Compact Distributed Interactive Proofs for the Recognition of Cographs and Distance-Hereditary Graphs*

Pedro Montealegre[1], Diego Ramírez-Romero[2,3], and Ivan Rapaport[3]

[1] Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Chile
p.montealegre@uai.cl
[2] Departamento de Ingeniería Matemática, Universidad de Chile, Chile
dramirez@dim.uchile.cl
[3] DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Chile
rapaport@dim.uchile.cl

Abstract. We present compact distributed interactive proofs for the recognition of two important graph classes, well-studied in the context of centralized algorithms, namely *complement reducible graphs* and *distance-hereditary graphs*. Complement reducible graphs (also called *cographs*) are defined as the graphs not containing a four-node path P_4 as an induced subgraph. Distance-hereditary graphs are a super-class of cographs, defined as the graphs where the distance (shortest paths) between any pair of vertices is the same on every induced connected subgraph.

First, we show that there exists a distributed interactive proof for the recognition of cographs with two rounds of interaction. More precisely, we give a dAM protocol with a proof size of $\mathcal{O}(\log n)$ bits that recognizes cographs with high probability. Moreover, our protocol can be adapted to verify any Turing-decidable predicate restricted to cographs in dAM with certificates of size $\mathcal{O}(\log n)$.

Second, we give a three-round, dMAM interactive protocol for the recognition of distance-hereditary graphs, still with a proof size of $\mathcal{O}(\log n)$ bits.

Finally, we show that any one-round (denoted dM) or two-round, dMA protocol for the recognition of cographs or distance-hereditary graphs requires certificates of size $\Omega(\log n)$ bits. Moreover, we show that any constant-round dAM protocol using shared randomness requires certificates of size $\Omega(\log \log n)$.

Keywords: Distributed interactive proofs · Distributed recognition of graph clases · Cographs · Distance hereditary graph

* Partially supported by CONICYT via PIA/ Apoyo a Centros Científicos y Tecnológicos de Excelencia AFB 170001 (P.M. and I.R.), FONDECYT 1170021 (D.R. and I.R.) and FONDECYT 11190482 (P.M.) and PAI + Convocatoria Nacional Subvención a la Incorporación en la Academia Año 2017 + PAI77170068 (P.M.).

1 Introduction

The study of graph classes provides important insights to address basic graph problems such as coloring, maximum independent set, dominating set, etc. Indeed, as such problems are hard in general, restricting the input to a particular graph-class is a natural approach in order to exploit structural properties for designing efficient algorithms.

A well-known example is the class of perfect graphs [20], i.e., the class of graphs satisfying that the chromatic number equals the size of the largest clique of every induced subgraph. Many NP-complete problems on general graphs, such as coloring, maximum clique, maximum independent set, etc., can be solved in polynomial-time when the input is known to be a perfect graph [22].

The design of efficient algorithms for particular graph-classes has also interest in the context of distributed algorithms. Besides the classes of sparse and bounded degree graphs, there are many examples of efficient distributed algorithms specially designed to run on planar graphs [19], interval graphs [23], chordal graphs [32] and others. It is therefore very important to efficiently *check the membership* of a graph to a given class. Through this checking procedure we make sure that the execution is performed in the right type of input, in order to avoid erroneous computations or even the lack of termination.

Distributed Interactive Proofs. Distributed decision refers to the task in which the nodes of a connected graph G have to collectively decide (whether G satisfies) some graph property [38]. For performing any such task, the nodes exchange messages through the edges of G. The input of distributed decision problems may also include labels given to the nodes and/or to the edges of G. For instance, the nodes could decide whether G is properly colored, or decide whether the graph belongs to a given graph-class.

Acceptance and rejection are defined as follows. If G satisfies the property, then all nodes must accept; otherwise, at least one node must reject. This type of algorithm could be used in distributed fault-tolerant computing, where the nodes, with some regularity, must check whether the current network configuration is in a legal state for some Boolean predicate [33]. Then, if the configuration becomes illegal at some point, the rejecting node(s) raise the alarm or launch a recovery procedure.

Deciding whether a given coloring is proper can be done locally, by exchanging messages between neighbors. These types of properties are called *locally decidable*. Nevertheless, some other properties, such as deciding whether G is a simple path, are not. As a remedy, the notion of *proof-labeling scheme* (PLS) was introduced [33]. Similar variants were also introduced: non-deterministic local decisions [16], locally checkable proofs [21], and others.

Roughly speaking, in all these models, a powerful prover gives to every node v a certificate $c(v)$. This provides G with a global distributed proof. Then, every node v performs a local verification using its local information together with $c(v)$. PLSs can be seen as a distributed counterpart to the class NP, where, thanks to nondeterminism, the power of distributed algorithms increases.

Just as it happened in the centralized framework a natural step forward is to consider a model where the nodes are allowed to have *more than one interaction round* with the prover. Interestingly, there is no gain when interactions are all deterministic. When there is no randomness, the prover, from the very beginning, has all the information required to simulate the interaction with the nodes. Then, in just one round, the prover could simply send to each node the transcript of the whole communication, and the nodes simply verify that the transcript is indeed consistent. A completely different situation occurs when the nodes have access to some kind of randomness [2, 18]. In that case, the exact interaction with the nodes is unknown to the prover until the nodes communicate the realization of their random variables. Adding a randomized phase to the non-deterministic phase gives more power to the model [2, 18].

