Distributedly Testing Cycle-Freeness

Heger Arfaoui, Pierre Fraigniaud, David Ilcinkas, Fabien Mathieu

Distributedly Testing Cycle-Freeness

Heger Arfaoui[1], Pierre Fraigniaud[1⋆], David Ilcinkas[2⋆⋆], and Fabien Mathieu[3]

[1] CNRS and University Paris Diderot, France
[2] CNRS and University of Bordeaux, France
[3] Alcatel-Lucent Bell Labs, France

Abstract. We tackle *local distributed testing* of graph properties. This framework is well suited to contexts in which data dispersed among the nodes of a network can be collected by some central authority (like in, e.g., sensor networks). In local distributed testing, each node can provide the central authority with just a few information about what it perceives from its neighboring environment, and, based on the collected information, the central authority is aiming at deciding whether or not the network satisfies some property. We analyze in depth the prominent example of checking *cycle-freeness*, and establish tight bounds on the amount of information to be transferred by each node to the central authority for deciding cycle-freeness. In particular, we show that distributedly testing cycle-freeness requires at least $\lceil \log d \rceil - 1$ bits of information per node in graphs with maximum degree d, even for connected graphs. Our proof is based on a novel version of the seminal result by Naor and Stockmeyer (1995) enabling to reduce the study of certain kinds of algorithms to order-invariant algorithms, and on an appropriate use of the known fact that every free group can be linearly ordered.

1 Introduction

1.1 Context and objective

We are interested in *monitoring* structural properties of networks. Our setting is the one of a large-scale distributed system in which nodes are linked together so as to form a network G. Our objective is to discuss the ability of the nodes to decide whether or not the network satisfies certain structural properties, which may in turn govern the ability of the network to perform certain tasks efficiently. Examples of such structural properties are, e.g., large expansion, which governs the ability to disseminate information quickly, or cycle-freeness, which prevents communication packets to enter into infinite loops. For the purpose of deciding structural properties of the network, each of its nodes can perform some

⋆ The first and second authors receive support from the ANR project DISPLEXITY, and from the INRIA project GANG.
⋆⋆ Partially supported by the ANR project DISPLEXITY. This study was carried out in the frame of the program "investment for the future" of Idex Bordeaux-CPU.

local computation, and eventually produce an individual output based on structural information gathered in its vicinity, and reflecting the local structure of the network around the node. This information is then transmitted to a central authority, which is in charge of taking the final decision about G, as a combination of all individual outputs produced by the nodes. The crucial point here is that the communication channel available between the central authority and each of the nodes is supposed to be narrow, and hence the amount of information that can be transmitted from each node to the central authority is limited. Yet, we want the central authority to be able to decide whether or not G satisfies some given structural property. A typical example of such a setting is a sensor network, in which the sensed data are gathered at a distant base station, either directly, or via intermediate routers and/or other sensors.

The setting presented above shares characteristics with both *property testing* [14] and *distributed decision* [12, 13]. Indeed, at a conceptual level, property testing on graphs can be viewed as: (1) querying a small number of nodes, typically $o(n)$, or even $O(1)$ nodes, in n-node networks; (2) extracting information from each query, typically $O(\log n)$ bits of information (e.g., the identity of a neighbor of the queried node); and (3) deciding whether the queried graph satisfies some given property $\mathcal{P}$, on the basis of the collection of information obtained from the queried nodes. It is the role of the tester algorithm to choose which nodes to query, and to eventually take the decision about the tested graph. The lack of information resulting from querying just a small subset of nodes is balanced by relaxing the decision requirement, which is subject to probabilistic errors, and does not impose to reject illegal instances that are "close" to legal instances.

Similarly, distributed decision in graphs [12, 13] can be viewed as (1) querying *all* nodes; (2) having every node providing a *single bit* of information (true or false) based on local information gathered in its vicinity; and (3) deciding whether the queried graph satisfies some given property $\mathcal{P}$, on the basis of the collection of boolean information obtained from the queried nodes. In distributed decision, the instance is accepted if and only if the *logical conjunction* of the boolean information computed at each node is true. That is, if the input graph satisfies $\mathcal{P}$, then every node must individually accept. Otherwise, at least one node must individually reject. Distributed decision assumes no gap between, on the one hand, the instances to be accepted, and, on the other hand, the ones to be rejected. Furthermore, the decision is usually deterministic and error-free.

The formal model used in this paper for monitoring structural properties of networks relaxes property testing in the sense that, as for distributed decision, all nodes are queried. It also relaxes distributed decision in the sense that, as for property testing, the decision is made by an algorithm taking as input structured information provided by the nodes (and not only boolean information). Therefore, as far as the computational constraints are concerned, our model is very liberal, by taking the best of property testing, and of distributed decision. On the other hand, the model is very conservative regarding the final output, by allowing no errors, and by requiring perfect dichotomy between the legal in-

	#queried nodes	amount of information	decision mechanism	gap	error
property testing [14]	$o(n)$	$O(\log n)$	algorithm	ϵ-far	yes
distributed decision [12, 13]	n	1	logical conjunction	none	no
distributed testing	n	$O(\log n)$	algorithm	none	no

Table 1. *Distributed testing*

stances and the illegal instances. We call this model *distributed testing*. Its main characteristics are summarized in Table 1.

