5. p_3 can detect this path as follows. Since p_3 knows $(p_1, p_2) \in F_{bad}$ and $(v, P_5) \notin E$, p_3 can conclude that $(p_1, p_5) \notin E$ due to Property 4.

Case 2: $p_4 \in V_1$. If $p_5 \in V_1$, then p_3 can detect the path due to Property 1. Assume $p_5 \in V_2$. Fix arbitrary node $v \in N(p_1) \cap N(p_2) \cap N(p_3) \cap V_1$. If $p_5 \in N(v)$, then v can detect the path. Otherwise, p_3 knows that $p_5 \notin N(v)$ and $(p_2, p_5) \notin E$. p_3 can detect the path since p_3 can verify that $(p_1, p_5) \notin E$ due to Property 4.

Case 3: $p_4 \in V_3$. If $p_5 \in V_3$, then p_3 can detect the path since p_1, p_2 do not have a neighbor in V_3. Assume $p_5 \in V_2$. Fix arbitrary node $v \in N(p_1) \cap N(p_2) \cap N(p_3)$. If $p_5 \in N(v)$, then v can detect the path. Otherwise, p_3 knows that $p_5 \notin N(v)$. From Property 4, we have $p_5 \notin N(p_1) \cup N(p_2)$. p_3 can detect the path.

Therefore, any induced P_5 that contains exactly one edge from F_{bad} can be detected by some node.

Now we consider a path $\{(p_i, p_{i+1})\}_{i\in\{1,2,3,4\}}$ containing at least two edges from F_{bad}. If there are two consecutive bad edges in the path, $(p_i, p_{i+1}), (p_{i+1}, p_{i+2})$, there is a node $u \in V_1 \cap N(p_i) \cap N(p_{i+1}) \cap N(p_{i+2})$ from Property 1. u is also connected to p_{i+3} when $i \leq 2$ or p_{i-1} when $i \geq 2$. u can detect the path. The remaining case is the path containing two bad edges that are not consecutive. We have two distinct cases.

Case 4: $(p_1, p_2), (p_3, p_4) \in F_{bad}$. From Property 1 and Property 4, there is a node $u \in N(p_1) \cap N(p_2) \cap N(p_3) \cap N(p_4)$ who can detect the path.

Case 5: $(p_1, p_2), (p_4, p_5) \in F_{bad}$. This is similar to a subcase of Case 1. For each node $v \in N(p_1) \cap V_1$, $v \in N(p_2) \cap N(p_3)$. If $(p_4, v) \in E$ or $(p_5, v) \in E$, v can detect the path. Otherwise $\{p_1, v, p_3, p_4, p_5\}$ is an induced P_5. p_3 can detect this path as follows. Since p_3 knows $(p_1, p_2) \in F_{bad}$ and $(v, P_5) \notin E$, p_3 can conclude that $(p_1, p_5) \notin E$ due to Property 4. ◀

▶ **Lemma 24.** *Assume that no node rejects in $\mathcal{P}_{\mathsf{collect}}$. If G contains an induced P_5 with at least one edge from $E_{3,3}$, then there is a node that detects an induced P_5.*

Proof. We write $\{(p_i, p_{i+1})\}_{i\in\{1,2,3,4\}}$ be the four edges constructing some induced P_5. Consider that $(p_2, p_3) \in E_{3,3}$ for instance. $p_1, p_4 \in V_3$ since Condition 4 of $\mathcal{P}_{\mathsf{collect}}$ does not hold. Similarly, since $(p_3, p_4) \in E_{3,3}$ we have $p_5 \in V_3$. Therefore there exists $v \in N(p_1) \cap N(p_2) \cap N(p_3) \cap N(p_4) \cap N(p_5) \cap V_2$ that can detect the path. The analysis of the other cases is done in the same way – we can show that all path edges are from $E_{3,3}$. ◀

Given Lemmas 23 and 24, we can detect any induced P_5 that involves edges from $F_{bad} \cup E_{3,3}$.

Our focus now shifts to the detection of an induced P_5 solely composed of edges from $E \backslash (F_{bad} \cup E_{3,3})$. In this context, we need to ensure that any P_5 induced by $E \backslash (F_{bad} \cup E_{3,3})$ does not actually arise as an artifact of the removal of $F_{bad} \cup E_{3,3}$, but is genuinely induced by E. To address this, we consider all five-node induced subgraphs of G. As illustrated in Figure 1, there are exactly 17 distinct five-node patterns $\mathcal{H} = \{H_1, \ldots, H_{17}\}$, each containing a P_5 as a subgraph. See Figure 2 for the illustration.

We introduce the concept of a "dangerous" induced subgraph. An induced copy of $H \in \mathcal{H}$ in the input network $G = (V, E)$ is *dangerous* iff it induces a P_5 in $E \backslash (F_{bad} \cup E_{3,3})$.

