Lower Bounds for Subgraph Detection in the CONGEST Model

Tzlil Gonen[1] **and Rotem Oshman**[2]

1 **Tel Aviv University, Tel Aviv, Israel**
`tzlilgon@mail.tau.ac.il`
2 **Tel Aviv University, Tel Aviv, Israel**
`roshman@mail.tau.ac.il`

Abstract

In the subgraph-freeness problem, we are given a constant-sized graph H, and wish to determine whether the network graph contains H as a subgraph or not. Until now, the only lower bounds on subgraph-freeness known for the CONGEST model were for cycles of length greater than 3; here we extend and generalize the cycle lower bound, and obtain polynomial lower bounds for subgraph-freeness in the CONGEST model for two classes of subgraphs.

The first class contains any graph obtained by starting from a 2-connected graph H for which we already know a lower bound, and replacing the vertices of H by arbitrary connected graphs. We show that the lower bound on H carries over to the new graph. The second class is constructed by starting from a cycle C_k of length $k \geq 4$, and constructing a graph $\tilde{H}$ from C_k by replacing each edge $\{i, (i+1) \bmod k\}$ of the cycle with a connected graph H_i, subject to some constraints on the graphs $H_0, \ldots, H_{k-1}$. In this case we obtain a polynomial lower bound for the new graph $\tilde{H}$, depending on the size of the shortest cycle in $\tilde{H}$ passing through the vertices of the original k-cycle.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity, F.2.0 General

Keywords and phrases subgraph freeness, CONGEST, lower bounds

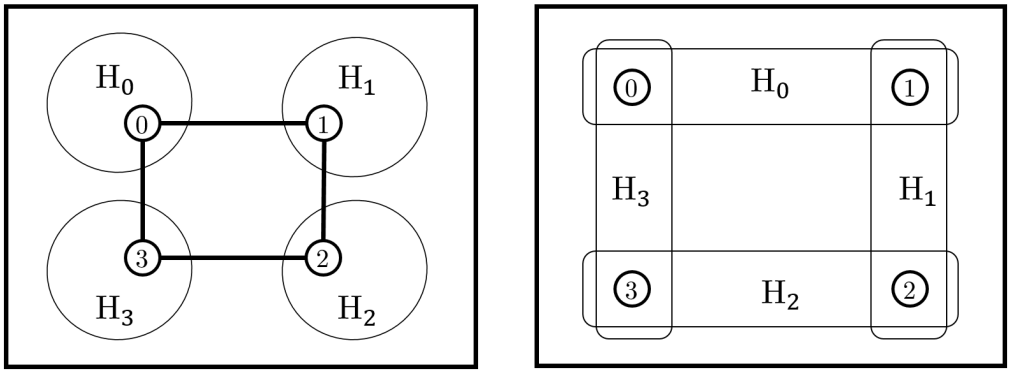
Digital Object Identifier 10.4230/LIPIcs.OPODIS.2017.6

1 Introduction

In the *subgraph-freeness* problem, we are given a constant-size graph H, and the goal is to determine whether the network graph contains a copy of H as a subgraph, or not. Subgraph-freeness in the CONGEST network model has recently received significant attention from the distributed computing community [4, 9, 10, 11, 12, 17], but until now, the only *lower bounds* in the literature have been for cycles: in [7] it was shown that checking C_k-freeness requires $\tilde{\Omega}(n)$ rounds for odd k and $\tilde{\Omega}(n^{2/k})$ rounds for even k. The lower bound for even cycles was recently strengthened to $\tilde{\Omega}(\sqrt{n})$ for any even k in [17]. In [1], it is shown that one-round deterministic algorithms for triangle-detection require bandwidth $\Omega(\Delta \log n)$, where Δ is the degree of the graph, and if we restrict to one bit per round, $\Omega(\log^* n)$ rounds are required.

In this work we seek to improve our understanding of the subgraph-freeness problem by broadening the class of graphs for which we know a polynomial lower bound, i.e., a lower bound of the form $\Omega(n^\delta)$ for some $\delta \in (0, 1]$. We give two classes of such graphs and prove polynomial lower bounds for them.

The first class of graphs comprises all graphs that can be constructed by starting from some 2-vertex-connected graph H for which we already *know* a polynomial lower bound



(a) Vertex replacement **(b)** Edge replacement

Figure 1

(regardless of how it was proven), and attaching an arbitrary graph to each vertex of H (see Fig. 1a). The graphs attached to different vertices may not share vertices or edges. We show that the new graph $\tilde{H}$ requires the same number of rounds as the graph H that we started with, up to a polylogarithmic factor, for both deterministic and randomized algorithms.

In particular, consider any connected graph G that is not a tree. (Trees are known to be easy: for any constant-sized tree T, the T-freeness problem can be deterministically solved in $O(1)$ rounds [8].) The *block cut tree* of G is a decomposition of G into a tree of maximal 2-connected components $C_1, \ldots, C_k$, where the pairwise intersection of any two components C_i, C_j (for $i \neq j$) is either empty or comprises a single vertex (called a *cut vertex*) [13]. Our first result shows that if for some i we know a lower bound of $\Omega(n^\delta)$ on checking C_i-freeness, then the entire graph G is also hard and requires $\tilde{\Omega}(n^\delta)$ rounds.

The second class of graphs takes a different approach: instead of replacing vertices, we replace edges (see Fig. 1b). We start with the 4-cycle[1] on $\{0, 1, 2, 3\}$, and replace each edge $\{i, (i+1) \bmod 4\}$ of the cycle with an arbitrary graph connecting vertices i and $(i+1) \bmod 4$, subject to two constraints: the resulting graph after replacing the edges must remain 2-connected, and no graph we added can contain the other three, connected to each other, as a subgraph (see Section 4 for a more formal definition). We show that checking subgraph-freeness for the resulting graph requires $\tilde{\Omega}(n^\delta)$ rounds, where the exponent $\delta \in (0, 1/2]$ depends on the graphs with which the edges of C_4 were replaced. For example, the second class of graphs includes any cycle of length at least 4 (for which our result is identical to [17]), as well as cycles with any number of chords, provided at least one of the smaller cycles created has length at least 4, and the smaller cycles are not too unbalanced in size.

The two classes complement each other: the graphs in the first class are obtained by starting from a 2-connected graph and replacing its vertices with other graphs, yielding a graph that is not 2-connected; the second class allows us to prove lower bounds on H-freeness for new subgraphs H that *are* 2-connected, and these can then be used to construct more graphs in the first class.

The reductions we use to show lower bounds for the two classes of graphs are fairly simple, but proving their correctness is non-trivial. For the first class, we take an algorithm for detecting the more complicated graph $\tilde{H}$, and transform it into an algorithm for the simpler graph H that we started with. This allows us to carry the lower bound over in the other

[1] We can also start from a larger cycle, but this gives us no additional power, because larger cycles can be constructed out of 4-cycles using our reduction. The lower bound we obtain would also not be higher.

direction. The transformation works by having each node in the network graph choose some vertex v in H, and "pretend" that it is the entire subgraph that we attached to vertex v in H when we constructed $\tilde{H}$. We must show that our algorithm does not inadvertently create copies of $\tilde{H}$ when the network graph does not contain a copy of H, and this is non-trivial. For the second class of graphs we extend the reduction from the two-party communication complexity of set disjointness used in [7] to show the hardness of C_4-freeness. Again, the difficulty lies in proving that in our reduction we do not create copies of $\tilde{H}$ when we do not mean to.