One illustrative example of the differences between, on the one hand, distributed testing, and, on the other hand, property testing and distributed decision, is *cycle-freeness* (see Table 2). For 2-sided error, it is known [15] that cycle-freeness in graphs with maximum degree d can be property tested with $O(\frac{1}{\epsilon^3} + \frac{d}{\epsilon^2})$ queries returning $\Theta(\log n)$ bits per queried node, where $\epsilon \in (0, 1)$ is the gap parameter between the legal and illegal instances. For 1-sided error, $\Omega(\sqrt{n})$ queries are required [15], and this is sufficient [8]. If one does not allow errors, it is folklore that, even in connected graphs, cycle-freeness cannot be distributedly decided[4]. It is however known [18] that cycle-freeness in connected graphs can be *verified* distributedly with the help of $O(\log n)$-bit additional information (i.e., *certificates*) per node. As for distributed decision, each node just outputs a boolean, and the global decision is the logical conjunction of these booleans. Moreover, [9, 19] proved that $\Omega(\log n)$-bit certificates are necessary for verifying trees. In [3], the size of the certificates is nevertheless decreased to constant, by allowing nodes to output just 2 bits instead of 1. (We call this latter setting distributed *certification*).

Of course, there is a very simple algorithm for distributedly testing cycle-freeness in connected graphs: each node v output its degree $\deg(v)$, and the central authority accepts if and only if $\sum_v \deg(v) = 2(n-1)$. The number of bits returned by each queried node is $\lceil \log d \rceil$ in graphs with maximum degree d. One question is: can we do better, i.e., with less bits of information transmitted from each node? Note that the answer is yes for *subdivided* graphs, where a subdivided graph [1] is a graph in which no two vertices of degree different from 2 are adjacent. Indeed, in such graphs with $n \geq 3$, the following algorithm works, with only four different kinds of outputs (i.e., 2-bit outputs): a node with degree $\neq 2$ outputs 0, and a node with degree 2 outputs 2, 3 or 4 depending on whether it is adjacent to 0, 1 or 2 nodes with degree $\neq 2$, respectively. The central authority then accepts if and only if the sum of the outputs equals $2(n-1)$.

[4] To see why, assume there exists a local algorithm $\mathcal{A}$ deciding cycle-freeness locally. Run $\mathcal{A}$ on the path with consecutive identities from 1 to n (nodes with identities 1 and n being the two extremities). In this configuration, the $n/2$ middle nodes output "true". Then, run $\mathcal{A}$ on the same path with identities $n/2, \ldots, n, 1, \ldots, n/2 - 1$. Again, the $n/2$ middle nodes output "true". Therefore, on the cycle with consecutive identities from 1 to n, all nodes output "true", yielding $\mathcal{A}$ to accept the cycle, a contradiction.

	#queried nodes	amount of information	success probability	Comments
property testing [15]	$O(\frac{1}{\epsilon^3} + \frac{d}{\epsilon^2})$	$\lceil \log n \rceil$	2-sided	ϵ-far
property testing [8, 15]	$\Theta(\sqrt{n})$	$\lceil \log n \rceil$	1-sided	ϵ-far
distributed decision [folklore]	n	1	impossible	–
distributed verification [9, 18, 19]	n	1	deterministic	$\Theta(\log n)$-bit certificates
distributed certification [3]	n	2	deterministic	$O(1)$-bit certificates
distributed testing [this paper]	n	$\log d \pm \Theta(1)$	deterministic	–

Table 2. *Monitoring cycle-freeness in n-node max-degree-d (connected) graphs*

In this paper, we question the existence of a distributed tester for cycle-freeness in arbitrary graphs, returning less than $\lceil \log d \rceil$ bits from each of the n queried nodes.

1.2 Our results

We prove that every distributed tester for cycle-freeness in graphs with maximum degree d requires that at least one node outputs at least $\lceil \log d \rceil - 1$ bits. Hence, the distributed tester in which every node simply outputs its degree is essentially optimal. This tight result completes the whole picture regarding checking cycle-freeness (see Table 2). That is, if one can stand errors and slacks then property-testing enables to query just a few nodes. On the other hand, if one insists on deterministically systematically rejecting graphs with cycles, and accepting graphs without cycles, then distributed testing seems to be the right option. Indeed, it consumes moderate bandwidth resources to gather the outputs of the nodes, and needs not to provide certificates (as opposed to distributed verification, and distributed certification).

Establishing that every distributed tester for cycle-freeness must output $\lceil \log d \rceil - 1$ bits at some node requires to combine several techniques. First, we show that one can reduce our concern to *order-invariant* testers, that is, roughly, to algorithms whose output at a node does not depend on the actual *value* of the identities of the nodes in its vicinity, but solely on the *relative order* of these values. The celebrated result by Naor and Stockmeyer [20] enabling to reduce the study of certain kinds of algorithms to order-invariant algorithms cannot be applied in our context because our instances are not necessarily in the class LCL of so-called *locally checkable languages*. Nevertheless, we were able to provide a novel reduction, that does not require LCL membership, by using the *infinite version* of Ramsey Theorem.