The notion of *distributed interactive protocols* was introduced by Kol, Oshman, and Saxena in [31] and further studied in [10, 17, 36, 37]. In such protocols, a centralized, untrustable prover with unlimited computation power, named Merlin, exchanges messages with a randomized distributed algorithm, named Arthur. Specifically, Arthur and Merlin perform a sequence of exchanges during which every node queries Merlin by sending a random bit-string, and Merlin replies to each node by sending a bit-string called proof. Neither the random strings nor the proofs need to be the same for each node. After a certain number of rounds, every node exchanges information with its neighbors in the network, and decides (i.e., it outputs accept or reject). For instance, a dMAM protocol involves three interactions: Merlin provides a certificate to Arthur, then Arthur queries Merlin by sending a random string. Finally, Merlin replies to the query by sending another certificate. Recall that this series of interactions is followed by a phase of distributed verification performed between every node and its neighbors.

When the number of interactions is k we refer to $\mathsf{dAM}[k]$ protocols (if the last player is Merlin) and $\mathsf{dMA}[k]$ protocols (otherwise). For instance, $\mathsf{dAM}[2] = \mathsf{dAM}$, $\mathsf{dMA}[3] = \mathsf{dAMA}$, etc. Also, the scenario of distributed verification, where there is no randomness and only Merlin interacts, corresponds dM. In other words, dM is the PLS model.

In distributed interactive proofs, Merlin tries to convince the nodes that G satisfies some property in a small number of rounds and through short messages. We say that an algorithm uses $\mathcal{O}(f(n))$ bits if the messages exchanged between the nodes (in the verification round) and also the messages exchanged between the nodes and the prover are upper bounded by $\mathcal{O}(f(n))$. We include this *bandwidth bound* in the notation, which becomes $\mathsf{dMA}[k, f(n)]$ and $\mathsf{dAM}[k, f(n)]$ for the corresponding protocols.

It is known that all Turing-decidable predicates on graphs admit a PLS with certificates of size $\mathcal{O}(n^2)$ bits [33]. Interestingly, some distributed problems are hard, even when a powerful prover provides the nodes with certificates. It is the case of SYMMETRY, the language of graphs having a non-trivial automorphism (i.e., a non-trivial one-to-one mapping from the set of nodes to itself preserving edges). Any PLS recognizing SYMMETRY requires certificates of size $\Omega(n^2)$ [21]. However, many problems requiring $\Omega(n^2)$-bit certificates in any PLS,

such as SYMMETRY, admit distributed interactive protocols with small certificates, and very few interactions. In fact, SYMMETRY is in both $\mathsf{dMAM}[\log n]$ and $\mathsf{dAM}[n \log n]$ [31].

Local Certification of Graph Classes. Regarding local certification of graph classes, there exist PLSs (with logarithmic-sized certificates) for the recognition of many graph classes such as acyclic graphs [33], planar graphs [15], graphs with bounded genus [14], etc. More recently, Busquet et al. [3] tackle the problem of locally certifying graphs classes defined by a finite set of minors.

Recently, Naor, Parter and Yogev defined in [37] a *compiler* which (1) turns any problem solved in NP in time $\tau(n)$ into a dMAM protocol using private randomness and bandwidth $\tau(n) \log n/n$ and; (2) turns any problem which can be solved in NC into a dAM protocol with private randomness, $\operatorname{poly}\log n$ rounds of interaction and bandwidth $\operatorname{poly}\log n$. This result has implications in the recognition of graph-classes. For example, it implies that any class of sparse graphs that can be recognized in linear time, can also be recognized by a dMAM protocol with logarithmic-sized certificates. This raises automatically the question of whether one can design, for the recognition of a given graph class, a distributed interactive proof based on fewer interactions than the interactions given by directly applying the compiler (while keeping the certificates as small as possible).

A graph-class is hereditary if the class is closed under vertex and edge deletion. Examples of hereditary graph classes include planar graphs, forests, bipartite graphs, perfect graphs, etc. Interestingly, all graph properties that are known to require large certificates (e.g. small diameter [6], non-3- colorability [21], having a non-trivial automorphism [21]), are non-hereditary.

Therefore, natural question is whether all hereditary graph-classes admit a distributed interactive proof with a constant number of interactions, and logarithmic-sized certificates. In this work we address the problem of the distributed recognition of two hereditary graph classes (which are in fact perfect graphs), namely *complement reducible graphs* and *distance-hereditary graphs*.

Cographs and Distance-Hereditary Graphs. The class of complement reducible graphs, or simply *cographs*, has several equivalent definitions, as it has been re-discovered in many different contexts [8, 28, 39, 40]. A graph is a cograph if it does not contain a four-node path P_4 as an induced subgraph. Equivalently, a graph is a cograph if it can be generated recursively from a single vertex by complementation and disjoint-union. A graph is a *distance-hereditary graph* if the distance between any two vertices is the same on every connected induced subgraph [25]. An equivalent definition is that every path between two vertices is a shortest path. It is known that every cograph is a distance-hereditary graph.

Many NP-complete problems are solvable in polynomial-time, or even linear time, when restricted to cographs and distance-hereditary graphs. For instance, maximum clique, maximum independent set, coloring (as distance-hereditary graphs are perfect [25]), hamiltonicity [27], Steiner tree and connected domination [13], computing the tree-width and minimum fill-in [5], among others. By a

result of Courcelle, Makowsky and Rotics [9], every decision problem expressible in a type of monadic second order logic can be solved in linear time on distance-hereditary graphs. Observe that all these results also apply to cographs. Other problems, like graph isomorphism, can be solved in linear time on cographs [8].

In the centralized setting, both cographs and distance-hereditary graphs can be recognized in linear time [12].

Respect to algorithms in the distributed setting, both recognition problems have also been addressed in the One-Round Broadcast Congested Clique Model (1BCC), also known as the Distributed Sketching Model [1]. In this model, the nodes of a graph send a single message to a *referee*, which initially has no information about the graph and, only using the received messages, has to decide a predicate of the input graph. 4 In [29] and [35], randomized protocols recognizing both classes of graphs in the 1BCC model are given. Interestingly, these protocols not only recognize the classes but *reconstruct them*, meaning that the referee learns all the edges of the input graph.

