Distributed Testing of Excluded Subgraphs[⋆]

Pierre Fraigniaud[1], Ivan Rapaport[2], Ville Salo[2], and Ioan Todinca[3]

[1] CNRS and University Paris Diderot, France
[2] DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Chile
[3] Université d'Orléans, France

Abstract. We study *property testing* in the context of *distributed computing*, under the classical CONGEST model. It is known that testing whether a graph is triangle-free can be done in a constant number of rounds, where the constant depends on how far the input graph is from being triangle-free. We show that, for every connected 4-node graph H, testing whether a graph is H-free can be done in a constant number of rounds too. The constant also depends on how far the input graph is from being H-free, and the dependence is identical to the one in the case of testing triangle-freeness. Hence, in particular, testing whether a graph is K_4-free, and testing whether a graph is C_4-free can be done in a constant number of rounds (where K_k denotes the k-node clique, and C_k denotes the k-node cycle). On the other hand, we show that testing K_k-freeness and C_k-freeness for $k \geq 5$ appear to be much harder. Specifically, we investigate two natural types of generic algorithms for testing H-freeness, called DFS tester and BFS tester. The latter captures the previously known algorithm to test the presence of triangles, while the former captures our generic algorithm to test the presence of a 4-node graph pattern H. We prove that both DFS and BFS testers fail to test K_k-freeness and C_k-freeness in a constant number of rounds for $k \geq 5$.

1 Introduction

Let $\mathcal{P}$ be a graph property, and let $0 < \epsilon < 1$ be a fixed parameter. According to the usual definition from *property testing* [16], an n-node m-edge graph G is ϵ-far from satisfying $\mathcal{P}$ if applying a sequence of at most ϵm edge-deletions or edge-additions to G cannot result in a graph satisfying $\mathcal{P}$. In the context of property testing, graphs are usually assumed to be stored using an adjacency list[4], and a centralized algorithm has the ability to probe nodes, via queries of the form (i, j) where $i \in \{1, \ldots, n\}$, and $j \geq 0$. The answer to a query $(i, 0)$ is the degree of node i, while the answer to a query (i, j) with $j > 0$ is the identity

[⋆] Additional support from ANR project DISPLEXITY, Inria project GANG, CONICYT via Basal in Applied Mathematics, Núcleo Milenio Información y Coordinación en Redes ICM/FIC RC130003, Fondecyt 1130061 and Fondecyt 3150552.

[4] Actually, property testing tackles graph problems in both the *dense* model (graphs represented by adjacency matrices) and the *sparse* model (graphs represented by adjacency lists). In this paper, we are interested in property testing in the sparse model.

of the jth neighbor of node i. After a small number of queries, the algorithm must output either accept or reject. An algorithm Seq is a testing algorithm for $\mathcal{P}$ if and only if, for every input graph G,

$$\begin{cases} G \text{ satisfies } \mathcal{P} \implies \Pr[\mathsf{Seq} \text{ accepts } G] \geq \frac{2}{3}; \\ G \text{ is } \epsilon\text{-far from satisfying } \mathcal{P} \implies \Pr[\mathsf{Seq} \text{ rejects } G] \geq \frac{2}{3}. \end{cases}$$

(An algorithm is 1-sided if it systematically accepts every graph satisfying $\mathcal{P}$). The challenge in this context is to design testing algorithms performing as few queries as possible.

In the context of *distributed* property testing [6], the challenge is not the number of queries (as all nodes perform their own queries in parallel), but the lack of global perspective on the input graph. The graph models a network. Every node of the network is a processor, and every processor can exchange messages with all processors corresponding to its neighboring nodes in the graph. After a certain number of rounds of computation, every node must output accept or reject. A distributed algorithm Dist is a distributed testing algorithm for $\mathcal{P}$ if and only if, for any graph G modeling the actual network,

$$\begin{cases} G \text{ satisfies } \mathcal{P} \implies \Pr[\mathsf{Dist} \text{ accepts } G \text{ in all nodes}] \geq \frac{2}{3}; \\ G \text{ is } \epsilon\text{-far from satisfying } \mathcal{P} \implies \Pr[\mathsf{Dist} \text{ rejects } G \text{ in at least one node}] \geq \frac{2}{3}. \end{cases}$$

The challenge is to use as few resources of the network as possible. In particular, it is desirable that every processor could take its decision (accept or reject) without requiring data from far away processors in the network, and that processors exchange messages of size respecting the inherent bounds imposed by the limited bandwidth of the links. These two constraints are well captured by the $\mathsf{CONGEST}$ model. This model is a classical model for distributed computation [27]. Processors are given distinct identities, that are supposed to have values in $[1, n^c]$ in n-node networks, for some constant $c \geq 1$. All processors start at the same time, and then proceed in synchronous rounds. At each round, every processor can send and receive messages to/from its neighbors, and perform some individual computation. All messages must be of size at most $O(\log n)$ bits. So, in particular, every message can include at most a constant number of processor identities. As a consequence, while every node can gather the identities of all its neighbors in just one round, a node with large degree that is aware of all the identities of its neighbors may not be able to send them all simultaneously to a given neighbor. The latter observation enforces strong constraints on distributed testing algorithms in the $\mathsf{CONGEST}$ model. For instance, while the LOCAL model allows every node to gather its t-neighborhood in t rounds, even just gathering the 2-neighborhood may require $\Omega(n)$ rounds in the $\mathsf{CONGEST}$ model (e.g., in the Lollipop graph, which is the graph obtained by joining a path to a clique). As a consequence, detecting the presence of even a small given pattern in a graph efficiently is not necessarily an easy task.

