Distributed Interactive Proofs for the Recognition of Some Geometric Intersection Graph Classes*

Benjamin Jauregui[1], Pedro Montealegre[2], and Ivan Rapaport[3]

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

Abstract. A graph $G = (V, E)$ is a geometric intersection graph if every node $v \in V$ is identified with a geometric object of some particular type, and two nodes are adjacent if the corresponding objects intersect. Geometric intersection graph classes have been studied from both the theoretical and practical point of view. On the one hand, many hard problems can be efficiently solved or approximated when the input graph is restricted to a geometric intersection class of graphs. On the other hand, these graphs appear naturally in many applications such as sensor networks, scheduling problems, and others. Recently, in the context of distributed certification and distributed interactive proofs, the recognition of graph classes has started to be intensively studied. Different results related to the recognition of trees, bipartite graphs, bounded diameter graphs, triangle-free graphs, planar graphs, bounded genus graphs, H-minor free graphs, etc., have been obtained.
The goal of the present work is to design efficient distributed protocols for the recognition of relevant geometric intersection graph classes, namely permutation graphs, trapezoid graphs, circle graphs and polygon-circle graphs. More precisely, for the two first classes we give proof labeling schemes recognizing them with logarithmic-sized certificates. For the other two classes, we give three-round distributed interactive protocols that use messages and certificates of size $\mathcal{O}(\log n)$. Finally, we provide logarithmic lower-bounds on the size of the certificates on the proof labeling schemes for the recognition of any of the aforementioned geometric intersection graph classes.

Keywords: Distributed decision; Proof-labeling scheme; Distributed interactive proofs; Intersection Graph Classes.

1 Introduction

This paper deals with the problem of designing compact distributed certificates and compact distributed interactive proofs for deciding graph properties. In these protocols, the nodes of a connected graph G have to decide, collectively, whether G itself belongs to a particular graph class. As in the centralized case, also in the distributed setting there exists a number of algorithms specially designed to decide whether G belongs to a particular graph class. The specific goal of this work is to decide, through proof-labeling schemes and the more general model of distributed interactive proofs, whether G belongs to relevant intersection graph classes. These classes have applications in topics like biology, ecology, computing, matrix analysis, circuit design, statistics, archaeology, etc. For a nice survey we refer to [35].

1.1 Proof-Labeling Schemes and Distributed Interactive Proofs

In locally decidable algorithms every node is just allowed to send messages to its neighbors, in one round (a less restrictive, but similar scenario, is where the number of rounds is constant, independent of the size of G, see [38]). Some very basic properties can be decided locally (with a local algorithm). For instance, deciding whether the graph G has bounded degree. More generally, if we do not impose bandwidth restrictions, then detecting the existence of any local structure (such as a triangle) can be solved through local algorithms.

In the aforementioned examples, acceptance and rejection are (implicitly) defined as follows. If G satisfies the property, then all nodes must accept; otherwise, at least one node must reject.

* This work was supported by Centro de Modelamiento Matemático (CMM), ACE210010 and FB210005, BASAL funds for centers of excellence from ANID-Chile, FONDECYT 11190482 and FONDECYT 1170021.

These very fast local algorithms 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 [29]. Then, if the configuration becomes at some point illegal, the rejecting node(s) raise the alarm or launch a recovery procedure. When there are distributed algorithms designed for particular graph classes, the use of an initial recognizing protocol could avoid the risk of running a distributed protocol for graphs that do not belong to the class for which the protocol was designed.

Of course, many simple properties cannot be decided with one round (or with a constant number of rounds) through local algorithms. In order to overcome this issue, the notion of proof-labeling scheme (PLS) was introduced [29]. PLSs can be seen as a distributed counterpart of the nondeterministic class NP. In fact, in a PLS, 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 $c(v)$.

Incorporating a powerful prover to the model is not just motivated by a purely theoretical interest. In fact, with the rise of the Internet, prover-assisted computing models are ubiquitous. Asymmetric applications –social networks, cloud computing, etc.– where a very powerful central entity stores and process large amounts of data, are already part of our everyday lives. A key issue, which is a central part of the PLS model, is that the devices of the network cannot trust the central entity and are forced to verify the correctness of the distributed proof.

The generalization of the class NP to interactive proof systems, a model where the prover and the verifier are allowed to interact was a breakthrough in computational complexity [3,18,19,32,40]. In the distributed framework, the notion of distributed interactive protocols was introduced in [27] and further studied in [9,14,36,37]. In such protocols, a centralized, untrustable prover with unlimited computation power, named Merlin, exchanges messages with a randomized distributed algorithm, named Arthur.

Let us illustrate the general idea of this model, and also some notation. If we consider, for instance, four interactions, then there are two possible protocols: a dAMAM protocol and dMAMA protocol. In a dAMAM protocol, also denoted $\mathsf{dAM}[4]$, the last interaction is performed by Merlin. In a dMAMA protocol, also denoted $\mathsf{dMA}[4]$, the last interaction is performed by Arthur. Note that $\mathsf{dAM}[1]{=}\mathsf{dM}$, and we recover exactly the PLS model.

Now, we are going to explain with more detail what happens in a particular case. Let us consider a three interaction, $\mathsf{dMAM}{=}\mathsf{dAM}[3]$ protocol. In this case Merlin starts, and he provides a certificate to Arthur (that is, certificates $c(v)$ for every node $v \in V$). Then, a random string (fresh randomness) is generated and made public to both Arthur (the nodes) and Merlin. This is the interaction performed by Arthur, and it should be interpreted as if Arthur was challenging Merlin. Note also that we are considering here the shared randomness setting.

Finally, Merlin replies to the query by sending another distributed certificate. After all these interactions, comes the deterministic distributed verification phase, performed between every node and its neighbors, after which every node decides whether to accept or reject.

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 certificates sent by the prover Merlin to the nodes, 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.

Interaction may decrease drastically the size of the messages needed to solve some problems. Consider, for instance, the problem symmetry, where the nodes are asked to decide whether the graph G has a non-trivial automorphism (i.e., a non-trivial one-to-one mapping from the set of nodes to itself preserving edges). Any PLS solving the symmetry problem requires certificates of size $\Omega(n^2)$ [21]. Nevertheless, this problem admits distributed interactive protocols with small certificates, and very few interactions. In fact, it can be solved with both a $\mathsf{dMAM}[\log n]$ protocol and a $\mathsf{dAM}[n \log n]$ protocol [27].

1.2 Geometric Intersection Graph Classes

A graph $G = (V, E)$ is a geometric intersection graph if every node $v \in V$ is identified with a geometric object of some particular type, and two vertices are adjacent if the corresponding objects intersect. The two simplest non-trivial, and arguably two of the most studied geometric intersection graphs are **interval graphs** and **permutation graphs**. In fact, most of the best-known geometric intersection graph classes are either generalizations of interval graphs or generalizations of permutation graphs. It comes as no surprise that many papers address different algorithmic and structural aspects, simultaneously, in both interval and permutation graph [2,23,30,43].

In both interval and permutation graphs, the intersecting objects are (line) segments, with different restrictions imposed on their positions. In interval graphs, the segments must all lie on the real line. In permutation graphs, the endpoints of the segments must lie on two separate, parallel real lines. In Figure 1 we show an example of a permutation graph.

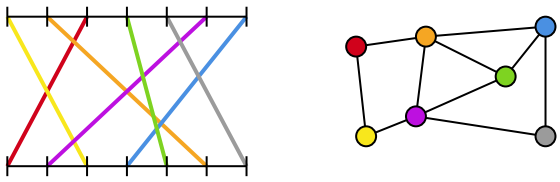

Fig. 1. An example of a permutation graph with its corresponding intersection model.