1.1 Related Work

The problem of subgraph-freeness (also called *excluded* or *forbidden subgraphs*) has been extensively studied in both the centralized and the distributed worlds. For the general problem of detecting whether a graph H is a subgraph of G, where both H and G are part of the input, the best known sequential algorithm is exponential [21]. When H is fixed and only G is the input, the problem becomes solvable in polynomial time. For example, using a technique called *color coding*, it is possible to detect a simple cycle of a specific size in expected time that is the running time of matrix multiplication [2]. We use the color coding technique in Section 3.

In the distributed setting, [10] and [11] very recently provided constant-round randomized and deterministic algorithms, respectively, for detecting a fixed tree in the CONGEST model. Both papers, as well as several others [3, 4, 12, 9] , also considered more general graphs, but with the exception of trees, they studied the *property testing* relaxation of the problem, where we only need to distinguish a graph that is H-free from a graph that is *far* from H-free. (In [9] there is also a property-testing algorithm for trees.) Here we consider the *exact* version.

Another recent work [15] gave randomized algorithms in the CONGEST model for triangle detection and triangle listing, with round complexity $\widetilde{O}(n^{2/3})$ and $\widetilde{O}(n^{3/4})$, respectively, and also established a lower bound of $\widetilde{\Omega}(n^{1/3})$ on the round complexity of triangle listing. There is also work on testing triangle-freeness in the congested clique model [5, 6] and in other, less directly related distributed models.

As for lower bounds on H-freeness in the CONGEST model, the only ones in the literature (to our knowledge) are for cycles. (In [7] there are lower bounds for other graphs, in a broadcast variant of the CONGEST model where nodes are required to send the *same* message on all their edges.) For any fixed $k > 3$, there is a polynomial lower bound for detecting the k-cycle C_k in the CONGEST model: it was first presented by [7], which showed that $\Omega(\mathrm{ex}(n, C_k)/\log(n))$ rounds are required, where $\mathrm{ex}(n, C_k)$ is the largest possible number of edges in a C_k-free graph over n vertices (see [14] for a survey on extremal graphs with forbidden subgraphs). In particular, for odd-length cycles, the lower bound of [7] is nearly linear. Very recently, [17] improved the lower bound for even-length cycles to $\Omega(\sqrt{n}/\log(n)))$.

2 Preliminaries

We generally work with undirected graphs, unless indicated otherwise.

The CONGEST model is a synchronous network model, where computation proceeds in *rounds*. In each round, each node of the network may send B bits on each of its edges, and these messages are received by neighbors in the current round. As is typical in the literature, we assume $B = O(\log n)$ here (the lower bounds generalize to other settings of B in a straightforward manner).

Notation. We let $V(G), E(G)$ denote the vertex and edge set of graph G, respectively, and use the following short-hand notation:

- $H \subseteq G$ for two graphs H, G stands for the subgraph relation: $H \subseteq G$ iff $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.
- We use $A \hookrightarrow B$ to denote a function mapping all elements of A into B. Specifically, if G contains a copy of H as a subgraph, we frequently let $\sigma : H \hookrightarrow G$ denote an isomorphism mapping the nodes of H into $V(G)$. When the isomorphism is *onto*, we use the usual $\rightarrow$, i.e., $\sigma : H \rightarrow H'$.
- If $\sigma : H \hookrightarrow G$ is an isomorphism mapping H onto a copy of H in G, we let $\sigma(H)$ denote the image of H under σ. We have $\sigma(H) \subseteq G$.

▶ **Definition 1** (Subgraph freeness)**.** Fix a graph H of constant size. In the *H-freeness* problem, the goal is to determine whether the input graph G contains a copy of H as a subgraph or not, that is, whether there is a subgraph $G' \subseteq G$ which is isomorphic to H.

We say that a distributed algorithm A *solves H-freeness with success probability p* if

- When A is executed in a graph containing a copy of H, the probability that all nodes accept is at least p.
- When A is executed in an H-free graph, the probability that at least one node rejects is at least p.

We typically assume constant p, e.g., $p = 2/3$.

The graph classes we define assume some amount of *vertex connectivity*, defined below.

▶ **Definition 2** (Vertex connectivity)**.** We say that a graph G is k-connected, for $k \geq 1$, if removing any k vertices from G (along with all their incident edges) does not disconnect G.

In Section 4 we rely on a lower bound for *two-party communication complexity*: we have two players, Alice and Bob, with private inputs X, Y, respectively. The players wish to compute a joint function $f(X, Y)$ of their inputs, and we are interested in the total number of bits they must exchange to do so (see the textbook [18] for more background on communication complexity).

In particular, we are interested in the *set disjointness* function, where the inputs X, Y are interpreted as subsets $X, Y \subseteq [n]$, and the goal of the players is to determine whether $X \cap Y = \emptyset$. The celebrated lower bound of [16, 20] shows that even for randomized communication protocols, the players must exchange $\Omega(n)$ bits to solve set disjointness with constant success probability.

3 Vertex-Replacement Reduction

In this section we describe a reduction that allows us to take any 2-connected graph H for which we know a polynomial lower bound, attach arbitrary connected graphs to the vertices of H to obtain a new graph $\tilde{H}$, and obtain the same lower bound on $\tilde{H}$ as for H, up to a logarithmic factor.

Let us describe more formally what we mean by "attaching graphs to the vertices of H".

▶ **Definition 3** (Graph attachment)**.** Fix two graphs G, H over vertex sets $V(G), V(H)$ with a unique intersection, $V(G) \cap V(H) = \{v\}$. We define an *attached graph*, $G \cup H$, as follows: $V(G \cup H) = V(G) \cup V(H)$, and $E(G \cup H) = E(G) \cup E(H)$. We refer to vertex v as the *attachment point* of $G \cup H$.

The attachment operation trivially has the following property, which we rely on in the sequel:

▶ **Property 4.** *Let $G = H_1 \cup H_2$, with $v \in V(H_1) \cap V(H_2)$ the attachment point. Then for any two vertices $w_1 \in V(H_1), w_2 \in V(H_2)$, any path $u_1 = w_1, u_2, \ldots, u_\ell = w_2$ from w_1 to w_2 in H must include v (that is, for some $1 \leq i \leq \ell$ we have $u_i = v$).*

▶ **Definition 5** (Compatible graphs)**.** Let G be a graph with $V(G) = [s]$. We say that a sequence of graphs $H_0, \ldots, H_{s-1}$ is *compatible with* G if

(1) For any $i \in [s]$ we have $V(H_i) \cap V(G) = \{i\}$, and

(2) For any $i \neq j$ we have $V(H_i) \cap V(H_j) = \emptyset$.

If $H_0, \ldots, H_{s-1}$ are compatible with G, we use the short-hand notation $G \cup \bigcup_{i=0}^{s-1} H_i$ to denote the graph defined by attaching each H_i to G (the order in which we attach $H_0, \ldots, H_{s-1}$ does not matter).

▶ **Definition 6** (The class $\mathcal{A}_H$)**.** Fix a graph H on vertices $V(H) = [s]$. The class $\mathcal{A}_H$ includes any graph given by $\tilde{H} = H \cup \bigcup_{i=0}^{s-1} H_i$ for $H_0, \ldots, H_{s-1}$ compatible with H.

See Figure 1a for an illustration.

Now we can state our main theorem for this section.