In this work, we focus on the recognition of cographs and distance-hereditary graphs in the model of distributed interactive proofs. We show that both classes can be recognized with compact certificates and constant (two or three) rounds of interaction.

Our Results. We show that the recognition of cographs is in $\mathsf{dAM}[\log n]$. Our result consists of adapting an algorithm given in [29, 35], originally designed for the 1BCC model. In this regard, we exploit the natural high connectivity of this class, combined with the use of non-determinism in order to route all messages in the network to a leader node, which is delegated to act as a *referee*. In fact, our protocol allows this leader to learn all the edges of the input graph. We use this fact to show that any Turing-decidable predicate restricted to cographs is decidable in $\mathsf{dAM}[\log n]$.

Interestingly, our results imply that any one-round deterministic protocol in the 1BCC model recognizing cographs, would immediately imply a dM (i.e. a PLS) protocol for the recognition of cographs. Unfortunately, up to our knowledge, it is not known whether recognizing cographs can be done through a deterministic 1BCC protocol.

Then, we adapt the protocol for the recognition of cographs and we combine it with a set of tools related to the structure of distance-hereditary graphs in order to show that the recognition of this class is in $\mathsf{dMAM}[\log n]$. In this case, we are not able to simulate the 1BCC protocol by gathering all the information in a single node representing the referee. Instead, we find a way to verify each step of the computation of the referee in a distributed manner, by choosing nodes that can receive (with the help of the prover) all necessary messages for performing the task.

We remark that our protocols beat the performance of the compiler of Naor, Parter and Yogev. In fact, both graph classes can have $\Theta(n^2)$ edges and, therefore, the use of the compiler shows that the recognition of these classes is in $\mathsf{dMAM}[n \log n]$ and in $\mathsf{dAM}[\operatorname{poly}\log n, \operatorname{poly}\log n]$ (note that cographs and

distance-hereditary graphs can be recognized in NC [11] and in linear time in the centralized setting [12], see the Related Work section).

Finally, we give some lower-bounds. More precisely, we show that any dM or dMA protocol for the recognition of cographs or distance-hereditary graphs, requires messages of size at least $\Omega(\log n)$. Our results are obtained extending a lower-bound technique described in [21], for the detection of a single leader in the context of locally checkable proofs. We note that our protocols use shared randomness. In that sense we prove that, any dAM protocol using shared randomness for the previous problems, requires messages of size at least $\Omega(\log \log n)$.

Related Work. The recognition of cographs and distance-hereditary graphs has been studied thoroughly in the parallel setting, where both problems have been shown to be in NC [11, 24, 30]. The currently best algorithms for the recognition of both classes run in time $\mathcal{O}(\log^2 n)$ and using a linear number of processors in a CREW-PRAM [11]. There also exist fast-parallel algorithms for NP-hard problems restricted to cographs and distance-hereditary graphs [26, 34].

Unfortunately, there are no much research regarding distributed algorithms specially designed for cographs and distance-hereditary. Nevertheless, the structural properties of distance-hereditary graphs have been used in the design of compact routing tables for interconnection networks [7].

Structure of the Article. Section 2 is the preliminary section, where we give some graph-theoretic background, including the formal definitions of cographs and distance-hereditary graphs. We also give the precise definition of distributed interactive proofs. In Section 3 we give the results regarding cographs, and in Section 4 we give the results regarding distance-hereditary graphs. Finally, in Section 5, we provide some lower-bounds. Due the lack of space, the results are only outlined in their corresponding sections, while the full proofs are detailed in the appendix

2 Preliminaries

Background on Cographs and Distance-Hereditary Graphs. All the graphs in this paper are simple and undirected. Let $G = (V, E)$ be a graph. For a set $U \subseteq V$, we define $G[U]$, the *induced subgraph* of $G = (V, E)$ according to U, as the graph $H = (U, E(U))$, where $E(U) = E \cap \binom{U}{2}$. We denote $H \subseteq G$ when H is s an induced subgraph of G. If, instead, we have a graph with vertex set U such that its edges are only contained in $E(U)$, we simply call it a *subgraph* of G. A *spanning subgraph* of G is a subgraph H with $V(H) = V(G)$. Given two nodes u, v of a connected graph H, the distance between them, denoted by $d_H(u, v)$, is defined as the length of the shortest path between u and v in H. A P_4 is an induced path of length four.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we define the *union* between both graphs, denoted by $G_1 \cup G_2$ as the graph $\tilde{G} = (\hat{V}, \tilde{E})$, with $\tilde{V} = V_1 \cup V_2$ and $\tilde{E} = E_1 \cup E_2$. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we

define the *join* between both graphs, denoted by $G_1 * G_2$ as the graph $\hat{G} = (\hat{V}, \hat{E})$, with $\hat{V} = V_1 \cup V_2$ and $\hat{E} = E_1 \cup E_2 \cup \{v_1 v_2 \text{ such that } v_1 \in V_1, v_2 \in V_2\}$.

The set of neighbors of a node u is denoted $N(u)$, and the closed neighborhood $N[u]$ is the set $N(u) \cup \{u\}$. A node v is said to be a *pending node* if it has a unique neighbor in the graph. A pair of nodes $u, v \in V$ are said to be *twins* if their neighborhoods are equal. That is, $N(u) = N(v)$ or $N[u] = N[v]$. In the case that u and v are adjacent ($N[u] = N[v]$) we refer to them as *true twins* and, otherwise, we refer to them as *false twins*.