The presence or absence of a certain given pattern (typically, paths, cycles or cliques of a given size) as a subgraph, induced subgraph, or minor, has a significant impact on graph properties, and/or on the ability to design efficient algorithms for hard problems. This paper investigates the existence of efficient distributed testing algorithms for the H-freeness property, depending on the given graph H. Recall that, given a graph H, a graph G is *H-free* if and only if H is not isomorphic to a subgraph of G, where H is a subgraph of a graph G if $V(H) \subseteq V(G)$, and $E(H) \subseteq E(G)$. Recently, Censor-Hillel et al. [6] established a series of results regarding distributed testing of different graph properties, including bipartiteness and triangle-freeness. A triangle-free graph is a K_3-free graph or, equivalently, a C_3-free graph, where K_k and C_k respectively denote the clique and the cycle on k vertices. The algorithm in [6] for testing bipartiteness (of bounded degree networks) requires $O(\log n)$ rounds in the $\mathsf{CONGEST}$ model, and the authors conjecture that this is optimal. However, quite interestingly, the algorithm for testing triangle-freeness requires only a constant number of rounds, $O(1/\epsilon^2)$, i.e., it depends only on the (fixed) parameter ϵ quantifying the ϵ-far relaxation.

In this paper, we investigate the following question: what are the (connected) graphs H for which testing H-freeness can be done in a constant number of rounds in the $\mathsf{CONGEST}$ model?

1.1 Our results

We show that, for every connected 4-node graph H, testing whether a graph is H-free can be done in $O(1/\epsilon^2)$ rounds. Hence, in particular, testing whether a graph is K_4-free, and testing whether a graph is C_4-free can be done in a constant number of rounds. Our algorithm is generic in the sense that, for all 4-node graphs H, the global communication structure of the algorithm is the same, with only a variant in the final decision for accepting or rejecting, which of course depends on H.

In fact, we identify two different natural generic types of testing algorithms for H-freeness. We call the first type DFS tester, and our algorithm for testing the presence of 4-node patterns is actually the DFS tester. Such an algorithm applies to Hamiltonian graphs H, i.e., graphs H containing a simple path spanning all its vertices (the only non-hamiltonian connected graph on 4 vertices is the star $K_{1,3}$, for which the problem is trivial). The DFS tester performs in $|H|-1$ rounds. Recall that, for a node set A, $G[A]$ denotes the subgraph of G induced by A. At each round t of the DFS tester, at every node u, and for each of its incident edges e, node u pushes a graph $G[A_t]$ (where, initially, A_0 is just the graph formed by u alone). The graph $G[A_t]$ is chosen u.a.r. among the sets of graphs received from the neighbors at the previous round. More specifically, upon reception of every graph $G[A_{t-1}]$ at round $t-1$, node u forms a graph $G[A_{t-1} \cup \{u\}]$, and the graph $G[A_t]$ pushed by u along e at round t is chosen u.a.r. among the collection of graphs $G[A_{t-1} \cup \{u\}]$ currently held by u. By repeating this algorithm $O(1/\epsilon^2)$ times we obtain the desired probability.

We call BFS tester the second type of generic testing algorithm for H-freeness. The algorithm in [6] for triangle-freeness is a simplified variant of the BFS tester, and we prove that the BFS tester can test K_4-freeness in $O(1/\epsilon^2)$ rounds. Instead of guessing a path spanning H (which may actually not exist for H large), the BFS tester aims at directly guessing all neighbors in H simultaneously. The algorithm performs in $D-1$ rounds, where D is the diameter of H. For the sake of simplicity, let us assume that H is d-regular. At each round, every node u forms groups of d neighboring nodes. These groups may overlap, but a neighbor should not participate to more than a constant number of groups. Then for every edge $e=\{u,v\}$ incident to u, node u pushes all partial graphs of the form $G[A_{t-1} \cup \{v_1,\dots,v_t\}]$ where $v \in \{v_1,\dots,v_t\}$, A_{t-1} is chosen u.a.r. among the graphs received at the previous round, and the v_i's will be in charge of checking the presence of edges between them and the other v_j's.

We prove that neither the DFS tester, nor the BFS tester can test K_k-freeness in a constant number of rounds for $k \geq 5$, and that the same holds for testing C_k-freeness (with the exception of a finite number of small values of k). This shows that testing K_k-freeness or C_k-freeness for $k \geq 5$ in a constant number of rounds requires, if at all possible, to design algorithms which explore G from each node in a way far more intricate than just parallel DFSs or parallel BFSs. Our impossibility results, although restricted to DFS and BFS testers, might be hints that testing K_k-freeness and C_k-freeness for $k \geq 5$ in n-node networks does require to perform a number of rounds which grows with the size n of the network.

1.2 Related work

The CONGEST model has become a standard model in distributed computation [27]. Most of the lower bounds in the CONGEST model have been obtained using reduction to communication complexity problems [8, 11, 23]. The so-called *congested clique* model is the CONGEST model in the complete graph K_n [10, 22, 24, 26]. There are extremely fast protocols for solving different types of graph problems in the congested clique, including finding ruling sets [20], constructing minimum spanning trees [19], and, closely related to our work, detecting small subgraphs [7, 9].

The distributed property testing framework in the CONGEST model was recently introduced in the aforementioned paper [6], inspired from classical property testing [16, 17]. Distributed property testing relaxes classical distributed decision [12, 13, 18], typically designed for the LOCAL model, by ignoring illegal instances which are less than ϵ-far from satisfying the considered property. Without such a relaxation, distributed decision in the CONGEST model requires non-constant number of rounds [8]. Other variants of local decision in the LOCAL model have been studied in [2, 3], where each process can output an arbitrary value (beyond just a single bit — accept or reject), and [4, 21], where nodes are bounded to perform a single round before to output at most $O(\log n)$ bits. Distributed decision has been also considered in other distributed environments, such as shared memory [14], with application to runtime verification [15], and networks of finite automata [28].

2 Detecting Small Graphs Using a DFS Approach

In this section, we establish our main positive result, which implies, in particular, that testing C_4-freeness and testing K_4-freeness can be done in constant time in the CONGEST model.