Our second main technique is the construction, for every order-invariant distributed tester supposed to decide cycle-freeness with too few information provided by each node, of two explicit instances, one with a cycle, and one without, that cannot be distinguished by the tester. This construction is difficult because, as mentioned before, cycle-freeness can be distributedly tested with just a 2-bit output per node in subdivided graphs. Nevertheless, by an appropriate use of the known fact that every free group can be linearly ordered, we were able to

construct legal and illegal instances that cannot be distinguished locally by the assumed order-invariant distributed tester.

1.3 Related work

Local computing is a wide domain of studies in distributed network computing, and the reader is referred to the textbook [21] for an excellent introduction to local computing, providing pointers to the most relevant techniques for solving prominent problems (e.g., MIS, coloring, etc.) locally. The question of what can be computed in a constant number of communication rounds was actually introduced in the seminal work by Naor and Stockmeyer in [20]. In particular, [20] introduced the class of *locally checkable languages* (LCL), and studied the question of how to deterministically or randomly construct instances of LCL languages in a constant number of rounds.

With the objective of providing distributed network computing with a complexity theory based on decision problems, following the guidelines of classical (sequential) complexity theory, [12, 13] introduced several decision classes for local computing, and studied the relationships between these classes (which are depending on the number of allowed rounds, on the potential access to oracles, on the potential use of non-determinism and/or of randomization, etc.). Paper [12] generated several following up contributions, including, e.g., studies on the impact of randomization [11], studies on the impact of node identifiers [10], studies on verification tasks where certificates include node IDs [17], etc. See also [16] for other forms of local checking, and for their impact on distributed graph-optimization problems.

Local distributed testing was introduced in [3] (although [3] does not use this terminology). Beside introducing complexity classes related to local distributed testing, and studying the relationships between these classes, [3] focused attention to verification, and on the size of the certificates involved in the verification. Our paper is also very much related to [4, 5], which use models that resemble local distributed testing, but where nodes are restricted to perform just one round of communication before outputting a value on $O(\log n)$ bits. In the restricted setting of [4, 5], even checking the presence of a 4-cycle in the network may not be feasible. Instead, in local distributed testing, the number of communication rounds is just restricted to be constant, but one aims at producing smaller output values, e.g., on $O(1)$ bits.

2 Local Distributed Testing

Let us consider a network modeled as a simple *connected* graph G. We are interested in the task consisting, for the nodes of G, to collectively decide whether G satisfies some given property $\mathcal{P}$, like, say, being planar, being a tree, etc. For this purpose, nodes can exchange information, so that every node eventually produces an output. The computational model considered in this paper is the classical $\mathcal{LOCAL}$ model [21], which is a standard distributed computing model

capturing the essence of locality. In this model, nodes have pairwise distinct identities (the identity of node v is denoted by $\mathsf{id}(v) \in \mathbb{N}$). They are woken up simultaneously, and computation proceeds in fault-free synchronous *rounds* during which every node exchanges messages of unlimited size with its neighbors in the underlying network G, and performs arbitrary individual computations on its data. The running time of an algorithm is defined as the maximum number of rounds it takes to terminate at all nodes, over all possible networks, and all possible identity assignments for the nodes in these networks. Similarly to [20], we consider algorithms whose running time is independent of the size of the network, and independent of the size of the identities. That is, they run in constant time.

Let $\mathsf{out}_{\mathcal{A}}(G, \mathsf{id}, v)$ denotes the output of node $v \in V(G)$ running Algorithm $\mathcal{A}$ in G with identity assignment id. We denote by $\mathsf{out}_{\mathcal{A}}(G, \mathsf{id})$ the global output, that is,

$$\mathsf{out}_{\mathcal{A}}(G, \mathsf{id}) = \{\mathsf{out}_{\mathcal{A}}(G, \mathsf{id}, v), v \in V(G)\}$$

is the multiset of all individual outputs (the same individual output may appear more than once in $\mathsf{out}_{\mathcal{A}}(G, \mathsf{id})$). By "collectively decide" a graph property $\mathcal{P}$, we mean the following. To each output corresponds a global state of the system. We question the ability to define two classes of global states, one called *accept*, and one called *reject*, so that the following holds. If G satisfies $\mathcal{P}$, then the nodes must compute outputs that yields the system to be in an accept state, while if G does not satisfy $\mathcal{P}$, then the nodes must compute outputs that yields the system to be in a reject state. More specifically, assume that each node of an n-node network can output one of the different values in a set S. Let $M_{n,S} = \binom{S}{n}$ be the set of all multisets of cardinality n, with elements taken from S. We say that a graph property $\mathcal{P}$ can be distributedly tested with output set S if there exists a local distributed algorithm $\mathcal{D}$, and a decomposition of $M_{n,S}$ for every $n \geq 1$, into two computable sets Y_n (the "yes"-set, or accept set) and $M_{n,S} \setminus Y_n$ (the reject set) such that, for every n-node graph G, and for every identity assignment id to the nodes in G, the following holds:

$$G \text{ satisfies } \mathcal{P} \iff \mathsf{out}_{\mathcal{D}}(G, \mathsf{id}) \in Y_n.$$

In other words, a distributed tester for $\mathcal{P}$ consists in a local distributed algorithm $\mathcal{D}$ producing an output at every node, coupled with a sequential algorithm $\mathcal{S}$ which takes as input the collection of all outputs produced by the nodes, and accepts or rejects, under the constraint that it must accept if and only if G satisfies $\mathcal{P}$.