▶ **Theorem 7.** *Let H be any 2-connected graph on $V(H) = [s]$. If solving H-freeness requires $\Omega(n^\delta)$ rounds in graphs of size n for deterministic algorithms, then for any $\tilde{H} \in \mathcal{A}_H$, solving $\tilde{H}$-freeness requires $\Omega(n^\delta / \log n)$ rounds for deterministic algorithms. The same relationship holds for randomized algorithms, except that if checking H-freeness requires $\Omega(n^\delta)$ rounds for randomized algorithms, then checking $\tilde{H}$-freeness requires $\Omega(n^\delta / \log^2 n)$ rounds.*

To prove Theorem 7, we describe a *randomized* reduction, which takes an algorithm for checking $\tilde{H}$-freeness, and constructs a randomized algorithm for checking H-freeness. This allows us to carry the lower bound over in the other direction, from checking H-freeness to checking $\tilde{H}$-freeness. This prove Theorem 7 for randomized algorithms; to prove it for deterministic algorithms, we derandomize our reduction.[2]

3.1 The Reduction

Fix an algorithm $\tilde{A}$ for checking $\tilde{H}$-freeness, with running time $t(n)$ and success probability p. We wish to use $\tilde{A}$ to check H-freeness. To do this, we have each node v of the network choose a "role in H", $c(v) \in V(H)$; we call $c(v)$ the *color* of v. Each node v then "imagines" that it is attached to a copy of $H_{c(v)}$, and we simulate the run of $\tilde{A}$ in the resulting network.

In the randomized reduction, each node chooses a *random* color; we call a copy of H *properly colored* if each node chose a color that matches the vertex it is mapped to in H.

▶ **Definition 8** (Properly-colored copies)**.** Fix graphs G, H, such that G contains a copy of H. Let $\sigma : H \hookrightarrow G$ map H onto its copy in G. We say that $\sigma(H)$ is *properly colored* by an assignment of colors $c : V(G) \to V(H)$ if for each $v \in \sigma(H)$ we have $c(v) = \sigma^{-1}(v)$.

Next we formally describe the algorithm $A(c)$ that is executed for a given color assignment $c : V(G) \to V(H)$, where G is the network graph.

[2] It is also possible to first derandomize the reduction and then apply it to either a randomized or deterministic algorithm. However, we must first reduce the error probability of the randomized algorithm, so either way we lose an additional $\log n$ factor.

3.1.1 Construction of $A(c)$

We define a "virtual graph", $\tilde{G}_c$, where

(1) Each vertex $v \in V(G)$ is replaced by $(v, c(v))$; let G_c be the resulting copy of G.

(2) Each $v \in V(G)$ creates a "virtual copy" $\tilde{G}_v$ of the graph $H_{c(v)}$, where each vertex $i \in H_{c(v)}$ is replaced by (v, i).

(3) $\tilde{G}_c = G_c \cup \bigcup_{v \in V(G)} \tilde{G}_v$, that is, $\tilde{G}_c$ is obtained by attaching the copies $\tilde{G}_v$ for each $v \in V(G)$ to G_c.

Let $\tilde{U}_v = V(\tilde{G}_v)$ denote the "virtual vertices" corresponding to the copy of $H_{c(v)}$ attached to vertex v (i.e., to vertex (v, c) in $\tilde{G}_c$).

In $A(c)$, we *simulate* the execution of $\tilde{A}$ in the virtual graph $\tilde{G}_c$, with each vertex $v \in G$ simulating all the vertices in $\tilde{U}_v$ (including "itself", vertex $(v, c(v))$. At the end of the execution, each vertex v *accepts* if all virtual vertices in $\tilde{U}_v$ accepted, and otherwise it *rejects*.

The running time of $A(c)$ in G is the same as the running time of $\tilde{A}$ in $\tilde{G}_c$. Let $a = \max_i |V(H_i)|$ be the maximum number of vertices simulated by a vertex of G in $\tilde{G}_c$. Then $|V(\tilde{G}_c)| \leq a|V(G)|$. Thus, the worst-case running time of A in G is bounded by $\tilde{t}(an)$.

The algorithm A itself simply has each node v choose a random color $c(v) \in V(H)$, and then runs $A(c)$. We will show that

- If the graph contains a copy of H, then A accepts with probability at least p/s^s;
- If the graph does not contain a copy of H, then A accepts with probability at most $1 - p$.

3.1.2 What Could Go Wrong?

Before proving A correct, let us illustrate why the requirement that H be 2-connected is necessary, and in general what could go wrong if we are not careful.

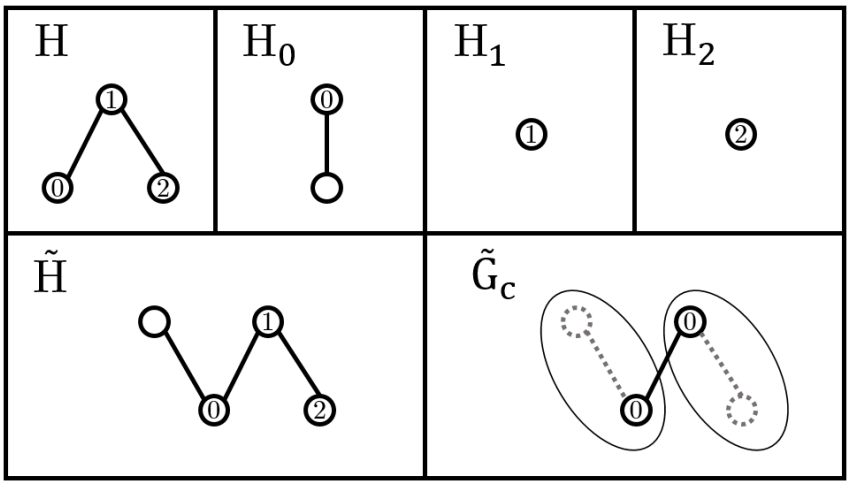


Figure 2 A bad example with a subgraph H that is not 2-connected.

Consider the graph H which is the line over vertices $\{0, 1, 2\}$, in increasing order. H is of course *not* 2-connected. Let H_0 be the graph containing only a single edge, $\{0, a\}$, and for each $i = 1, 2$, let H_i be the single vertex i. The graph $\tilde{H}$ we obtain by attaching H_0, H_1, H_2 to H is the line of length 3 (i.e., 3 edges).

Now let G be a graph comprising a single edge, $\{a, b\}$, and suppose that we were unlucky and chose the color assignment $c(a) = c(b) = 0$. Then $\tilde{G}_c$ is a line of length 3, i.e., it contains

$\tilde{H}$ as a subgraph. But G does *not* contain H as a subgraph, so our reduction would be unsound if we applied it with this H.

Next we show that for subgraphs H that *are* 2-connected, our reduction is complete and sound: the graph we construct contains a copy of $\tilde{H}$ iff the original graph contained a properly-colored copy of H.

3.1.3 Completeness of A

Suppose that the original graph G contains a copy H, and let $\sigma : H \hookrightarrow G$ be an isomorphism mapping H into G. Suppose further that $\sigma(H)$ is properly colored by c. Conditioned on this event, the new graph $\tilde{G}_c$ contains a copy of $\tilde{H}$, witnessed by the following isomorphism σ_c: for each $x \in \tilde{H}$, if $x \in H_j$, then $\sigma_c(x) = (\sigma(j), x)$. Thus, conditioned on the event that we get a good coloring c, $A(c)$ accepts with probability at least p. The overall acceptance probability of A is at least p/s^s.

3.1.4 Soundness of A

Now suppose that G does *not* contain a copy of H; we show that for any coloring c of G, the corresponding graph $\tilde{G}_c$ does not contain a copy of $\tilde{H}$. Therefore A rejects with the same probability that $\tilde{A}$ rejects, i.e., at least p.

Suppose for the sake of contradiction that there is a coloring c of G such that $\tilde{G}_c$ does contain a copy of $\tilde{H}$, and let us abuse notation by denoting $\tilde{G} = \tilde{G}_c$. We will show that there is some "virtual copy" $\tilde{G}_v$ which contains infinitely many distinct copies of H, which is, of course, impossible.