As we mentioned in the introduction, a cograph is a graph that does not contain a P_4 as an induced subgraph (i.e. it is P_4-*free*). Another equivalent definition states that cographs are the graphs which can be obtained recursively following three rules: (1) A single vertex is a cograph, (2) the disjoint union between two cographs is a cograph and (3) the join of two cographs is a cograph. An advantage of cographs is that they admit other characterizations that may be useful for local verification. First, we define a *twin ordering* as an ordering $(v_i)_{i=1}^n$ of the nodes of V such that, for each $j \geq 2$, v_j has a twin in $G[\{v_1, \dots v_j\}]$.

Proposition 1 ([29]). *Given a graph G the following are equivalent:*

1. *G is a cograph.*
2. *Each non trivial induced subgraph of G has a pair of* twins.
3. *G is P_4-*free.
4. *G admits a* twin ordering.

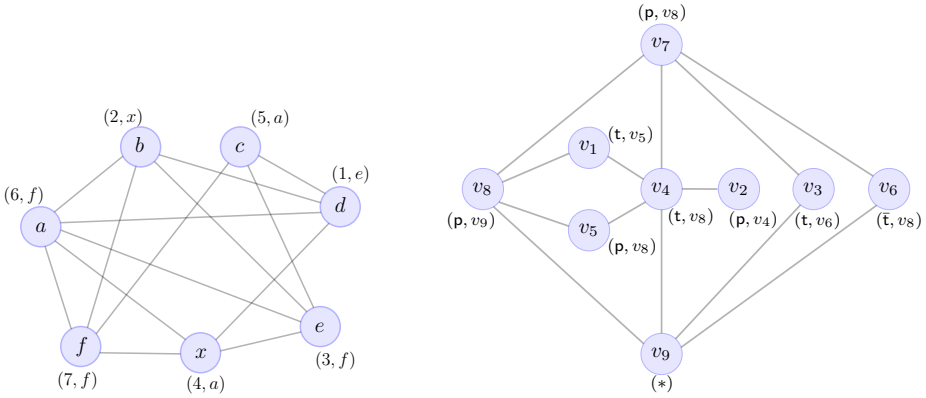

Fig. 1: Left: A cograph with labels according to a twin ordering. The first entry represents the step at which they are removed, while the second entry indicates the node's twin at such step. Right: A distance-hereditary graph with labels according to its ordering. The first entry indicates whether it is removed as a *true twin* ($\bar{\mathsf{t}}$), as a*false twin* (t) or as a pending node (p).

A graph G is said to be distance-hereditary if for any induced subgraph $H \subseteq G$ and any pair $u, v \in H$ satisfy that $d_H(u, v) = d_G(u, v)$. That is, any induced

path between a pair of nodes is a shortest path. A relevant characterization for this class is the following.

Proposition 2 ([4]). *An n-node graph G is distance hereditary iff there exists an ordering $(v_i)_{i=1}^n$ such that, for any $i \in [n]$, either there exists $j < i$ such that v_i and v_j are* twins *in $G_i = G[\{v_1, ..v_i]\}$ or v_i is a pending node at G_i.*

Model Definitions. Let G be a simple connected n-node graph, let $I : V(G) \to \{0,1\}^*$ be an input function assigning labels to the nodes of G, where the size of all inputs is polynomially bounded on n. Let $\mathsf{id} : V(G) \to \{1, \ldots, \text{poly}(n)\}$ be a one-to-one function assigning identifiers to the nodes. A *distributed language* $\mathcal{L}$ is a (Turing-decidable) collection of triples (G, id, I), called *network configurations*.

A distributed interactive protocol consists of a constant series of interactions between a *prover* called Merlin, and a *verifier* called Arthur. The prover Merlin is centralized, has unlimited computing power and knows the complete configuration (G, id, I). However, he cannot be trusted. On the other hand, the verifier Arthur is distributed, represented by the nodes in G, and has limited knowledge. In fact, at each node v, Arthur is initially aware only of his identity $\mathsf{id}(v)$, and his label $I(v)$. He does not know the exact value of n, but he knows that there exists a constant c such that $\mathsf{id}(v) \leq n^c$. Therefore, for instance, if one node v wants to communicate $\mathsf{id}(v)$ to its neighbors, then the message is of size $\mathcal{O}(\log n)$.

Given any network configuration (G, id, I), the nodes of G must collectively decide whether (G, id, I) belongs to some distributed language $\mathcal{L}$. If this is indeed the case, then all nodes must accept; otherwise, at least one node must reject (with certain probabilities, depending on the precise specifications we are considering).

There are two types of interactive protocols: Arthur-Merlin and Merlin-Arthur. Both types of protocols have two phases: an interactive phase and a verification phase. Let us define first *Arthur-Merlin interactive protocols*. If Arthur is the party that starts the interactive phase, he picks a random string $r_1(v)$ at each node v of G (this string could be either private or shared) and send them to Merlin. Merlin receives r_1, the collection of these n strings, and provides every node v with a certificate $c_1(v)$ that is a function of v, r_1 and (G, id, I). Then again Arthur picks a random string $r_2(v)$ at each node v of G and sends r_2 to Merlin, who, in his turn, provides every node v with a certificate $c_2(v)$ that is a function of v, r_1, r_2 and (G, id, I). This process continues for a fixed number of rounds. If Merlin is the party that starts the interactive phase, then he provides at the beginning every node v with a certificate $c_0(v)$ that is a function of v and (G, id, I), and the interactive process continues as explained before. In Arthur-Merlin protocols, the process ends with Merlin. More precisely, in the last, k-th round, Merlin provides every node v with a certificate $c_{\lceil k/2 \rceil}(v)$. Then, the verification phase begins. This phase is a one-round deterministic algorithm executed at each node. More precisely, every node v broadcasts a message M_v to its neighbors. This message may depend on $\mathsf{id}(v)$, $I(v)$, all random strings generated by Arthur at v, and all certificates received by v from Merlin. Finally, based on all the knowledge accumulated by v (i.e., its identity, its input label, the

generated random strings, the certificates received from Merlin, and all the messages received from its neighbors), the protocol either accepts or rejects at node v. Note that Merlin knows the messages each node broadcasts to its neighbors because there is no randomness in this last verification round.