An example (borrowed from [13]) of a graph property that can be distributedly tested is: clique-width at most 2. Indeed, a graph has clique-width at most 2 if and only if it is a cograph – see [7]. Now, cographs have also been characterized as the family of P_4-free graphs (i.e., the graphs which do not contain the path on 4 nodes as an induced subgraph) – see [6]. Hence, to decide clique-width at most 2, each node can simply communicate at bounded distance to check whether it contains an induced P_4 in its vicinity, and output 0 if it is the case, and 1 otherwise. The accept set is simply defined as

$Y_n = \{\{x_1, \ldots, x_n\} \in M_{n,\{0,1\}} : \prod_{i=1}^{n} x_i = 1\}$. Distributed testing restricted to this latter class of accept sets actually reduces to distributed decision [13]. (See [2] for the difficulty of property testing cographs).

The size of S needs not be constant, and may actually vary with n. This is for instance the case of the aforementioned task of testing cycle-freeness, for which every node simply returns its degree. In this case, $Y_n = \{\{x_1, \ldots, x_n\} \in M_{n,[n]} : \sum_{i=1}^{n} x_i = 2(n-1)\}$, where $[n] = \{1, \ldots, n\}$.

Note that every computable graph property $\mathcal{P}$ can be distributedly tested in this model. Indeed, each node can just output its identity, and the set of all the identities of its neighbors. In other words, each node v outputs its identity and its adjacency list L_v in the current graph G. In this case,

$$Y_n = \{\{L_1, \ldots, L_n\} : \text{graph } (L_1, \ldots, L_n) \text{ satisfies } \mathcal{P}\}$$

would enable to distinguish graphs that satisfy $\mathcal{P}$ from those that do not. However, such a trivial solution involves individual outputs of size $\Omega(n \log n)$ bits in dense graphs. Our objective is to study the ability of distributedly testing graph properties with individual outputs having size as small as possible, ideally constant, independent of the network size, and of the range of identities. We define the *output size* of a distributed tester in a graph family $\mathcal{G}$ as the maximum, taken over all instances (G, id) where $G \in \mathcal{G}$, of the maximum number of bits outputted by a node in this instance.

3 Order-invariance revisited

Our first result in the paper is a key ingredient for the proof of our main result. This ingredient may have its interest on its own, and it is worth dedicating an entire section to it. We show that, w.l.o.g., one can consider only *order-invariant* distributed testers. Recall that an order-invariant distributed algorithm is a distributed algorithm for which the output at any given node does not depend on the actual values of the identities of the nodes in its vicinity, but only on the relative order of these identities. More precisely, let $B_G(v,t)$ be the ball of radius t around node v in graph G, that is, $B_G(v,t)$ is the subgraph of G induced by all nodes at distance at most t from v, excluding the edges between the nodes at distance exactly t from v. An algorithm $\mathcal{A}$ is order-invariant if the following holds: for any graph G, for any node v, and for any two identity assignments id and id' of the nodes in G, if the ordering of the nodes in $B_G(v,t)$ induced by id, and the one induced by id' are identical, then the output of $\mathcal{A}$ at node v is the same in both (G, id) and (G, id').

A *distributed language* $\mathcal{L}$ is defined by a collection of labeled graphs, or *configurations* (G, ℓ) where G is a connected graph, and $\ell : V(G) \to \{0,1\}^*$ is a function that labels each node v with the label $\ell(v)$. The *construction* task defined by a language $\mathcal{L}$ consists, for every node v of every graph G, to compute $\mathsf{out}(v)$ such that the global output satisfies $(G, \mathsf{out}) \in \mathcal{L}$. In their seminal paper, Naor and Stockmeyer [20] consider the subclass LCL of *locally checkable languages*. Languages in LCL are distributed languages that are defined on graph

families with constant maximum degree, and with constant label size at every node (i.e., $|\ell(v)| = O(1)$ for every node v). A language $\mathcal{L}$ is locally checkable, or, alternatively, is in LD according to the terminology of [13], if there exists a distributed algorithm performing in a constant number of rounds such that, for any $(G, \ell) \in \mathcal{L}$, all nodes accept, and, for any $(G, \ell) \notin \mathcal{L}$, at least one node rejects. (See [13, 20] for more details). Theorem 3.3 in [20] establishes that, for every language $\mathcal{L} \in$ LCL, if there exists a construction algorithm for $\mathcal{L}$ performing in $t = O(1)$ rounds, then there exists a t-round order-invariant construction algorithm for $\mathcal{L}$.