▶ **Lemma 25.** *Let $\mathcal{H} = \{H_i\}_{i\in[17]}$ be a set of five-node graphs that contain P_5 as a (non-necessarily induced) subgraph as in Figure 1. Then, for any $H \in \mathcal{H}$, the number of induced copies of H in the graph that are dangerous can be counted (by the node r) in $O(n)$ rounds.*

Proof. We first note that there is no dangerous copy of $H \in \mathcal{H}$ including edges from both F_{bad} and $E_{3,3}$, since two endpoints of a bad edge do not have neighbors in V_3 due to Property 3 of F_{bad}.

We consider the case of H_1 in Figure 1, i.e., 5-node cycles C_5. Let

$$\{(c_1, c_2), (c_2, c_3), (c_3, c_4), (c_4, c_5), (c_5, c_1)\}$$

be five edges that form a dangerous C_5 where $(c_5, c_1) \in F_{bad}$ and $(c_i, c_{i+1}) \in E \setminus (F_{bad} \cup E_{3,3})$ for $i \in \{1, 2, 3, 4\}$. Then, from Property 1 and Property 3 of F_{bad}, we can assume that $c_2, c_4 \in V_2$. Moreover, from Property 4 all the nodes in $N(c_1) \cap V_1 = N(c_5) \cap V_1$ are neighbors of c_2 and c_4. Let u be the node in $N(c_1) \cap V_1 = N(c_5) \cap V_1$ with minimum identifier. Then u can detect the cycle as it knows all the neighbors of c_1, c_2, c_4, c_5. Each node counts the number of such cycles, and sends it to r. In this way, r can count the number of dangerous C_5 in the graph. Note that it is impossible to have $(c_5, c_1) \in E_{3,3}$ due to the fact that Condition 4 of $\mathcal{P}_{\mathsf{collect}}$ does not hold: If $(c_5, c_1) \in E_{3,3}$, then $c_2, c_4 \in V_3$. It contradicts the assumption that $(c_1, c_2) \notin E_{3,3}$.

For any other pattern graph from $\{H_2, \ldots, H_{17}\}$ illustrated in Figure 1, we can see that at least one node of the pattern is connected to at least three other nodes of the pattern who can detect the pattern. This completes the proof. ◀

Proof of Theorem 3. In $O(n)$ rounds, we can compute the diameter of the graph [24]. If the diameter exceeds 3, the network immediately rejects. If the diameter is two, we apply Theorem 17, which establishes that P_5-freeness can be verified in $O(n \log^2 n)$ rounds with high probability. So we focus on the case where the diameter is exactly 3.

First, if the number of edges in the graph satisfies $m \leq 400n \log^2 n$, the node r collects all edges in $O(m)$ rounds and decide if G is P_5-free. Otherwise, the network runs $\mathcal{P}_{\mathsf{collect}}$ in $O(n)$ rounds. Assuming $\mathcal{P}_{\mathsf{collect}}$ does not reject, r knows all the edges in $E \backslash (F_{bad} \cup E_{3,3})$ as $E_{2,2} \backslash \tilde{E}_{2,2} = F_{bad}$.

Next, we assume that there is no induced P_5 that contains at least one edge from $F_{bad} \cup E_{3,3}$ as such a P_5 can be detected by some node due to Lemma 23 and Lemma 24.

For each $H_i \in \mathcal{H}$, the node r counts the number of induced copies of H_i that are dangerous as in Lemma 25. Let $t(H_i)$ be the count and let $t = \sum_{i \in [17]} t(H_i)$ be the sum of them.

Finally, r counts the number of P_5's induced by $E \setminus (F_{bad} \cup E_{3,3})$. If it is equal to t, we can conclude that G is P_5-free as removing $F_{bad} \cup E_{3,3}$ from G increases the number of induced P_5's by t. Otherwise, if the count is strictly larger than t, G is not P_5-free. ◀

4.3 Local certification of P_5-freeness

We use the following standard technique.

▶ **Lemma 26.** *Let $G = (V, E)$ be an n-node connected graph, and T be a spanning tree rooted at arbitrary node t. For each $u \in V$, let $a(u) \in [\mathrm{poly}(n)]$ be an integer assigned to u. Then, there is a PLS_1 that computes $\sum_{u \in V} a(u)$ with certificates of size $O(\log n)$.*

Proof. Let T_u be a subtree of T rooted at u. The certificate to u consists of two $O(\log n)$-bit values b and $b(u)$. If u is a leaf, u rejects when $a(u) \neq b(u)$. If u is not a leaf, u rejects when $b(u) \neq a(u) + \sum_{v \in N(u) \cap T_u} b(v)$. The root node t rejects when $b \neq b(t)$. If no node rejects, $b = \sum_{u \in V} a(u)$. ◀

Proof of Theorem 4. We first describe the certificates and then explain the verification process and its correctness.

The certification comprises the following components, each represented by $O(n \log n)$ bits. We assume that $\sqrt{m/n}$ is an integer (otherwise, we use $\lceil \sqrt{m/n} \rceil$).