First we show that any copy of H in $\tilde{G}$ must be entirely contained in some $\tilde{G}_v$.

▶ **Observation 9.** *For any isomorphism $\sigma : H \hookrightarrow \tilde{G}$, there is a vertex $v \in V(G)$ such that $\sigma(H) \subseteq \tilde{U}_v$.*

Proof. From our assumption, there is no copy of H in G, and therefore $\sigma(H) \not\subseteq G$; that is, $\sigma(H)$ must contain some "virtual nodes". Accordingly, there must be some "virtual part", $\tilde{U}_v$, such that $\sigma(H) \cap \tilde{U}_v \setminus \{v\} \neq \emptyset$. If $\sigma(H)$ is only *partially* contained in $\tilde{U}_v$ (i.e., if $\sigma(H) \not\subseteq \tilde{U}_v$), then removing $(v, c(v))$ from $\tilde{G}$ disconnects $\sigma(H)$, because any path from vertices in $\tilde{U}_v$ to vertices outside $\tilde{U}_v$ must pass through $(v, c(v))$ (Property 4). But H is 2-connected, and therefore $\sigma(H)$ must be entirely contained in $\tilde{U}_v$. ◀

Now fix an isomorphism $\sigma : \tilde{H} \hookrightarrow \tilde{G}$, and let v be the vertex from Observation 9, such that $\sigma(H) \subseteq \tilde{U}_v$. Let $i = c(v)$.

▶ **Observation 10.** $\sigma(H_i) \not\subseteq \tilde{G}_v$.

Proof. Recall that $\tilde{U}_v$ is the vertex set of $\tilde{G}_v$, which is a copy of H_i; thus, $|\tilde{U}_v| = |V(H_i)|$, and we can only have $\sigma(H_i) \subseteq \tilde{U}_v$ if $\sigma(H_i) = \tilde{U}_v$. But $\tilde{U}_v$ contains at least one vertex which is not in $\sigma(H_i)$: take any vertex $j \in V(H) \setminus \{i\}$ (which exists because $|V(H)| > 1$), and since $\sigma(H) \subseteq \tilde{U}_v$ and $V(H) \cap V(H_i) = \{i\}$, we have $\sigma(j) \in \tilde{U}_v \setminus \sigma(H_i)$. Therefore $\sigma(H_i) \neq \tilde{U}_v$ and $\sigma(H_i) \not\subseteq \tilde{U}_v$. ◀

▶ **Observation 11.** *We have $(v, i) \in \sigma(H_i)$.*

Proof. From Observation 10, there is some $u \in H_i$ such that $\sigma(u) \notin \tilde{U}_v$. In H_i, there is a path π from u to i, because $u, i \in H_i$ and H_i was assumed connected; the isomorphism σ maps π onto a path $\sigma(\pi)$ in $\tilde{G}$ from $\sigma(u) \notin \tilde{U}_v$ to $\sigma(i) \in \sigma(H) \subseteq \tilde{U}_v$. By Property 4, the path $\sigma(\pi)$ must include (v, i), and therefore $(v, i) \in \sigma(H_i)$. ◀

For a vertex $(v,x) \in \tilde{G}_v$, let σ' be the isomorphism between $\tilde{G}_v$ and H_i given by $\sigma'(v,x) = \sigma(x)$. Note that $\sigma'(\tilde{G}_v) \subseteq \sigma(H_i) \subseteq \tilde{G}_v$.

We now construct a sequence of isomorphisms $\sigma_1, \sigma_2, \ldots : H \to \tilde{G}_v$ as follows:

- $\sigma_0 = \sigma$,
- $\sigma_{j+1}(x) = \sigma'(\sigma_j(x))$ for any $j \geq 0, x \in H$.

Note that a-priori, this sequence is not necessarily well-defined, because σ' can map nodes of $\tilde{G}_v$ to nodes outside $\tilde{G}_v$. We will show that this does not happen to nodes of H, so that we get infinitely many copies of H inside $\tilde{G}_v$.

For any $j > 0$, let

$$d_j = \mathrm{dist}_{\tilde{G}_v}((v,i), \sigma_j(i)).$$

Also, let π_j be a path of length d_j between (v,i) and $\sigma_j(i)$ in $\tilde{G}_v$. (These definitions assume that the sequence $\sigma_1, \ldots, \sigma_j$ is well-defined up to index j, and accordingly we will only use them under this assumption.)

▶ **Observation 12.** *Let $\pi \subseteq \tilde{G}_v$ be a simple path between two vertices $(v,x),(v,y) \in \tilde{G}_v$, such that $\sigma'(v,x), \sigma'(v,y) \in \tilde{G}_v$ as well. Then $\sigma'(\pi) \subseteq \tilde{G}_v$.*

Proof. Assume for contradiction that $\sigma'(\pi) \not\subseteq \tilde{G}_v$, and let $(v,w) \in \pi$ be some vertex such that $\sigma'(v,w) \notin \tilde{G}_v$. Split π into two sub-paths, π_1 and π_2, where π_1 connects (v,x) to (v,w), and π_2 connects (v,w) to (v,y). The isomorphism σ' maps π_1, π_2 into two simple paths $\sigma'(\pi_1), \sigma'(\pi_2)$ connecting $\sigma'(v,x)$ to $\sigma'(v,w)$ and $\sigma'(v,w)$ to $\sigma'(v,y)$, respectively. Since $\sigma'(v,w) \notin \tilde{G}_v$ and $\sigma'(v,x), \sigma'(v,y) \in \tilde{G}_v$, Property 4 asserts that $\sigma(\pi_1)$ and $\sigma(\pi_2)$ both include node (v,i). But this means that $\sigma(\pi) = \sigma(\pi_1)\sigma(\pi_2)$ is not a simple path, which is a contradiction, as π is simple and σ' is bijective. ◀

For convenience, let us denote $\sigma_0(i) = (v,i)$.

▶ **Claim 13.** *Fix $k > 0$, and assume that $\sigma_j(H) \subseteq \tilde{G}_v$ for each $j < k$. Then for any $0 \leq j < k$ and $0 < \ell < k$ with $j + \ell < k$, there is a simple path $\pi \subseteq \tilde{G}_v$ of length d_ℓ connecting $\sigma_j(i)$ and $\sigma_{j+\ell}(i)$.*

Proof. By induction on j.

For $j = 0$, this is immediate from the definition of d_ℓ as the distance in $\tilde{G}_v$ between $\sigma_0(i) = (v,i)$ and $\sigma_\ell(i)$.