In the context of this paper, the language corresponding to a distributed tester $(\mathcal{D}, \mathcal{S})$ is determined by the accept set, that is by the set of multisets of outputs that $\mathcal{S}$ accepts. In general, the language corresponding to a distributed tester $(\mathcal{D}, \mathcal{S})$ for a property $\mathcal{P}$ is

$$\mathcal{L} = \{(G, \ell) : \mathcal{S} \text{ accepts } \ell \iff G \text{ satisfies } \mathcal{P}\}$$

where ℓ is the collection of values $\ell(v)$, $v \in V(G)$, and $\ell(v)$ is the value owned by node v. In particular, the language corresponding to the distributed tester $(\mathcal{D}, \mathcal{S})$ for cycle-freeness where $\mathcal{D}$ outputs $\deg(v)$ at each node v is

$$\mathcal{L}_{\text{cycle-free}} = \{(G, \ell) : \textstyle\sum_{v \in VG} \ell(v) = 2(n-1) \iff G \text{ is an } n\text{-node tree}\}.$$

Observe that such languages are not necessarily locally checkable. For instance, $\mathcal{L}_{\text{cycle-free}} \notin$ LCL (even if restricted on graphs with maximum degree d, for some constant d). Hence, Theorem 3.3 in [20] does not apply to our setting. The result below extends this latter theorem to non locally checkable languages. We define the *domain* of a language $\mathcal{L}$ as the set of all values taken by the labels $\ell(v)$ in $\mathcal{L}$, for all graphs G, and all nodes v of G. Note that, in a construction task defined by a distributed language $\mathcal{L}$, nodes may be a priori provided with inputs. In this context, $x(v)$ denotes the input to node v, and every node v has to compute an output $y(v)$ such that $(G, (x, y)) \in \mathcal{L}$.

Theorem 1. *For every non-negative integers k, t, d, and every language $\mathcal{L}$ defined on connected graphs with maximum degree d, and k-valued domain, if there exists a t-round construction algorithm $\mathcal{A}$ for $\mathcal{L}$, then there is a t-round order-invariant construction algorithm $\mathcal{A}'$ for $\mathcal{L}$.*

Proof. For any set X, and any positive integer r, let us denote by $X^{(r)}$ the set of all subsets of X with size exactly r. Let X be a countably infinite set, let r and s be two positive integers, and let $c : X^{(r)} \to [s]$ be a "coloring" of each set in $X^{(r)}$ by an integer in $[s] = \{1, \ldots, s\}$. Recall that (the infinite version of) Ramsey's Theorem states that there exists an infinite set $Y \subseteq X$ such that the image by c of $Y^{(r)}$ is a singleton (that is, all sets in $Y^{(r)}$ are colored the same by c). We make use of this theorem as follows.

Let us consider the collection $\mathcal{B}$ of all graphs isomorphic to some ball $B_G(v, t)$ of radius t, centered at some node v in some graph G with maximum degree d. If the language $\mathcal{L}$ is described by labels encoding input-output relations, then $\mathcal{B}$

is the collection of all labeled graphs isomorphic to some labeled ball $B_G(v,t)$. Since the domain of the labels has k values, there are at most k different inputs, and thus there is a finite number β of pairwise non-isomorphic balls in $\mathcal{B}$.

We enumerate these (labeled) balls from 1 to β, and let n_i be the number of vertices in the ith ball, for $i = 1, \ldots, \beta$. For every i, the vertices of the ith ball can be ordered in $n_i!$ different manners, corresponding to the $n_i!$ permutations in Σ_{n_i}. We consider the $N = \sum_{i=1}^{\beta} n_i!$ ordered balls $B_{i,\sigma}$, for $i = 1, \ldots, \beta$, and $\sigma \in \Sigma_{n_i}$, and we enumerate these ordered balls as $\mathbf{B}_1, \ldots, \mathbf{B}_N$ in an arbitrary order. Using these balls, we define an infinite set $\mathcal{I}$ of identities as follows.

Let $X_0 = \mathbb{N}$, and assume that we have already secured the existence of a sequence of infinite sets $X_0 \supseteq X_1 \supseteq \ldots \supseteq X_j$, $0 \leq j < N$, such that, for every i, $1 \leq i \leq j$, the output of $\mathcal{A}$ at the center of $\mathbf{B}_i$ is the same for all possible identity assignments to the nodes in $\mathbf{B}_i$ with values in X_i, and respecting the ordering of the nodes in $\mathbf{B}_i$. We define the coloring $c : X_j^{(r)} \to [k]$ where r is the number of nodes in $\mathbf{B}_{j+1}$, as follows: for each r-element set $I \in X_j^{(r)}$, assign r pairwise distinct identities to the nodes of $\mathbf{B}_{j+1}$ using the r values in I, and respecting the order of the nodes in $\mathbf{B}_{j+1}$. Then, define $c(I)$ as the output of Algorithm $\mathcal{A}$ at the center of $\mathbf{B}_{j+1}$ under this identity assignment to the nodes of $\mathbf{B}_{j+1}$. By Ramsey's Theorem, there exists an infinite set $Y_j \subseteq X_j$ such that all r-element sets $I \in Y_j^{(r)}$ are given the same color. We set $X_{j+1} = Y_j$. We proceed that way until we exhaust all balls $\mathbf{B}_i$, $i = 1, \ldots, N$, and we set $\mathcal{I} = X_N$.