- Neighbors: For each node u, $\mathsf{Neighbors}(u)$ consists of a set of neighbors of u.
- Diameter: $\mathsf{Diameter}(u)$ consists of a BFS tree starting from u.
- SpanningTree: $\mathsf{SpanningTree}(u)$ describes a spanning tree rooted at the node with maximum degree, denoted r. Thus, it is identical to $\mathsf{Diameter}(r)$.
- Partition: $\mathsf{Partition}(u)$ provides a partition of nodes. The prover divides V into $\sqrt{m/n}$ subsets $\mathcal{V} = \{V^i\}_{i \in [\sqrt{m/n}]}$ so that the number of edges between V^i and V^j is $O(n)$ for all $i, j \in [\sqrt{m/n}]$. This partition exists due to Lemma 16. $\mathsf{Partition}(u)$ includes the partition $\mathcal{V}$ and a set containing all the neighbors of r in ascending order with respect to their IDs. Additionally, $\mathsf{Partition}(u)$ includes encoded values in $O(\log n)$-bit: $\tilde{m}_{2,2}$, m_{bad}, $m_{2,3}$, $m_{3,3}$, $\tilde{m}_{2,2}(u)$, $m_{bad}(u)$, $m_{2,3}(u)$, $m_{3,3}(u)$. These values are used to verify the size of edge subsets. Finally, $\mathsf{Partition}(u)$ includes, for each $H \in \mathcal{H}$, the numbers $c(H)$ and $c_u(H)$. These values are used to count the number of dangerous copy of H in the graph.

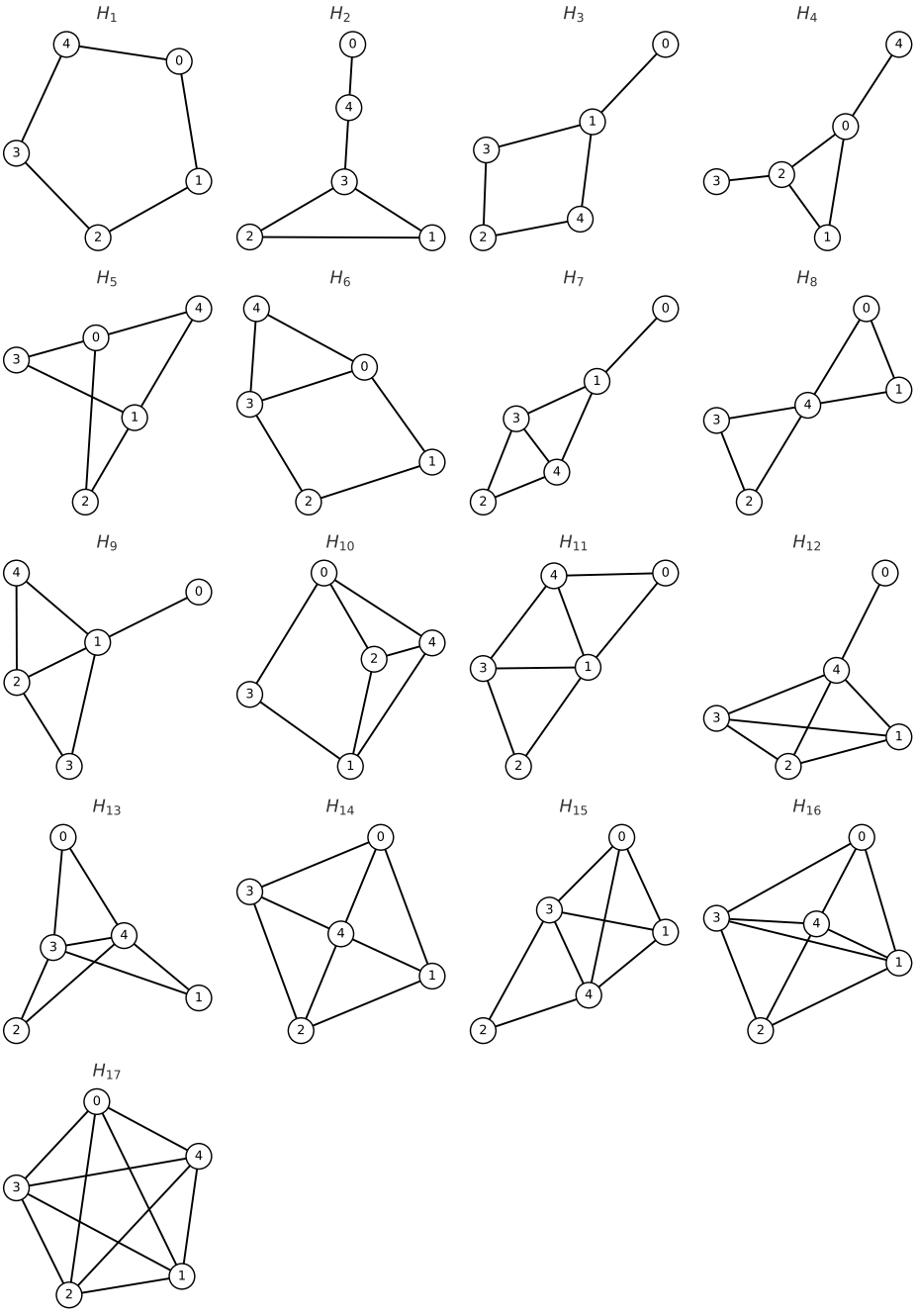

Figure 1 All different 5-node graphs that contain a P_5 as a subgraph.