Theorem 1. *Let H be a connected graph on four vertices. There is a 1-sided error distributed property-testing algorithm for H-freeness, performing in a constant number of rounds in the* CONGEST *model.*

Proof. All 4-node connected graphs H contain a P_4 (a path on four vertices) as a subgraph, with the only exception of the *star* $K_{1,3}$ (a.k.a. the *claw*). Nevertheless, testing whether a graph G is $K_{1,3}$-free is trivial (every node rejects whenever its degree is at least three). Therefore, we are going to show the theorem by exhibiting a generic distributed testing protocol for testing H-freeness, that applies to any graph H on four vertices containing a P_4 as a subgraph. The core of this algorithm is presented as Algorithm 1. Note that the test $H \subseteq G[u, u', v', w']$ at step 4 of Algorithm 1 can be performed thanks to the bit b that tells about the only edge that node u is not directly aware of. Algorithm 1 performs in just two rounds (if we omit the round used to acquire the identities of the neighbors), and that a single $O(\log n)$-bit message is sent through every edge at each round. Clearly, if G is H-free, then all nodes accept.

Algorithm 1: Testing H-freeness for 4-node Hamiltonian H. Instructions for node u.

```
1 Send $\mathrm{id}(u)$ to all neighbors;
2 For every neighbor $v$, choose a received identity $\mathrm{id}(w)$ u.a.r., and send
  $(\mathrm{id}(w), \mathrm{id}(u))$ to $v$;
3 For every neighbor $v$, choose a received pair $(\mathrm{id}(w'), \mathrm{id}(u'))$ u.a.r., and send
  $(\mathrm{id}(w'), \mathrm{id}(u'), \mathrm{id}(u), b)$ to $v$, where $b = 1$ if $w'$ is a neighbor of $u$, and $b = 0$
  otherwise;
4 For every received 4-tuple $(\mathrm{id}(w'), \mathrm{id}(v'), \mathrm{id}(u'), b)$, check whether
  $H \subseteq G[u, u', v', w']$;
5 If $H \subseteq G[u, u', v', w']$ for one such 4-tuple then reject else accept.
```

In order to analyze the efficiency of Algorithm 1 in case G is ϵ-far from being H-free, let us consider a subgraph $G[\{u_1, u_2, u_3, u_4\}]$ of G containing H, such that (u_1, u_2, u_3, u_4) is a P_4 spanning H. Let $\mathcal{E}$ be the event "at step 2, vertex u_2 sends $(\mathrm{id}(u_1), \mathrm{id}(u_2))$ to its neighbor u_3". We have $\Pr[\mathcal{E}] = 1/d(u_2)$. Similarly, let $\mathcal{E}'$ be the event "at step 3, vertex u_3 sends $(\mathrm{id}(u_1), \mathrm{id}(u_2), \mathrm{id}(u_3))$ to its neighbor u_4". We have $\Pr[\mathcal{E}'|\mathcal{E}] = 1/d(u_3)$. Since $\Pr[\mathcal{E} \wedge \mathcal{E}'] = \Pr[\mathcal{E}'|\mathcal{E}] \cdot \Pr[\mathcal{E}]$, it follows that

$$\Pr[H \text{ is detected by } u_4 \text{ while performing Algorithm 1}] \geq \frac{1}{d(u_2)d(u_3)}. \quad (1)$$

Note that the events $\mathcal{E}$ and $\mathcal{E}'$ only depend on the choices made by u_2 for the edge $\{u_2, u_3\}$ and by u_3 for the edge $\{u_3, u_4\}$, in steps 3 and 4 of Algorithm 1, respectively. Since these choices are performed independently at all nodes, it follows that if H_1 and H_2 are edge-disjoint copies of H in G, then the events $\mathcal{E}_1$ and $\mathcal{E}_2$ associated to them are independent, as well as the events $\mathcal{E}'_1$ and $\mathcal{E}'_2$.

The following result will be used throughout the paper, so we state it as a lemma for further references.

Lemma 1. *Let G be ϵ-far from being H-free. Then G contains at least $\epsilon m/|E(H)|$ edge-disjoint copies of H.*

Proof of Lemma 1. Let $S = \{e_1, \ldots, e_k\}$ be a smallest set of edges whose removal from G results in an H-free graph. We have $k \geq \epsilon m$. Let us then remove these edges from G according to the following process. The edges are removed in arbitrary order. Each time an edge e is removed from S, we select an arbitrary copy H_e of H containing e, we remove all the edges of H_e from G, and we reset S as $S \setminus E(H_e)$. We proceed as such until we have exhausted all the edges of S. Note that each time we pick an edge $e \in S$, there always exists a copy H_e of H containing e. Indeed, otherwise, $S \setminus \{e\}$ would also be a set whose removal from G results in an H-free graph, contradicting the minimality of $|S|$. After at most k such removals, we get a graph that is H-free, and, by construction, for every two edges $e, e' \in S$, we have that H_e and $H_{e'}$ are edge-disjoint. Every step of this process removes at most $|E(H)|$ edges from S, hence the process performs at least $\epsilon m/|E(H)|$ steps before exhausting all edges in S. Lemma 1 follows. □

Let us now define an edge $\{u, v\}$ as *important* if it is the middle-edge of a P_4 in one of the $\epsilon m/|E(H)|$ edge-disjoint copies of H constructed in the proof of Lemma 1. We denote by $I(G)$ the set of all important edges. Let N_0 be the random variable counting the number of distinct copies of H that are detected by Algorithm 1. As a direct consequence of Eq (1), we get that

$$\mathbf{E}(N_0) \geq \sum_{\{u,v\} \in I(G)} \frac{1}{d(u)d(v)}.$$

Define an edge $\{u, v\}$ of G as *good* if $d(u)d(v) \leq 4m|E(H)|/\epsilon$, and let $g(G)$ denote the set of good edges. Note that if there exists a constant $\gamma > 0$ such that $|I(G) \cap g(G)| \geq \gamma m$, then the expected number of copies of H detected during a phase is

$$\mathbf{E}(N_0) \geq \sum_{\{u,v\} \in I(G) \cap g(G)} \frac{1}{d(u)d(v)} \geq \gamma m \, \frac{1}{4m|E(H)|/\epsilon} = \frac{\gamma\epsilon}{4|E(H)|}. \qquad (2)$$