A *Merlin-Arthur interactive protocols* of k interactions is an Arthur-Merlin protocol with $k-1$ interactions, but where the verification round is randomized. More precisely, Arthur is in charge of the k-th interaction, which includes the verification algorithm. The protocol ends when Arthur picks a random string $r(v)$ at every node v and uses it to perform a (randomized) verification algorithm. In other words, each node v randomly chooses a message M_v from a distribution specified by the protocol, and broadcast M_v to its neighbors. Finally, as explained before, the protocol either accepts or rejects at node v. Note that, in this case, Merlin does not know the messages each node broadcasts to its neighbors (because they are randomly generated). If $k = 1$, a distributed Merlin-Arthur protocol is a (1-round) randomized decision algorithm; if $k = 2$, it can be viewed as the non-deterministic version of randomized decision, etc.

Definition 1. *Let $\mathcal{V}$ be a verifier and $\mathcal{M}$ a prover of a distributed interactive proof protocol for languages over graphs of n nodes. If $(\mathcal{V}, \mathcal{M})$ corresponds to an Arthur-Merlin (resp. Merlin Arthur) k-round, $\mathcal{O}(f(n))$ bandwidth protocol, we write $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ (resp. $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$).*

Definition 2. *Let $\varepsilon \leq 1/3$. The class $\mathsf{dAM}_{\varepsilon}[k, f(n)]$ (resp. $\mathsf{dMA}_{\varepsilon}[k, f(n)]$) is the class of languages $\mathcal{L}$ over graphs of n nodes for which there exists a verifier $\mathcal{V}$ such that, for every configuration (G, id, I) of size n, the two following conditions are satisfied.*

Completeness. *If $(G, \mathsf{id}, I) \in \mathcal{L}$ then, there exists a prover $\mathcal{M}$ such that $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ (resp. $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$) and*

$$\mathbf{Pr}\Big[\mathcal{V} \text{ accepts } (G, \mathsf{id}, I) \text{ in every node given } \mathcal{M}\Big] \geq 1 - \varepsilon.$$

Soundness. *If $(G, \mathsf{id}, I) \notin \mathcal{L}$ then, for every prover $\mathcal{M}$ such that $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ (resp. $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$),*

$$\mathbf{Pr}\Big[\mathcal{V} \text{ rejects } (G, \mathsf{id}, I) \text{ in at least one nodes given } \mathcal{M}\Big] \geq 1 - \varepsilon.$$

We also denote $\mathsf{dAM}[k, f(n)] = \mathsf{dAM}_{1/3}[k, f(n)]$ and $\mathsf{dMA} = \mathsf{dMA}_{1/3}[k, f(n)]$, and omit the subindex ε when its value is obvious from the context.

In this paper, we are interested mainly in two languages, that we call COGRAPH and DIST-HEREDITARY which are the languages of graphs that are cographs and distance-hereditary graphs, respectively. Formally,

- COGRAPH $= \{\langle G, \mathsf{id}\rangle$ s.t. G is a cograph$\}$.
- DIST-HEREDITARY $= \{\langle G, \mathsf{id}\rangle$ s.t. G is distance-hereditary$\}$.

Also, for a distributed language $\mathcal{L}$, the *restriction of $\mathcal{L}$ to cographs*, denoted $\mathcal{L}_{\text{COGRAPH}}$ is the subset of network configurations $(G, \mathsf{id}, I) \in \mathcal{L}$ such that G is a cograph.

3 Cographs

We first show a way to distribute the proofs received by the network in such a way that we can centralize the verification process.

Lemma 1. *Given a cograph G, it is possible to construct a spanning tree T of* depth *two, such that each node at depth one, has at most one child.*

By the previous lemma, we know that for any two round protocol $\mathcal{P}$ over a cograph G with cost $\Omega(\log n)$ bits, we may assume, without loss of generality, that there is a root ρ with access to all coins and messages received by the whole network. We simply construct the spanning tree given by Lemma 1, by choosing the root ρ as follows. First, a bipartite graph can be easily verified with two colors, and ρ can be chosen to be the node in G_2 with the smallest identifier. Then, it suffices to assign to each node u of depth one in the spanning tree, both its proof and the proof received by its child w, along with the random coin it drew. Then, the nodes can locally verify the consistency of this message and the root will have received all the messages in the network.

Lemma 2. *Given any* dM *(resp.* dAM*) protocol with bandwidth L that runs over a cograph, we can construct a* dM *(resp.* dAM*) protocol with bandwidth cost $L + \mathcal{O}(\log n)$ and where there exists a node ρ which has access to all messages (resp. all messages and coins) in the network.*

An advantage of this procedure is that we may simulate any protocol in the (non-deterministic) One-Round Broadcast Congested Clique model (by using the root ρ as referee) by either using one round of interaction (if the simulated protocol is deterministic) or two rounds (when the simulated protocol is randomized). From here it follows that we can use the protocol by [29] to recognize cographs, therefore constructing a protocol for cograph detection in two rounds of interaction and $\mathcal{O}(\log n)$ bits. That is, COGRAPH $\in$ dAM$[\log n]$. For the sake of completeness, we now describe the protocol of [29].