Although the class of interval graphs is quite restrictive, there are a number of practical applications and specialized algorithms for interval graphs [20,22,28]. Moreover, for several applications, the subclass of unit interval graphs (the situation where all the intervals have the same length) turns out to be extremely useful as well [4,25].

A natural generalization of interval graphs are **circular arc graphs**, where the segments, instead of lying on a line, lie on a circle. More precisely, a circular arc graph is the intersection graph of arcs of a circle. Although circular arc graphs look similar to interval graphs, several combinatorial problems behave very differently on these two classes of graphs. For example, the coloring problem is NP-complete for circular arc graphs while it can be solved in linear time on interval graphs [16]. Recognizing circular-arc graphs can be done in linear-time [24,34].

The class of permutation graphs behaves as the class of interval graphs in the sense that, on one hand, permutation graphs can be recognized in linear time [30] and, on the other hand, many NP-complete problems can be solved efficiently when the input is restricted to permutation graphs [8,31]. A graph G is a **circle graph** if G is the intersection model of a collection of chords in a circle (see Figure 2).

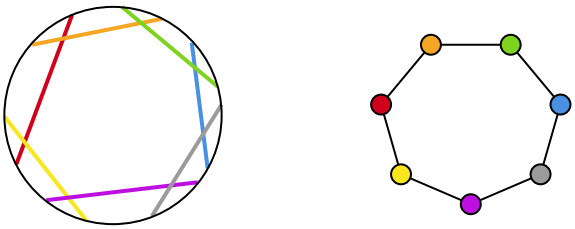

Fig. 2. An example of a circle graph with its corresponding intersection model.

Clearly, circle graphs are a generalization of permutation graphs. In fact, permutation graphs can be characterized as circle graphs that admit an *equator*, i.e., an additional chord that inter-

sects every other chord. Circle graphs can be recognized in time $\mathcal{O}(n^2)$ [41]. Many NP-complete problems can be solve in polynomial time when restricted to circle graphs [26,42].

The well-known class of **trapezoid graphs** is a generalization of both interval graphs and permutation graphs. A trapezoid graph is defined as the intersection graph of trapezoids between two horizontal lines (see Figure 3). Ma and Spinrad [33] showed that trapezoid graphs can be recognized in $\mathcal{O}(n^2)$ time. Trapezoid graphs were applied in various contexts such as VLSI design [10] and bioinformatics [1]. Note that trapezoid and circle graphs are incomparable: the trapezoid graph of Figure 3 is not a circle graph, while the circle graph of Figure 2 is not a trapezoid graph.

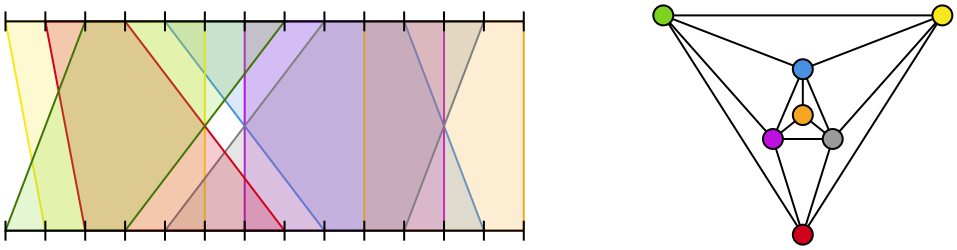

Fig. 3. An example of a permutation graph with its corresponding intersection model.

Recall that the way permutation graphs were generalized to circular graphs is by placing the ends of the segments in a circle (as chords) instead of placing the ends of the segments in two parallel lines. The same approach is used to generalize trapezoidal graphs and thus introducing polygon circle graphs.

More precisely, a **polygon circle graph** is the intersection graph of convex polygons of k sides, all of whose vertices lie on a circle. In this case we refer to a k-polygon circle graphs. In Figure 4 we show an example of a 3-polygon circle graph. Both trapezoid graphs and circle graphs are proper subclasses of polygon circle graphs. Note that the polygon circle graph of Figure 4 is neither a trapezoid graph nor a circle graph.

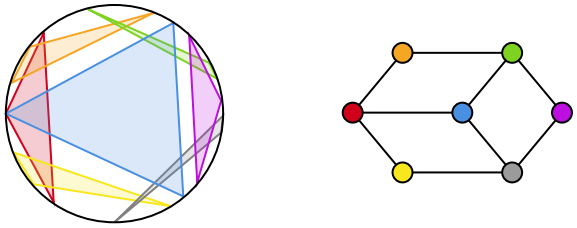

Fig. 4. An example of a 3-polygon circle graph with its corresponding intersection model.

The problem of recognizing whether a graph is a k-polygon circle graphs, for any $k \geq 3$, is NP-complete [39]. Nevertheless, many NP-complete problems have polynomial time algorithms when restricted to polygon circle graphs [16,17]. In an unpublished result, M. Fellows proved that the class of polygon circle graphs is closed under taking induced minors.

1.3 Our Results

In Section 3 we recall, explain and develop some tools, building blocks for the rest of the paper. In Section 4 we prove that trapezoid graphs can be recognized by PLSs with certificates of size $\mathcal{O}(\log n)$, and then we obtain the result for permutation graphs as a corollary. Then, in Section 5 we prove that k-polygon circle graphs can be recognized with a three round, dMAM protocol woth proof-size $\mathcal{O}(\log n)$, and we obtain the result for circle graphs as a particular case

when $k = 2$. Finally, in Section 6, we prove that any PLS for recognizing permutation graphs, trapezoid graphs, circle graphs or polygon circle graphs requires certificates of size $\Omega(\log n)$.

1.4 Related Work

We already know the existence of PLSs (with logarithmic size certificates) for the recognition of many graph classes such as acyclic graphs [29], planar graphs [13], graphs with bounded genus [12], graph classes defined by a finite set of forbidden minors [5], etc.

The distributed interactive proof model is clearly more powerful than the PLS model. Some problems, for which any PLS requires huge certificates, can be solved with a distributed interactive protocol with small certificates and, in fact, with very few interactions. This is the case of problem symmetry, where the system must decide whether a graphs has a non-trivial automorphism. Problem symmetry is in $\mathsf{dMAM}[\log n]$ and also in $\mathsf{dAM}[n \log n]$, while the size of any certificate in a PLS must be of size at least $\Omega(n^2)$ [27].

It is always important to keep in mind the compiler defined in [37] which turns, automatically, any problem solved in NP in time $\tau(n)$ into a dMAM protocol that uses bandwidth $\tau(n) \log n/n$. Therefore, any class of sparse graphs that can be recognized in linear time, can also be recognized by a dMAM protocol with logarithmic-sized certificates.

Any geometric intersection graph class is hereditary (a graph class is hereditary if the class is closed under taking induced subgraphs). Examples of hereditary graph classes include planar graphs, forests, bipartite graphs, perfect graphs, etc. Interestingly, the only graph properties that are known to require PLSs with large certificates (e.g. small diameter [7], non-3-colorability [21], having a non-trivial automorphism [21]), are non-hereditary. This raises the question of whether we can improve the result of this paper and design a PLS for the recognition of any geometric intersection graph.

2 Preliminaries

Let G be a simple connected n-node graph, let $I : V(G) \to \{0,1\}^\star$ 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).

Interactive protocols have two phases: an interactive phase and a verification phase. If Arthur is the party that starts the interactive phase, he picks a random string r_1 (known to all nodes of G because we are considering the shared randomness setting) and send it to Merlin. Merlin receives r_1 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 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 the last interaction round, 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, 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.

