Possibilities and Impossibilities for Distributed Subgraph Detection

Orr Fischer
Tel-Aviv University
Tel-Aviv, Israel
orrfischer@mail.tau.ac.il

Tzlil Gonen
Tel-Aviv University
Tel-Aviv, Israel
tzlilgon@mail.tau.ac.il

Fabian Kuhn
University of Freiburg
Freiburg, Germany
kuhn@cs.uni-freiburg.de

Rotem Oshman
Tel-Aviv University
Tel-Aviv, Israel
roshman@tau.ac.il

ABSTRACT

In the distributed subgraph detection problem, we are given a fixed subgraph H, and the network must decide whether the network graph contains a copy of H or not. Subgraph detection can be solved in a constant number of rounds if message size is unbounded, but in the CONGEST model, where each message has bounded size, it can have high round complexity. Distributed subgraph detection has received significant attention recently, with new upper and lower bounds, but several fundamental questions remain open.

In this paper we prove new possibility and impossibility results for subgraph detection in the CONGEST model. We show for the first time that some subgraphs require superlinear — in fact, nearly quadratic — running time, even in small-diameter networks. We also study cycle-detection, and show that any even cycle can be detected in sublinear time (in contrast to odd cycles, which require linear time). For the special case of triangle-detection, we show that deterministic algorithms require $\Omega(\log n)$ total communication even in graphs of degree 2, and that one-round randomized algorithms must send $\Omega(\Delta)$ bits in graphs of degree Δ, improving on the recent results of [Abboud et. al.]. Finally, we extend a recent lower bound of [Izumi, Le Gall] on listing all triangles to cliques of any size.

CCS CONCEPTS

• **Networks** → **Network algorithms**; • **Theory of computation** → **Distributed algorithms**; *Lower bounds and information complexity*;

ACM Reference Format:
Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. 2018. Possibilities and Impossibilities for Distributed Subgraph Detection. In *SPAA '18: 30th ACM Symposium on Parallelism in Algorithms and Architectures, July 16–18, 2018, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3210377.3210401

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SPAA '18, July 16–18, 2018, Vienna, Austria
© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-5799-9/18/07…$15.00
https://doi.org/10.1145/3210377.3210401

1 INTRODUCTION

In the *subgraph detection* problem (also called *subgraph freeness*), 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-detection is an extremely local problem: in the LOCAL model of distributed computing, where network nodes can send messages of unrestricted size in each round, the H-detection problem for any graph H of size k can be solved in at most $O(k)$ rounds — we simply have each node collect its entire k-neighborhood and check if it contains a copy of H. However, in the CONGEST model, where the bandwidth on each edge *is* restricted, subgraph detection can be much harder.

Subgraph-freeness in the CONGEST network model has recently received significant attention from the distributed computing community [1, 6, 10, 12–15, 18]. Several linear and sublinear upper and lower bounds have been shown for special cases, including triangles and larger cycles [1, 6, 10, 13]; trees, which can be detected in $O(1)$ rounds [12]; cliques and complete bipartite subgraphs, which can be detected in $O(n)$ rounds [10]; and more complicated classes of graphs, for which in some cases we can prove a lower bound of the form $\Omega(n^{\delta})$, where $\delta < 1$ [15].

In this work we aim to better understand several fundamental questions regarding the subgraph-freeness problem.

1.1 Our Results

Solving C_{2k}-detection in sublinear time. In [10] it was shown that for any odd cycle C_k for $k \geq 5$, the C_k-detection problem takes $\widetilde{\Omega}(n)$ rounds. It is easy to see that $O(n)$ rounds suffice, so this bound is nearly tight. For *even* cycles, on the other hand, the best known lower bound is $\Omega(\sqrt{n})$ [10, 18], but no sublinear upper bounds were known for $k \geq 6$, so it was unclear whether even cycles require nearly-linear time, or whether they can be detected faster. We show that in fact any even cycle can be detected in sublinear time:

Theorem 1.1. *For any fixed $k \geq 2$, C_{2k}-detection can be solved in $O(n^{1-1/(k(k-1))})$ rounds.*

(Our algorithm is randomized, but it is easily de-randomized using standard techniques, at the cost of an additional $O(\log n)$ factor in the running time. (See, e.g, [15] for a similar argument.)

Near-quadratic lower bounds on subgraph-freeness. A wide variety of subgraphs, such as cycles and cliques, can be detected in a linear number of rounds. A natural question that arises is whether *any* constant-sized subgraph H be detected in $O(n)$ rounds. We show that the answer is negative:

Theorem 1.2. *For any $k, n \geq 2$, there is a graph H_k of size $O(k)$ and diameter 3, such that H_k-freeness requires $\Omega(n^{2-1/k}/(Bk))$ rounds in the* CONGEST *model, even when the network diameter is 3.*

Our lower bound provides another "natural" problem which has a superlinear lower bound in the CONGEST model; to our knowledge, previously the only examples of such problems were found in [8]. Since the subgraph-freeness problem is also extremely local, we get a separation between the CONGEST and LOCAL models, nearly the largest possible: by Theorem 1.2, if we take $k = \Theta(\log n)$, we get a graph that requires $\widetilde{\Omega}(n^2)$ rounds to detect, but only $O(\log n)$ rounds in the LOCAL model.

We also show a variant of this theorem for a *bipartite* subgraph. We note that any bipartite graph H has Túran number $\text{ex}(n, H) \leq O(n^{2-\Omega(1)})$: any graph on n vertices that does *not* contain a copy of H cannot have more than $O(n^{2-\Omega(1)})$ edges. Therefore, any such graph can be detected in sub-quadratic time, by simply collecting all edges of the network graph. Thus, the H-detection problem for bipartite H is a somewhat natural intermediate problem in CONGEST: it is strongly sub-quadratic, yet super-linear.

Improved lower bounds on triangle-detection. Recently, [1] showed that for deterministic algorithms, if the bandwidth is $B = 1$ (i.e., one bit per round), then $\Omega(\log^* n)$ rounds are required, even in constant-degree graphs; and if the algorithm only has one round, and *each triangle node* must announce that it is in a triangle, then the bandwidth must be $\Omega(\Delta \log n)$, where Δ is the maximum degree of the graph.

Here we show:

(a) In any deterministic triangle-detection algorithm, even when the maximum degree is constant, $\Omega(\log n)$ bits must be sent on some edge, if the namespace is of size $\Theta(n)$; and
(b) Randomized one-round triangle-detection requires bandwidth $\Omega(\Delta)$ in graphs of maximum degree Δ.

The first result is shown using a technique similar to [1], but we strengthen the lower bound from $\Omega(\log^* n)$ to $\Omega(\log n)$, which is tight for the lower bound graph. The second result, for randomized algorithms, uses standard information theory instead of the fooling views technique introduced in [1] for their deterministic lower bound. This allows us to both generalize from deterministic to randomized algorithms, and lift the restriction that all three triangle nodes must detect that they are in a triangle (instead, we use the usual definition, where it suffices if one node detects the triangle).

One might wonder what happened to the $\log n$ factor that is present in the deterministic $\Omega(\Delta \log n)$ lower bound from [1] but missing in our $\Omega(\Delta)$ randomized lower bound. It is not clear whether randomized algorithms need to pay this logarithmic factor or not; we leave this question for future work.

Lower bound on listing s-cliques in the Congested Clique. Finally, we extend a recent lower bound of $\widetilde{\Omega}(n^{1/3})$ rounds on listing all triangles in the graph [16, 20], to a lower bound of $\widetilde{\Omega}(n^{1-2/s})$ for listing all s-cliques, for any $s \geq 3$. This lower bound holds even when each node can send $O(\log n)$ bits to each other node in every round. For lack of space, the details of this lower bound are relegated to the full paper. The proof is similar to the one in [16], and the main difference is that we need to prove the following lemma:

Lemma 1.3. *For $s \geq 2$, any graph on m edges has at most $O(m^{s/2})$ copies of K_s.*

This lemma generalizes a corresponding lemma for triangles from [23], which was used in the lower bound of [16].

For lack of space, many technical details and full proofs are omitted in the body of the paper; they appear in the full paper.

1.2 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 [24]. When H is fixed and only G is the input, the problem becomes solvable in polynomial time.

In the distributed setting, [12] very recently provided a constant-round deterministic algorithm for detecting a fixed tree in the CONGEST model. This paper, as well as several others [4, 6, 14] , 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. Here we consider the *exact* version.