For the induction step, suppose that the claim holds for j: there is a simple path $\pi \subseteq \tilde{G}_v$ of length d_ℓ connecting $\sigma_j(i)$ and $\sigma_{j+\ell}(i)$. Assume that $j + 1 + \ell < k$, and recall that we assumed $\sigma_r(H) \subseteq \tilde{G}_v$ for each $0 \leq r < k$. Then in particular, $\sigma_{j+1}(i) = \sigma'(\sigma_j(i)) \in \tilde{G}_v$ and $\sigma_{j+\ell+1}(i) = \sigma'(\sigma_{j+\ell}(i)) \in \tilde{G}_v$. Thus we can apply Observation 12 to get that $\sigma'(\pi) \subseteq \tilde{G}_v$. This proves the claim, because $\sigma'(\pi)$ connects $\sigma_{j+1}(i)$ and $\sigma_{j+\ell+1}(i)$ and has length d_ℓ. ◀

▶ **Corollary 14.** *Fix $k > 0$, and assume that $\sigma_j(i) \in \tilde{G}_v$ for each $j < k$. Then for any $0 \leq j < k$ and $0 < \ell < k$ with $j + \ell + 1 \leq k$, there is a simple path of length d_ℓ connecting $\sigma_j(i)$ and $\sigma_{j+\ell}(i)$.*

Proof. If $j + \ell + 1 < k$, this is Claim 13. If $j = 0$, this follows from the definition of d_j, as in the base case of Claim 13. Finally, if $j + \ell + 1 = k$, then Claim 13 shows that there is a path of length d_ℓ between $\sigma_{j-1}(i) \in \tilde{G}_v$ and $\sigma_{j+\ell-1}(i) \in \tilde{G}_v$, and applying σ' once more yields a path of length d_ℓ between $\sigma'(\sigma_{j-1}(i)) = \sigma_j(i)$ and $\sigma'(\sigma_{j+\ell-1}(i)) = \sigma_{j+\ell}(i)$. ◀

▶ **Claim 15.** *For each $k \geq 0$ we have $\sigma_k(H) \subseteq \tilde{G}_v$.*

Proof. By induction on k. The base case, $k = 0$, is by choice of v, i.

For the induction step, let $k \geq 1$, and assume that $\sigma_j(H) \subseteq \tilde{G}_v$ for each $j < k$.

It suffices to find some $j < k$ such that there is a path of length at most d_j between $\sigma_j(i)$ and $\sigma_k(i)$: since d_j is the distance from $\sigma_j(i)$ to (v, i), any path of length at most d_j starting at $\sigma_j(i)$ ends inside $\tilde{G}_v$. We therefore get that $\sigma_k(i) \in \tilde{G}_v$, and since any copy of H is entirely contained in some $\tilde{G}_w$ for $w \in V(G)$ (Observation 9), $\sigma_k(H) \subseteq \tilde{G}_v$.

Let us write $k = 2j + b$ for $j > 0$ and $b \in \{0, 1\}$. By Corollary 14, there is a path $\pi \subseteq \tilde{G}_v$ of length d_j connecting $\sigma_{j+b}(i)$ and $\sigma_{2j+b}(i) = \sigma_k(i)$. If k is even, i.e., $b = 0$, then we are done, as there is a path of length d_j between $\sigma_j(i)$ and $\sigma_k(i)$.

Suppose now that k is odd, i.e., $b = 1$. Then we just showed that there is a path of length d_j connecting $\sigma_{j+1}(i)$ and $\sigma_k(i)$. If $d_j \leq d_{j+1}$, then we are done. Otherwise, $d_j > d_{j+1}$. By Corollary 14, there is a path of length $d_{j+1} < d_j$ between $\sigma_j(i)$ and $\sigma_{2j+1}(i) = \sigma_k(i)$, so again we are done. ◀

Next we show that the copies $\sigma_1(H), \sigma_2(H), \ldots$ cannot overlap completely.

▶ **Claim 16.** *For each $1 \leq j < k$ we have $\sigma_j(H) \neq \sigma_k(H)$.*

Proof. By induction on j.

For the base case, observe that for any $k > 1$ we have $\sigma(H) \neq \sigma_k(H)$: since $H \cap H_i = \{i\}$, we know that $\sigma(H) \cap \sigma(H_i) = \{\sigma(i)\}$, but for all $k > 1$ we have $\sigma_k(H) \subseteq \sigma(H_i)$. Therefore $\sigma(H) \cap \sigma_k(H) \subseteq \{\sigma(i)\}$, and since $|H| > 1$ we get that $\sigma(H) \neq \sigma_k(H)$.

Now suppose the claim holds for j, and suppose for the sake of contradiction that $\sigma_{j+1}(H) = \sigma_k(H)$ for some $k > j+1$. Since $\sigma_{j+1}(H) = \sigma'(\sigma_j(H))$ and $\sigma_k(H) = \sigma'(\sigma_{k-1}(H))$, and σ' is bijective, we get that $\sigma_j(H) = \sigma_{k-1}(H)$, which contradicts the induction hypothesis (as $k > j - 1$). ◀

We have reached a contradiction: we have an infinite sequence of copies $\sigma_1(H), \sigma_2(H), \ldots$ all contained in $\tilde{G}_v$ but no two are identical. This is impossible, because $\tilde{G}_v$ is finite.

3.2 Lower Bound for Randomized Algorithms

We can now use the reduction described above to prove Theorem 7 for randomized algorithms.

Proof of Theorem 7 for randomized algorithms. Fix an algorithm $\tilde{A}$ for solving $\tilde{H}$-freeness, with success probability $p > 1/2$ and running time $t(n)$.

Our first step is to construct from $\tilde{A}$ an algorithm $\tilde{A}'$, such that in graphs of size $s \cdot n$, the success probability of $\tilde{A}'$ in solving $\tilde{H}$-freeness is at least $p' \geq p$ such that $1 - p' < p'/s^s$, that is, $p' > 1/(1 + 1/s^s)$. To do this, we execute $\tilde{A}$ sequentially $C \cdot \log n$ times for some constant C, and have each vertex accept iff the majority of iterations of $\tilde{A}$ accepted. If the graph is not $\tilde{H}$-free, the expected number of executions that accept is at least $Cp \log n$, and if the graph is $\tilde{H}$-free, the expected number of iterations that accept is at most $C(1 - p) \log n$. We choose C a sufficiently large constant so that by Chernoff, if the graph contains a copy of $\tilde{H}$, then a given vertex rejects with probability at most $p'/(sn)$, and if the graph is $\tilde{H}$-free, then the probability that all vertices accept is at most p'. A union bound then gives the desired success probability for $\tilde{A}'$.

Next, we use the transformation described above to obtain an algorithm A for solving H-freeness with running time $(Ct(sn) \log(n))$ in graphs of size sn. In graphs that contain a copy of H, the probability that A accepts is at least p'/s^s, and in graphs that are H-free the probability that A accepts is at most $1 - p'$. Since we chose p' so that $1 - p' < p'/s^s$, we can again use $C' \cdot \log n$ iterations of A for some constant C' to increase the success probability to $2/3$. The resulting running time is $C \cdot C' \cdot t(sn) \log^2 n$, and we know that $\Omega(n^\delta)$ rounds are required for solving H-freeness; therefore, $t(N) = \Omega(N^\delta / \log^2 N)$. ◀

3.3 Derandomizing the Reduction

To obtain a deterministic reduction, we use a trick often used in the context of graph coloring. We fix a set of L color assignments $c_1, \ldots, c_L : [n] \to [s]$ with the following property:[3]

▶ **Property 17.** *Take any graph G on n vertices containing a copy of H, and let $\sigma : H \hookrightarrow G$ map H onto its copy in G. Then $\sigma(H)$ is properly colored by at least one color assignment c_i for $i \in [L]$.*

▶ **Lemma 18.** *There is a list $c_1, \ldots, c_L$ with $L = O(\log n)$ satisfying Property 17.*

Proof. We choose L random color assignments, and show that for $L = O(\log n)$, the probability that the list we sampled satisfies Property 17 is greater than zero.