We now show that the number of edges that are both important and good is indeed at least a fraction γ of the edges, for some constant $\gamma > 0$. We first show that G has at least $(1 - \frac{3}{4|E(H)|}\epsilon)m$ good edges. Recall that $\sum_{u \in V(G)} d(u) = 2m$, and define $N(u)$ as the set of all neighbors of node u. We have

$$\sum_{\{u,v\} \in E(G)} d(u)d(v) = \frac{1}{2} \sum_{u \in V(G)} d(u) \sum_{v \in N(u)} d(v) \leq \sum_{u \in V(G)} d(u) \; m \leq 2m^2.$$

Thus G must have at least $(1-\frac{3}{4|E(H)|}\epsilon)m$ good edges, since otherwise

$$\sum_{\{u,v\}\in E(G)} d(u)d(v) \geq \sum_{\{u,v\}\in E(G)\setminus g(G)} d(u)d(v) > \frac{3}{4|E(H)|}\epsilon m \frac{4m|E(H)|}{\epsilon} = 3m^2,$$

contradicting the aforementioned $2m^2$ upper bound. Thus, G has at least $(1-\frac{3}{4|E(H)|}\epsilon)m$ good edges. On the other hand, since the number of important edges is at least the number of edge-disjoint copies of H in G, there are at least $\epsilon m/|E(H)|$ important edges. It follows that the number of edges that are both important and good is at least $\frac{\epsilon}{4|E(H)|}m$. Therefore, by Eq. (2), we get that $\mathbf{E}(N_0) \geq \left(\frac{\epsilon}{4|E(H)|}\right)^2$.

All the above calculations were made on the $\epsilon m/|E(H)|$ edge-disjoint copies of H constructed in the proof of Lemma 1. Therefore, if $X_i^{(0)}$ denotes the random variable satisfying $X_i^{(0)} = 1$ if the ith copy H is detected, and $X_i^{(0)} = 0$ otherwise, then we have $N_0 = \sum_{i=1}^{\epsilon m/|E(H)|} X_i^{(0)}$, and the variables $X_i^{(0)}$, $i = 1,\dots,\epsilon m/|E(H)|$, are mutually independent. Let $T = 8\ln 3\,\left(\frac{4|E(H)|}{\epsilon}\right)^2$.

By repeating the algorithm T times, and defining $N = \sum_{t=0}^{T-1} N_t$ where N_t denotes the number of copies of H detected at the tth independent repetition, we get $\mathbf{E}(N) \geq 8\ln 3$. In fact, we also have $N = \sum_{t=0}^{T-1}\sum_{i=1}^{\epsilon m/|E(H)|} X_i^{(t)}$ where $X_i^{(t)} = 1$ if the ith copy H is detected at the tth iteration of the algorithm, and $X_i^{(t)} = 0$ otherwise. All these variables are mutually independent, as there is mutual independence within each iteration, and all iterations are performed independently. Therefore, Chernoff bound applies (see Theorem 4.5 in [25]), and so, for every $0 < \delta < 1$, we have $\Pr[N \leq (1-\delta)\mathbf{E}[N]] \leq e^{-\delta^2 \mathbf{E}[N]/2}$.

By taking $\delta = \frac{1}{2}$ we get $\Pr[N \leq 4\ln 3] \leq \frac{1}{3}$. Therefore, a copy of H is detected with probability at least $\frac{2}{3}$, which completes the proof of Theorem 1. □

3 Limits of the DFS Approach

Algorithm 1 can be extended in a natural way to any k-node graph H containing a Hamiltonian path, as depicted in Algorithm 2. At the first round, every vertex u sends its identifier to its neighbors, and composes the $d(u)$ graphs formed by the edge $\{u,v\}$, one for every neighbor v. Then, during the $k-2$ following rounds, every node u forwards through each of its edges one of the graphs formed a the previous round.

Let $(u_1, u_2, \dots, u_k)$ be a simple path in G, and assume that $G[\{u_1, u_2, \dots, u_k\}]$ contains H. If, at each round i, $2 \leq i < k$, vertex u_i sends to u_{i+1} the graph $G[\{u_1, \dots, u_i\}]$, then, when the repeat-loop completes, vertex u_k will test precisely the graph $G[\{u_1, u_2, \dots, u_k\}]$, and thus H will be detected by the algorithm. Theorem 1 states that Algorithm 2 works fine for 4-node graphs H. We

Algorithm 2: Testing H-freeness: Hamiltonian H, $|V(H)| = k$. Instructions for node u.

```
send the 1-node graph $G[u]$ to every neighbor $v$;
form the graph $G[\{u,v\}]$ for every neighbor $v$;
repeat $k-2$ times
    for every neighbor $v$ do
        choose a graph $G[A]$ u.a.r. among those formed during the previous round;
        send $G[A]$ to $v$;
        receive the graph $G[A']$ from $v$;
        form the graph $G[A' \cup \{u\}]$;
if $H \subseteq G[A]$ for one of the graphs formed at the last round then reject else accept.
```

show that, $k = 4$ is precisely the limit of detection for graphs that are ϵ-far from being H-free, even for the cliques and the cycles.

Theorem 2. *Let $H = K_k$ for arbitrary $k \geq 5$, or $H = C_k$ for arbitrary odd $k \geq 5$. There exists a graph G that is ϵ-far from being H-free in which any constant number of repetitions of Algorithm 2 fails to detect H, with probability at least $1 - o(1)$.*

For the purpose of proving Theorem 2, we use the following combinatorial result, which extends Lemma 7 of [1], where the corresponding claim was proved for $k = 3$, with a similar proof. The bound on p' is not even nearly optimal in Lemma 2 below, but it is good enough for our purpose[5].