Another recent work [16] 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. In [16, 20] a lower bound of $\widetilde{\Omega}(n^{1/3})$ on the round complexity of triangle listing in the congested clique is shown. There is also work on testing triangle-freeness in the congested clique model [7, 9] 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 [10] and restricted lower bounds for triangles [1], and the reductions from [15], which construct hard graphs from other hard graphs by replacing their vertices or their edges with other graphs. (In [10] 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 [10], which showed that $\Omega(\text{ex}(n, C_k)/B)$ rounds are required, where $\text{ex}(n, C_k)$ is the largest possible number of edges in a C_k-free graph over n vertices. In particular, for odd-length cycles, the lower bound of [10] is nearly linear. Very recently, [18] improved the lower bound for even-length cycles to $\Omega(\sqrt{n}/B)$. All lower bounds mentioned above are linear or sublinear.

In a recent work [8], the first near-quadratic lower bounds (and indeed the first superlinear lower bounds) in CONGEST were shown. The problems addressed in [8] include some NP-hard problems such as minimum vertex cover and graph coloring, as well as a weighted variant of C_8-freeness, called *weighted cycle detection*: given a weight $W \in [0, \text{poly}(n)]$, the goal is to determine whether the graph has a cycle of length 8 and weight exactly W.

2 PRELIMINARIES

Notation. We let $V(G), E(G)$ denote the vertex and edge set of graph G, respectively. In Section 5, which uses information theory, we use the convention that bold-face letters denote random variables, while normal letters denote concrete (scalar) values.

The CONGEST *model.* 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. Typically, B is taken to be logarithmic in the size n of the network graph.

Subgraph detection. We are interested in the *subgraph-detection* problem (also called *subgraph-freeness*), defined as follows:

Definition 1 (Subgraph detection)**.** *Fix a graph H. In the H-*detection *problem, the goal is to determine whether the input graph $G = (V, E)$ contains a copy of H as a subgraph or not; that is, whether there are subsets $U \subseteq V, F \subseteq E$ of vertices and edges, respectively, such that (U, F) is isomorphic to H.*

We say that a distributed algorithm A *solves H-detection with success probability p* if whenever A is executed in a graph that contains a copy of H, with probability at least p some node rejects; and on the other hand, when A is executed in a graph that is H-free, with probability at least p all nodes accept. We typically assume constant p, e.g., $p = 2/3$.

Túran numbers. For a fixed graph H, the *Túran number* of H, denoted $\mathrm{ex}(n, H)$, is the maximum number of edges in any H-free graph on n vertices.

Two-party communication complexity. Our superlinear lower bound is shown by reduction from *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 [19] 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 [17, 22] shows that even for randomized communication protocols, the players must exchange $\Omega(n)$ bits to solve set disjointness with constant success probability.

Information theory. In Section 5 we use information theory to prove a randomized lower bound on 1-round triangle-detection. The basic concept we need is the *mutual information* between two variables: if $\boldsymbol{X}, \boldsymbol{Y}$ are two possibly-dependent random variables, then their mutual information is defined to be $\mathrm{I}(\boldsymbol{X}; \boldsymbol{Y}) = \mathrm{H}(\boldsymbol{X}) - \mathrm{H}(\boldsymbol{X}|\boldsymbol{Y})$. Here, $\mathrm{H}(\boldsymbol{X})$ is the Shannon entropy of $\boldsymbol{X}$, and $\mathrm{H}(\boldsymbol{X}|\boldsymbol{Y}) = \mathrm{E}_y\left[\mathrm{H}(\boldsymbol{X}|\boldsymbol{Y} = y)\right]$ is the conditional Shannon entropy — the average entropy remaining in $\boldsymbol{X}$, after the value of $\boldsymbol{Y} = y$ is revealed. Intuitively, the mutual information $\mathrm{I}(\boldsymbol{X}; \boldsymbol{Y})$ measures the *uncertainly loss* of $\boldsymbol{X}$ when we learn $\boldsymbol{Y}$. (It is symmetric in $\boldsymbol{X}, \boldsymbol{Y}$.)

The *conditional* mutual information of $\boldsymbol{X}$ and $\boldsymbol{Y}$ given a third random variable $\boldsymbol{Z}$ is defined as the loss of conditional entropy: $\mathrm{I}(\boldsymbol{X}; \boldsymbol{Y}|\boldsymbol{Z}) = \mathrm{H}(\boldsymbol{X}|\boldsymbol{Z}) - \mathrm{H}(\boldsymbol{X}|\boldsymbol{Y}, \boldsymbol{Z})$. We sometimes abuse notation by writing *events* on the right-hand side instead of random variables, e.g., $\mathrm{I}(\boldsymbol{X}; \boldsymbol{Y}|\boldsymbol{Z} = z)$. When we do this, we are referring to the mutual information between $\boldsymbol{X}$ and $\boldsymbol{Y}$ when their joint distribution is conditioned on $\boldsymbol{Z} = z$.

3 SUPERLINEAR LOWER BOUNDS ON SUBGRAPH DETECTION

We begin by proving Theorem 1.2, namely, that some graphs require nearly-quadratic running time to detect.

We give here informal descriptions of the subgraph H_k and the lower bound graph family $\mathcal{G}$ in which we show that H_k-detection is hard. The formal constructions can be found in the full paper.

3.1 Informal Description of H_k

The graph H_k is composed of several "parts" (see Figure 1):

Cliques. We put in five cliques, one of each size $s = 6, \ldots, 10$. We pick one special vertex v_s from each clique, and connect $\{s_6, \ldots, s_{10}\}$ in a 5-clique. Each remaining (non-clique) vertex of H_k is connected to exactly one special clique vertex v_s, and no other clique vertices.

The cliques serve two purposes. First, they reduce the diameter of H_k to 3: each vertex in H_k is connected to some special clique vertex v_s, and all special clique vertices are connected to each other. Second, the cliques serve to "mark" the different parts of H_k: each "part" of H_k (except for the cliques themselves) is connected to exactly one s-clique for $s \in \{6, \ldots, 10\}$. When we construct the network graph G in which we show the lower bound on H_k-freeness, we will make sure G also contains exactly one copy of each s-clique for $s = 6, \ldots, 10$, so that any isomorphism mapping H_k into G must map the s-clique of H_k onto the s-clique in G. The "parts" of G will echo the "parts" of H_k, and the connections between the cliques and the other vertices will force any isomorphism from H_k into G to respect this logical partition.

Top and bottom. The remainder of H_k consists of two identical copies of a graph H. We call the two copies "top" and "bottom", respectively. The subgraph H consists of k triangles $\mathsf{Tri}_1, \ldots, \mathsf{Tri}_k$, and two additional "endpoint nodes", which we call A and B. In each triangle Tri_i, we have three vertices denoted $(i, \mathsf{A}), (i, \mathsf{B}), (i, \mathsf{Mid})$. Endpoint A is connected to all triangle vertices in $\{(i, \mathsf{A}) \mid i \in [k]\}$, endpoint B is connected to all triangle vertices in $\{(i, \mathsf{B}) \mid i \in [k]\}$, and the "middle vertices" of the triangles are not connected to either endpoint. (The triangles ensure that a copy of H cannot be embedded in any bipartite graph; we will need this for the lower bound.)

We refer to the triangles as the *body* of H_k, and the top and bottom endpoint nodes on both sides (A and B) as *endpoints*. There are no edges between the top and the bottom copy of H, except for exactly two edges: the top and bottom A-endpoints are connected by an edge, and the top and bottom B-endpoints are connected by an edge.

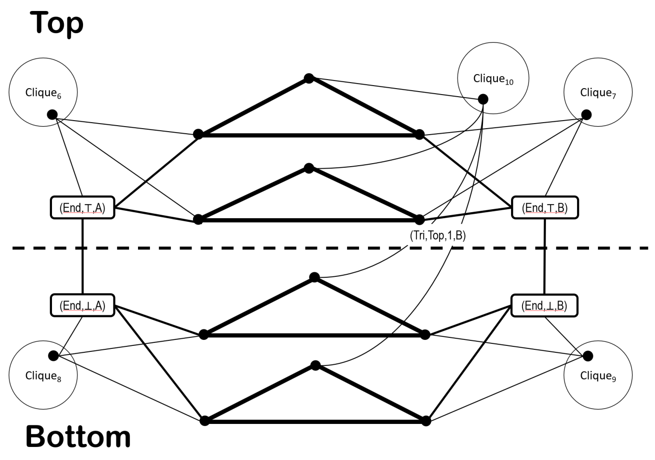

Figure 1: The graph H_k for $k = 2$. In addition to the edges shown here, the five clique nodes $\{(\text{Clique}_s, 0) \mid 6 \le s \le 10\}$ are connected to each other; we omit these edges from the figure for clarity.