By construction, the set $\mathcal{I}$ satisfies that, for every ball $B_{i,\sigma}$, for $i = 1, \ldots, \beta$, and $\sigma \in \Sigma_{n_i}$, the output of $\mathcal{A}$ at the center of $B_{i,\sigma}$ is the same for all identity assignments to the nodes of $B_{i,\sigma}$ with identities taken from $\mathcal{I}$ and assigned to the nodes in the order σ.

We now define the order-invariant algorithm $\mathcal{A}'$ as follows. Every node v inspects its radius-t ball $B_G(v,t)$ around it in the actual graph G. In particular, it collects the identities of the nodes in that ball. Let σ be the ordering of the nodes in $B_G(v,t)$ induced by their identities. Node v simulates $\mathcal{A}$ by reassigning identities to the nodes of $B_G(v,t)$ using the $r = |B_G(v,t)|$ smallest values in $\mathcal{I}$, in the order specified by σ, and outputs what would have outputted $\mathcal{A}$ if nodes were given these identities.

$\mathcal{A}'$ is well defined, as nodes can be provided with the $\nu = \sum_{i=0}^{t} d^i$ smallest integers in the set $\mathcal{I}$. (I.e., nodes do not need to know the entire set $\mathcal{I}$, but only a finite number of values in $\mathcal{I}$). Also, by construction, $\mathcal{A}'$ is order-invariant. To establish that $\mathcal{A}'$ is correct, let us consider some n-node input graph G, with nodes provided with pairwise distinct identities in $\mathcal{I}$, and let $\mathsf{out} = \{\mathsf{out}(v), v \in V(G)\}$ be the output of $\mathcal{A}$ in this context. This output is precisely the multi-set outputted by $\mathcal{A}'$ in G. Indeed, every node v relabels its radius-t ball with identities in $\mathcal{I}$, respecting the order induced by the original identities in $\mathcal{I}$, and $\mathcal{I}$ is precisely defined so that the output of v will be the same in both cases. In other words, the output of $\mathcal{A}'$ is precisely the output of $\mathcal{A}$ if nodes were assigned identities restricted to be in $\mathcal{I}$. Hence, since $\mathcal{A}$ is correct, it follows that $\mathcal{A}'$ is correct as well. □

4 Distributedly testing cycle-freeness

In this section, we prove our main result:

Theorem 2. *For any positive even integer d, every distributed tester for cycle-freeness in connected graphs with maximum degree at most d has output size at least $\lceil \log d \rceil - 1$ bits.*

The rest of the section is entirely dedicated to prove this result. The proof is by contradiction. Let d be a positive even integer. We assume the existence of a distributed tester $(\mathcal{D}, \mathcal{S})$ for cycle-freeness in connected graphs with maximum degree d, where $\mathcal{D}$ runs in t rounds, for some constant $t \geq 0$, and outputs at most $\lceil \log d \rceil - 2$ bits at each node. We first start by shrinking the set of candidate algorithms $\mathcal{D}$. Indeed, as a direct consequence of Theorem 1, we get the following:

Corollary 1. *If there exists a t-round distributed tester $(\mathcal{D}, \mathcal{S})$ for cycle-freeness with k-valued outputs in connected graphs with maximum degree d, then there is distributed tester $(\mathcal{D}', \mathcal{S})$ satisfying the same, but where $\mathcal{D}'$ is order-invariant.*

Based on this latter result, we now show that every distributed tester $(\mathcal{D}, \mathcal{S})$ for cycle-freeness in connected graphs with maximum degree at most d, where $\mathcal{D}$ is order-invariant, has output size at least $\lceil \log d \rceil - 1$ bits.

The intuition is as follows. We will focus our attention on so-called type-i nodes, with i being an even integer between 2 and d. Intuitively, such nodes are defined as nodes of degree i that only "see" a tree of nodes of degree i in their neighborhood up to distance t, and with a particular ordering of their identities. We will construct two (connected) graphs, with their corresponding identity assignments, such that only one of this two graphs is a tree, and the multi-set of the local views gathered by the nodes in the two graphs will only differ by their numbers of type-i and type-j nodes, for some $i \neq j$. Any distributed tester for cycle-freeness has to distinguish the two graphs and has thus to give different output values to type-i and type-j nodes. This will prove that any distributed tester for cycle-freeness must have at least $d/2$ different output values, thus proving our main theorem.