Let $s = |V(H)|$, and let $L = \alpha \log n$, where $\alpha \geq 1$ is a constant we will fix later. Consider a specific copy $\sigma(H)$ of H, identified by the vertices $v_1, \ldots, v_s \in [n]$. The probability that a random assignment $c : V(G) \to V(H)$ colors $\sigma(H)$ properly is $1/s^s$. The probability that c *fails* to color $\sigma(H)$ properly is $1 - 1/s^s$, and the probability that L random assignments all fail to color $\sigma(H)$ properly is

$$\left(1 - \frac{1}{s^s}\right)^L \leq \left(e^{-1/s^s}\right)^L = e^{-L/s^s} = \left(\frac{1}{n}\right)^{\alpha/s^s}.$$

By union bound, the probability that a list of L random color assignments is bad for *any* copy of H is bounded by $n^s \cdot (1/n)^{\alpha/s^s}$, and choosing α a sufficiently large constant, this probability is strictly smaller than one. ◀

▶ **Corollary 19.** *If solving H-freeness requires $\Omega(n^\delta)$ rounds in graphs of size n for deterministic algorithms, then solving $\tilde{H}$-freeness also requires $\Omega(n^\delta / \log n)$ rounds for deterministic algorithms.*

Proof. Fix $c_1, \ldots, c_L : V(G) \to V(H)$ satisfying Property 17, with $L = O(\log n)$.

Given a deterministic algorithm $\tilde{A}$ for testing $\tilde{H}$-freeness, we construct a deterministic algorithm A for testing H-freeness, by repeating the simulation from the proof of Theorem 7 with each c_i for $i \in [L]$. If at least one iteration of $\tilde{A}$ accepts, then A accepts.

If $\tilde{A}$ requires $t(n)$ rounds in graphs of size n, then A requires $t(sn) \log n$ rounds.

When we run A in a graph that does not contain a copy of H, we showed that *no* color assignment produces a copy of $\tilde{H}$, and therefore $\tilde{A}$ rejects in all of its iterations. Therefore, A rejects as well.

When we run A in a graph that contains a copy of H, there is some i such that c_i colors the copy of H properly, and in this case we showed that the virtual graph $\tilde{G}$ contains a copy of $\tilde{H}$. In iteration i, $\tilde{A}$ accepts, and therefore so does A. ◀

4 Edge-Replacement Reduction

In this section we describe another reduction, which replaces the *edges* of a graph with other graphs. We choose to work with the 4-cycle C_4 as our starting graph whose edges we replace. (Any larger cycle would do just as well but would not give us any additional results, as larger cycles can be constructed from C_4 by replacing a single edge with a line comprising several edges.)

[3] Here we assume that the nodes of the graph in which our algorithm is executed have IDs $\{1, \ldots, n\}$, but this assumption is not necessary. The IDs can be drawn from a polynomially-large namespace; the proof of Lemma 18 still goes through, with a larger constant.

▶ **Definition 20** (The class $\mathcal{B}$)**.** A graph H on vertices $V = \{0, \ldots, k-1\}$ is in $\mathcal{B}$ if it satisfies the following conditions:

(1) H is 2-connected, and

(2) There are four subsets $V_0, \ldots, V_3 \subseteq V$ such that

- **(a)** $V_i \cap V_{(i+1) \bmod 4} = \{(i+1) \bmod 4\}$ for each $i \in \{0, \ldots, 3\}$,
- **(b)** $V_i \cap V_j = \emptyset$ for each $i, j \in \{0, \ldots, 3\}$ where $j \neq (i+1) \bmod 4$,
- **(c)** The subgraph H_i induced by H on V_i is connected,
- **(d)** For each $i \in [3]$, the graph H_i does not contain a copy of the other three attached to each other, $\bigcup_{j \neq i} H_j$.

For an illustration, see Figure 1b. We believe the last requirement is not necessary for the reduction, but our current proof of soundness requires it.

Our approach for proving a lower bound for graphs in the class $\mathcal{B}$ is to modify the reduction from [7], which was originally used to show a lower bound of $\tilde{\Omega}(\sqrt{n})$ for checking C_4-freeness. As in [7], we require dense graphs that are free of cycles up to some length. We use the construction of [19], as stated in [14], Theorem 4.47:

▶ **Theorem 21** ([19])**.** *For any $n, g \geq 1$ there is a bipartite graph on n vertices, with girth at least $2g+2$, and $\Omega(n^{1+\epsilon(g)})$ edges, where*

$$\epsilon(g) = \begin{cases} 2/(3g-3), & \text{if } g \text{ is odd,} \\ 2/(3g-3+1), & \text{if } g \text{ is even.} \end{cases}$$

Now we can state our result:

▶ **Theorem 22.** *For each $H \in \mathcal{B}$ there is some $\delta \in (0, 1/2]$ such that testing H-freeness requires $\Omega(n^{\delta})$ rounds for randomized algorithms.*

The proof is a reduction from two-party communication complexity: we show that if there is a fast algorithm for H-freeness for some $H \in \mathcal{B}$, then we can construct from this algorithm a communication-efficient protocol for set disjointness. Because we know that set disjointness requires $\Omega(n)$ bits of communication, we obtain a lower bound on the number of rounds required for H-freeness for $H \in \mathcal{B}$.

Proof of Theorem 22. For each $i = 0, \ldots, 3$, let k_i be the distance between i and $(i+1) \bmod 4$ in H_i, and let $k = \sum_{i=0}^{3} k_i$. Assume w.l.o.g. that $k_0 + k_2 \geq k_1 + k_3$ and $k_2 \geq k_0$. Observe that H contains the cycle C_k, passing through nodes $0, 1, 2, 3$ in this order.

Fix a bipartite graph F with girth greater than $\lfloor k/k_0 \rfloor$ and with m edges, $\{e_1, \ldots, e_m\}$. Direct the edges from one side to the other, and let us denote $(u, v)^{-1} = (v, u)$.

We reduce from set disjointness on m elements as follows.

Given inputs $X, Y \subseteq [m]$, Alice and Bob jointly construct a graph $G_{X,Y}$, which consists of many copies of $H_0, \ldots, H_3$, where

- Alice decides which copies of H_0 to include in $G_{X,Y}$, depending on her input X;
- Bob decides which copies of H_1 to include in $G_{X,Y}$, depending on his input Y;
- The set of copies of H_1, H_3 that are included in $G_{X,Y}$ is fixed and does not depend on the input.

The various copies of $H_0, \ldots, H_3$ are connected to each other at vertices $0, \ldots, 3$ in such a way that a copy of H appears iff $X \cap Y \neq \emptyset$.

Each of the copies of H_i is associated with a directed edge $e \in [n]^2$, and accordingly, we name each copy H_i^e, where i is the index of the graph H_i of which H_i^e is a copy, and $e \in [n]^2$ is the edge with which it is associated.

We now describe the construction formally.

Step 1: Creating copies of $H_0, \ldots, H_3$

For an edge $e = (u, v) \in [n]^2$ and a graph H_i where $i \in [3]$, we create a *copy of* H_i denoted $H_i^{(u,v)}$, over the vertices

$$V\left(H_i^{(u,v)}\right) = \{(i, u), ((i+1) \bmod 4, v)\} \cup \{(x, i, (u, v)) \mid x \in V(H_i) \setminus \{i, (i+1) \bmod 4\}\}.$$

The original H_i is mapped onto its copy $H_i^{(u,v)}$ by the isomorphism $\sigma_i^{(u,v)}$, defined as follows:

- The vertices i and $(i+1) \bmod 4$ of H_i are mapped by $\sigma_i^{(u,v)}$ to $(i, u), ((i+1) \bmod 4, v)$ in $H_i^{(u,v)}$, respectively. Vertices y are called the *endpoints* of $H_i^{(u,v)}$. Note that these vertices can be *shared* between different copies $H_i^{e_1}$ and $H_j^{e_2}$; we elaborate below, in Property 23.
- Any other vertex, $x \in V(H_i) \setminus \{i, (i+1) \bmod 4\}$, is mapped by $\sigma_i^{(u,v)}$ to a "fresh vertex" denoted $(x, i, (u, v))$, which is not shared with any other copy.