3.2 The Lower Bound Family $\mathcal{G}_{k,n}$

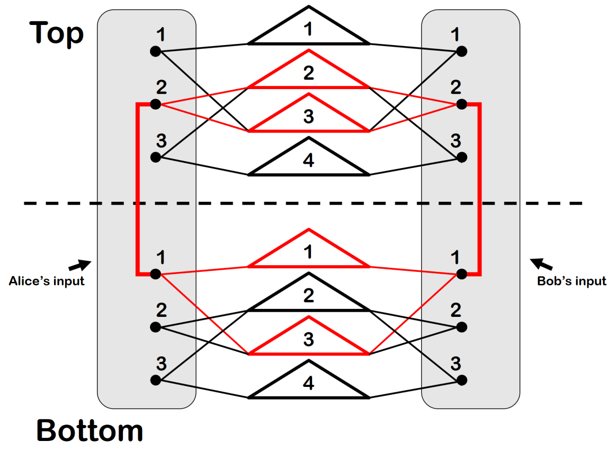

Figure 2: The graph $G_{X,Y} \in \mathcal{G}_{k,n}$ for $n = 3, k = 2$, so that $m = k\lceil n^{1/k}\rceil = 2 \cdot \lceil 3^{1/2}\rceil = 4$. For clarity, the figure omits the five cliques $\{\text{Clique}'_s \mid 6 \le s \le 10\}$ and their edges. The inputs X, Y whose graph $G_{X,Y}$ is depicted here both include $(2,1)$ (that is $(2,1) \in X \cap Y$). As a result, a copy of H_k appears in $G_{X,Y}$, highlighted in red.

Let us now describe the graph family $\mathcal{G}_{k,n}$, in which we show that H_k-detection requires $\Omega(n^{2-1/k}/(Bk))$ rounds. (Here, $\Theta(n)$ is the number of vertices in each graph $G \in \mathcal{G}_{k,n}$.)

Our lower bound graph family $\mathcal{G}_{k,n}$ echoes the graph H_k: it comprises many "potential copies" of the subgraph H_k, each containing all parts of H_k *except* the edges between the top and bottom endpoints. When we reduce from set disjointness, Alice and Bob will use their inputs to decide which endpoints to connect to which other endpoints: Alice will put in top-to-bottom edges between the A-endpoints, and Bob will do the same for the B-endpoints, in such a way that a complete copy of H_k appears iff the players' inputs are not disjoint. The players will then simulate the execution of an H_k-detection algorithm on the graph they constructed, with Alice simulating all the A-vertices, Bob simulating all the B-vertices, and both players simulating the "middle" vertices of the triangles.

The naïve way to construct $\mathcal{G}_{k,n}$ would be to take n disjoint copies of H_k, each missing the top-bottom endpoint edges, and have the players add the endpoint edges as we described above. This gives us n^2 *pairs* of endpoints on each side (A and B), so we can reduce from a disjointness instance of size n^2, and achieve our goal of having a copy of H_k appear iff the inputs are not disjoint. However, taking n disjoint copies of H_k would result in a *large cut*, of linear size, between the A-part of the graph and the B-part of the graph. This would make simulating the distributed algorithm too expensive: simulating one round would cost us $O(n \cdot B)$ bits, so the best round lower bound we could hope for would be $\Omega(n^2/(nB)) = \Omega(n/B)$.

To reduce the cut size, we do not take n *fully* disjoint copies of H_k. Instead, for top and bottom respectively, we take only $\Theta(kn^{1/k})$ disjoint triangles. Now we can mix-and-match among the $\Theta(kn^{1/k})$ triangles to form a copy of the body of H (which has k triangles). However, we still take n copies of the two endpoints of H (A/B), because we want to work with a disjointness universe of size n^2. Now we must connect the n endpoint-copies to the $\Theta(kn^{1/k}) \ll n$ triangles, in such a way that a complete copy of H_k appears iff Alice and Bob connected the same pair (i, j) of endpoint-copies on their respective sides (A and B).

Recall that in H we have k triangles, and the A- and B-endpoints are connected to opposite ends of each triangle, so the degree of each endpoint is k. Let us *encode* each index $i \in [n]$ as a subset P_i of k elements from the universe $\lceil kn^{1/k}\rceil$, in such a way that no two indices in $[n]$ have the same encoding. (This is possible, because $\binom{\lceil kn^{1/k}\rceil}{k} \ge n$.) Now, we connect the i-th endpoint to the k copies of H in its encoding: the i-th endpoint on side A (resp. B) is connected to the A-vertex (resp. B-vertex) of triangle j iff $j \in P_i$. Since the encoding is *unique*, and in H_k each of the top/bottom endpoints is connected to k top/bottom triangles (resp.), a complete copy of H_k can appear iff there is a pair $(i, j) \in [n]^2$, such that Alice connected the i-th copy of the top-A endpoint to the j-th copy of the bottom-A endpoint, and Bob did the same on his side (B).

The advantage of working with only $\Theta(kn^{1/k})$ triangles is that now the cut size is also $\Theta(kn^{1/k})$: the cut between the players' sides "cuts through" each triangle.

We now describe the construction formally. Let $m = k\lceil n^{1/k}\rceil$. Fix an ordering $Q_1, \ldots, Q_N$ of the subsets of size k of $[m]$, where $N = \binom{m}{k} = \binom{k\lceil n^{1/k}\rceil}{k}$. Note that

$$N = \binom{k\lceil n^{1/k}\rceil}{k} \ge \left(\frac{kn^{1/k}}{k}\right)^k = \frac{k^k n}{k^k} = n.$$

For each $i \in [N]$, let us denote $Q_i = \left\{q_i^1, \ldots, q_i^k\right\}$. (We will only use the first n subsets, $Q_1, \ldots, Q_n$.)

Definition 2 (The graph family $\mathcal{G}_{k,n}$). *Fix integers k, n. A graph G is in the family $\mathcal{G}_{k,n}$ if it has the following structure.*

First, the graph contains the following "components":

- *n "potential top endpoints" and "potential bottom endpoints" of H_k, denoted $\mathsf{End}' \times [n]$. For $S \in \{\top, \bot\}$ and $P \in \{\mathsf{A}, \mathsf{B}\}$, we denote*

$$\mathsf{End}'_{S,P} = \{(\mathsf{End}', S, P, i) \mid i \in [n]\}.$$

- *$2m$ triangles, indexed by $\{\top, \bot\} \times [m]$. Triangle (S, i) comprises vertices (Tri, S, i, P) for each $P \in \{\mathsf{A}, \mathsf{B}, \mathsf{Mid}\}$.*
- *Copies of each of the cliques in H_k.*

The graph contains the following edges between the components, and no other edges:

- *For each "direction" $(S, P) \in \{\top, \bot\} \times \{\mathsf{A}, \mathsf{B}\}$, all vertices in $\mathsf{End}'_{S,P}$ are connected to one fixed vertex of the $c_{S,P}$-clique.*
- *For each "direction" $(S, X) \in \{\top, \bot\} \times \{\mathsf{A}, \mathsf{B}, \mathsf{Mid}\}$, all vertices in $\mathsf{Tri}'_{S,X}$ are connected to one fixed vertex of the $c_{S,P}$-clique.*
- *The fixed vertices of the five cliques are connected to each other.*
- *For each $S \in \{\top, \bot\}$, $P \in \{\mathsf{A}, \mathsf{B}\}$, and $i \in [n]$, the endpoint-copy (End', S, P, i) is connected to each of the k triangle nodes (Tri, S, j, P) where $j \in Q_i$.*
- *Finally, for each $P \in \{\mathsf{A}, \mathsf{B}\}$, the graph may contain an arbitrary subset of the edges in $\mathsf{End}'_{\top,P} \times \mathsf{End}'_{\bot,P}$, depending on Alice and Bob's inputs..*

Property 1. *Any graph in $\mathcal{G}_{k,n}$ has diameter 3 and size $O(n)$.*

Crucially, the only copies of H_k that can appear in $\mathcal{G}_{k,n}$ have very specific structure: the endpoints of H_k must be mapped onto endpoints in $\mathcal{G}_{k,n}$, respecting the "top/down" partition and also the "directions" A, B. Consequently, we get:

Lemma 3.1. *A graph $G \in \mathcal{G}_{k,n}$ contains H_k as a subgraph iff there exist $i_\top, i_\bot \in [n]$ such that $(\mathsf{End}', \top, \mathsf{A}, i_\top), (\mathsf{End}', \bot, \mathsf{A}, i_\bot) \in E(G)$ and $(\mathsf{End}', \top, \mathsf{B}, i_\top), (\mathsf{End}', \bot, \mathsf{B}, i_\bot) \in E(G)$.*

3.3 The Lower Bound

To prove that H_k-detection requires $\Omega(n^{2-1/k}/(Bk))$ rounds, we give a reduction from 2-party set disjointness on the universe $[n]^2$.