Definition 3. *Given a cograph $G = (V, E)$, we can define its* canonical order *as follows. We start by choosing the smallest pair of* twins *(those with the smallest identifiers in lexicographic order) which we know to exist by Proposition 1. From there we choose and remove the smallest node from this pairing. Then, we repeat this process by finding another pair and removing one of its members until we end up with a single node.*

Let p be a prime and $\phi = (\phi_w)_{w \in V}$ be a family of linearly independent polynomials in $\mathbb{Z}_p[x]$. Given $w \in V$ we define, $q_w = \sum_{w' \in N(w)} \phi_{w'}$ and $\bar{q}_w = q_w + \phi_w$. We also define the *derived polynomials* of ϕ as the collection

$$\alpha_{u,v} = \phi_u - \phi_v \qquad \beta_{u,v} = q_u - q_v, \qquad \gamma_{u,v} = \bar{q}_u - \bar{q}_v, \quad u.v \in V$$

Now, given a pair of *twins* u and v, we assign to $G - v$ the polynomials $\{\phi'_w\}_{w \in V - v}$ defined as

$$\phi'_w = \begin{cases} \phi_w & \text{if } w \neq u \\ \phi_u + \phi_v & \text{if } w = u \end{cases}$$

With this construction, from $\phi_u(x) = x^{\mathsf{id}(u)}$ it is possible to construct a sequence of polynomials ϕ_u^i for $i \in [n]$ according to the *canonical order* $\{v_i\}_{i=1}^n$ and u in the graph $G - \{v_j\}_{j=i+1}^n$. We call these functions the *basic* polynomials of G. And so the *canonical family* of polynomials of G is defined as the union between its basic and derived polynomials. It follows that this family of functions has at most $3n^3$ elements.

Definition 4. *Let G be a cograph. We say that a vector $m = ((a_w, b_w))_{w \in V} \in (\mathbb{Z}_p)^{2n}$ is* valid for G in $t \in \mathbb{Z}_p$ *if there exists a family of linearly independent polynomials $(\phi_w)_{w \in V}$ in $\mathbb{Z}_p[X]$ such that $a_w = \phi_w(t)$ and $b_w = q_w(t)$ for each $w \in V$.*

Lemma 3. *Let $m = ((a_w, b_w))_{w \in V} \in (\mathbb{Z}_p)^{2n}$ be a valid vector for G in t. Consider u, v to be a pair of twins in G such that $a_u \neq a_v$. Then, the vector $m' = ((a'_w, b'_w))_{w \in V - v} \in (\mathbb{Z}_p)^{2n-2}$ is valid for $G - v$ in t, where its coordinates are given by*

$$(a'_w,\, b'_w) = \begin{cases} (a_w,\, b_w) & \text{if } w \in V - \{u, v\} \\ (a_u + a_v,\, b_u - a_v \delta_{uv}) & \text{if } w = u \end{cases}$$

with δ_{uv} equals one if and only if $a_u + b_u = a_v + b_v$

With this lemma now we can proceed to describe the protocol.

Theorem 1. *There is a distributed interactive proof with two rounds for the recognition of cographs, i.e.,* COGRAPH $\in \mathsf{dAM}[\log n]$. *Moreover, the protocol uses shared randomness and gives the correct answer with high probability.*

Proof. Let $G = (V, E)$ be an n-node graph. Without loss of generality we may assume the graph has identifiers in $[n]$ as, following Lemma 1, it is possible to implement a PERMUTATION protocol in a single round: Merlin sends to each node v an identifier $\overline{\mathsf{id}} : V \to [n]$ and the root, by receiving all proofs, can see that they all received distinct identifiers which are consistent with their original ones.

Let p be a prime such that $3n^{c+4} \leq p \leq 6n^{c+4}$. The protocol is the following: All nodes collectively generate a seed $t \in \mathbb{F}_p$ uniformly at random. Then Merlin sends to each node w a message m_w such that $m = (m_w)_{w \in V}$ is a valid vector for G at t. Each node then computes such message by defining $\phi_w(x) = x^{\overline{\mathsf{id}}(w)}$.

After the nodes exchange messages, following Lemma 1 we obtain that the root ρ owns a vector $m \in \mathbb{F}_p^{2n}$. From here, the root repeats the following procedure at most $n-1$ times trying to construct a canonical ordering $\{v_i\}_{i=1}^n$ for G. At step i, it starts at graph G^i and a vector $m_i \in \mathbb{F}_p^{2(n-i+1)}$ (where $G^1 = G$ and $m^1 = m$) and looks for a pair of nodes u, v in G^i such that $a_u^i \neq a_v^i$ and either $b_u^i = b_v^i$ or $a_u^i + b_u^i = a_v^i + b_b^i$. Then it chooses, among all pairs it has found, the first in lexicographic order. If no such pair exists, then he rejects. On the contrary, he defines $G^{i+1} = G^i - v$, and setting $v_{n-i+1} = v$ (without loss of generality we assume that $\mathsf{id}(v) < \mathsf{id}(u)$). Then the root computes m^{i+1} from the previous vector m^i following Lemma 3. If the root reaches step $n-1$ then it accepts.

Completeness and **Soundness.** It follows then that as the messages depend on the original identifiers and the root ρ has access to all messages, then both acceptance errors depend solely in the 1BCC construction. Now, by Lemma 3 it follows that the only point at which the protocol might fail is if the chosen t turns out to be a root for any of the polynomials in the canonical family from Definition 3. As there are at most $3n^3$ such polynomials, each of degree at most n , we have that the acceptance error is at most $3n^4/3n^{c+4} = 1/n^c$ and the theorem follows. □

As we mentioned in the Introduction, the result obtained in [29] is much stronger than just recognizing cographs. In fact, the referee not only can recognize a cograph but actually can *reconstruct* it. In other words, when the input graph is a cograph, after the communication round, the referee learns all the edges. In our context, this implies that the root ρ not only recognizes cographs, but also can recognize any distributed language restricted to them.