Lemma 2. *Let k be a constant. For any sufficiently large p, there exists a set $X \subset \{0,\dots,p-1\}$ of size $p' \geq p^{1-\frac{\log\log\log p+4}{\log\log p}}$ such that, for any k elements $x_1, x_2, \dots, x_k$ of X, $\sum_{i=1}^{k-1} x_i \equiv (k-1)x_k \pmod{p} \implies x_1 = \cdots = x_{k-1} = x_k$.*

Proof. Let $b = \lfloor \log p \rfloor$ and $a = \left\lfloor \frac{\log p}{\log\log p} \right\rfloor$. Take p sufficiently large so that $a < b/k$ is satisfied. X is a set of integers encoded in base b, on a b-ary digits, such that the digits of each $x \in X$ are a permutation of $\{0, 1, \dots, a-1\}$. More formally, for any permutation π over $\{0,\dots,a-1\}$, let $N_\pi = \sum_{i=0}^{a-1} \pi(i) b^i$. Then, let us set $X = \{N_\pi \mid \pi \text{ is a permutation of } \{0,\dots,a-1\}\}$. Observe that different permutations π and π' yield different numbers N_π and $N_{\pi'}$ because these numbers have different digits in base b. Hence X has $p' = a!$ elements. Using the inequality $z! > (z/e)^z$ as in [1] (Lemma 7), we get that $a! \geq p^{1-\frac{\log\log\log p+4}{\log\log p}}$, as desired.

Now, for any $x \in X$, we have $x \leq p/k$. Indeed, $x < a \cdot b^{a-1} \leq \frac{1}{k} b^a$, and $b^a \leq (\log p)^{\frac{\log p}{\log\log p}} = p$. Consequently, the modulo in the statement of

[5] The interested reader can consult [29] for the state-of-the-art on such combinatorial constructions, in particular constructions for $p' \geq p^{1-c/\sqrt{\log p}}$, for a constant c depending on k.

the Lemma becomes irrelevant, and we will simply consider integer sums. Let $x_1, \dots, x_{k-1}, x_k$ in X, such that $\sum_{l=1}^{k-1} x_i = (k-1)x_k$. Viewing the x_i's as integers in base b, and having in mind that all digits are smaller than b/k, we get that the equality must hold coordinate-wise. For every $1 \leq l \leq k$, let π_l be the permutation such that $x_l = N_{\pi_l}$. For every $i \in 0, \dots, a-1$, we have $\sum_{l=1}^{k-1} \pi_l(i) = (k-1)\pi_k(i)$. By the Cauchy-Schwarz inequality applied to vectors $(\pi_1(i), \dots, \pi_{k-1}(i))$ and $(1, \dots, 1)$, for every $i \in \{0, \dots, a-1\}$, we also have $\sum_{l=1}^{k-1} (\pi_l(i))^2 \geq (k-1)(\pi_k(i))^2$. Moreover equality holds if and only if $\pi_1(i) = \cdots = \pi_{k-1}(i) = \pi_k(i)$. By summing up the a inequalities induced by the a coordinates, observe that both sides sum to exactly $(k-1)\sum_{i=0}^{a-1} i^2$. Therefore, for every i, the Cauchy-Schwarz inequality is actually an equality, implying that the ith digit is identical in all the k integers $x_1, \dots, x_k$. As a consequence, $x_1 = \cdots = x_{k-1} = x_k$, which completes the proof. □

Proof of Theorem 2. Assume that $G[\{u_1, u_2, \dots, u_k\}]$ contains H, where the sequence $(u_1, u_2, \dots, u_k)$ is a path of G. For $2 \leq i \leq k-1$, let us consider the event "at round i, vertex u_i sends the graph $G[u_1, \dots, u_i\}]$ to u_{i+1}". Observe that this event happens with probability $\frac{1}{d(u_i)}$ because u_i choses which subgraph to send uniformly at random among the $d(u_i)$ constructed subgraphs. With the same arguments as the ones used to establish Eq. (1), we get

$$\Pr[H \text{ is detected along the path } (u_1, \dots, u_k)] = \frac{1}{d(u_2)d(u_3)\dots d(u_{k-1})}. \quad (3)$$

We construct families of graphs which will allow us to show, based on that latter equality, that the probability to detect a copy of H vanishes with the size of the input graph G. We actually use a variant of the so-called *Behrend graphs* (see, e.g., [1, 5]), and we construct graph families indexed by k, and by a parameter p, that we denote by $BC(k,p)$ for the case of cycles, and by $BK(k,p)$ for the case of cliques. We prove that these graphs are ϵ-far from being H-free, while the probability that Algorithm 2 detects a copy of H in these graphs goes to 0.

Let us begin with the case of testing cycles. Let p be a large prime number, and let X be a subset of $\{0, \dots, p-1\}$ of size $p' \geq p^{1-\frac{\log\log\log p+4}{\log\log p}}$, where p' is as defined in Lemma 2. Graph $BC(k,p)$ is then constructed as follows. The vertex set V is the disjoint union of an odd number k of sets, $V^0, V^1, \dots V^{k-1}$, of p elements each. For every l, $0 \leq l \leq k-1$, let u_i^l, $i = 0, \dots, p-1$ be the nodes in V^l so that $V^l = \{u_i^l \mid i \in \{0, \dots, p-1\}\}$. For every $i \in \{0, \dots, p-1\}$ and every $x \in X$, edges in $BC(k,p)$ form a cycle

$$C_{i,x} = (u_i^0, \dots, u_{i+lx}^l, u_{i+(l+1)x}^{l+1}, \dots, u_{i+(k-1)x}^{k-1}),$$

where the indices are taken modulo p. The cycles $C_{i,x}$ form a set of pp' edge-disjoint copies of C_k in $BC(k,p)$. Indeed, for any two distinct pairs $(i,x) \neq (i',x')$, the cycles $C_{i,x}$ and $C_{i',x'}$ are edge-disjoint. Otherwise there exists a common edge e between the two cycles. It can be either between two consecutive layers V^l and V^{l+1}, or between V^0 and V^{k-1}. There are two cases. If $e = \{u_y^l, u_z^{l+1}\}$

we must have $y = i + lx = i' + lx'$ and $z = i + (l+1)x = i' + (l+1)x'$, where equalities are taken modulo p, and, as a consequence, $(i, x) = (i', x')$. If $e = \{u^0_y, u^{k-1}_z\}$ then we have $y = i = i'$ and $z = i + (k-1)x' = i' + (k-1)x'$, and, since p is prime, we also conclude that $(i, x) = (i', x')$.