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 k-round, $\mathcal{O}(f(n))$ bandwidth protocol, we write $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$.*

Definition 2. *Let $\varepsilon \leq 1/3$. The class $\mathsf{dAM}_{\varepsilon}[k, f(n)]$ 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 $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ 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 $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$,*

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

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

For small values of k, instead of writing $\mathsf{dAM}[k, f(n)]$, we alternate M's and A's. For instance: $\mathsf{dMAM}[f(n)] = \mathsf{dAM}[3, f(n)]$. In particular $\mathsf{dAM}[f(n)] = \mathsf{dAM}[2, f(n)]$. Moreover, we denote $\mathsf{dM}[f(n)]$ the model where only Merlin provides a certificate, and no randomness is allowed (in other words, the model dM is the PLS model).

In this paper, we are interested mainly in the languages of graphs that are permutation, trapezoid, circle, polygon-circle and unit square. Formally,

Permutation-Recognition $= \{\langle G, \mathsf{id}\rangle$ s.t. G is a permutation graph$\}$.
Trapezoid-Recognition $= \{\langle G, \mathsf{id}\rangle$ s.t. G is a trapezoid graph$\}$.
Circle-Recognition $= \{\langle G, \mathsf{id}\rangle$ s.t. G is a circle graph$\}$.
k-Polygon-Circle-Recognition $= \{\langle G, \mathsf{id}\rangle$ s.t. G is a k-polygon-circle graph$\}$.

We denote by $[n]$ the set $\{0, \ldots, n-1\}$ and S_n the set of permutations of $[n]$. In the following, all graphs $G = (V, E)$ are simple and undirected. When the nodes of an n-node graph are enumerated with unique values in $[n]$, we denote $G = ([n], E)$. In a distributed problem, we always assume that the input graph is connected. We use the standard definitions and notations for (induced) subgraph, neighborhood, path, cycle, tree, clique, etc. For more details we refer to the textbook of Diestel [11].

3 Toolbox

In our results, we use some previously defined protocols as subroutines. In some cases, we consider protocols that solve problems which are more general than just decision problems (as, for instance, the construction of a spanning tree).

3.1 Spanning Tree and Related Problems

The construction of a spanning tree is an important building block for several protocols in the PLS model. Given a network configuration $\langle G, \mathsf{id}\rangle$, the Spanning-Tree problem asks to construct a spanning tree T of G, where each node has to end up knowing which of its incident edges belong to T.

Proposition 1. *There is a 1-round protocol for* Spanning-Tree *with certificates of size* $\mathcal{O}(\log n)$.

From the protocol of Proposition 1 it is easy to construct another one for problem Size, where the nodes, given the input graph $G = (V, E)$, have to verify the precise value of $|V|$ (recall that we are assuming that the nodes are only aware of a polynomial upper bound on $n = |V|$).

Proposition 2. *[29] There is a 1-round protocol for* Size *with certificates of size* $\mathcal{O}(\log n)$.

Finally, for two fixed nodes $s, t \in V$, problem $s,t-$Path is defined in the usual way: given a network configuration $\langle G, \mathsf{id}\rangle$, the output is a path P that goes from s to t. In other words, each node must end up knowing whether it belongs to P, and, in the positive cases, which of its neighbors are its predecessor and successor in P.

Proposition 3. *[29] There is a 1-round protocol for* $s,t-$Path *with certificates of size* $\mathcal{O}(\log n)$.

3.2 Problems Equality and Permutation

A second important building block, this time for interactive protocols, is a protocol for solving problem Equality, which is defined as follows. Given G a connected n-node graph, each node v receives two natural numbers $a(v)$ and $b(v)$, both of them encoded with $\mathcal{O}(log(n))$ bits. The problem Equality consists of verifying whether the multi-sets $\mathcal{A} = \{a(v)\}_{v\in V}$ and $\mathcal{B} = \{b(v)\}_{v\in V}$ are equal.

Proposition 4. *[37] Problem* Equality *belongs to* $\mathsf{dAM}_{1/3}[\log n]$.

A closely related problem is Permutation, where some function π is given as input, and the nodes must verify whether π is indeed a permutation (a bijective function from V to $[n]$). Note that the input is given in a distributed way, by given $\pi(v)$ to each node $v \in V$. Using the protocol for Equality as subroutine, it is possible to solve Permutation with certificates of size $\mathcal{O}(\log n)$.

Proposition 5. *[37] Problem* Permutation *belongs to* $\mathsf{dMAM}_{1/3}[\log n]$.

We now introduce a new problem called Corresponding Order, which is defined for inputs of the form $\langle G = (V, E), \mathsf{id}, (x, \pi)\rangle$, where the nodes must verify that: (i) π is a bijection from V to $[n]$; (ii) x is an injective function from V to $[N]$, where $N \geq n$; and (iii) for every $u, v \in V$, $\pi(u) \geq \pi(v) \iff x(u) \geq x(v)$.

Proposition 6. *Problem* Corresponding Order *belongs to* $\mathsf{dMAM}[\log N]$.

Proof. The following is a protocol for Corresponding Order. In the first round, each node v receives from the prover:

- The certification for the size of V and that π is an injective function.
- $a(v) = (x(v), \pi(v))$ and $b(v) = (y(v), \pi(v) + 1 \mod n)$, with $y(v) \in [N]$.

Suppose that π is an injective function and that n is known by all the nodes. Observe that, if $\{a(v)\}_{v\in V}$ and $\{b(v)\}_{v\in V}$ are equal, then $y(v) = x(u)$, where u is the successor of v, i.e. $\pi(u) = \pi(v) + 1$. Then, $\langle G = (V, E), \mathsf{id}, (x, \pi)\rangle$ is a yes-instance of Corresponding Order if and only if π is an injective function, $\{a(v)\}_{v\in V} = \{b(v)\}_{v\in V}$, and $x(v) \leq y(v)$ for each node v such that $\pi(v) < n - 1$.

Then, in the two remaining rounds, the nodes interact with the prover in order to prove that π is an injective function and that $\{a(v)\}_{v\in V}$, $\{b(v)\}_{v\in V}$ are equal multi-sets, using the protocols for Permutation and Equality, respectively. The nodes also check that $x(v) \leq y(v)$ (except for the node v such that $\pi(v) = n - 1$). The communication bounds, as well as the correctness and soundness of the protocol follows from the ones of protocols for Spanning-Tree, Size, Equality and Permutation. □

Note that our protocol for Corresponding Order can be easily extended to the case when the range of function x is a set S of size N that admits a total order.

4 Permutation and Trapezoid Graphs

To recognize trapezoid graphs, first we present a useful characterization of them that are used to build a compact one-round PLS for Trapezoid-Recognition, from which we derive a protocol for Permutation.