Proof of Theorem 1.2. Fix an algorithm A for which solves H_k-freeness in the class $\mathcal{G}_{k,n}$, and let us construct from A a protocol for disjointness. Given inputs $X, Y \subseteq [n]^2$, Alice and Bob construct a graph $G_{X,Y} \in \mathcal{G}_{k,n}$. The only freedom when constructing a graph in $\mathcal{G}_{k,n}$ is the choice of the edges we take from $\mathsf{End}'_{\top,\mathsf{A}} \times \mathsf{End}'_{\bot,\mathsf{A}}$ and from $\mathsf{End}'_{\top,\mathsf{B}} \times \mathsf{End}'_{\bot,\mathsf{B}}$. For this choice the players use their inputs:

- Edge $\{(\mathsf{End}', \top, \mathsf{A}, i), (\mathsf{End}', \bot, \mathsf{A}, j)\}$ is included in $G_{X,Y}$ iff $(i, j) \in X$, and
- Edge $\{(\mathsf{End}', \top, \mathsf{B}, i), (\mathsf{End}', \bot, \mathsf{B}, j)\}$ is included in $G_{X,Y}$ iff $(i, j) \in Y$.

By Lemma 3.1, the graph $G_{X,Y}$ includes a copy of H_k as a subgraph iff $X \cap Y \neq \emptyset$. Thus, to solve their disjointness instance, Alice and Bob simulate the execution of A in $G_{X,Y}$, and output "$X \cap Y = \emptyset$" iff A rejects.

Let us describe the simulation. We partition $V(G_{X,Y})$ into three parts: Alice's part,

$$V_A = \bigcup_{S \in \{\top,\bot\}} \left(\mathsf{End}'_{S,\mathsf{A}} \times [n] \cup \mathsf{Tri}'_{S,\mathsf{A}}\right) \cup \mathsf{Clique}'_6 \cup \mathsf{Clique}'_8,$$

Bob's part,

$$V_B = \bigcup_{S \in \{\top,\bot\}} \left(\mathsf{End}'_{S,\mathsf{B}} \times [n] \cup \mathsf{Tri}'_{S,\mathsf{B}}\right) \cup \mathsf{Clique}'_7 \cup \mathsf{Clique}'_9,$$

, and a shared part, U, comprising the remaining vertices, $U = \mathsf{Tri}'_{\top,\mathsf{Mid}} \cup \mathsf{Tri}'_{\bot,\mathsf{Mid}} \cup \mathsf{Clique}'_{10}$. Note that each player knows all edges of $G_{X,Y}$, except those edges that are internal to the other player's part — those are the only edges that depend on the other player's input. For example, Alice knows all edges in $(V_A \cup V_B \cup U) \times (V_A \cup U)$. Thus, the players only need to tell each other about messages crossing the cut between the part only they simulate, and the rest of the graph. By construction of the family $\mathcal{G}_{k,n}$, the cut is of size $O(kn^{1/k})$, so the cost of the simulation is $O(kn^{1/k} \cdot B)$ bits per round. (We omit the details of the simulation here, as they are fairly standard.)

If A runs for R rounds, then the total cost of the simulation is $O(R \cdot kn^{1/k} \cdot B)$. Solving disjointness on $[n]^2$ requires $\Omega(n^2)$ bits. Thus, we must have $R = \Omega\left(n^2 / \left(kn^{1/k} \cdot B\right)\right) = \Omega(n^{2-1/k}/(Bk))$. □

3.4 A Superlinear Lower Bound for Bipartite Subgraphs

The superlinear lower bound described above is for a specific graph H_k which is not bipartite: we relied on the fact that the body of H_k contained *triangles*, while the subgraph connecting the top/bottom endpoints on each side is *bipartite*. This ensured that any embedding of H_k into the larger graph G had to "use" the triangles in G "in the role of" the triangles of H_k. We also used cliques (also not bipartite, of course) to "mark" the various parts of H_k.

Could it be that all bipartite subgraphs *can* be detected in linear time? It turns out that the answer is no: we show that for any $s, k > 1$, there is a bipartite graph $H_{s,k}$ of size $\Theta((s!)^2 \cdot k)$, such that B-freeness requires $\Omega(n^{2-1/k-1/s}/(Bk))$ rounds in graphs of size $\Theta(n)$. This construction follows the same approach as the non-bipartite one, but it is much more involved, because we cannot use non-bipartite "components" (e.g., triangles) to constrain an embedding of the subgraph into the graph $G_{X,Y}$ simulated by Alice and Bob. Instead, we rely on the fact that each endpoint in $G_{X,Y}$ has degree k. We restrict the edges that Alice and Bob can receive, and construct a bipartite "gadget" that cannot be embedded in $G_{X,Y}$ without using two endpoints from Alice's side and two endpoints from Bob's side, as intended. This gadget replaces the triangles, allowing the reduction to go through (with more effort).

4 LOWER BOUND ON DETERMINISTIC TRIANGLE-DETECTION

In this section we show:

Theorem 4.1. *Distinguishing a triangle from a 6-cycle deterministically requires $\Omega(\log(N))$ bits of communication in the* CONGEST *model, given a namespace of size N.*

As in the proof of the $\Omega(\log^* n)$ lower bound from [1], we rely on the following result from extremal graph theory: let $K^{(r)}(\ell)$ be the complete r-uniform r-partite hypergraph, where each side has size ℓ, (Formally, $K^{(r)}(\ell)$ is defined as follows: let $V_1, ..., V_r$ be disjoint vertex sets of size ℓ each. Then $K^{(r)}(\ell)$ is the hypergraph with vertex set $\bigcup_{i=1}^{r} V_i$ and edge set $E = \{(v_1, ..., v_r) \mid v_i \in V_i\}$.)

Theorem 4.2 ([11], Theorem 1). *If G is an r-uniform hypergraph that does not contain $K^{(r)}(\ell)$ as a subgraph, then $|E(G)| \le n^{r-1/(\ell^{r-1})}$.*

For our purposes, we take $r = 3$ and $\ell = 2$, and obtain that if G is 3-regular hypergraph with at least $n^{2.75}$ edges, then G contains the complete 3-uniform 3-partite graph where each side is of size 2.

We make two assumptions about the algorithms we consider: first, we assume that in each round, each node sends at least one bit. If we do not impose this restriction, nodes might communicate information to each other 'for free' by remaining silent in a given round (we do not charge them for *not* sending bits). Second, we assume that the algorithm's messages are self-delimiting, that is, they form a *prefix code*: if M_1 and M_2 are possible messages the algorithm can send, then M_1 is not a prefix of M_2 and vice-versa. Any algorithm can be transformed into an algorithm that has this property by at most doubling its total communication cost.

The intuition behind the lower bound is that if the bandwidth is so small that nodes cannot even send their neighbors' identifiers, then for some adversarial choice of identifiers, the algorithm cannot distinguish between a *triangle* and a *hexagon* (a 6-cycle). This idea was used in [1], and its roots go back to the proof of impossibility of Byzantine consensus with 3 processors of which one may be faulty [21].

The approach of [1] was *recursive*, showing that after each bit sent, the adversary can still use many identifiers to "fool" the algorithm. However, if before the i-th bit the adversary had n identifiers available, then after the i-th bit, [1] shows that it has roughly $\log n$; because of this fast shrinkage rate, the lower bound of [1] is $\Omega(\log^* n)$.

Here we do not take the recursive approach. Instead we argue that there is a *complete transcript* t of all messages sent by the algorithm during its execution, which occurs for many triangles (where each triangle is defined by the choice of identifiers for its nodes). As in [1], we construct a 3-regular hypergraph representing the adversary's choices, with nodes representing identifiers and each edge representing a triangle that yields the transcript t. Then we use Theorem 4.2 to argue that the adversary can choose two "compatible" triangles and combine them to form a hexagon, in a way that guarantees that each node's view is consistent with some triangle that generates transcript t. Thus, the adversary can make the algorithm reject the hexagon (which it is supposed to accept).