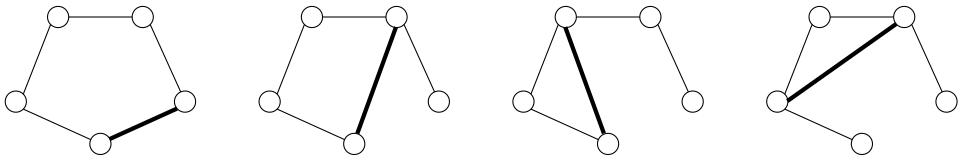

Figure 2 An illustration of H that contains exactly one bad edge considered in Lemma 25. Thick edges represent bad edges. For instance, an induced C_5 (the leftmost graph) contains exactly one edge from $F_{bad} \cup E_{3,3}$, and the remaining edges from $E \backslash (F_{bad} \cup E_{3,3})$, then removing $F_{bad} \cup E_{3,3}$ from the graph creates a new induced P_5.

- Edges: Let v_k be the neighbor of r whose degree is the k-th largest among all the neighbors of r for $k \in [m/n]$. $\mathsf{Edges}(v_k)$ then contains $O(n \log n)$-bit descriptions of all edges between V^i and V^j where $k = i + (j-1)\sqrt{m/n}$.

Verification

Diameter is used to certificate the graph diameter is at most 3. This is done by checking the structure of the trees as in [7].

Certifying the partition of nodes and edges. Each node u checks that the partition of nodes and the set representing all the neighbors of r in ascending order with respect to their IDs written in $\mathsf{Partition}(u)$, and the spanning tree written in $\mathsf{SpanningTree}(u)$ are the same as of its neighbors, so that all nodes agree the same partition, and the distance from r for all nodes. Now, given a partition of nodes in Partition and the spanning tree in SpanningTree, endpoints of each edge e determine which subset of

$$E_{1,1} \cup E_{1,2} \cup \tilde{E}_{2,2} \cup F_{bad} \cup E_{2,3} \cup E_{3,3}$$

the edge e belongs to. Next, each node computes the sizes of edge sets $|E_{1,1}|$, $|E_{1,2}|$, $|\tilde{E}_{2,2}|$, $|F_{bad}|$, $|E_{2,3}|$, $|E_{3,3}|$ using the protocol of Lemma 26 and the values written in Partition as follows: For example, we can compute $|E_{2,3}|$ by setting $b = |E_{2,3}|$, $b(u) = m_{2,3}(u)$, and $a(u)$ be the number of edges in $E_{2,3}$ incident to u.

Certifying the assigned edges. Let v_k be the neighbor of r that has the k-th minimum ID among all the neighbors of r. Since all nodes agree on the set representing all the neighbors of r in ascending order with respect to their IDs, all nodes know which is v_k. For each v_k, the validity of $\mathsf{Edges}(v_k)$ can be checked as follows. First, each v_k checks if the following holds.

1. $\mathsf{Edges}(v_k) \subseteq E \cap (V^i \times V^j)$ for $k = i + (j-1)\sqrt{m/n}$. v_k can check this since all nodes agree on the partition $\mathcal{V}$.
2. For each edge $e \in \mathsf{Edges}(v_k)$, at least one of its endpoints is at most distance 2 from v_k. v_k can check this since v_k knows nodes with distance 2 from v_k by looking at Neighbors of its neighbors.

If these conditions do not hold then v_k rejects. Each neighbor u of v_k rejects if $\mathsf{Edges}(v_k)$ contains non-edge that includes u' for some $u' \in N(u)$ by looking at $\mathsf{Neighbors}(u')$. This ensures that the edge set $\mathsf{Edges}(v_k)$ is indeed the set of edges between V^i and V^j such that at least one of the endpoints are a 2-hop neighbor of v_k. Each node then checks whether the four conditions described in $\mathcal{P}_{\mathsf{collect}}$ hold: r checks Condition 1, and each node u checks Condition 2,3, and 4 for each edge (u, v). If any condition holds, the network rejects. This part of verification ensures that r learns $E \backslash (F_{bad} \cup E_{3,3})$.

Detecting an induced P_5 containing $F_{bad} \cup E_{3,3}$. Assuming Conditions 1-4 of $\mathcal{P}_{\mathsf{collect}}$ are not met, some node can now detect an induced P_5 that contains at least one edge from $F_{bad} \cup E_{3,3}$ if exists, as in Lemma 23 and Lemma 24 since all necessary information from $\mathcal{P}_{\mathsf{collect}}$ is included in its certificate or the certificates of its neighbors.