We now show that any k-cycle has exactly one vertex in each set V^l, for $0 \leq l < k-1$. For this purpose, we focus on the parity of the layers formed by consecutive vertices of a cycle. The “short” edges (i.e., ones between consecutive layers) change the parity of the layer, and hence every cycle must include “long” edges (i.e., ones between layers 0 and $k-1$). However, long edges do not change the parity of the layer. Therefore, every cycle contains an odd number of long edges, and an even number of short edges. For this to occur, the only possibility is that the cycle contains a vertex from each layer.

Next, we show that any cycle of k vertices in $BC(k, p)$ must be of the form $C_{i,x}$ for some pair (i, x). Let $C = (u^0_{y_0}, \dots u^l_{y_l}, u^{l+1}_{y_{l+1}}, \dots, u^{k-1}_{y_{k-1}})$ be a cycle in $BC(k, p)$. For every $l = 1, \dots, k-1$, let $x_l = y_l - y_{l-1} \bmod p$. We have $x_l \in X$ because the edge $\{u^{l-1}_{y_{l-1}}, u^l_{y_l}\}$ is in some cycle C_{i,x_l}. Also set $x_k \in X$ such that the edge $\{u^0_{y_0}, u^{k-1}_{y_{k-1}}\}$ is in the cycle C_{y_0,x_k}. In particular we must have that $y_{k-1} = y_0 + (k-1)x_k$. It follows that $y_{k-1} = y_0 + (k-1)x_k = y_0 + x_1 + x_2 + \cdots + x_{k-1}$. By Lemma 2, we must have $x_1 = \cdots = x_{k-1} = x_k$, so C is of the form $C_{i,x}$.

It follows from the above that $BC(k, p)$ has exactly pp' edge-disjoint cycles of k vertices. Since $BC(k, p)$ has $n = kp$ vertices, and $m = kpp'$ edges, $BC(k, p)$ is ϵ-far from being C_k-free, for $\epsilon = \frac{1}{k}$. Also, each vertex of $BC(k, p)$ is of degree $2p'$ because each vertex belongs to p' edge-disjoint cycles.

Let us now consider an execution of Algorithm 2 for input $BC(k, p)$. As $BC(k, p)$ is regular of degree $d = 2p'$, this execution has probability at most $\frac{2k}{d^{k-2}}$ to detect any given cycle C of k vertices. Indeed, C must be detected along one of the paths formed by its vertices in graph $BC(k, p)$, there are at most $2k$ such paths in C (because all C_k's in $BC(k, p)$ are induced subgraphs, and paths are oriented), and, by Eq. (3), the probability of detecting the cycle along one of its paths is $\frac{1}{d^{k-2}}$. Therefore, applying the union bound, the probability of detecting a given cycle C is at most $\frac{2k}{d^{k-2}}$.

Since there are pp' edge-disjoint cycles, the expected number of cycles detected in one execution of Algorithm 2 is at most $\frac{2kpp'}{(2p')^{k-2}}$. It follows that the expected number of cycles detected by repeating the algorithm T times is at most $\frac{2kpT}{2(2p')^{k-3}}$. Consequently, the probability that the algorithm detects a cycle is at most $\frac{2kpT}{2(2p')^{k-3}}$. Plugging in the fact that, by Lemma 2, $p' = p^{1-o(1)}$, we conclude that, for any constant T, the probability that T repetitions of Algorithm 2 detect a cycle goes to 0 when p goes to ∞, as claimed.

The case of the complete graph is treated similarly. Graphs $BK(k, p)$ are constructed in a way similar to $BC(k, p)$, as k-partite graphs with p vertices in each partition (in particular, $BK(k, p)$ also has $n = kp$ vertices). The difference with $BC(k, p)$ is that, for each pair $(i, x) \in \{0, \dots, p-1\} \times X$, we do not add a cycle, but a complete graph $K_{i,x}$ on the vertex set $\{u^0_i, \dots, u^l_{i+lx}, u^{l+1}_{i+(l+1)x}, \dots, u^{k-1}_{i+(k-1)x}\}$. By the same arguments as for $BC(k, p)$, $BK(k, p)$ contains contains exactly pp' edge-disjoint copies of K_k (namely $K_{i,x}$, for each pair (i,x)). This fact holds even for even values of k, because any k-clique must have a vertex in each layer, no matter the parity of k. Thus, in particular $BK(k,p)$ has $m = \binom{k}{2}pp'$ edges, and every vertex is of degree $d = (k-1)p'$. The graph $BK(k,p)$ is ϵ-far from being K_k-free, for $\epsilon = \frac{2}{k(k-1)}$. The probability that Algorithm 2 detects a given copy of K_k is at most $\frac{k!}{d^{k-2}}$. Indeed, a given K_k has $k!$ (oriented) paths of length k, and, by Eq. (3), the probability that the algorithm detects this copy along a given path is $\frac{1}{d^{k-2}}$. The expected number $\mathbf{E}[N]$ of K_k's detected in T runs of Algorithm 2 is, as for $BC(k,p)$, at most $\frac{k!\,Tpp'}{d^{k-2}} = \frac{k!\,pT}{(k-1)^{k-2}(p')^{k-3}}$. Therefore, since $\Pr[N \neq 0] \leq \mathbf{E}[N]$, we get that the probability that the algorithm detects some K_k goes to 0 as p goes to ∞. It follows that the algorithm fails to detect K_k, as claimed. □

Remark. The proof that Algorithm 2 fails to detect C_k for odd $k \geq 5$, can be extended to all (odd or even) $k \geq 13$, as well as to $k = 10$. The cases of C_6, C_8, are C_{12} are open, although we strongly believe that Algorithm 2 also fails to detect these cycles in some graphs.