There are several subtle points in the proof: we must set up our construction carefully, to ensure that the algorithm's transcript really does capture "everything there is to know" about the algorithm's execution. In particular, from the transcript we must be able to read off the source and destination of each message. Because we are working with total communication less than $\log N$, we cannot use the naïve solution of assuming w.l.o.g. that this information is contained in the messages themselves, so we are careful about the order of messages when we define the representation of a transcript.

We remark that although the lower bound specifically shows that it is impossible to distinguish two constant-sized shapes (a triangle and a hexagon), it is easy to "pad" the graph to any desired size of at most $n < N/3$ nodes, by, e.g., attaching a fixed line of $\Theta(n)$ nodes to one of the triangle or hexagon nodes.

Proof of Theorem 4.1. Let A be deterministic algorithm that solves the triangle-freeness problem in the CONGEST model given a namespace of size $N = 3n$. We split N into three equal-size disjoint subsets, N_0, N_1, N_2, and define $S := N_0 \times N_1 \times N_2$. In the lower bound, we are interested in the class of graphs

$$\mathcal{G}_\triangle := \{\triangle(u_0, u_1, u_2) \mid (u_0, u_1, u_2) \in S\},$$

where $\triangle(u_0, u_1, u_2)$ is the graph consisting of a single triangle on nodes u_0, u_1, u_2.

Let C be the worst-case number of bits sent by a node in any execution of A. Assume for the sake of contradiction that $C \le \log(n)/60$.

Making decision values uniform. Our first step is to transform A into a modified algorithm A', which has one extra round. In A', nodes first execute A, and then in the extra round they send their decision according to A (i.e., whether they decided to accept or reject) to all their neighbors. Finally, each node v accepts iff under A, node v and all its neighbors decided to accept.

The communication cost of A' is $C + 1$. The transformation preserves the correctness of A, because A' accepts iff all nodes of A accept, and rejects iff some node rejects in A. It also has the following property, which will be important for our proof:

Claim 4.3. *When A' is executed in a graph that contains exactly one triangle, all nodes of the triangle reject.*

Transcripts. Next we formally define the *transcript* generated by a node executing A' in some triangle $\triangle(u, v, w)$. Although it is intuitively obvious "what this should mean", some care is required to ensure that transcripts can be parsed in a unique way, otherwise our proof does not go through.

Consider the graph $\triangle(u_0, u_1, u_2)$ for $(u_0, u_1, u_2) \in S$. (Recall that $S = N_0 \times N_1 \times N_2$.) We define *the transcript of node u_i in $\triangle(u_0, u_1, u_2)$*, denoted $\mathrm{Tr}_{u_0,u_1,u_2}(u_i)$, to be the binary string obtained by first writing all the messages sent by u_i to $u_{(i+1) \bmod 3}$ (in order, round after round), and then writing all the messages sent by u_i to $u_{(i+2) \bmod 3}$.

We define *the transcript of A' in* $\triangle(u_0, u_1, u_2)$ to be the concatenation of the nodes' transcripts:

$$\begin{aligned}\mathrm{Tr}(u_0, u_1, u_2) &= \\ &= \mathrm{Tr}_{u_0,u_1,u_2}(u_0)\,\mathrm{Tr}_{u_0,u_1,u_2}(u_1)\,\mathrm{Tr}_{u_0,u_1,u_2}(u_2).\end{aligned}$$

Note that the nodes' transcripts are written in order of the part of the namespace to which the nodes belong: first the transcript of u_0, which is a node from N_0, then the transcript of $u_1 \in N_1$ and finally $u_2 \in N_2$.

With this encoding, transcripts can be parsed uniquely: given the binary transcript and three nodes $(u_0, u_1, u_2) \in S$, we know exactly which message was sent by which node in which round. This is the reason we insisted that A's messages be a prefix code, partitioned the namespace into disjoint subsets, and took care to define the order in which messages appear in the transcript.

Since the total number of bits sent by any node to any other node is at most $C + 1$ in any execution of A', and the transcript contains all the messages sent by any node to any other node with no bits added, the length of any transcript generated by A' is at most $6(C + 1)$.

For each binary string t of length $6(C+1)$, define the subset $S_t \subseteq S$ to be the set of all triples in S on which A' produces transcript t: $S_t := \{(u_0, u_1, u_2) \in S \mid \text{Tr}(u_0, u_1, u_2) = t\}$.

Now fix t to be the transcript with the largest set S_t (or one of the transcripts maximizing $|S_t|$, if there is more than one). Observe that

$$|S_t| \geq \frac{|S|}{2^{6(C+1)}} = \frac{n^3}{2^{6(C+1)}}.$$

Consider the undirected 3-uniform hypergraph H with N as the vertex set, and edges given by the triplets in S_t. Note that although S_t is defined as a set of *ordered* triplets, because $S_t \subseteq N_0 \times N_1 \times N_2$, and N_0, N_1, N_2 are mutually disjoint, we may as well think of S_t as a set of unordered triplets.

The number of edges in H is $|E(H)| = |S_t| \geq n^3/2^{(6(C+1))}$. By assumption, $C \leq \log(n)/60 - 1$, so

$$|E(H)| \geq \frac{n^3}{2^{6 \cdot \log(n)/60}} = n^{2.9}.$$

By Theorem 4.2, if H has at least $\Omega(n^{2.75})$ edges, then it must contain a complete 3-partite 3-uniform hypergraph such that each part contains exactly two vertices. Let

$$F = (\{u_0, u_0'\}, \{u_1, u_1'\}, \{u_2, u_2'\})$$

and $E(F)$ be one such hypergraph, with the edge set $E(F)$ containing *every triplet* $\{x, y, z\}$ such that $x \in \{u_0, u_0'\}, y \in \{u_1, u_1'\}, z \in \{u_2, u_2'\}$. Since $E(F) \subseteq E(H) = S_t$, each side of F is a subset of N_i for some i; without lose of generality we assume that $u_0, u_0' \in N_0, u_1, u_1' \in N_1$ and $u_2, u_2' \in N_2$.

Now consider the execution of A' on the 6-cycle Q with vertices $u_0, u_1, u_2, u_0', u_1', u_2'$ in this order. Let $\text{Tr}_Q(u)$ denote the transcript generated by node u in the execution of A' in Q: if $u \in N_i$, then $\text{Tr}_Q(u)$ is the binary string obtained by first writing all the messages sent by u to its neighbor in $N_{(i+1) \bmod 3}$ (in order, round after round), followed by all the messages sent by u to its neighbor in $N_{(i+2) \bmod 3}$.

Claim 4.4. *Let us write $t = t_0 t_1 t_2$, where $t_i = \text{Tr}_{u_0,u_1,u_2}(u_i) = \text{Tr}_{u_0',u_1',u_2'}(u_i')$ for each $i \in \{0, 1, 2\}$. Then for each $i \in \{0, 1, 2\}$ we have $\text{Tr}_Q(u_i) = \text{Tr}_Q(u_i') = t_i$.*

Proof sketch. By induction on rounds. In the step we use the fact that for each node v in Q, if x and y are v's neighbors in Q, then $\{x, v, y\}$ is an edge of F, meaning that the transcript generated by A' in $\triangle(x, v, y)$ is the same transcript t we work with; therefore, v's view in Q stays consistent its view in $\triangle(x, v, y)$. □

Now consider node u_1's view at the end of the execution of A' in Q. By the claim above, the messages u_1 receives throughout the execution in Q are the same messages it would receive in $\triangle(u_0, u_1, u_2)$. Also, the neighbors of u_1 in Q are the same neighbors it has in $\triangle(u_0, u_1, u_2)$. Thus, the state of u_1 is the same at the end of the execution in Q and in $\triangle(u_0, u_1, u_2)$. In $\triangle(u_0, u_1, u_2)$ we know that u_1 rejects (by Claim 4.3); therefore it also rejects in Q, violating the correctness of A' and hence also of A. □

5 BANDWIDTH LOWER BOUND ON 1-ROUND TRIANGLE-DETECTION

In this section we show:

Theorem 5.1. *There exist a constant $\epsilon \in (0, 1)$ such that any protocol solving triangle-detection in one round with error probability at most ϵ requires bandwidth $B = \Omega(n)$ in graphs of maximum degree $\Theta(n)$.*

To generalize to any degree Δ, we can embed our lower bound construction in a larger graph, and trivially obtain the lower bound of $\Omega(\Delta)$ for maximum degree Δ.