We now define formally two families of trees, which will be used as building blocks in our constructions. (See Fig. 1). Let i, $1 \leq i \leq d$, be an even integer. We define the trees T_i and T_i' as follows. For T_i, we start from one single node, called the *downtown* node (this node is considered as a leaf). For T_i', we start from $i+1$ nodes organized as a star (i.e., with one center and i leaves), and also called *downtown* nodes. Then, we replace each of the leaves of these two "seeds" by a $(i-1)$-ary tree of height t. As a consequence, all the internal nodes of the resulting two trees are of degree i. Moreover, the downtown node closest to every leaf is at distance exactly $t+1$. The internal nodes of the resulting trees that are not downtown nodes are called *suburb* nodes. For the ease of description of our constructions, and for simplifying our arguments, we assign numbers to the edges incident to the internal nodes of these two trees. More specifically, for each internal node v, a distinct label between 1 and $\deg(v)$ is assigned to every edge e

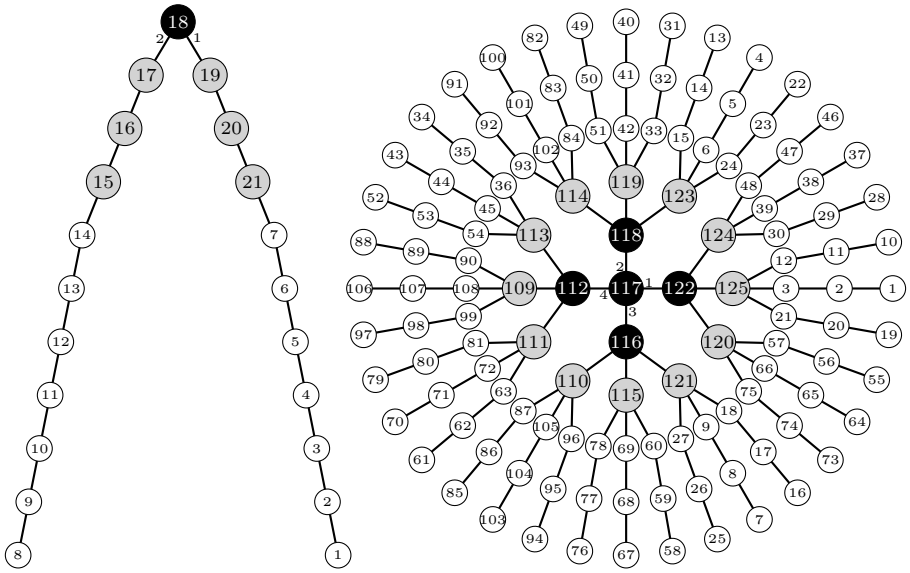

Fig. 1. The trees T_2, for $t = 3$, and T'_4, for $t = 1$. The downtown, suburb, and countryside nodes are depicted as black, grey, and white nodes respectively. Port numbers are only indicated for the central downtown node. The other port numbers can be inferred using the same cyclic ordering of port numbers around the nodes.

incident to v. This label is called the *port number* of the edge e at node v. The port numbers are assigned in such a way that, for each edge whose extremities are two nodes with the same degree i, if $p \in [1, i]$ is one of the port numbers assigned to the edge, then $i - p + 1$ is the other port number assigned to this edge. Then, we apply another transformation, which consists in replacing every edge of these two trees that is incident to a leaf by a path of length $2t + 1$. The trees T_i and T'_i are the trees resulting from this second transformation. In these two trees, all the nodes that are neither downtown nodes, nor suburb nodes, are called *countryside* nodes.

Before describing the identity assignments for these trees, let us make the following observations. An infinite regular tree of degree i can be viewed as the Cayley graph of the free group of rank $i/2$. More precisely, let $\{a_1, a_2, \ldots, a_{i/2}\}$ be the set of the $i/2$ generators of the free group $(F, \star)$ of rank $i/2$. The Cayley graph associated to this group is a directed arc-labeled graph with the following properties: the set of nodes is the set F, and there is an arc with label a_p, with $1 \leq p \leq i/2$ from node g to node g' if and only if $g' = g \star a_p$. If each arc (g, g') with label a_p is replaced by an (undirected) edge $\{u, v\}$ with port label p at u and $i - p + 1$ at v, then we get the infinite regular tree of degree i with a port-labeling similar to the one we have described for T_i and T'_i. More precisely, for each node in this infinite tree, the edges incident to it are assigned a local

port number from 1 to i such that if p is one of the port numbers assigned to an edge, then $i - p + 1$ is the other port number assigned to this edge. Besides, any finitely generated free group is bi-ordered, i.e., admits a total order $\preceq$ such that, for any three elements a, b, and c of the group, if $a \preceq b$, then $a \star c \preceq b \star c$ and $c \star a \preceq c \star b$.

Let us now describe the identity assignments for the trees T_i and T'_i. The construction is similar in both cases. The countryside nodes will receive the lower identities, while the suburb and downtown nodes will receive the larger identities. More specifically, to every countryside node u, we associate its distance j to the closest leaf, and the sequence s of the $t + 1$ port numbers describing the path going from the closest downtown node to u. The countryside nodes are assigned identities respecting the lexicographic order of their pair (s, j), with ties broken arbitrarily. The suburb and downtown nodes are assigned identities that are compatible with the total order $\preceq$ of the corresponding free group. Again, see Fig. 1 for examples of these identity assignments.

A node having the same local view up to distance t as the local view up to distance t of a downtown node of T_i, except for actual identity values but respecting the order of these identities, is called a type-i node.