Remember that in a model of a trapezoid graph, there are two parallel lines $\mathcal{L}_t$ and $\mathcal{L}_b$. We denote this lines the *top and bottom lines*, respectively. Each trapezoid has sides contained in each line, and then defined by four vertices, two in the top line, and two in the bottom line. Formally, each trapezoid T is defined by the set $T = \{t_1, t_2, b_1, b_2\}$, where $t_1 < t_2$ and $b_1 < b_2$, with $t_1, t_2 \in \mathcal{L}_t$ and $b_1, b_2 \in \mathcal{L}_b$ (see Figure 5). The definition of a trapezoid graph can be restated

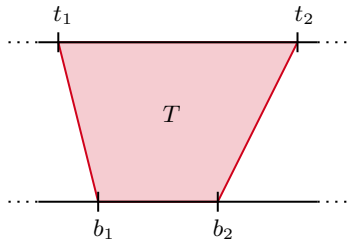

Fig. 5. Each trapezoid T is defined by the set $T = \{b_1, b_2, t_1, t_2\}$.

as follows (see [6]): A trapezoid graph $G = (V, E)$ is the intersection graph of a set of trapezoids $\{T_v\}_{v\in V}$ satisfying the following conditions. The vertices of each trapezoid have values in $[2n]$, two corresponding to the upper line and the other to the bottom line. The vertices defining the set $\{T_v\}_{v\in V}$, are all different, i.e., no pair of trapezoids share vertices. Therefore, in both the top and the bottom lines, each element in $[2n]$ correspond to a vertex of some trapezoid. The trapezoid model in the example of reffig:Extrapezoid satisfies these conditions.

For $v \in V$, we call $\{t_1(v), t_2(v), b_1(v), b_2(v)\}$ the vertices of T_v. Moreover, we say that $\{t_1(v), t_2(v), b_1(v), b_2(v)\}$ are the *vertices* of node v. In the following, a trapezoid model satisfying the abode conditions is called a *proper trapezoid model* for G. Given a graph $G = (V, E)$ (that is not necessarily a trapezoid graph), a *semi-proper trapezoid model* for G is a set of trapezoids $\{T_v\}_{v\in V}$ satisfying previous conditions, such that, for every $\{u, v\} \in E$, the trapezoids T_v and T_u have nonempty intersection. The difference between a proper and a semi-proper model is that in the first we also ask every pair of non-adjacent edges have non-intersecting trapezoids.

Given a trapezoid graph $G = (V, E)$ and a proper trapezoid model $\{T_v\}_{v\in V}$, we define the following sets for each $v \in V$:

$$F_t(v) = \{i \in [2n] \mid i < t_1(v) \text{ and } i \in \{t_1(w), t_2(w)\} \text{ for some } w \notin N(v)\}$$
$$F_b(v) = \{i \in [2n] \mid i < b_1(v) \text{ and } i \in \{b_1(w), b_2(w)\} \text{ for some } w \notin N(v)\}$$

We also call $f_t(v) = |F_t(v)|$ and $f_b(v) = |F_b(v)|$. The following lemmas characterize trapezoid graphs.

Lemma 1. *Let $G = (V, E)$ an connected trapezoid a graph with n nodes. Then each proper trapezoid model $\{T_v\}_{v \in V}$ of G satisfies for every $v \in V$:*

$$b_1(v) - f_b(v) = t_1(v) - f_t(v)$$

Proof. Let $\{T_v\}_{v \in V}$ be a proper trapezoid model of G. Then, given a node $v \in V$, all the coordinates in $F_t(v)$ are vertices of some $w \neq N(v)$. Such trapezoids T_w have their two upper vertices in the set $[t_1(v)]$ and their two lower vertices in $[b_1(v)]$, as otherwise T_w and T_v would intersect. Then, the cardinality of the set $[t_1(v)] \setminus F_t(v)$ is even, and the same holds for $[b_1(v)] \setminus F_b(v)$. Moreover, the cardinality of the set $[t_1(v)] \setminus F_t(v)$ equals the cardinality of $[b_1(v)] \setminus F_b(v)$, as every position in $[2n]$ corresponds to a vertex of some trapezoid's node. We deduce that

$$t_1(v) - f_t(v) = |\{1, \ldots, t_1(v)\} \setminus F_t(v)| = |\{1, \ldots, b_1(v)\} \setminus F_b(v)| = b_1(v) - f_b(v).$$

Lemma 2. *Let $G = (V, E)$ be a n-node graph that is not a trapezoid graph. Then, for every semi-proper trapezoid model $\{T_v\}_{v \in V}$ of G, at least one of the following conditions is true:*

1. *$\exists v \in V$ such that some value in $\{b_1(v), \ldots, b_2(v)\}$ or $\{t_1(v), \ldots, t_2(v)\}$ is a vertex of $\omega \notin N(v)$.*
2. *$\exists v \in V$ such that $b_1(v) - f_b(v) \neq t_1(v) - f_t(v)$.*

Proof. Let G be a graph that is not a trapezoid graph and $\{T_v\}_{v \in V}$ a semi-proper trapezoid model. As G is not a permutation graph, by definition necessarily there exist a pair $\{v, \omega\} \notin E$ such that $T_v \cap T_\omega \neq \emptyset$. We distinguish two possible cases (see Figure 6):

1. $[b_1(v), b_2(v)]_{\mathbb{N}} \cap [b_1(\omega), b_2(\omega)]_{\mathbb{N}} \neq \emptyset$ or $[t_1(v), t_2(v)]_{\mathbb{N}} \cap [t_1(\omega), t_2(\omega)]_{\mathbb{N}} \neq \emptyset$.
2. $[b_1(v), b_2(v)]_{\mathbb{N}} \cap [b_1(\omega), b_2(\omega)]_{\mathbb{N}} = \emptyset$ and $[t_1(v), t_2(v)]_{\mathbb{N}} \cap [t_1(\omega), t_2(\omega)]_{\mathbb{N}} = \emptyset$.

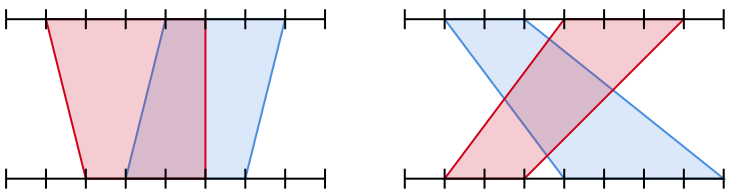

Fig. 6. A representation of the two possible cases. In the first case, depicted in left, at least one vertex of a trapezoid is contained in the other. In the second case, in the right hand, the trapezoids intersect, but not in the vertices.

Clearly if the first case holds, then condition 1 is satisfied. Suppose then that there is no pair $\{v, \omega\} \notin E$ such that $T_v \cap T_\omega \neq \emptyset$ satisfying the first case. Then necessarily the second case holds. Let u be a node for which exists $\omega \in V \setminus N(u)$ such that $T_u \cap T_w \neq \emptyset$. For all possible choices of u, let us pick the one such that $b_1(u)$ is minimum. Then u satisfies the following conditions:

- **(a)** Exists a node $\omega \in V$ such that $\omega \notin N(v)$ and $T_u \cap T_\omega \neq \emptyset$
- **(b)** All nodes $\omega \in V$ such that $\omega \notin N(v)$ and $T_u \cap T_\omega \neq \emptyset$ satisfy that $t_2(\omega) < t_1(u)$ and $b_2(u) < b_1(\omega)$
- **(c)** None of the positions in $\{1, \ldots, b_1(u)\}$ is occupied by a vertex of a node ω such that $\{u, \omega\} \notin E$ and $T_u \cap T_\omega \neq \emptyset$.

Observe that conditions **(a)** and **(b)** imply that $t_1(u) - f_t(u) > 0$, while condition **(c)** implies that $b_1(u) - f_b(u) = 0$. We deduce that condition 2 holds by u. □

We are now ready to define our protocol and main result regarding Trapezoid-Recognition.

Theorem 1. *There is a 1-round proof labelling scheme for* Trapezoid-Recognition *with certificates of size* $\mathcal{O}(\log n)$.

Proof. The following is a one-round PLS for Trapezoid-Recognition

Given an instance $\langle G = (V, E), \mathsf{id}\rangle$, the certificate provided by the prover to node $v \in V$ is interpreted as follows.