Theorem 2. *For every distributed language $\mathcal{L}$, there is a distributed interactive proof with two rounds for its restriction to cographs, i.e. $\mathcal{L}_{\text{COGRAPH}} \in \mathsf{dAM}[\log n]$. Moreover, the protocol uses shared randomness and gives the correct answer with high probability.*

Proof. It is sufficient to notice that the tree-root ρ in the construction from Lemma 1 has access to all proofs in the network. In particular, the id's and positions for each node in the twin-ordering π. As such, ρ has knowledge of the entire topology of the network and its inputs (provided that these are of size $\mathcal{O}(\log n)$) and can compute any property related to them, with the acceptance error matching that of the verification procedure in Theorem 1. As for the rest of the nodes, they simply accept and delegate this decision to the root. □

4 Distance-Hereditary Graphs

Following the protocol described for cographs, it is possible to derive an interactive protocol for distance-hereditary graphs, which admit a similar construction. Indeed, as described before, any distance-hereditary graph can be constructed by sequentially adding twins or pending nodes. Notice that for the protocol in Theorem 1, the verification process is done by the root as it prunes the graph in $n-1$ steps. This leads to an order by which the nodes were selected, and we call it *canonical ordering*. While we cannot delegate the verification routine to a single node (as distance-hereditary graphs can have arbitrarily large diameter), we can distribute the verification process by letting different nodes check different steps of the computation. As the rule described in Lemma 3 for pruning the graph involves only the pair of twins at each step, we only need to find nodes that, for a fixed node v, can receive all the proofs sent by v, its twins and its pending nodes.

In order to prune the graph in this new setting, we need a rule for pruning pending nodes from a graph and updating the vectors of each node accordingly. Here, we use the definition of a *valid vector* as described in Section 3.

Lemma 4. *Let $m = ((a_w, b_w))_{w \in V} \in (\mathbb{Z}_p)^{2n}$ be a valid vector for G at some point t. If $u \in G$ has v as a pending node adjacent to it, then, the vector $m' = ((a'_w, b'_w))_{w \in V-v} \in (\mathbb{Z}_p)^{2n-2}$ is valid for $G - v$ in t, where the coordinates of m' are given by*

$$(a'_w,\, b'_w) = \begin{cases} (a_w,\, b_w) & \text{if } w \in V - \{u, v\} \\ (a_w,\, b_w - a_v) & \text{if } w = u \end{cases}$$

In order to distribute the verification procedure, for any fixed v we wish to set a node to compute the correctness of the vectors of all nodes assigned as twins of v. Indeed, for a fixed ordering π for pruning the graph and a node v with $\pi_v < n$, consider the *predecessor* of v, denoted by $\mathsf{ant}(v)$, to be v's neighbor whose value for $\pi(\cdot)$ is immediately after that of v among its neighbors. As all previous nodes in the order which are twins of v have the same neighborhood, it follows that all these nodes must be adjacent to $\mathsf{ant}(v)$. In case that no such a node exists, by assuming that G is connected, it follows that the last node according to π which is assigned as a twin of v must be a *true* twin and, therefore, be adjacent to him. And the same reasoning holds.

Thus, the main strategy of our protocol is that, given an initial vector (a_v, b_v) for a node v in the graph, each node $\mathsf{ant}(v)$ has the task of updating this vector until it obtains the vector v that the referee should have at the time the node is pruned from the graph, which we denote by (a^π_v, b^π_v). Then, each node u which is a twin of v provides its vector (a^π_u, b^π_u) (which is proved to be correct by some other node) and so the predecessor of v compares and updates v's vector according to the rules from Lemmas 3 and 4.

Theorem 3. *There is a distributed interactive proof with three rounds of interaction for the recognition of distance-hereditary graphs, i.e.,* DIST-HEREDITARY $\in$ $\mathsf{dMAM}[\log n]$. *Moreover, the protocol uses shared randomness and gives the correct answer with high probability.*

5 Lower Bounds

In this section, we provide lower-bounds on the certificate size of distributed interactive proofs for COGRAPH or DIST-HEREDITARY. Due the lack of space, the proof of the result on this section are detailed in the appendix. The following result is based on a construction by [21].

Theorem 4. *If* COGRAPH *or* DIST-HEREDITARY *belongs to the class* $\mathsf{dM}[f(n)]$, *then* $f(n) = \Omega(\log n)$. *Moreover, for any fixed k, if* COGRAPH *or* DIST-HEREDITARY *belongs to* $\mathsf{dAM}^{\text{pub}}[k, g(n)]$, *then* $g(n) = \Omega(\log \log n)$.

Following an approach introduced by Fraigniaud et al. [17], we obtain that the graph constructions used in the proof of Theorem 4 can be adapted in order to obtain lower-bounds for the models dMA.

Corollary 1. *If any of the problems* COGRAPH *or* DIST-HEREDITARY *belongs to* $\mathsf{dMA}_{1/7}[f(n)]$, *then* $f(n) = \Omega(\log n)$.