Detecting an induced P_5 solely composed of $E \backslash (F_{bad} \cup E_{3,3})$. The final components of $\mathsf{Partition}(u)$ are used to count the number of dangerous H for $H \in \mathcal{H}$, as described in Lemma 25. This is accomplished using Lemma 26 as follows. For simplicity, let us consider the case of induced C_5 (H_1 in Figure 1). In our CONGEST algorithm, as in the proof of Lemma 25, for each such cycle, the node in V_1 with minimum identifier that is incident to the cycle detects it, and report the number of detected cycles to the node r through the BFS tree to compute the entire number of dangerous C_5's in the graph. Setting $b = c(H_1)$, $b(u) = c_u(H_1)$, and $a(u)$ be the number of dangerous C_5's detected by u in Lemma 26, we can certify it. r then determines the number of P_5's that increase by removing $F_{bad} \cup E_{3,3}$ (i.e., the number of dangerous copies of H for all $H \in \mathcal{H}$) and compares this with the number of induced P_5's in $(V, E \backslash (F_{bad} \cup E_{3,3}))$. Based on this comparison, r can decide if G is P_5-free. ◀

5 Lower bounds for P_k-freeness

5.1 Proof of Theorem 5

Our proof leverages a reduction from the three-party nondeterministic communication complexity of the set-disjointness problem in the number-on-forehead (NOF) model. In this model, each player sees the inputs of the other two players but not their own. The players communicate by writing bit strings on a shared blackboard, and the communication cost of the protocol is the total number of bits written.

As a first step, we use the reduction from set-disjointness to triangle-freeness established by Drucker et al.[13]. The key graph used in their reduction is a tripartite graph $G_{RS} = (A \cup B \cup C, E)$ where (1) $|A| = |B| = n$ and $|C| = n/3$ for arbitrary integer n; (2) G_{RS} contains $t = \frac{n^2}{e^{O(\sqrt{\log n})}}$ triangles $\mathcal{T} = \{T_1, \ldots, T_t\}$; (3) each edge of the graph belongs to exactly one triangle. This graph construction relates to the Ruzsa–Szemerédi problem[26] in extremal graph theory and has been well studied.

For $X_A, X_B, X_C \in \{0,1\}^t$, we define $G(X_A, X_B, X_C)$ as the graph obtained by removing edges from G_{RS} as follows. Let $T_i = (e_A, e_B, e_C)$ be a triangle such that e_A is an edge between B and C, e_B is an edge between A and C, and e_C is an edge between A and B. For $P \in \{A, B, C\}$, we remove the edge e_P iff the i-th bit of X_P is 0.

We then construct another graph $G^*(X_A, X_B, X_C)$ as follows. This construction is obtained by tweaking the construction in [12, 34]. Let $\overline{G}(X_A, X_B, X_C)$ be the complement graph of $G(X_A, X_B, X_C)$. For each node u in $\overline{G}(X_A, X_B, X_C)$, we have a triangle u_1, u_2, u_3 in $G^*(X_A, X_B, X_C)$. We add edges (u_i, v_i) for all $i \in \{1, 2, 3\}$. For each edge (u, v) in $\overline{G}(X_A, X_B, X_C)$, we add edges (u_i, v_j) for all $i, j \in \{1, 2, 3\}$ in $G^*(X_A, X_B, X_C)$. Additionally, there are two extra nodes x, y. x is connected to u_1 and u_2 for all $u \in V(\overline{G}(X_A, X_B, X_C))$. y is connected to u_2 and u_3 for all $u \in V(\overline{G}(X_A, X_B, X_C))$. Thus, the number of nodes in $G^*(X_A, X_B, X_C)$ is $O(n)$.

▶ **Lemma 27.** *$G^*(X_A, X_B, X_C))$ contains an induced P_5 if and only if X_A, X_B and X_C are not disjoint.*

Proof. Suppose that there is an index i such that the i-th bit of X_A, X_B and X_C are all 1. Then we have a triangle T_i in the graph $G(X_A, X_B, X_C)$ and thus have an independent set $\{u, v, w\}$ of size 3 in $\overline{G}(X_A, X_B, X_C)$. $\{u_1, x, v_2, y, w_3\}$ induces P_5 in $G^*(X_A, X_B, X_C)$.

Conversely, assume that $G^*(X_A, X_B, X_C)$ contains an induced P_5. Let us focus on an independent set of size 3 in the path. The set does not contain x and y at the same time, since each other node is connected at least one of x and y. Moreover, the set cannot contain x, as otherwise the other two nodes are written u_3 and v_3, but it is impossible since u_3 and v_3 are always connected. A similar contradiction occurs if the set contains y. Therefore three nodes in the set are written u_1, v_2, w_3. From the construction u, v, w in $\overline{G}(X_A, X_B, X_C)$ is an independent set. This means that X_A, X_B and X_C are not disjoint. ◀

We are now ready to prove Theorem 5.

Proof of Theorem 5. Suppose that P_5-freeness can be certified with certificates of size s. We construct a non-deterministic communication protocol for set-disjointness with communication complexity $O(ns)$. Suppose that Alice, Bob, and Charlie, who receive the inputs X_A, X_B, and X_C, respectively, simulate $G^*(X_A, X_B, X_C)$ as follows:

- For all $u \in A$ in $\overline{G}(X_A, X_B, X_C)$, Alice simulates nodes u_1, u_2, u_3 in $G^*(X_A, X_B, X_C)$.
- For all $u \in B$ in $\overline{G}(X_A, X_B, X_C)$, Bob simulates nodes u_1, u_2, u_3 in $G^*(X_A, X_B, X_C)$.
- For all $u \in C$ in $\overline{G}(X_A, X_B, X_C)$, Charlie simulates nodes u_1, u_2, u_3 in $G^*(X_A, X_B, X_C)$.
- All players simulates x and y.