1. The certification of the total number of nodes n, according to some protocol for Size.
2. Values $b_1(v), b_2(v), t_1(v), t_2(v) \in [2n]$, such that $b_1(v) < b_2(v)$ and $t_1(v) < t_2(v)$, representing the vertices of a trapezoid T_v.
3. Value p_v corresponding to the minimum position in the upper line greater that $t_1(v)$ that is not a vertex of a neighbor of v.
4. Value q_v corresponding minimum position in the lower line grater than $b_1(v)$ that is not a vertex of a neighbor of v.
5. The certification of a path P_t between the node with vertex 0 and the node with vertex $2n-1$ in the upper line (respecting assignment in 2.) and a path P_b between the node node with vertex 0 and the node with vertex $2n-1$ in the lower line. Both paths according to a protocol for $s, t -$ Path.

Then, in the verification round, each node shares with its neighbors their certificates. Using that information each node v can compute $f_t(v)$ and $f_b(v)$, and check the following conditions:

- **a.** The correctness of the value of n, according to some protocol for Size.
- **b.** The correctness of the paths P_b and P_t, according to a protocol for $s, t -$ Path.
- **c.** The vertices of the trapezoid of v are in $[2n]$.
- **d.** $T_v \cap T_\omega \neq \emptyset$ for all $\omega \in N(v)$.
- **e.** All values in $\{t_1(v)+1, \ldots, t_2(v)-1\}$ and $\{b_1(v)+1, \ldots, b_2(v)-1\}$ are a vertex of some neighbor of v.
- **f.** $t_2(v) < p_v$ and $b_2(v) < q_v$.
- **g.** If $\omega \in N(v)$ and $p_\omega < t_2(v)$, then v verifies that p_ω is a vertex of some other neighbor.
- **h.** If $\omega \in N(v)$ and $q_\omega < b_2(v)$, then v verifies that q_ω is a vertex of some other neighbor.
- **i.** $b_1(v) - f_b(v) = t_1(v) - f_t(v)$.

We now analyze the soundness and completeness of our protocol.

Completeness: Suppose that G is a trapezoid graph. An honest prover just has to send the real number of nodes n, a trapezoid model $\{T_v\}_{v\in V}$ of G and valid paths P_b and P_t according the trapezoid model. Then, the nodes will verify **a**, **b** by the completeness of the protocols for Size and $s, t -$ Path. Conditions **c**, **d**, **e** ,**f**, **g** and **h** are verified by the correctness of the model $\{T_v\}_{v\in V}$. Condition **i** is also verified, by Lemma 1.

Soundness: Suppose G is not a trapezoid graph. If a dishonest prover provides a wrong value of n, or wrong paths P_t or P_b, then at least one node will reject verifying **a** or **b**. Then, we assume that the prover cannot cheat on these values.

Suppose that the prover gives values $\{T_v\}_{v\in V}$ such that is fulfilled $\bigcup_{v\in V}\{t_1(v), t_2(v)\} \neq [2n]$. If some vertex of a node is not in the set $[2n]$, then that node fails to verify condition **c** and rejects. Without loss of generality, we can assume that there exists a $j \in [2n]$ such that $t_1(v), t_2(v) \neq j$, for every $v \in V$. If a node ω satisfies that $t_1(\omega) < j < t_2(\omega)$, then node ω fails to verify condition **e** and rejects. Then j is not contained in any trapezoid. As P_t is correct, j must be different than 1 and $2n$. Also by the correctness of P_t, there exist a pair of adjacent nodes $u, v \in V$ such that $t_2(u) < j < t_1(v)$. From all possible choices for u and v, we pick the one such that $t_2(u)$ is maximum. We claim that v fails to check condition **g**. Since j is not a vertex of any node, then

$p_u \leq j$. If v verifies condition **g**, then necessarily $p_u < j$. Then, there must exist a node $\omega \in N(v)$ such that $p_u = t_1(\omega)$. But since we are assuming that j is not contained in any trapezoid, we have that $t_2(\omega) < j$, contradicting the choice of u.

Therefore, if conditions **a** - **h** are verified, we can assume that the nodes are given a semi-proper trapezoid model of G. Since we are assuming that G is not a trapezoid graph, by Lemma 2 we deduce that condition **i** cannot be satisfied and some node rejects.

We now analyze the communication complexity of the protocol: the certification for Size and $s,t-$Path is $\mathcal{O}(\log n)$, given by refprop:sizeofG and refprop:stpath. On the other hand, for each $v \in V$, the values $b_1(v)$, $b_2(v)$, $t_1(v)$, $t_2(v)$, p_v, q_v are computable in $\mathcal{O}(\log n)$ space as they are numbers in $[2n]$. Overall the total communication is $\mathcal{O}(\log n)$. □

Observe that permutation graphs are exactly the trapezoid graphs that admit a proper model where $t_2(v) = t_1(v) + 1$ and $b_2(v) = b_1(v) + 1$, for every $v \in V$. Then, previous protocol can be adapted to solve Permutation-Recognition, simply asking the nodes to accept only the models that satisfy previous condition. We conclude that Permutation-Recognition admits a PLS with certificates of size $\mathcal{O}(\log n)$.

5 Circle and Polygon Circle Graphs

In this section, we give a three-round protocol for the recognition of polygon-circle graph. This extension is based in a non-trivial extension of the properties of circle graphs.

Remember that a n-node graph $G = (V, E)$ is a k-polygon-circle graph if and only if G is the intersection model of a set of n polygons of k vertices inscribed in a circle, namely $\{P_v\}_{v\in V}$. Further, every k-polygon-circle graph admits a model satisfying the following conditions [6]: (1) for each $v \in V$, the polygon P_v is represented as a set of k vertices $\{p_0(v), \ldots, p_{k-1}(v)\}$ such that, for each $i \in [k-1]$, $1 \leq p_i(v) < p_{i+1}(v) \leq n \cdot k$, and (2) $\bigcup_{v\in V} \bigcup_{i\in[k]} \{p_i(v)\} = [n \cdot k]$. In other words, each value in $[k \cdot n]$ corresponds to a unique vertice of some polygon. A set of polygons satisfying conditions (1) and (2) are called a *proper polygon model* for G. Similar to previous cases, when we just ask that adjacent nodes have intersecting polygons (but not necessarily the reciprocal) we say that the model is a *semi-proper polygon model* for G.

Let G be a graph and $\{P_v\}$ be a semi-proper model for G. For each $v \in V$. Let us call $\alpha(v)$ the set of points in $\{1, \ldots, p_1(v)\} \cup \{p_k(v), \ldots, kn\}$ that do not correspond to a neighbor of v. For each $i \in [k]$, we also call $\beta_i(v)$ the set of nodes $w \notin N(v)$ such that $p_i(w) \in \alpha(v)$. Formally,

$$\alpha(v) = \{i \in [0, p_1(v)]_{\mathbb{N}} \cup [p_k(v), kn-1]_{\mathbb{N}} : \forall u \in N(v), i \notin P_u\}$$
$$\beta_i(v) = \{w \in V : p_i(w) \in \alpha(v)\}$$

Lemma 3. *Let $G = (V, E)$ be a graph, and let $\{P_v\}_{v\in V}$ a semi-proper model for G. Then $\{P_v\}_{v\in V}$ is a proper model for G if and only if $|\alpha(v)| = k|\beta_1(v)|$ for every $v \in V$.*

Proof. Let us suppose first that $\{P_v\}_{v\in V}$ is a proper model for G and v be an arbitrary node. If $\alpha(v) = \emptyset$ the result is direct. Then, let us suppose that $\alpha(v) \neq \emptyset$, and let us pick $q \in \alpha(v)$. Then necessarily there exists $i \in [k]$ such that q belongs to $\beta_i(v)$. Observe that for each node w in $\beta_i(v)$, all the vertices of the polygon P_w are contained α_v. Otherwise, the polygons P_w and P_v would have non-empty intersection, which contradicts the fact that $\{P_v\}_{v\in V}$ is a proper model. This implies that $|\alpha(v)| = k|\beta_i(v)|$, for every $i \in [k]$. In particular $|\alpha(v)| = k|\beta_1(v)|$.