As we said, some of the copies share vertices; the following property characterizes the copies that do and do not share vertices, and which vertices are shared.

▶ **Property 23.** *For any* $(i, u_i, v_i) \neq (j, u_j, v_j)$,

- *If* $j = (i+1) \bmod 4$ *and* $v_i = u_j$, *then* $V(H_i^{(u_i,v_i)}) \cap V(H_j^{(u_j,v_j)}) = \{((i+1) \bmod 4, v_i)\}$;
- *If* $i = (j+1) \bmod 4$ *and* $v_j = u_i$, *then* $V(H_i^{(u_i,v_i)}) \cap V(H_j^{(u_j,v_j)}) = \{((j+1) \bmod 4, v_j)\}$;
- *If neither condition above holds, then* $V(H_i^{(u_i,v_i)}) \cap V(H_j^{(u_j,v_j)}) = \emptyset$.

Selecting copies for $G_{X,Y}$

Not all the copies H_i^e described above actually appear in the graph $G_{X,Y}$ constructed by the players; as we said, the players choose which copies to include based on their inputs.

Recall that F is a bipartite graph with girth greater than $\lfloor k/k_0 \rfloor$ and with m edges, $\{e_1, \ldots, e_m\}$, which we directed from one side to the other. Recall also that $X, Y \subseteq [m]$, so Alice and Bob can view their inputs as sets of edges from F. The graph $G_{X,Y}$ is constructed by attaching together four sets of copies:

$$G_{X,Y} = \bigcup_{i=0}^{3} \left\{ H_i^e \mid e \in E_i^{X,Y} \right\},$$

where the edge set $E_i^{X,Y}$ is given by:

- For $i \in \{1, 3\}$ we define $E_i^{X,Y} = \{(u, u) \mid u \in [n]\}$,
- For $i = 0$ we define $E_0^{X,Y} = \{e_x \mid x \in X\}$, and
- For $i = 2$ we define $E_2^{X,Y} = \left\{e_y^{-1} \mid y \in Y\right\}$.

The Simulation

We describe how Alice and Bob simulate the execution of a distributed algorithm for testing H-freeness in the graph $G_{X,Y}$. Let G_A be the subgraph of $G_{X,Y}$ induced by all the nodes participating in copies H_0^e for some $e \in F$, and let G_B be the subgraph of $G_{X,Y}$ induced by all the nodes participating in copies $H_2^{e^{-1}}$ for some $e \in F$. Alice simulates all the nodes in G_A, and Bob simulates the nodes in G_B; the remaining nodes are simulated by both players, using public randomness to agree on their random choices. Note that by construction, each player knows all internal edges on their side of the graph (G_A, G_B respectively), and both players know the structure of the shared part of the graph.

In each round, Alice sends Bob the messages sent on each edge adjacent to nodes $(0, u)$, $(1, u)$ for each $u \in [n]$, and Bob sends Alice the messages sent on each edge adjacent to nodes $(2, u)$, $(3, u)$ for each $u \in [n]$. The players then feed these messages to the nodes they simulate, and also compute the messages sent on internal edges in G_A (for Alice) or G_B (for Bob), and feed them in as well.

If Δ is the total degree of nodes $0, 1, 2, 3$ in H, then the total number of bits required for each round of the simulation is $\Delta n \log n = O(n \log n)$.

Correctness of the Reduction

First, suppose $x \in X \cap Y$, and let $e_x = (u, v)$. Then Alice and Bob both add copies $H_0^{(u,v)}, H_2^{(v,u)}$, with $H_0^{(u,v)}$ connecting nodes $(0, u)$ and $(1, v)$, and $H_2^{(v,u)}$ connecting nodes $(2, v)$ and $(3, u)$. Because in $G_{X,Y}$ there is always a copy $H_1^{(v,v)}$ of H_1 connecting $(1, v)$ to $(2, v)$, and there is always a copy $H_3^{(u,u)}$ of H_3 connecting $(3, u)$ to $(0, u)$, we get a complete copy of H.

Now suppose that $X \cap Y = \emptyset$, and assume for the sake of contradiction that $G_{X,Y}$ contains a copy of H.

▶ **Observation 24.** *In $G_{X,Y}$ there is no copy of H which intersects at most one copy $H_i^{(u,v)}$ for each $i \in [3]$.*

Proof. Suppose there is such a copy. Because $|V(H)| = \sum_{i=0}^{3} |V_i|$, this copy of H must also intersect at *least* one copy $H_i^{(u_i,v_i)}$ for each $i \in [3]$, and since H is connected, these four subgraphs must share vertices (as there are no edges between the non-shared vertices of two distinct copies). By Property 23, we must have $v_i = u_{(i+1) \bmod 4}$ for each $i \in [3]$. In particular, $v_0 = u_1 = v_1 = u_2$, and $v_2 = u_3 = v_3 = u_0$. But such copies would only be added to the graph if there exist edges $e_x = (u_0, v_0) \in E(F)$ with $x \in X$, and $e_y = (v_2, u_2) \in E(F)$ with $y \in Y$. Since $v_2 = u_0$ and $u_2 = v_0$, we get that $x = y$ and hence $X \cap Y \neq \emptyset$, contrary to our assumption. ◀

▶ **Claim 25.** *In $G_{X,Y}$, any cycle C_d with $d \leq k$ is entirely contained in some copy H_i^e for $i \in [3]$ and $e \in [n]^2$.*

Proof. Consider a cycle C_d in $G_{X,Y}$ which is not entirely contained in any copy H_i^x. Let $(i_1, e_1), \ldots, (i_a, e_a)$ be the sequence of copies of $H_{i_j}^{e_j}$ through which C_d passes. Then $C' = e_1, \ldots, e_a$ is a cycle: the various copies connect to each other only at their endpoints, and if C_d enters some copy through one endpoint it must exit at the other (it is a simple cycle, so it cannot exit through the vertex where it entered).

Recall that F is a bipartite graph with girth greater than $\lfloor k/k_0 \rfloor$.

Consider first a cycle C_d completely contained in Alice's side of the graph, G_A. Then each edge e_j of C' corresponds to a passage through copy $H_0^{e_j}$ and adds at least k_0 edges, as this is the distance between vertices 0 and 1 inside H_0. However, the girth of F is greater than $\lfloor k/k_0 \rfloor$, so $|C'| > \lfloor k/k_0 \rfloor$, that is, C_d must pass through at least $\lfloor k/k_0 \rfloor + 1$ copies before the cycle can close. This yields a total length of at least $(\lfloor k/k_0 \rfloor + 1) \cdot k_0 > k \geq d$.

Next, suppose that C_d is entirely contained in G_B. The same argument holds, except that each passage through a copy of H_2 adds k_2 edges instead of k_0. Since $k_2 \geq k_0$ by assumption, we still get a total length of at least $(\lfloor k/k_0 \rfloor + 1) \cdot k_2 \geq (\lfloor k/k_0 \rfloor + 1) \cdot k_0 > k \geq d$.

Finally, suppose C_d is not entirely contained in G_A or in G_B. Because the copies $\left\{H_1^{(u,u)}, H_3^{(u,u)}\right\}_{u \in [n]}$ are not connected to each other, and C_d is not contained in any single

copy, it must go through some copies of H_0 and H_2 as well. We know from Observation 24 that C_d *cannot* use only one copy of H_i for each $i \in [3]$.