Since the incident edges of simulated nodes by each player are independent from its input, each player can construct the inputs of its simulated nodes locally. Then the players simulate the certification for P_5-freeness. Since the certificate size is s and each player simulates $O(n)$ nodes, the length of the certificates for each player is $O(ns)$. In order to determine the outputs of the simulated nodes, each player writes its certificates to the shared blackboard. From Lemma 27, the players can compute set-disjointness of size $\frac{n^2}{e^{O(\sqrt{\log n})}}$. We now obtain $s = \Omega\left(\frac{n}{e^{O(\sqrt{\log n})}}\right) = \Omega(n^{1-o(1)})$ as desired by the nondeterministic multiparty communication complexity of set-disjointness in the number-on-forehead model [37]. To extend the lower bound for larger locality, we just replace edges incident to x and y by longer paths. Thus we can increase the locality by 1 at the cost of increasing the path length by 4. ◀

5.2 Lower bounds for P_k-freeness: general k

Technical challenges and the general proof idea for Theorem 6

We use a standard reduction from two-party set-disjointness function for the proof of Theorem 6 (and Theorem 7). Alice and Bob jointly construct some graph which depends on their inputs so that the constructed graph is P_k-free iff the output of some function (in our case, the set-disjointness function) is 1.

The main challenge in this proof is to ensure that the cut of the constructed graph (i.e., the number of edges between nodes held by Alice and those held by Bob) is as sparse as possible. This was not an issue in the proof of Theorem 5, which uses the non-deterministic communication complexity of the NOF model or in the proof of the lower bound given in [2], which uses counting arguments. In fact, the constructions used in these proofs do not achieve a sparse cut, making them unsuitable for the proof of Theorem 6 that holds for the CONGEST model.

Fortunately, in the simplest case where $d \geq 3$, we can utilize the construction from Ref. [29] with a slight modification, taking advantage of the graph structure that holds for all $d \geq 3$. However, for the cases where $d = 1$ and $d = 2$, the construction from Ref. [29], which

is applicable to induced k-cycles for every $k \geq 4$, fails for induced k-paths. This is intuitively because we lose several symmetric properties of induce k-cycles by removing one edge from the cycle. We thus construct two other graph families from scratch. The main challenge in establishing these constructions is that both edges and non-edges must be carefully chosen to avoid creating unexpected induced k-paths. Our proof requires a detailed analysis to ensure this, as the general structure valid for $d \geq 3$ does not apply for $d \leq 2$.

6 Ordered path detection and applications

In this section, we address the problem of detecting an ordered P_k. In this problem, each node of the graph is assigned a color from $\{1, \ldots, k\}$, and the objective is to detect an induced path $\{(p_i, p_{i+1})\}_{i \in \{1,\ldots,k-1\}}$ on k nodes $\{p_1, \ldots p_k\}$, where each node p_i is colored with i. Our first result is that detecting an ordered P_5 is already challenging, unlike the case of P_k-freeness where we only know nontrivial lower bound for $k \geq 11$ as in Theorem 6.

▶ **Theorem 28.** *Any randomized algorithm that solves ordered P_k detection for $k \geq 5$ requires $\tilde{\Omega}(n)$ rounds in the* CONGEST *model.*

The proof of this theorem leverages a reduction from two-party communication complexity, with a construction that is notably simpler compared to the proof of Theorem 6. The detailed proof is provided in the full version.

Our next insight regarding ordered path detection is the connection between proving non-trivial lower bounds for ordered P_3 detection and a long-standing open problem in circuit complexity.[2] Moreover, this result has implications for the detection of induced C_4 in the quantum CONGEST model. Specifically, the following problem has remained unresolved for decades:

▶ **Open Problem 1.** *Construct an explicit family of boolean functions $f_n : \{0,1\}^n \to \{0,1\}$ such that there exist constants $\delta_1, \delta_2 > 0$ such that any family of circuits $\{C_n\}_{n \in \mathbb{N}}$, where each circuit C_n is depth $O(n^{\delta_1})$ and consists of $O(n^{1+\delta_2})$ gates with constant fan-in and fan-out, cannot compute $\{f_n\}_{n \in \mathbb{N}}$.*

The following theorem formalizes this relationship:

▶ **Theorem 29** (The formal version of Theorem 12)**.** *Let $\varepsilon > 0$ be a constant. Then proving $\Omega(n^{\varepsilon})$ lower bound for ordered P_3 detection in the* CONGEST *model or $\Omega(n^{1/2+\varepsilon})$ lower bound for induced C_4 detection in the quantum* CONGEST *model solves Open Problem 1.*

▶ Remark 30. One might think that ordered P_3 detection can be solved in $O(1)$ rounds as the case of P_3-freeness. However, it is easily shown that the round complexity of ordered P_3 detection in the CONGEST model is actually at least the round complexity of triangle detection in the congested clique model: Let G' be the graph resulting by deleting all edges in the input graph G between a node colored by 1 and a node colored by 3, and replacing non-edges between them by edges. Then any CONGEST algorithm for ordered P_3 detection is simulated in the congested clique model with the input graph G' where each ordered P_3 in G corresponds to a multicolored triangle in G'. Note that multicolored triangle detection can be reduced to (the standard) triangle detection in this model [34].