The idea behind the lower bound is to take three "special" nodes, with randomly chosen identifiers, and connect every pair of them with iid probability $1/2$, so that a triangle appears w.p. $1/8$. The error of the protocol is assumed to be much smaller, $\epsilon \ll 1/8$, so the protocol must reject with good probability in this case. In addition to the three special nodes, we give each of them $\Theta(n)$ non-special neighbors, in such a way that no special node can tell a-priori whether a particular neighbor is another special node or not. This way, the triangle edges are "hidden" among many other edges, which look the same to each special node.

We do not prevent a special node from *learning* the identities of the other two special nodes after it receives their messages, but by this point it is too late for the protocol: because each special node did not know a-priori which edges it should care about, we show that its message does not convey much information about the presence or absence of the potential triangle edges (i.e., the edges between special nodes).

Let us now describe the graph construction more formally, and proceed to sketch the proof of the lower bound.

The template graph. In our lower bound, we work with random subgraphs of the following graph G_T.

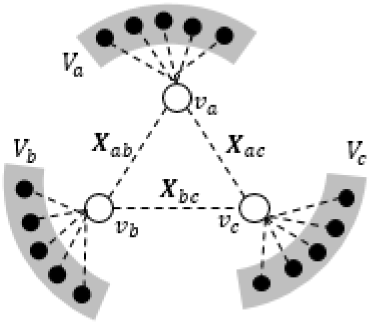

Figure 3: The template graph G_T.

Let $V_a = \{v_1^a, \ldots, v_n^a\}$, $V_b = \{v_1^b, \ldots, v_n^b\}$, $V_c = \{v_1^c, \ldots, v_n^c\}$ be three disjoint vertex sets with $|A| = |B| = |C| = n$. The vertex set of G_T is $V(G_T) = V_a \cup V_b \cup V_c \cup \{v_a, v_b, v_c\}$, and the edges are the following:

- Nodes v_a, v_b, v_c are connected in a triangle, and
- For each $s \in \{a, b, c\}$, every $u \in V_s$ is connected to v_s.

We refer to v_a, v_b, v_c as *special nodes*; they are the only nodes that are part of a triangle in G_T. Throughout this proof sketch, we ignore the non-special nodes, because their input gives them no information about the presence of a triangle. (We account for the non-special nodes more formally in the full proof, see the full version for details.)

In a departure from the rest of the paper, here we treat a *graph node* as a separate entity from its *identifier*: we construct a random graph $\boldsymbol{G}$ by taking a random subgraph of G_T as we describe below, and assigning to each node a random identifier. The algorithm executed at node $v \in V(G)$ knows only the identifier assigned to node v, and the identifiers of v's neighbors; it does not know v. This is merely a technical convenience due to the fact that the graphs we consider are all subgraphs of G_T, and does not weaken the lower bound.

The input distribution. We define $\boldsymbol{G}$ to be a random subgraph of G_T, where each edge of G_T is included with iid probability $1/2$. Identifiers are also assigned at random: for each $v \in V(G_T)$, we choose an identifier $\boldsymbol{id}(v)$ uniformly at random from $[n^3]$.

Remark. Since we choose the identifiers independently from $[n^3]$, there is a small but non-zero chance that two vertices get the same identifier. In this case we do not know what the algorithm will do, as it is not obligated to work when identifiers are not unique. However, the probability of this event is so tiny that it does not affect us by much.

For each $s \in \{a, b, c\}$, let $\boldsymbol{u}_s = \boldsymbol{id}(v_s)$ be the identifier assigned to v_s.

Input representation. It is technically convenient to represent the input to each node as the sequence of identifiers assigned to its neighbors in G_T (i.e., possibly *more* neighbors than it has in $\boldsymbol{G}$), together with a bit-vector indicating for each one whether it is a neighbor in $\boldsymbol{G}$ or not. The algorithm is free to ignore the extra information, so this only strengthens the lower bound. This representation makes it technically easy to view the potential triangle edges as "hidden" among $\Theta(n)$ other iid Bernoulli random variables.

Thus, we give each special node v_s the sequence

$$\boldsymbol{U}_s = \{\boldsymbol{id}(w) \mid \{w, v_s\} \in E(G_T)\}$$

of identifiers of v_s's neighbors in G_T, but we apply a random permutation $\boldsymbol{\pi}_s$ to the sequence, so that v_s cannot tell which node in G_T corresponds to which identifier in $\boldsymbol{G}$. In addition, v_s receives the bit-vector $\boldsymbol{X}_s$ indicating for each potential neighbor w in G_T whether the edge $\{v_s, w\}$ is in $\boldsymbol{G}$; the vector $\boldsymbol{X}_s$ is also scrambled using the same permutation $\boldsymbol{\pi}_s$, so that its order matches the order of the identifiers in $\boldsymbol{U}_s$. And finally, v_s receives its own identifier, $\boldsymbol{u}_s$.

For $s \neq t \in \{a, b, c\}$, let $\boldsymbol{i}_s(t)$ be the index in $\boldsymbol{X}_s$ indicating whether the potential triangle edge $\{v_s, v_t\}$ is in $\boldsymbol{G}$ or not. (This is a uniformly random index from v_s's perspective.) We use the following short-hand notation: $\boldsymbol{N}_s = (\boldsymbol{U}_s, \boldsymbol{X}_s, \boldsymbol{u}_s)$ is the input to v_s, and $\boldsymbol{X}_{st} = \boldsymbol{X}_s(\boldsymbol{i}_s(t))$ is the bit indicating whether the edge $\{v_s, v_t\}$ is in $\boldsymbol{G}$.

Observation 5.2. *G contains a triangle if and only if $\boldsymbol{X}_{ab} \wedge \boldsymbol{X}_{bc} \wedge \boldsymbol{X}_{ac} = 1$.*

Now, fix an algorithm A with distributional error at most $1/1000$ on μ; that is, when we draw $\boldsymbol{G} \sim \mu$, the probability that A outputs the correct answer is at least $999/1000$. (The constant is fairly arbitrary.)

Because a triangle appears with probability $1/8$, and A has overall error at most $1/1000$, it is easy to verify that *conditioned* on the presence of the triangle, A rejects with probability at least $1 - 8/1000 > 99/100$. This means that at least one of the three special vertices, w.l.o.g. v_a, rejects w.p. $> 33/100$ in this case. On the other hand, if $\boldsymbol{X}_{ab} = \boldsymbol{X}_{ac} = 1$ but $\boldsymbol{X}_{bc} = 0$, then v_a must accept with probability at least $1 - 8/1000 > 99/100$. So, conditioned on the event $\boldsymbol{X}_{ab} = \boldsymbol{X}_{ac} = 1$, node v_a must *learn* some information about $\boldsymbol{X}_{bc}$, and this is formalized by the following lemma. For two vertices u, w, Let $\boldsymbol{M}_{uw}$ denote the messages sent by u to w.

Lemma 5.3. *If Π is a protocol that solves triangle-detection, then either*

$$\mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{M}_{ba}, \boldsymbol{M}_{ca} | \boldsymbol{N}_a, \boldsymbol{X}_{ab} = 1, \boldsymbol{X}_{ac} = 1) \geq 0.3,$$

or a symmetric claim holds for v_b or for v_c.

Proof sketch. Let $\boldsymbol{acc}_a$ be an indicator for the event that v_a accepts. By the data processing inequality, since v_a's decision is a function of its input and the messages it receives,

$$\mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{M}_{ba}, \boldsymbol{M}_{ca} | \boldsymbol{N}_a, \boldsymbol{X}_{ab} = 1, \boldsymbol{X}_{ac} = 1) \geq \mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{acc}_a | \boldsymbol{N}_a, \boldsymbol{X}_{ab} = 1, \boldsymbol{X}_{ac} = 1),$$

so it suffices to lower-bound the right-hand side.

As we said above, the distribution of $\boldsymbol{acc}_a$ changes noticeably depending on the value of $\boldsymbol{X}_{bc} = 1$: when $\boldsymbol{X}_{bc} = 0$, the probability that v_a accepts is at least $99/100$. However, the *prior* probability that v_a accepts, when $\boldsymbol{X}_{bc} \sim \text{Bernoulli}(1/2)$, is at most $(1/2) \cdot (99/100) + (1/2) \cdot (67/100) < 8/10$. This "change in behavior" translates to a lower bound on mutual information (see the full proof for the details). □

On the other hand, we claim that nodes v_b and v_c cannot convey this much information about $\boldsymbol{X}_{bc}$ to v_a.