Since F is bipartite, C_d must use at least two copies of H_0 from Alice's side, two copies of H_2 from Bob's side, and either two copies of H_1 or two copies of H_3. Therefore the length of the cycle is at least $2(k_0 + k_2 + \min(k_1, k_3)) \geq k + 2\min(k_1, k_3) > k \geq d$. ◀

▶ **Corollary 26.** *There exist $i \in [3]$ and $u, v \in [n]$ such that for each $j \neq i, j \in [3]$ we have $\sigma(H_j) \subseteq H_i^{(u,v)}$.*

Proof. From the definition of the class $\mathcal{B}$, the graph H contains a cycle C_k passing through vertices $0, 1, 2, 3$, and from Observation 25, σ maps this cycle into some copy, that is, $\sigma(C_k) \subseteq H_i^{(u,v)}$ for some $i \in [3]$ and $u, v \in [n]$.

Recall that $H_i^{(u,v)}$ is connected to the rest of $G_{X,Y}$ only at its own endpoints (i, u) and $((i+1) \bmod 4, v)$, and that in H, each subgraph H_j connects to the next subgraph $H_{(j+1) \bmod 4}$ at the endpoint $(j+1) \bmod 4$. This means that there is at most one $j \in [3]$ such that $\sigma(H_j) \not\subseteq H_i^{(u,v)}$: all paths from $\sigma(0), \ldots, \sigma(3)$ to vertices outside $H_i^{(u,v)}$ must pass through an endpoint of $H_i^{(u,v)}$, either (i, u) or $((i+1) \bmod 4, v)$. Thus, if there is some $x \in H_j$ such that $\sigma(x) \notin H_i^{(u,v)}$, then any path from $\sigma(x)$ to $\sigma(j)$ must pass through (i, u) or through $((i+1) \bmod 4, v)$. In fact, because H is 2-connected, there must be *two* paths from $\sigma(x)$ to $\sigma(j)$, one using (i, u) and one using $((i+1) \bmod 4, v)$, otherwise removing one endpoint would disconnect H. But this implies that $(i, u), ((i+1) \bmod 4, v) \in \sigma(H_j)$, and hence for any $j' \neq j$ there is no path inside $\sigma(H_{j'})$ from $\sigma(j')$ to any node outside $H_i^{(u,v)}$ Because $H_{j'}$ is connected, we get that $\sigma(H_{j'}) \subseteq H_i^{(u,v)}$.

We have now shown that there is at most one $j \in [3]$ such that $\sigma(H_j) \not\subseteq H_i^{(u,v)}$. Note in addition that we cannot have $\sigma(H_i) \subseteq H_i^{(u,v)}$, as $H_i^{(u,v)}$ is isomorphic to H_i, and in addition at least two nodes in $H_i^{(u,v)}$ are not in the σ-image of H_i (nodes $(i+2) \bmod 4$ and $(i+3) \bmod 4$). It follows that if some H_j "escapes" $H_i^{(u,v)}$, i.e., if $\sigma(H_j) \not\subseteq H_i^{(u,v)}$, then $j = i$. In other words, for each $j \neq i$ we have $\sigma(H_j) \subseteq H_i^{(u,v)}$. ◀

From the corollary we get that there is some $i \in [3]$ which contains a copy of $\bigcup_{j \neq i} H_j$, contradicting condition (d) of the class $\mathcal{B}$. This concludes the proof of soundness for the reduction.

Putting Everything Together

Let $g = \lfloor k/k_0 \rfloor$. By Theorem 21, there is a bipartite graph with girth greater than g and $m = \Omega(n^{1+\epsilon})$ edges, where $\epsilon = \Theta(1/g)$. Set F to be such a graph.

The size of the graph $G_{X,Y}$ is bounded by $m \cdot (|V_0| + |V2|) + n \cdot (|V_1| + |V_3|) \leq n^{1+\epsilon} \cdot s$, where $s = |V(H)|$. We know that to solve disjointness over m elements we require a total of $\Omega(m)$ bits, and our simulation requires $O(n \log n)$ bits per round, so the number of rounds of any randomized algorithm testing H-freeness in $G_{X,Y}$ is $\Omega(m/(n \log n)) = \Omega(n^{\epsilon}/\log n)$ in the worst case.

Now let us express this result in terms of the size of the graph $G_{X,Y}$: set $N = |V(G_{X,Y})| \leq n^{1+\epsilon}s$, where s is the size of H (a constant). Then $n = \Theta(N^{1/(1+\epsilon)})$, and the running time we get is $\Omega(N^{\epsilon/(1+\epsilon)}/\log N)$ rounds. ◀

5 Conclusion

In this paper we made a step towards understanding which subgraphs are hard to detect in the distributed setting: we showed that if H is a 2-connected graph that is hard to detect, then any graph obtained from H by replacing its vertices with other graphs will also be hard; and that if we take a cycle and replace its edges with other graphs, then under some conditions, then resulting graph will still be hard.

In our view, the main open question raised by our work is the following: *is there a 2-connected graph H such that H-freeness can be solved in sub-polynomial time? Or are all 2-connected graphs hard?*

As we pointed out in Section 1, if indeed all 2-connected graphs are polynomially hard, then our first reduction implies that *any graph that is not a tree is polynomially hard*, yielding a strong dichotomy between trees, which only require $O(1)$ rounds to detect [8], and any other connected graph.

References

1 Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Christoph Lenzen. Fooling views: A new lower bound technique for distributed computations under congestion. *CoRR*, abs/1711.01623, 2017.

2 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

3 Zvika Brakerski and Boaz Patt-Shamir. Distributed discovery of large near-cliques. *Distributed Computing*, 24(2):79–89, 2011.

4 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. *Fast Distributed Algorithms for Testing Graph Properties*, pages 43–56. Springer, Berlin, Heidelberg, 2016. doi:10.1007/978-3-662-53426-7_4.

5 Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015*, pages 143–152, 2015.

6 Danny Dolev, Christoph Lenzen, and Shir Peled. *"Tri, Tri Again": Finding Triangles and Small Subgraphs in a Distributed Setting*, pages 195–209. Springer, Berlin, Heidelberg, 2012. doi:10.1007/978-3-642-33651-5_14.

7 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 '14, pages 367–376, 2014.

8 Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. Three Notes on Distributed Property Testing. In *31st International Symposium on Distributed Computing (DISC 2017)*, volume 91 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:30, 2017.

9 Guy Even, Reut Levi, and Moti Medina. Faster and simpler distributed algorithms for testing and correcting graph properties in the congest-model. *CoRR*, abs/1705.04898, 2017.

10 Orr Fischer, Tzlil Gonen, and Rotem Oshman. Distributed property testing for subgraph-freeness revisited. *CoRR*, abs/1705.04033, 2017.

11 Pierre Fraigniaud, Pedro Montealegre, Dennis Olivetti, Ivan Rapaport, and Ioan Todinca. Distributed subgraph detection. *CoRR*, abs/1706.03996, 2017.

12 Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. *Distributed Testing of Excluded Subgraphs*, pages 342–356. Springer, Berlin, Heidelberg, 2016. doi:10.1007/978-3-662-53426-7_25.

13 Eugene C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32(4):755–761, 1985.

14 Zoltán Füredi and Miklós Simonovits. *The History of Degenerate (Bipartite) Extremal Graph Problems*, pages 169–264. Springer, Berlin, Heidelberg, 2013.

15 Taisuke Izumi and François Le Gall. Triangle finding and listing in congest networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, pages 381–389, 2017.

16 Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

17 Janne H. Korhonen and Joel Rybicki. Deterministic subgraph detection in broadcast CONGEST. *CoRR*, abs/1705.10195, 2017.

18 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

19 Felix Lazebnik, Vasiliy A. Ustimenko, and Andrew J. Woldar. A new series of dense graphs of high girth. *Bull. Amer. Math. Soc.*, 32(1):73–39, 1995.

20 Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

21 J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.