[2] This kind of connection is already known for several subgraphs such as triangles and 6-cycles [15, 14].

Proof sketch of Theorem 29. The first part of the statement builds on the known hardness results for triangle detection (and counting) established by [14]. We demonstrate that ordered P_3 detection is at least as difficult as certain variants of triangle counting or ordered P_3 counting in high-conductance graphs. The full proof is deferred to the full version.

For the second part, we use the framework of distributed quantum search [30]. Consider a function $f : X \to \{0, 1\}$ defined on some finite set X, and assume a T-round CONGEST algorithm allows a fixed node u to output $f(x)$ with constant probability, given input $x \in X$. In the quantum CONGEST model, we can find an element $x \in X$ such that $f(x) = 1$ (if such an element exists) with constant probability in $\tilde{O}(\sqrt{|X|} \cdot T)$ rounds.

Now, assume that ordered P_3 detection can be solved in T rounds in the CONGEST model. We can show that induced C_4 detection can be solved in $\tilde{O}(\sqrt{n} \cdot T)$ rounds in the quantum CONGEST model. The statement then follows from the first part of the theorem, as an $\Omega(n^{1/2+\varepsilon})$ lower bound for induced C_4 detection implies an $\Omega(n^{\varepsilon})$ lower bound for ordered P_3 detection.

Specifically, let u be an arbitrary node, and define $M(u) = \{v : \mathrm{dist}(u, v) = 2\}$. Consider the subgraph $G[N(u) \cup M(u)]$ induced by $N(u)$ and $M(u)$. Nodes in $M(u)$ are colored with color 2, while nodes in $N(u)$ randomly choose their color from 1 and 3. Note that $G[N(u) \cup M(u)]$ contains an ordered P_3 (p_1, p_2, p_3) if G contains an induced C_4 (u, p_1, p_2, p_3). Since p_1 and p_3 randomly choose their colors, the probability that (p_1, p_2, p_3) is properly colored is constant. Therefore, in $O(T)$ rounds, we can determine if u belongs to an induced C_4. By using distributed quantum search for a function $f : V \to \{0, 1\}$, which outputs $f(u) = 1$ if and only if u belongs to an induced C_4, we can solve induced C_4 detection in $\tilde{O}(\sqrt{n} \cdot T)$ rounds. ◀

References

1. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
2. Nicolas Bousquet, Linda Cook, Laurent Feuilloley, Théo Pierron, and Sébastien Zeitoun. Local certification of forbidden subgraphs. *arXiv preprint arXiv:2402.12148*, 2024. doi:10.48550/arXiv.2402.12148.
3. Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Local Certification of Graph Decompositions and Applications to Minor-Free Classes. In *25th International Conference on Principles of Distributed Systems (OPODIS 2021)*, pages 22:1–22:17, 2022. doi:10.4230/LIPIcs.OPODIS.2021.22.
4. Nicolas Bousquet and Sébastien Zeitoun. A subquadratic certification scheme for p5-free graphs. *Theoretical Computer Science*, page 115091, 2025. doi:10.1016/J.TCS.2025.115091.
5. Christoph Brause, Ingo Schiermeyer, Přemysl Holub, Zdeněk Ryjáček, Petr Vrána, and Rastislav Krivoš-Belluš. 4-colorability of P_6-free graphs. *Electronic Notes in Discrete Mathematics*, 49:37–42, 2015.
6. Keren Censor-Hillel, François Le Gall, and Dean Leitersdorf. On distributed listing of cliques. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 474–482, 2020. doi:10.1145/3382734.3405742.
7. Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theoretical Computer Science*, 811:112–124, 2020. doi:10.1016/J.TCS.2018.08.020.
8. Yi-Jun Chang, Seth Pettie, and Hengjie Zhang. Distributed triangle detection via expander decomposition. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 821–840. SIAM, 2019. doi:10.1137/1.9781611975482.51.
9. Yi-Jun Chang and Thatchaphol Saranurak. Improved distributed expander decomposition and nearly optimal triangle enumeration. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 66–73, 2019. doi:10.1145/3293611.3331618.

10 Derek G Corneil, Helmut Lerchs, and L Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981. doi:10.1016/0166-218X(81)90013-5.

11 Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-Offs in Distributed Interactive Proofs. In *33rd International Symposium on Distributed Computing (DISC 2019)*, pages 13:1–13:17, 2019. doi:10.4230/LIPICS.DISC.2019.13.

12 Mina Dalirrooyfard, Thuy Duong Vuong, and Virginia Vassilevska Williams. Graph pattern detection: Hardness for all induced patterns and faster non-induced cycles. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1167–1178, 2019. doi:10.1145/3313276.3316329.