For the purpose of contradiction, assume that there exists an order-invariant distributed tester for cycle-freeness using less than $d/2$ output values. Therefore there must exist two even integers i, j with $i < j \leq d$, such that the distributed tester outputs the same value for type-i and type-j nodes. Hence, let G'_1 be the graph formed by the disjoint union of one copy of T'_i and $j - 1$ copies of T_j, with disjoint ranges of identity values assigned to the nodes in these copies. Connect $j - 1$ disjoint pairs of leaves by an edge to make the graph connected. We denote by G_1 the resulting graph. Note that G_1 is a tree. Similarly, let G'_2 be the graph formed by the disjoint union of $i - 1$ copies of T_i and one copy of T'_j, with disjoint ranges of identities assigned to the nodes in these copies. Connect $i - 1$ disjoint pairs of leaves by as many edges to make the graph connected. Further, connect $j - i$ other pairs of leaves to create cycles. We denote by G_2 the resulting graph. Note that G_2 is connected but is not a tree.

The multiset of local views up to distance t, ignoring the identity values but taking into account their relative order, is exactly the same in both graphs G_1 and G_2, with the only exception that G_1 has two more type-i nodes than G_2, and two less type-j nodes than G_2. The distributed tester $(\mathcal{D}, \mathcal{S})$ will thus take the same decision for both graphs, which contradicts the fact that it is a correct distributed tester for cycle-freeness. This contradiction completes the proof of the theorem. □

References

1. N. Alon. Subdivided graphs have linear ramsey numbers. *Journal of Graph Theory* **18**(4): 343-347 (1994)
2. N. Alon and A. Shapira. A Characterization of Easily Testable Induced Subgraphs. *Combinatorics, Probability & Computing* **15**(6): 791-805 (2006)

3. H. Arfaoui, P. Fraigniaud, and A. Pelc. Local Decision and Verification with Bounded-Size Outputs. *Proc. 15th Symp. on Stabilization, Safety, and Security of Distributed Systems* (SSS), pp133-147, 2013.
4. F. Becker, A. Kosowski, N. Nisse, I. Rapaport, and K. Suchan. Allowing each node to communicate only once in a distributed system: shared whiteboard models. *Proc. 24th ACM Symp. on Parallelism in Alg. and Architectures* (SPAA), pp11-17, 2012.
5. F. Becker, M. Matamala, N. Nisse, I. Rapaport, K. Suchan, and I. Todinca. Adding a referee to an interconnection network: What can(not) be computed in one round. *Proc. 25th IEEE Int. Symp. on Parallel and Dist. Proc.* (IPDPS), pp508-514, 2011.
6. D. Corneil, H. Lerchs, L. Burlingham. Complement reducible graphs. *Discrete Applied Mathematics* **3**(3): 163-174 (1981)
7. B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics* **101**(13): 77-144 (2000)
8. A. Czumaj, O. Goldreich, D. Ron, C. Seshadhri, A. Shapira, and C Sohler. Finding Cycles and Trees in Sublinear Time. *Proc. Electronic Colloquium on Computational Complexity* (ECCC) **19**: 35 (2012)
9. S. Dolev, M. G. Gouda, and M. Schneider. Memory Requirements for Silent Stabilization. *Acta Inf.* **36**(6):447-462 (1999)
10. P. Fraigniaud, M. Göös, A. Korman, J. Suomela. What can be decided locally without identifiers? *Proc. 32nd ACM Symp. on Principles of Distributed Computing* (PODC), pp157-165, 2013.
11. P. Fraigniaud, A. Korman, M. Parter, D. Peleg. Randomized Distributed Decision. *Proc. 26th Int. Symp. on Distributed Computing* (DISC), pp371-385, 2012.
12. P. Fraigniaud, A. Korman, and D. Peleg. Local Distributed Decision. *Proc. 52nd Annual IEEE Symp. on Foundations of Comp. Science* (FOCS), pp708-717, 2011.
13. P. Fraigniaud, A. Korman, and D. Peleg. Towards a complexity theory for local distributed computing. *J. ACM* **60**(5):35 (2013)
14. O. Goldreich (Ed.). Property Testing — Current Research and Surveys. LNCS 6390, Springer, 2010.
15. O. Goldreich and D. Ron. Property Testing in Bounded Degree Graphs. *Algorithmica* **32**(2): 302-343, 2002.
16. M. Göös, J. Hirvonen, and J. Suomela. Lower bounds for local approximation. *J. ACM* **60**(5):39 (2013)
17. M. Göös, J. Suomela. Locally checkable proofs. *Proc. 30th ACM Symp. on Principles of Distributed Computing* (PODC), pp159-168, 2011.
18. G. Itkis, L. A. Levin. Fast and Lean Self-Stabilizing Asynchronous Protocols. In *35th IEEE Symp. on Foundations of Computer Science* (FOCS), pp226-239 (1994)
19. A. Korman, S. Kutten, and D. Peleg. Proof labeling schemes. *Distributed Computing* **22**(4): 215-233 (2010)
20. M. Naor and L. Stockmeyer. What can be computed locally? *SIAM J. Comput.* **24**(6): 1259–1277 (1995).
21. D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.