References

1. Assadi, S., Kol, G., Oshman, R.: Lower bounds for distributed sketching of maximal matchings and maximal independent sets. In: Emek, Y., Cachin, C. (eds.) PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020. pp. 79–88. ACM (2020)
2. Baruch, M., Fraigniaud, P., Patt-Shamir, B.: Randomized proof-labeling schemes. In: Symposium on Principles of Distributed Computing. pp. 315–324 (2015)
3. Bousquet, N., Feuilloley, L., Pierron, T.: Local certification of graph decompositions and applications to minor-free classes. arXiv preprint arXiv:2108.00059 (2021)
4. Brandstadt, A., Spinrad, J.P., et al.: Graph classes: a survey, vol. 3. Siam (1999)
5. Broersma, H., Dahlhaus, E., Kloks, T.: A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. Discrete Applied Mathematics **99**(1-3), 367–400 (Feb 2000)
6. Censor-Hillel, K., Paz, A., Perry, M.: Approximate proof-labeling schemes. Theoretical Computer Science **811**, 112–124 (2020)
7. Cicerone, S., Di Stefano, G., Flammini, M.: Compact-port routing models and applications to distance-hereditary graphs. Journal of Parallel and Distributed Computing **61**(10), 1472–1488 (2001)
8. Corneil, D., Lerchs, H., Burlingham, L.: Complement reducible graphs. Discrete Applied Mathematics **3**(3), 163–174 (Jul 1981)
9. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. Theory of Computing Systems **33**(2), 125–150 (Mar 2000)
10. Crescenzi, P., Fraigniaud, P., Paz, A.: Trade-offs in distributed interactive proofs. In: 33rd International Symposium on Distributed Computing (DISC 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
11. Dahlhaus, E.: Efficient parallel recognition algorithms of cographs and distance hereditary graphs. Discrete applied mathematics **57**(1), 29–44 (1995)
12. Damiand, G., Habib, M., Paul, C.: A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. Theoretical Computer Science **263**(1-2), 99–111 (Jul 2001)
13. D'Atri, A., Moscarini, M.: Distance-hereditary graphs, steiner trees, and connected domination. SIAM Journal on Computing **17**(3), 521–538 (Jun 1988)
14. Feuilloley, L., Fraigniaud, P., Montealegre, P., Rapaport, I., Rémila, É., Todinca, I.: Local certification of graphs with bounded genus. arXiv preprint arXiv:2007.08084 (2020)
15. Feuilloley, L., Fraigniaud, P., Montealegre, P., Rapaport, I., Rémila, É., Todinca, I.: Compact distributed certification of planar graphs. Algorithmica pp. 1–30 (2021)
16. Fraigniaud, P., Korman, A., Peleg, D.: Towards a complexity theory for local distributed computing. Journal of the ACM (JACM) **60**(5), 1–26 (2013)
17. Fraigniaud, P., Montealegre, P., Oshman, R., Rapaport, I., Todinca, I.: On Distributed Merlin-Arthur Decision Protocols. In: International Colloquium on Structural Information and Communication Complexity. pp. 230–245. Springer (2019)
18. Fraigniaud, P., Patt-Shamir, B., Perry, M.: Randomized proof-labeling schemes. Distributed Computing **32**(3), 217–234 (2019)
19. Ghaffari, M., Haeupler, B.: Distributed algorithms for planar networks i: Planar embedding. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing. pp. 29–38 (2016)

20. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57). North-Holland Publishing Co., NLD (2004)
21. Göös, M., Suomela, J.: Locally checkable proofs in distributed computing. Theory of Computing **12**(1), 1–33 (2016)
22. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer Berlin Heidelberg (1993)
23. Halldórsson, M.M., Konrad, C.: Distributed algorithms for coloring interval graphs. In: International Symposium on Distributed Computing. pp. 454–468. Springer (2014)
24. He, X.: Parallel algorithm for cograph recognition with applications. Journal of Algorithms **15**(2), 284–313 (1993)
25. Howorka, E.: A characterization of distance-hereditary graphs. The Quarterly Journal of Mathematics **28**(4), 417–420 (1977)
26. Hsieh, S.Y., Ho, C.W., Hsu, T.S., Ko, M.T., Chen, G.H.: Efficient parallel algorithms on Distance hereditary graphs. Parallel Processing Letters **09**(01), 43–52 (Mar 1999)
27. Hung, R.W., Chang, M.S.: Linear-time algorithms for the hamiltonian problems on distance-hereditary graphs. Theoretical Computer Science **341**(1-3), 411–440 (Sep 2005)
28. Jung, H.: On a class of posets and the corresponding comparability graphs. Journal of Combinatorial Theory, Series B **24**(2), 125–133 (Apr 1978)
29. Kari, J., Matamala, M., Rapaport, I., Salo, V.: Solving the induced subgraph problem in the randomized multiparty simultaneous messages model. In: International Colloquium on Structural Information and Communication Complexity. pp. 370–384. Springer (2015)
30. Kirkpatrick, D.G., Przytycka, T.: Parallel recognition of complement reducible graphs and cotree construction. Discrete applied mathematics **29**(1), 79–96 (1990)
31. Kol, G., Oshman, R., Saxena, R.R.: Interactive distributed proofs. In: ACM Symposium on Principles of Distributed Computing. pp. 255–264. ACM (2018)
32. Konrad, C., Zamaraev, V.: Brief announcement: Distributed minimum vertex coloring and maximum independent set in chordal graphs. In: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing. pp. 159–161 (2018)
33. Korman, A., Kutten, S., Peleg, D.: Proof labeling schemes. Distributed Computing **22**(4), 215–233 (2010)
34. Lin, R., Olariu, S.: Fast parallel algorithms for cographs. In: Lecture Notes in Computer Science, pp. 176–189. Springer Berlin Heidelberg (1990)
35. Montealegre, P., Perez-Salazar, S., Rapaport, I., Todinca, I.: Graph reconstruction in the congested clique. Journal of Computer and System Sciences (2020)
36. Montealegre, P., Ramírez-Romero, D., Rapaport, I.: Shared vs private randomness in distributed interactive proofs. LIPIcs, vol. 181, pp. 51:1–51:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
37. Naor, M., Parter, M., Yogev, E.: The power of distributed verifiers in interactive proofs. In: ACM-SIAM Symposium on Discrete Algorithms. pp. 1096–115. SIAM (2020)
38. Naor, M., Stockmeyer, L.: What can be computed locally? SIAM Journal on Computing **24**(6), 1259–1277 (1995)
39. Seinsche, D.: On a property of the class of n-colorable graphs. Journal of Combinatorial Theory, Series B **16**(2), 191–193 (Apr 1974)
40. Sumner, D.P.: Dacey graphs. Journal of the Australian Mathematical Society **18**(4), 492–502 (Dec 1974)