Let us suppose now that $\{P_v\}_{v\in V}$ is not a proper model for G. Let us define the set C of vertices having non-neighbor with intersecting polygons, formally

$$C = \{v \in V : \exists w \in V, \{v, w\} \notin E \text{ and } P_v \cap P_w \neq \emptyset\}.$$

Now pick $v \in C$ such that $p_1(v)$ is maximum, and call C_v the set of non-neighbors of v whose polygons intersect with P_v. Let w be an arbitrary node in C_v. By the maximality of $p_1(v)$, we

know that $p_1(w) \in \beta_1(v)$. But since $P_w \cap P_v \neq \emptyset$, there must exist $i \in [k]$ such that $p_i(w) \notin \beta_i(v)$. This implies that $|\beta_1(v)| \geq |\beta_i(v)|$ for every $i \in [k]$, and at least one of these inequalities is strict. Since $|\alpha(v)| = \sum_{i\in[k]} |\beta_i(v)|$, we deduce that $k|\beta_1(v)| > |\alpha(v)|$. □

Let $G = (V, E)$ be a graph, and $\{P_v\}_{v\in V}$ be a semi-proper polygon model for G. For each $i \in [k]$ and $v \in V$, we denote by $\pi_i(v)$ the cardinality of the set $\{u \in V : p_i(u) < p_i(v)\}$, and denote by $\sigma_i(v)$ the cardinality of the set $\{q < p_i(v) : \exists u \in V : p_1(u) = q \vee p_k(u) = q\}$.For a node v, we denote $N_{1,k}(v)$ the number vertices of polygons corresponding to neighbors of v, that are contained $[0, p(v_1)]_{\mathbb{N}} \cup [p(v_k), kn]_{\mathbb{N}}$. Formally,

$$N_{1,k}(v) = |\{q \in [0, p(v_1)]_{\mathbb{N}} \cup [p(v_k), kn]_{\mathbb{N}} : \exists w \in N(v), q \in P_w\}|$$

Lemma 4. *Let $G = (V, E)$ be a graph, and $\{P_v\}_{v\in V}$ be a semi-proper polygon model for G. Then, $|\alpha(v)| = kn - p_k(v) + p_1(v) - 1 - N_{1,k}(v)$, and $|\beta_1(v)| = n - \sigma_k(v) + \pi_k(v) + \pi_1(v)$.*

Proof. Let $\{P_v\}_{v\in V}$ be a semi-proper polygon model for G and v be an arbitrary node. First, observe that there are $p_1(v)$ integer positions for vertices in $[p_1(v)]$ and $(kn - 1) - p_k$ positions for vertices in $[p_k, kn - 1]$. Then, there are $kn - p_k + p_1(v) - 1$ available integer positions in $[p_1(v)] \cup [p_k(v), kn-1]_{\mathbb{N}}$. Since $N_{1,k}(v)$ of these positions are occupied by a polygon corresponding some neighbor of v, we deduce that $\alpha(v) = kn - p_k + p_1(v) - 1 - N_{1,k}(v)$.

Second, observe that the set $\{p_1(u), p_k(u)\}_{u\in V}$ uses $2n$ of the kn possible positions. Then, there are $2n-\sigma_k(v)$ positions used the elements of $\bigcup_{u\in V}\{p_1(u), p_k(u)\} \cap [p_k(v)+1, kn-1]_{\mathbb{N}}$. On the other hand, there are $n-\pi_k$ positions used by vertices in $\bigcup_{u\in V}\{p_k(u)\} \cap [p_k(v)+1, kn-1]_{\mathbb{N}}$. Therefore, there are $n - \sigma_k(v) + \pi_k(v)$ positions used by $\bigcup_{u\in V}\{p_1(u)\} \cap [p_k(v) + 1, kn - 1]_{\mathbb{N}}$. Finally, noticing that there are $\pi_1(v)$ positions used by $\bigcup_{u\in V}\{p_1(u)\} \cap [0, p_1(v) - 1]_{\mathbb{N}}$, we deduce that $|\beta_1(v)| = n - \sigma_k(v) + \pi_k(v) + \pi_1(v)$. □

We are now ready to give the main result of this section.

Theorem 2. *k-Polygon-Circle-Recognition belongs to $\mathsf{dMAM}[\log n]$.*

Proof. Consider the protocols for Size, Permutation and Corresponding Orderof Propositions 2, 5 and 6.Given an instance $\langle G, \mathsf{id}\rangle$, consider the following three round protocol. In the first round, the prover provides each node v with the following information:

1. The certification of the total number of nodes n, according to the protocol for Size.
2. The vertices of the polygon P_v, denoted $V(P_v) = \{p_1(v), \ldots, p_k(v)\}$.
3. The values of $\pi_1(v)$, $\pi_k(v)$ and $\sigma(v)$.
4. The certification of $\bigcup_v V(P_v) = [k \cdot n]$ according to the protocol for Permutation.
5. The certification of the correctness of $\{(p_1(v), \pi_1(v))\}_{v\in V}$ according to the protocol for Corresponding Order.
6. The certification of the correctness of $\{(p_k(v), \pi_k(v))\}_{v\in V}$ according to the protocol for Corresponding Order.
7. The certification of the correctness of $\{(p_1(v), \sigma_1(v)\}_{v\in V}$ and the collection $\{p_k(v), \sigma_k(v)\}_{v\in V}$ according to the protocol for Corresponding Order.

Then, in the second and third round the nodes perform the remaining interactions of the protocols for Permutation and Corresponding Order. In the verification round, the nodes first check the correctness of 1-7 according to the verification rounds for Size, Permutation and Corresponding Order.

Remark: in order to check 7, each node has to play the role of two different nodes v', v'', one to verify $\{(p_1(v), \sigma(v')\}_{v\in V}$, and the other one to verifies $\{(p_k(v), \sigma(v'')\}_{v\in V}$, where $\sigma(v') = \sigma_1(v)$ and $\sigma(v'') = \sigma_k(v)$. To do so, Merlin gives v the certificates of v' and v'', and v answers with the

random bits as if they would be generated by v' and v''. Obviously, this increases the communication cost by a factor of 2.

Then, each node v computes $|\beta(v)|$ and $|\alpha(v)|$ according to the expressions of lemma 4, and checks the following conditions:

a. $\forall u \in N(v)$, $P_u \cap P_v \neq \emptyset$.
b. $|\alpha(v)| = k|\beta_1(v)|$.

We now analyze completeness and soundness.

Completeness: Suppose that input graph G is a k-polygon-circle graph. Then Merlin gives a proper polygon model $\{P_v\}_{v\in V}$ for G. Merlin also provides the correct number of nodes n, correct orders $\{\pi_1(v)\}_{v\in V}$ and $\{\pi_k(v)\}_{v\in V}$, $\{\sigma_1(v)\}_{v\in V}$ and $\{\sigma_k(v)\}_{v\in V}$, and the certificates required in the corresponding sub-routines. Then, the nodes verify correctness of **1**-**7** with probability greater than 2/3, by the correctness of the protocols for Size, Permutation and Corresponding Order. Finally, condition **a** is verified by definition of a proper model, and condition **b** is verified by lemma 3. We deduce that every node accepts with probability greater than 2/3.