Lemma 5.4. *The information node v_a learns about $\boldsymbol{X}_{bc}$ given its input and the presence of edges $\{v_a, v_b\}, \{v_a, v_c\}$ is at most:*

$$\mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{M}_{ba}, \boldsymbol{M}_{ca} | \boldsymbol{N}_a, \boldsymbol{X}_{ab} = 1, \boldsymbol{X}_{ac} = 1) \leq \frac{4(|\boldsymbol{M}_{ca}| + |\boldsymbol{M}_{ba}|)}{n+1} + \frac{2}{n}.$$

Symmetric claims hold for v_b and v_c.

Proof sketch. First, since $\Pr(\boldsymbol{X}_{ab} = 1, \boldsymbol{X}_{ac} = 1) = 1/4$ and mutual information is non-negative,

$$\mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{M}_{ba}, \boldsymbol{M}_{ca} | \boldsymbol{N}_a) \geq (1/4)\, \mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{M}_{ba}, \boldsymbol{M}_{ca} | \boldsymbol{N}_a, \boldsymbol{X}_{ab} = 1, \boldsymbol{X}_{ac} = 1).$$

(This can be seen by writing the expectation on $\boldsymbol{N}_a$ as an expectation on $\boldsymbol{X}_{ab}, \boldsymbol{X}_{ac}$ and the rest of $\boldsymbol{N}_a$, and expanding out the expectation on $\boldsymbol{X}_{ab}, \boldsymbol{X}_{ac}$.)

We bound the left-hand side. It expresses the information v_b and v_c's messages *together* convey about $\boldsymbol{X}_{bc}$. However, because the two nodes' inputs are independent given the conditioning and $\boldsymbol{X}_{bc}$, we can actually break it up into the sum of what each message conveys by itself: we show that

$$\mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{M}_{ba}, \boldsymbol{M}_{ca} | \boldsymbol{N}_a) \leq \mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{M}_{ba} | \boldsymbol{N}_a, \boldsymbol{u}_c) + \mathrm{I}(\boldsymbol{X}_{bc}; \boldsymbol{M}_{ca} | \boldsymbol{N}_a, \boldsymbol{u}_b).$$

We bound each term separately.

Consider the first term, $\mathrm{I}(\boldsymbol{X}_{bc};\boldsymbol{M}_{ba}|\boldsymbol{N}_a,\boldsymbol{u}_c)$ (the other term is similar). This term essentially measures the information v_b sends to a about $\boldsymbol{X}_{bc}$, for a "typical" identifier $\boldsymbol{u}_c$. Crucially, when v_b sends its message to v_a, it does not *know* the identity of $\boldsymbol{u}_c$; that is, its message is *independent* of the choice of $\boldsymbol{u}_c$, conditioned on its input. Consequently, we can argue that $\boldsymbol{M}_{bc}$ is independent of the index $\boldsymbol{i}_b(c)$ where we hid the bit $\boldsymbol{X}_{bc}$. (This is slightly delicate, because we could have duplicate identifiers; $\boldsymbol{u}_c$ does not uniquely specify $\boldsymbol{i}_b(c)$. We handle this in the proof by conditioning on the high-probability event that there are no duplicate identifiers; hence the additive $O(1/n)$ term in the bound we get.)

Now we have reduced everything to the following question: how much does v_b's message $\boldsymbol{M}_{ba}$ reveal about a *random* coordinate $\boldsymbol{X}_b(\boldsymbol{i})$ of its input, where the index $\boldsymbol{i}$ is uniformly random on $n+1$ locations,[1] and not known to v_b in advance? Using standard arguments, we show that the answer is $O(|\boldsymbol{M}_{ba}|/(n+1))$. The claim follows. □

Corollary 5.5. *If* $|\boldsymbol{M_{ba}}|, |\boldsymbol{M_{ca}}| < n/60$ *then*

$$\mathrm{I}(\boldsymbol{X_{bc}};\boldsymbol{M_{ba}},\boldsymbol{M_{ca}}|\boldsymbol{N_a},\boldsymbol{X_{ab}}=1,\boldsymbol{X_{ac}}=1) < 0.1.$$

By combining the two lemmas above, and dealing with the low-probability event that $\boldsymbol{G}$ contains duplicate identifiers, we get Theorem 5.1.

6 SUBLINEAR-TIME PROTOCOL FOR C_{2k}

In this section we show that for any $k \geq 2$, the C_{2k}-subgraph detection problem can be solved in $O(n^{1-1/(k(k-1))})$ rounds. E.g., C_4 can be detected in in $O(n^{1/2})$ rounds (which was already known [10]); C_6 can be detected in $O(n^{5/6})$ rounds; and so on. Our algorithm combines several techniques: color coding [2], pipelining, and a particular decomposition of low-arboricity graphs (previously used in, e.g., [3]). We also rely on the fact that the Túran number of an even cycle C_{2k} is $\mathrm{ex}(n, C_{2k}) = O(n^{1+1/k})$ (see, e.g., [5]).

Let $M = O(n^{1+1/k})$ be an upper bound on $\mathrm{ex}(n, C_{2k})$. We assume for simplicity that the bandwidth is sufficiently large to send a sequence of $2k$ identifiers in one message. (Note that k is treated as a constant throughout.)

Phase I: dealing with high-degree nodes. Our goal in this step is to find a copy of C_{2k} that includes at least one "high-degree node": a node with degree at least n^δ, where $\delta = 1/(k-1)$. If there is such a C_{2k} in the graph, then in this phase at least one node rejects. Otherwise we will search for cycles comprising only low-degree nodes, in the next phase of the algorithm.

We rely on the fact that if $|E(G)| \leq M$, then there cannot be more than $\lceil M/n^\delta \rceil$ nodes with degree at least n^δ. On the other hand, if $|E(G)| > M$, then we may fail, but in this case we know that the graph contains a $2k$-cycle, because it has too many edges (more than $\mathrm{ex}(n, C_{2k})$).

We first color-code the graph, assigning each node u a random color $c(u) \in \{0, \ldots, 2k-1\}$. Then we look for a *properly-colored* copy of C_{2k}: a cycle $u_0, \ldots, u_{2k-1}$, where $c(u_i) = i$ for each $i = 0, \ldots, 2k-1$. To search for such cycles, we start a "color-coded" BFS to depth $2k$, from *every node* that has degree at least n^δ and color 0. A color-coded BFS is like a regular BFS, except that the BFS token is only allowed to move from node u to a neighbor v if $c(v) = c(u) + 1$. In this way we "unravel" paths of nodes with colors $0, 1, \ldots, 2k-1$ respectively, until finally the last node, whose color is $2k-1$, sends the BFS token back to node 0, and then node 0 detects the cycle and rejects.

The color-coded BFS is executed in parallel for all nodes, as follows. Each node u maintains a queue of messages it needs to send; initially, if $c(u) = 0$ and $\deg(u) \geq n^\delta$, then u adds the message $(\mathsf{ColorBFS}, u, 0)$ to its queue, and otherwise u's queue is initially empty.

In each round, every node chooses an arbitrary message from its queue, sends it to all its neighbors, and removes it from its queue. Upon receipt of a message $(\mathsf{ColorBFS}, u, i)$, node v does the following:

- If node v previously received the same message, it discards the new copy.
- If $u = v$ (i.e., node u received its own message back), and $i = 2k-1$, then we have found a $2k$-cycle, and node u rejects.
- If $c(v) \neq i + 1$, it discards the message.
- Otherwise, v adds a message $(\mathsf{ColorBFS}, u, i+1)$ to its queue.

Phase I takes $R_1 := \lceil M/n^\delta \rceil + 2k$ rounds. At the end of R_1 rounds, if any node has a non-empty message queue, it rejects.

A standard pipelining argument shows:

Lemma 6.1. *If* $|E(G)| \leq M$*, then after* $R_1 = \lceil M/n^\delta \rceil + 2k$ *rounds, all nodes have empty queues.*

Intuitively, this is because at most $\lceil M/n^\delta \rceil$ BFS tokens can "stand in the way" of any given token (as this is an upper bound on the number of high-degree nodes), and each token only travels to distance $2k$.

Consequently, at the end of phase I, if $|E(G)| \leq M$ then all BFS instances complete, and we get:

Corollary 6.2. *If* $|E(G)| \leq M$*, and* G *contains a copy of* C_{2k} *that includes a node with degree at least* n^δ*, then in phase I, with probability at least* $1/(2k)^{2k}$*, some node rejects.*

What happens if $|E(G)| > M$? In this case we know that the graph *must* contain a $2k$-cycle, but still, we might not find it. However, we are guaranteed that if after R_1 rounds some node has a non-empty queue, then the graph has more than M edges, and it contains a $2k$-cycle. Hence:

Lemma 6.3. *If in phase I some node rejects, then* G *contains a copy of* C_{2k}*.*

Phase II: dealing with the remaining graph. At the end of phase I, any node with degree at least n^δ removes itself and all its edges from the graph. Now we search for copies of C_{2k} among the remaining nodes.