13 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the Power of the Congested Clique Model. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing (PODC 2014)*, pages 367–376, 2014. doi:10.1145/2611462.2611493.

14 Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. Sublinear-time distributed algorithms for detecting small cliques and even cycles. *Distributed Computing*, pages 1–28, 2022.

15 Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and impossibilities for distributed subgraph detection. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 153–162, 2018. doi:10.1145/3210377.3210401.

16 Pierre Fraigniaud, Maël Luce, Frederic Magniez, and Ioan Todinca. Even-cycle detection in the randomized and quantum congest model. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing (PODC 2024)*, pages 209–219, 2024.

17 Pierre Fraigniaud, Frédéric Mazoit, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. Distributed certification for classes of dense graphs. In *Proceedings of the 37th International Symposium on Distributed Computing (DISC 2023)*, pages 20:1–20:17, 2023. doi:10.4230/LIPICS.DISC.2023.20.

18 Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. A meta-theorem for distributed certification. *Algorithmica*, 86(2):585–612, 2024. doi:10.1007/S00453-023-01185-1.

19 Pierre Fraigniaud and Dennis Olivetti. Distributed detection of cycles. *ACM Transactions on Parallel Computing (TOPC)*, 6(3):1–20, 2019. doi:10.1145/3322811.

20 Peter Gartland and Daniel Lokshtanov. Independent Set on P_k-Free Graphs in Quasi-Polynomial Time. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 613–624. IEEE, 2020.

21 Gaston H Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the ACM (JACM)*, 28(2):289–304, 1981. doi:10.1145/322248.322254.

22 Andrzej Grzesik, Tereza Klimošová, Marcin Pilipczuk, and Michał Pilipczuk. Polynomial-time algorithm for maximum weight independent set on P_6-free graphs. *ACM Transactions on Algorithms (TALG)*, 18(1):1–57, 2022. doi:10.1145/3414473.

23 Chính T Hoàng, Marcin Kamiński, Vadim Lozin, Joe Sawada, and Xiao Shu. Deciding k-colorability of P_5-free graphs in polynomial time. *Algorithmica*, 57:74–81, 2010. doi:10.1007/S00453-008-9197-8.

24 Stephan Holzer and Roger Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 355–364, 2012. doi:10.1145/2332432.2332504.

25 Jarkko Kari, Martín Matamala, Ivan Rapaport, and Ville Salo. Solving the induced subgraph problem in the randomized multiparty simultaneous messages model. In *International Colloquium on Structural Information and Communication Complexity (SIROCCO 2015)*, pages 370–384, 2015. doi:10.1007/978-3-319-25258-2_26.

26 János Komlós and Miklós Simonovits. Szemeredi"s regularity lemma and its applications in graph theory, 1995.

27 Janne H. Korhonen and Joel Rybicki. Deterministic Subgraph Detection in Broadcast CONGEST. In *Proceedings of the 21st International Conference on Principles of Distributed Systems (OPODIS 2017)*, pages 4:1–4:16, 2018. doi:10.4230/LIPIcs.OPODIS.2017.4.

28 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010. doi:10.1007/S00446-010-0095-3.

29 François Le Gall and Masayuki Miyamoto. Lower Bounds for Induced Cycle Detection in Distributed Computing. In *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, pages 58:1–58:19, 2021. doi:10.4230/LIPICS.ISAAC.2021.58.

30 François Le Gall and Frédéric Magniez. Sublinear-time quantum computation of the diameter in CONGEST networks. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC 2018)*, pages 337–346, 2018. URL: https://dl.acm.org/citation.cfm?id=3212744.

31 Daniel Lokshantov, Martin Vatshelle, and Yngve Villanger. Independent set in P_5-free graphs in polynomial time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms*, pages 570–581. SIAM, 2014. doi:10.1137/1.9781611973402.43.

32 Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. Mst construction in o (log log n) communication rounds. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 94–100, 2003. doi:10.1145/777412.777428.

33 Pedro Montealegre, Diego Ramírez-Romero, and Ivan Rapaport. Compact distributed interactive proofs for the recognition of cographs and distance-hereditary graphs. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems (SSS 2021)*, pages 395–409, 2021. doi:10.1007/978-3-030-91081-5_26.

34 Amir Nikabadi and Janne Korhonen. Beyond distributed subgraph detection: Induced subgraphs, multicolored problems and graph parameters. In *25th International Conference on Principles of Distributed Systems (OPODIS 2021)*, volume 217, 2022. doi:10.4230/LIPIcs.OPODIS.2021.15.

35 David Peleg. *Distributed computing: a locality-sensitive approach*. SIAM, 2000.

36 Bert Randerath and Ingo Schiermeyer. 3-Colorability $\in P$ for P_6-free graphs. *Discrete Applied Mathematics*, 136(2-3):299–313, 2004.

37 Anup Rao and Amir Yehudayoff. Simplified lower bounds on the multiparty communication complexity of disjointness. In *30th Conference on Computational Complexity (CCC 2015)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CCC.2015.88.