Soundness: Suppose now that G is not a k-polygon-circle graph. By the soundness of the protocols for Size, Permutation and corresponding order, we now that at least one node rejects the certificates not satisfying **1**-**7**, with probability greater than 2/3. Suppose then that conditions **1**-**7** are verified. Observe that every set of polygons satisfying condition **a** form a semi-proper polygon model for G. Since G is not a k-polygon-circle graph, by lemma 3 we deduce that at least one node fails to verify **a** or **b**. All together, we deduce that at least one node rejects with probability greater than 2/3.

We now analyze the communication complexity of the protocol: the certification for Size, Permutation and Corresponding Order is $\mathcal{O}(\log n)$, given by proposition 2, 5 and 6. On the other hand, for each $v \in V$, the values $\pi_1(v), \pi_k(v), \sigma_1(v), \sigma_2(v), V(P_v)$ can be encoded in $\mathcal{O}(\log n)$ as they are numbers in $[n]$, $[2n]$ or $[kn]$. Overall the total communication is $\mathcal{O}(\log n)$.

Since circle graphs are 2-polygon-circle graphs, we deduce that Circle-Recognition is in $\mathsf{dMAM}[\log n]$.

6 Lower Bounds

In this section we give logarithmic lower-bounds in the certificate sizes of any PLS that recognizes the class of permutation, trapezoid, circle or polygon-circle graphs. In order to so, we use a technique given by Fraigniaud et al [15], called *crossing edge*, and which we detail as follows. Let $G = (V, E)$ be a graph and let $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ be two subgraphs of G. We say that H_1 and H_2 are independent if and only if $V_1 \cap V_2 = \emptyset$ and $E \cap (V_1 \times V_2) = \emptyset$.

Definition 3 ([15]). *Let $G = (V, E)$ be a graph and let $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ be two independent isomorphic subgraphs of G with isomorphism $\sigma\colon V_1 \to V_2$. The* crossing *of G induced by σ, denoted by $\sigma_{\bowtie}(G)$, is the graph obtained from G by replacing every pair of edges $\{u, v\} \in E_1$ and $\{\sigma(u), \sigma(v)\} \in E_2$, by the pair $\{u, \sigma(v)\}$ and $\{\sigma(u), v\}$.*

Then, the tool that we use to build our lower-bounds is the following.

Theorem 3 ([15]). *Let $\mathcal{F}$ be a family of network configurations, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Suppose that there is a configuration $G_s \in \mathcal{F}$ satisfying that (1) G contains as subgraphs r pairwise independent isomorphic copies $H_1, ..., H_r$ with s edges each, and (2) there*

exists r port-preserving isomorphisms $\sigma_i \colon V(H_1) \to V(H_i)$ such that for every $i \neq j$, the isomorphism $\sigma^{ij} = \sigma_i \circ \sigma_j^{-1}$ satisfies $\mathcal{P}(G_s) \neq \mathcal{P}(\sigma^{ij}_{\bowtie}(G)_s)$. Then, the verification complexity of any PLS for $\mathcal{P}$ and $\mathcal{F}$ is $\Omega\left(\frac{log(r)}{s}\right)$.

Let us consider first permutation and trapezoid graphs. Let $\mathcal{F}$ the family of instances of Permutation-Recognition, induced by the family of graphs $\{Q_n\}_{n>0}$. Each graph Q_n consists of $5n$ nodes forming a path $\{v_1, \ldots, v_{5n}\}$ where we add the edge $\{v_{5i-3}, v_{5i-1}\}$, for each $i \in [n]$. It is easy to see that for each $n > 0$, Q_n is a permutation graph (and then also a trapezoid graph). In fig. 7 is depicted the graph Q_3 and its corresponding model.

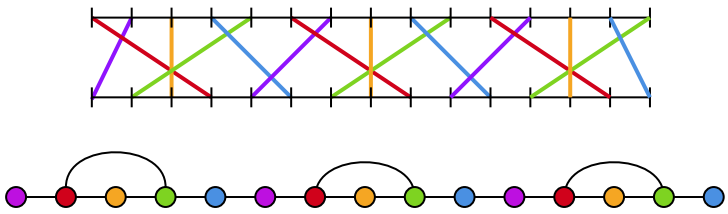

Fig. 7. Graph Q_3 and a permutation model for Q_3.

Given Q_n defined above, consider the subgraphs $H_i = \{v_{5i-2}, v_{5i-1}\}$, for each $i \in [n]$, and the isomorphism $\sigma_i \colon V(H_1) \to V(H_i)$ such that $\sigma_i(v_3) = v_{5i-2}$ and $\sigma_i(v_4) = v_{5i-1}$.

Lemma 5. *For each $i \neq j$, the graph $\sigma^{ij}_{\bowtie}(Q_n)$ it is not a trapezoid graph.*

Proof. Given $i < j$, observe that in $\sigma^{ij} \colon V(H_j) \to V(H_i)$, the nodes v_{5j-3}, v_{5j-2}, v_{5i-1}, v_{5i-3}, v_{5i-2}, v_{5j-1} form an induced cycle of length 6 (see fig. 8 for an example).

Fig. 8. Graph $\sigma^{12}_{\bowtie}(Q_3)$, where in red are represented the crossing edges. Observe that this graph is not a trapezoid graph, as it contains an induced cycle of length 6.

As a trapezoid graph have induced cycles of length at most 6, we deduce that $\sigma^{ij}_{\bowtie}(Q_n)$ is not a trapezoid graph.

By theorem 3 and the abode result, the lower bound result is direct.

Theorem 4. *Any PLS for* Permutation-Recognition *or* Trapezoid-Recognition *has proof-size of $\Omega(\log n)$ bits.*

We now tackle the lower-bound for circle and polygon-circle graphs. Let $\mathcal{G}$ the family of instances of Circle-Recognition, defined by the family of graphs $\{M_n\}_{n>2}$. Each graph M_n consists of $6n$ nodes, where $4n$ nodes form a path $\{v_1, \ldots, v_{4n}\}$ where we add, for each $i \in [n]$, the edges $\{v_{4i-3}, v_{4n+i}\}$, $\{v_{4i-2}, v_{5n+i}\}$ and $\{v_{4n+i}, v_{5n+i}\}$. It is easy to see that for each $n > 0$, M_n is a circle graph (and then also a polygon-circle graph). In Figure 9 is depicted the graph M_4 and its corresponding model.

Given M_n defined above, consider the subgraphs $H_i = \{v_{4n+i}, v_{5n+i}\}$, for each $i \in [n]$, and the isomorphism $\sigma_i \colon V(H_1) \to V(H_i)$ such that $\sigma_i(4n+1) = v_{5n+i}$ and $\sigma_i(5n+1) = v_{4n+i}$.

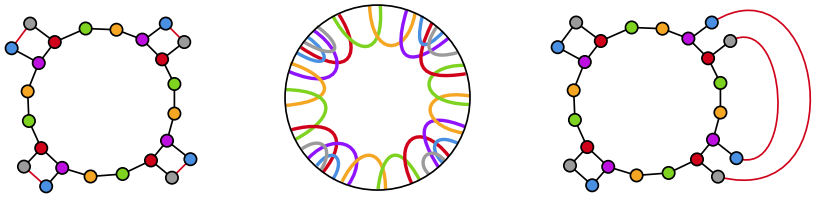

Fig. 9. Left: Graph M_4. Middle: a permutation model for M_4. Right: Graph $\sigma_{\bowtie}^{i,i+1}(M_4)$, where in red are represented the crossing edges.