4 Detecting Small Graphs Using a BFS Approach

We discuss here another very natural approach, extending the algorithm proposed by Censor-Hillel et al. [6] for testing triangle-freeness. In the protocol of [6], each node u samples two neighbors v_1 and v_2 uniformly at random, and asks them to check the presence of an edge between them. We generalize this protocol as follows. Assume that the objective is to test H-freeness, for a graph H containing a universal vertex (a vertex adjacent to every other). Each node u samples $d(u)$ sets $S_1, \ldots, S_{d(u)}$, of $|V(H)| - 1$ neighbors each. For each $i = 1, \ldots, d(u)$, node u sends S_i to all its neighbors in S_i, asking them to check the presence of edges between them, and collecting their answers. Based on these answers, node u can tell whether $G[S_i \cup \{u\}]$ contains H. We show that this very simple algorithm can be used for testing K_4-freeness.

Theorem 3. *There is a 1-sided error distributed property-testing algorithm for K_4-freeness, performing in a constant number of rounds in the* CONGEST *model.*

Again, we show the theorem by exhibiting a generic distributed testing protocol for testing H-freeness, that applies to any graph H on four vertices with a universal vertex. The core of this algorithm is presented as Algorithm 3 where all calculations on indices are performed modulo $d = d(u)$ at node u. This algorithm is presented for a graph H with k nodes.

At Step 3, node u picks a permutation π u.a.r., in order to compose the $d(u)$ sets $S_1, \ldots, S_{d(u)}$, which are sent in parallel at Steps 4-5. At Step 8, every node u considers separately each of the $k-1$ tuples of size $k-1$ received from each of its neighbors, checks the presence of edges between u and each of the nodes in that tuple, and sends back the result to the neighbor from which it received

Algorithm 3: Testing H-freeness for H with a universal vertex. Instructions for node u of degree d. We let $k = |V(H)|$.

```
1 send $\mathrm{id}(u)$ to all neighbors;
2 index the $d$ neighbors $v_0, \dots, v_{d-1}$ in increasing order of their IDs;
3 pick a permutation $\pi \in \Sigma_d$ of $\{0, 1, \dots, d-1\}$, u.a.r.;
4 for each $i \in \{0, \dots, d-1\}$ do
5     Send $(\mathrm{id}(v_{\pi(i)}), \mathrm{id}(v_{\pi(i+1)}), \dots, \mathrm{id}(v_{\pi(i+k-2)}))$ to $v_{\pi(i)}, v_{\pi(i+1)}, \dots, v_{\pi(i+k-2)}$;
6 for each $i \in \{0, \dots, d-1\}$ do
7     for each of the $k-1$ tuples $(\mathrm{id}(w^{(1)}), \dots, \mathrm{id}(w^{(k-1)}))$ received from $v_i$ do
8         Send $(b^{(1)}, \dots, b^{(k-1)})$ to $v_i$ where $b^{(j)} = 1$ iff $u = w^{(j)}$ or $\{u, w^{(j)}\} \in E$;
9 If $\exists i \in \{0, \dots, d-1\}$ s.t. $H \subseteq G[u, v_{\pi(i)}, \dots, v_{\pi(i+k-2)}]$ then reject else accept.
```

the tuple. Finally, the tests $H \subseteq G[u, v_{\pi(i)}, v_{\pi(i+1)}, \dots, v_{\pi(i+k-2)}]$ performed at the last step is achieved thanks to the $(k-1)$-tuple of bits received from each of the neighbors $v_{\pi(i)}, v_{\pi(i+1)}, \dots, v_{\pi(i+k-2)}$, indicating the presence or absence of all the edges between these nodes. Note that exactly $2k-5$ IDs are actually sent through each edges at Steps 4-5, because of the permutation shifts. Similarly, $2k-5$ bits are sent through each edge at Steps 6-8. Therefore Algorithm 3 runs in a constant number of rounds in the CONGEST model. The proof of the following result will appear in a full version of this paper. Among others, it relies on the observation that two disjoint copies of K_4 share at most one vertex (which does not hold for other graphs H).

Lemma 3. *Let G be ϵ-far from being K_4-free. Algorithm 3 for $H = K_4$ rejects G with constant probability.*

5 Limits of the BFS Approach

As it happened with the DFS-based approach, the BFS-based approach fails to generalize to large graphs H. Actually, it already fails for K_5.

Theorem 4. *Let $k \geq 5$. There exists a graph G that is ϵ-far from being K_k-free in which any constant number of repetitions of Algorithm 3 fails to detect any copy of K_k, with probability at least $1 - o(1)$.*

Proof. The family of graphs $BK(k,p)$ constructed in the proof of Theorem 2 for defeating Algorithm 2 can also be used to defeat Algorithm 3. Recall that those graphs have $n = kp$ vertices, $m = \binom{k}{2} pp'$ edges, and every vertex is of degree $d = (k-1)p'$, for $p' = p^{1-o(1)}$. Moreover, they have exactly pp' copies of K_k, which are pairwise edge-disjoint. $BK(k,p)$ is ϵ-far from being K_k-free with $\epsilon = 1/\binom{k}{2}$. For each copy K of K_k, and for every $u \in K$, the probability that u detects K is $d/\binom{d}{k-1} \leq \frac{\alpha}{d^{k-2}}$ for some constant $\alpha > 0$. Therefore, when running the algorithm T times, it follows from the union bound that the expected number of detected copies of K_k is at most $\frac{\alpha k T p p'}{d^{k-2}}$, which tends to 0 when $p \to \infty$, for any $k \geq 5$. Consequently, the probability of detects at least one copy of K_k also tends to 0. □

6 Conclusion and Further Work

The lower bound techniques for the CONGEST model are essentially based on reductions to communication complexity problems. Such an approach does not seem to apply easily in the context of distributed testing. The question of whether the presence of large cliques (or cycles) can be tested in $O(1)$ rounds in the CONGEST model is an intriguing open problem.