Let $d = \lceil M/2n \rceil$. Using the decomposition from [3], *if* the graph is C_{2k}-free (which we do not know), we can decompose it into $\lceil \log n \rceil$ layers $V_1, \ldots, V_L$, with the property that for each layer ℓ and node $v \in V_\ell$, the number of edges going "up" from v to nodes in equal-or-higher layers $V_\ell, V_{\ell+1}, \ldots, V_L$ is at most $d = \lceil M/2n \rceil$. Essentially, we recursively remove from the graph nodes with degree

[1] One location is taken up by the bit $\boldsymbol{X}_{ab}$; since we condition here on $\boldsymbol{u}_a$, as it is part of v_a's input $\boldsymbol{N}_a$, only $n+1$ locations are possible for $\boldsymbol{X}_{bc}$.

sufficiently small in the remaining graph, relying on the fact that if there is no $2k$-cycle, then at each point the average degree cannot exceed $\lceil M/n \rceil$. After $\lceil \log n \rceil$ such steps, if there is some node that has not been assigned to any layer, this node rejects, as we know that there is a $2k$-cycle.

Otherwise we continue as follows: for each $v \in V$, let $\ell(v)$ denote the layer of node v. We randomly assign each node v a color $c(v) \in [2k]$. We look only for properly-colored $2k$-cyles $u_0, \ldots, u_{2k-1}$ where $c(u_0) = 0, \ldots, c(u_{2k-1}) = 2k - 1$ and furthermore, u_0 is on the highest layer out of all the cycle nodes. (That is, for each $i = 1, \ldots, 2k - 1$ we have $\ell(u_i) \leq \ell(u_0)$.)

For this type of cycle, we have the advantage that from the perspective of nodes u_1 and u_{2k-1}, the number of neighbors they need to "consider for the role of u_0" is bounded by their "up-degree" d — the number of neighbors they have in equal-or-higher layers.

Let us call a sequence $u_0, \ldots, u_i$ a *length-i increasing prefix* if $c(u_0) = 0, c(u_1) = 1, \ldots, c(u_i) = i$, or a *length-$i$ decreasing prefix* if $c(u_0) = 0, c(u_1) = 2k-1, c(u_2) = 2k-2, \ldots, c(u_i) = 2k-i$. Our goal now is to have node u_k (the "midpoint" of the $2k$-cycle) receive a length-k increasing sequence *and* a length-k decreasing sequence, both starting from node u_0 (the highest-layer node). Node u_k will then detect the cycle and reject.

We propagate prefixes of increasing length, as follows:

(1) First, each v with $c(v) = 0$ sends the message $(v, \ell(v))$.
(2) Next, each node v colored 1 or $2k - 1$ sends the following: for each message $(u, \ell(u))$ received in the previous round, *if* $\ell(u) \geq \ell(v)$, then v sends out the prefix (u, v). Since node v has at most d neighbors in equal-or-higher layers, this requires at most d rounds.
(3) For each $i \in \{2, \ldots, k - 1\}$, nodes colored i (resp. $2k - i$) send all the length-$(i - 1)$ increasing (resp. decreasing) prefixes they received, appending their name at the end to form length-i increasing (resp. decreasing) prefixes. This takes at most $d \cdot n^{\delta \cdot (k-2)}$ rounds.
(4) Finally, nodes v colored k collect the length-k increasing and decreasing prefixes they received, and check if there is some node u_0 that starts both an increasing and a decreasing prefix ending at v. If so, they have found a $2k$-cycle, and they reject.

Claim 6.4. *Assume that $|E(G)| \leq M$. Then (a) all nodes are assigned to a layer after $\lceil \log n \rceil$ steps of layer assignment; and (b) if G contains a $2k$-cycle, then with probability at least $1/(2k)^{2k}$ some node rejects in phase II.*

The running time of phase II is $R_2 = \lceil \log n \rceil + (2k) \cdot d \cdot n^{\delta \cdot (k-2)}$. Balancing the running times R_1, R_2 of the two phases, we see that when we take $\delta = 1/(k-1)$, both running times are $O(n^{1-1/(k(k-1))})$, as desired.

Putting everything together. Phases I and II together give us the following guarantee:

- If $|E(G)| \leq M$, and the graph contains a $2k$-cycle, then with probability at least $1/(2k)^(2k)$ some node rejects.
- If some node rejects, then either $|E(G)| > M$, or the graph contains a $2k$-cycle.

Of course, if $|E(G)| > M$, then we know that the graph *must* contain a $2k$-cycle, even though we did not necessarily detect a specific one. Therefore it is proper that the algorithm reject in this case.

To construct an algorithm that detects a $2k$-cycle with *constant* probability, we simply repeat each phase $O\left(2k^{2k}\right)$ times, each iteration using independently-chosen colors for the nodes.

7 ACKNOWLEDGMENTS

We would like to thank Dan Karliner for fruitful discussions, and for pointing out Theorem 4.2 to us.

REFERENCES

[1] Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Christoph Lenzen. 2017. 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. 1995. Color-coding. *J. ACM* 42, 4 (1995), 844–856.
[3] Leonid Barenboim and Michael Elkin. 2008. Sublogarithmic Distributed MIS Algorithm for Sparse Graphs Using Nash-williams Decomposition *(PODC '08)*. 25–34.
[4] Zvika Brakerski and Boaz Patt-Shamir. 2011. Distributed discovery of large near-cliques. *Distributed Computing* 24, 2 (2011), 79–89.
[5] Boris Bukh and Zilin Jiang. 2017. A bound on the number of edges in graphs without an even cycle. *Combinatorics, Probability and Computing* 26, 1 (2017), 1–15.
[6] Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. 2016. *Fast Distributed Algorithms for Testing Graph Properties.* 43–56.
[7] Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. 2015. Algebraic Methods in the Congested Clique. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015.* 143–152.
[8] Keren Censor-Hillel, Seri Khoury, and Ami Paz. 2017. Quadratic and Near-Quadratic Lower Bounds for the CONGEST Model. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria.* 10:1–10:16.
[9] Danny Dolev, Christoph Lenzen, and Shir Peled. 2012. *"Tri, Tri Again": Finding Triangles and Small Subgraphs in a Distributed Setting.* 195–209.
[10] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. 2014. On the Power of the Congested Clique Model. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing (PODC '14).* 367–376.
[11] P. Erdös. 1964. On extremal problems of graphs and generalized graphs. *IJoM* (1964).
[12] Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. 2017. Three Notes on Distributed Property Testing. In *31st International Symposium on Distributed Computing (DISC 2017) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 91. 15:1–15:30.
[13] Pierre Fraigniaud and Dennis Olivetti. 2017. Distributed Detection of Cycles. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017.* 153–162.
[14] Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. 2016. *Distributed Testing of Excluded Subgraphs.* 342–356.
[15] Tzlil Gonen and Rotem Oshman. 2017. Lower Bounds for Subgraph Detection in the CONGEST Model. To appear in OPODIS 2017.
[16] Taisuke Izumi and François Le Gall. 2017. Triangle Finding and Listing in CONGEST Networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC '17).* 381–389.
[17] Bala Kalyanasundaram and Georg Schnitger. 1992. The Probabilistic Communication Complexity of Set Intersection. *SIAM J. Discrete Math.* 5, 4 (1992), 545–557.
[18] Janne H. Korhonen and Joel Rybicki. 2017. Deterministic Subgraph Detection in Broadcast CONGEST. In *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017.* 4:1–4:16.
[19] Eyal Kushilevitz and Noam Nisan. 1997. *Communication Complexity.* Cambridge University Press.
[20] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. 2016. Tight Bounds for Distributed Graph Computations. *CoRR* (2016). http://arxiv.org/abs/1602.08481
[21] M. Pease, R. Shostak, and L. Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (1980), 228–234.
[22] Alexander A. Razborov. 1992. On the Distributional Complexity of Disjointness. *Theor. Comput. Sci.* 106, 2 (1992), 385–390.
[23] Igor Rivin. 2002. Counting Cycles and Finite Dimensional Lp Norms. *Adv. Appl. Math.* 29, 4 (2002), 647–662.
[24] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM* 23, 1 (1976), 31–42.