Lemma 6. *For every $k > 0$ and each $i \neq j$, the graph $\sigma_{\bowtie}^{ij}(M_n)$ it is not a k-polygon-circle graph.*

Proof. First, observe that in $\sigma_{\bowtie}^{ij}(M_n)$ we have two induced cycles defined by $C_1 = v_1, \ldots, v_{4n}$, and $C_2 = v_{4j-3}, v_{4n+j}, v_{5n+i}, v_{4i-2}, v_{4i-3}, v_{4n+i}, v_{5n+j}$. Moreover $|V(C_1) \cap V(C_2)| = 4$, $|V(C_1) - V(C_2)| = 4n - 4$ and $|V(C_2) - V(C_1)| = 4$. See fig. 9 for a representation of $\sigma_{\bowtie}^{i,i+1}(M_4)$.

Claim. Every graph G consisting in two graphs C_1 and C_2 such that $|V(C_1) \cap V(C_2)| \geq 4$, $|V(C_1) - V(C_2)| \geq 2$ and $|V(C_2) - V(C_1)| \geq 2$ is not a k-polygon-circle graph, for every $k > 0$.

Let us denote by v_i and v_f two special nodes with degree 3, connecting the two cycles. Suppose there exists a k-polygon-circle model for G. Observe that, if we delete all polygons corresponding to nodes of $V(C_2) - V(C_1)$, we obtain a polygon model for C_1. However, the cycle C_1 has at least 4 nodes, because $|V(C_1) - V(C_2)| \geq 2$ and $|V(C_1) \cap V(C_2)| \geq 4$. Then, there is no way to add the removed polygons corresponding to $V(C_2) - V(C_1)$, without intersecting a polygon of $V(C_1) - V(C_2)$.

Then, by section 6, we deduce that the graph induced by $C_1 \cup C_2$ is not a k-polygon-cycle graph. Since the class of polygon-circle graphs is hereditary, we deduce that $\sigma_{\bowtie}^{ij}(M_n)$ it is not a k-polygon-circle graph. □

Direct by Theorem 3 and Lemma 6 we deduce the following result.

Theorem 5. *Any PLS for* Circle-Recognition *or k-*Polygon-Circle-Recognition *has proof-size of $\Omega(\log n)$ bits.*

References

1. Mohamed Ibrahim Abouelhoda and Enno Ohlebusch. Chaining algorithms for multiple genome comparison. *Journal of Discrete Algorithms*, 3(2-4):321–341, 2005.
2. Katerina Asdre, Kyriaki Ioannidou, and Stavros D Nikolopoulos. The harmonious coloring problem is np-complete for interval and permutation graphs. *Discrete Applied Mathematics*, 155(17):2377–2382, 2007.
3. László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.
4. Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM (JACM)*, 30(3):479–513, 1983.
5. Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Local certification of graph decompositions and applications to minor-free classes. *arXiv preprint arXiv:2108.00059*, 2021.
6. Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, January 1999.
7. Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theoretical Computer Science*, 811:112–124, 2020.
8. HS Chao, Fang-Rong Hsu, and Richard C. T. Lee. An optimal algorithm for finding the minimum cardinality dominating set on permutation graphs. *Discrete Applied Mathematics*, 102(3):159–173, 2000.
9. Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In *33rd International Symposium on Distributed Computing (DISC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

10. Ido Dagan, Martin Charles Golumbic, and Ron Yair Pinter. Trapezoid graphs and their coloring. *Discrete Applied Mathematics*, 21(1):35–46, 1988.
11. Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173, 2005.
12. Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and Ioan Todinca. Local certification of graphs with bounded genus. *arXiv preprint arXiv:2007.08084*, 2020.
13. Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and Ioan Todinca. Compact distributed certification of planar graphs. *Algorithmica*, pages 1–30, 2021.
14. Pierre Fraigniaud, Pedro Montealegre, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. On Distributed Merlin-Arthur Decision Protocols. In *International Colloquium on Structural Information and Communication Complexity*, pages 230–245. Springer, 2019.
15. Pierre Fraigniaud, Boaz Patt-Shamir, and Mor Perry. Randomized proof-labeling schemes. *Distributed Computing*, 32(3):217–234, 2019.
16. Michael R Garey, David S Johnson, Gary L Miller, and Christos H Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980.
17. Fanica Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73(5-6):181–188, 2000.
18. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.
19. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.
20. Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.
21. Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory of Computing*, 12(1):1–33, 2016.
22. Magnús M Halldórsson and Christian Konrad. Improved distributed algorithms for coloring interval graphs with application to multicoloring trees. *Theoretical Computer Science*, 811:29–41, 2020.
23. Mamadou Moustapha Kanté, Vincent Limouzy, Arnaud Mary, Lhouari Nourine, and Takeaki Uno. On the enumeration and counting of minimal dominating sets in interval and permutation graphs. In *International Symposium on Algorithms and Computation*, pages 339–349. Springer, 2013.
24. Haim Kaplan and Yahav Nussbaum. A simpler linear-time recognition of circular-arc graphs. In *Scandinavian Workshop on Algorithm Theory*, pages 41–52. Springer, 2006.
25. Haim Kaplan and Ron Shamir. Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, 1996.
26. Ton Kloks. Treewidth of circle graphs. In *International Symposium on Algorithms and Computation*, pages 108–117. Springer, 1993.
27. Gillat Kol, Rotem Oshman, and Raghuvansh R Saxena. Interactive distributed proofs. In *ACM Symposium on Principles of Distributed Computing*, pages 255–264. ACM, 2018.
28. Christian Konrad and Viktor Zamaraev. Distributed minimum vertex coloring and maximum independent set in chordal graphs. In *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
29. Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.
30. Dieter Kratsch, Ross M McConnell, Kurt Mehlhorn, and Jeremy P Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM Journal on Computing*, 36(2):326–353, 2006.
31. Evaggelos Lappas, Stavros D Nikolopoulos, and Leonidas Palios. An o (n)-time algorithm for the paired domination problem on permutation graphs. *European Journal of Combinatorics*, 34(3):593–608, 2013.
32. Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM (JACM)*, 39(4):859–868, 1992.
33. Tze-Heng Ma and Jeremy P Spinrad. On the 2-chain subgraph cover and related problems. *Journal of Algorithms*, 17(2):251–268, 1994.
34. Ross M McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003.
35. Terry A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. Society for Industrial and Applied Mathematics, January 1999. doi:10.1137/1.9780898719802.
36. Pedro Montealegre, Diego Ramírez-Romero, and Ivan Rapaport. Shared vs private randomness in distributed interactive proofs. volume 181 of *LIPIcs*, pages 51:1–51:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
37. Moni Naor, Merav Parter, and Eylon Yogev. The power of distributed verifiers in interactive proofs. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1096–115. SIAM, 2020.
38. Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.
39. Martin Pergel. Recognition of polygon-circle graphs and graphs of interval filaments is np-complete. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 238–247. Springer, 2007.
40. Adi Shamir. Ip= pspace. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.
41. Jeremy Spinrad. Recognition of circle graphs. *Journal of Algorithms*, 16(2):264–282, 1994.
42. Alexander Tiskin. Fast distance multiplication of unit-monge matrices. *Algorithmica*, 71(4):859–888, 2015.
43. Kazuaki Yamazaki, Toshiki Saitoh, Masashi Kiyomi, and Ryuhei Uehara. Enumeration of nonisomorphic interval graphs and nonisomorphic permutation graphs. *Theoretical Computer Science*, 806:310–322, 2020.