It is worth mentioning that our algorithms generalize to testing the presence of *induced* subgraphs. Indeed, if the input graph G contains at least ϵm edge-disjoint *induced* copies of H, for a graph H on four vertices containing a Hamiltonian path, then Algorithm 1 detects an induced subgraph H with constant probability (the only difference is that, in the last line of the algorithm, we check for an induced subgraph instead of just a subgraph). Moreover, if the input contains ϵm edge-disjoint induced claws (i.e., induced subgraphs $K_{1,3}$), then Algorithm 3 detects one of them with constant probability. Thus, for any connected graph H on four vertices, distinguishing between graphs that do not have H as induced subgraph, and those who have ϵm edge-disjoint induced copies of H can be done in $O(1)$ rounds in the CONGEST model. However, we point out that, unlike in the case of subgraphs, a graph that is ϵ-far from having H as induced subgraph may not have many edge-disjoint induced copies of H.

References

1. Alon, N., Kaufman, T., Krivelevich, M., Ron, D.: Testing triangle-freeness in general graphs. SIAM J. Discrete Math. 22(2), 786–819 (2008)
2. Arfaoui, H., Fraigniaud, P., Ilcinkas, D., Mathieu, F.: Distributedly testing cycle-freeness. In: Proc. of WG 2014. pp. 15–28 (2014)
3. Arfaoui, H., Fraigniaud, P., Pelc, A.: Local decision and verification with bounded-size outputs. In: Proc. of SSS 2013. pp. 133–147 (2013)
4. Becker, F., Kosowski, A., Matamala, M., Nisse, N., Rapaport, I., Suchan, K., Todinca, I.: Allowing each node to communicate only once in a distributed system: shared whiteboard models. Distributed Computing 28(3), 189–200 (2015)
5. Behrend, F.A.: On sets of integers which contain no three terms in arithmetical progression. Proceedings of the National Academy of Sciences 32(12), 331 (1946)
6. Censor-Hillel, K., Fischer, E., Schwartzman, G., Vasudev, Y.: Fast distributed algorithms for testing graph properties. CoRR abs/1602.03718 (Feb 2016)
7. Censor-Hillel, K., Kaski, P., Korhonen, J.H., Lenzen, C., Paz, A., Suomela, J.: Algebraic methods in the congested clique. In: Proc. of PODC 2015. pp. 143–152 (2015)
8. Das-Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. SIAM J. Comput. 41(5), 1235–1265 (2012)

9. Dolev, D., Lenzen, C., Peled, S.: "Tri, Tri Again": Finding triangles and small subgraphs in a distributed setting. In: Proc. of DISC 2012. pp. 195–209 (2012)
10. Drucker, A., Kuhn, F., Oshman, R.: On the power of the congested clique model. In: Proc. of PODC 2014. pp. 367–376 (2014)
11. Elkin, M.: An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. SIAM Journal on Computing 36(2), 433–456 (2006)
12. Fraigniaud, P., Göös, M., Korman, A., Suomela, J.: What can be decided locally without identifiers? In: Proc. of PODC 2013. pp. 157–165 (2013)
13. Fraigniaud, P., Korman, A., Peleg, D.: Local distributed decision. In: Proc. of FOCS 2011. pp. 708–717 (2011)
14. Fraigniaud, P., Rajsbaum, S., Travers, C.: Locality and checkability in wait-free computing. Distributed Computing 26(4), 223–242 (2013)
15. Fraigniaud, P., Rajsbaum, S., Travers, C.: On the number of opinions needed for fault-tolerant run-time monitoring in distributed systems. In: Proc. of RV 2014. pp. 92–107 (2014)
16. Goldreich, O. (ed.): Property Testing - Current Research and Surveys, LNCS, vol. 6390. Springer (2010)
17. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. Journal of the ACM 45(4), 653–750 (1998)
18. Göös, M., Jukka Suomela, J.: Locally checkable proofs. In: Proc. of PODC 2011. pp. 159–168 (2011)
19. Hegeman, J.W., Pandurangan, G., Pemmaraju, S.V., Sardeshmukh, V.B., Scquizzato, M.: Toward optimal bounds in the congested clique: Graph connectivity and MST. In: Proc. of PODC 2015. pp. 91–100 (2015)
20. Hegeman, J.W., Pemmaraju, S.V., Sardeshmukh, V.: Near-constant-time distributed algorithms on a congested clique. In: Proc. of DISC 2014. pp. 514–530 (2014)
21. Kari, J., Matamala, M., Rapaport, I., Salo, V.: Solving the induced subgraph problem in the randomized multiparty simultaneous messages model. In: Proc. of SIROCCO 2015. pp. 370–384 (2015)
22. Lenzen, C.: Optimal deterministic routing and sorting on the congested clique. In: Proc. of PODC 2013. pp. 42–50 (2013)
23. Lotker, Z., Patt-Shamir, B., Peleg, D.: Distributed MST for constant diameter graphs. Distributed Computing 18(6), 453–460 (2006)
24. Lotker, Z., Pavlov, E., Patt-Shamir, B., Peleg, D.: MST construction in $\mathcal{O}(\log\log n)$ communication rounds. In: Proc. of SPAA 2003. pp. 94–100 (2003)
25. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press (2005)
26. Patt-Shamir, B., Teplitsky, M.: The round complexity of distributed sorting. In: Proc. of PODC 2011. pp. 249–256 (2011)
27. Peleg, D.: Distributed Computing: A Locality-sensitive Approach. SIAM, Philadelphia, PA, USA (2000)
28. Reiter, F.: Distributed graph automata. In: Proc. of LICS 2015. pp. 192–201 (2015)
29. Schoen, T., Shkredov, I.D.: Roth's theorem in many variables. Israel Journal of Mathematics 199(1), 287–308 (2014)