UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

ESTUDIO DE PROTOCOLOS DISTRIBUIDOS INTERACTIVOS SOBRE CLASES DE GRAFOS HEREDITARIOS.

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO

BENJAMÍN ANTONIO JÁUREGUI FLORES

PROFESOR GUÍA:
IVÁN RAPAPORT ZIMERMANN
PROFESOR GUÍA 2:
PEDRO MONTEALEGRE BARBA

COMISIÓN:
MARTÍN MATAMALA VÁSQUEZ

Este trabajo ha sido parcialmente financiado por Proyecto Fondecyt 11190482, CMM ANID PIA AFB170001, CMM ANID BASAL ACE210010 y CMM ANID BASAL FB210005

SANTIAGO DE CHILE
2022

Resumen

ESTUDIO DE PROTOCOLOS DISTRIBUIDOS INTERACTIVOS SOBRE CLASES DE GRAFOS HEREDITARIOS.

En el presente trabajo se estudian protocolos distribuidos para el reconocimiento de ciertas clases de grafos geométricos. En estos protocolos, existe un probador con poder ilimitado pero no confiable, llamado Merlín, que le entrega mensajes a los nodos de un grafo, buscando convencerlos de que el grafo cumple un cierto predicado. Posterior a que Merlín entrega los mensajes, los nodos pueden entrar de inmediato en la etapa de verificación, durante la cual intercambian los mensajes que poseen con sus vecinos y, cada uno, de forma local, decide con esta colección de mensajes si acepta o rechaza. Se debe cumplir que si el grafo satisface el predicado, entonces existe una asignación de mensajes de Merlín a los nodos que hace que todos acepten. Si no cumple el predicado, para cualquier colección de mensajes entregados por Merlín, al menos un nodo debe rechazar. Este modelo, donde inmediatamente después de recibir los mensajes dados por Merlín los nodos entran en la etapa de verificación, es el modelo llamado *proof labeling schemes*. Si se permite que haya más rondas de interacción entre los nodos y Merlín, al modelo lo llamamos *protocolos interactivos distribuidos*.

El primer capítulo se dedica a definir toda la teoría necesaria de Teoría de Grafos y algunos resultados de Teoría de Números, junto con protocolos distribuidos para resolver ciertos problemas particulares que son usados como subrutinas en los protocolos distribuidos aquí diseñados. También se introducen las clases de grafos geométricos de intersección que son estudiados en este trabajo: Grafos de permutación, trapezoidales, circulares, k-circulares-poligonales y cuadrado unitario.

En el segundo capítulo, se estudian los grafos de permutación y trapezoidales, donde se da una caracterización de ellos y un protocolo (*proof labeling scheme*) para reconocerlos.

En el tercer capítulo se introducen y resuelven de forma distribuida dos problemas independientes, llamados CORRESPONDING ORDER y LINEAR ASSIGNATION, que permiten compartir información verificable entre nodos que no son vecinos.

En el cuarto capítulo se estudian las clases de grafos circulares, k-circulares-poligonales y de cuadrado-unitario. Para estas tres clases se presentan algoritmos distribuidos interactivos que necesitan de 3 rondas de interacción alternada entre Merlín y los nodos para reconocerlas.

Finalmente, en el quinto capítulo se prueba que el largo de los mensajes que son necesarios para reconocer cualquiera de las clases estudiadas en este trabajo es de $\Omega(\log n)$.

THE STUDY OF INTERACTIVE DISTRIBUTED PROTOCOLS ON HEREDITARY GRAPH CLASSES.

In the present work distributed protocols for the recognition of certain classes of geometric graphs are studied. In these protocols, there is an unreliable prover with unlimited computational power, called Merlin, who sends messages to the nodes of a graph, seeking to convince them that the graph meets a certain predicate. Once Merlin sends the messages, the nodes can immediately enter the verification stage, during which they exchange the information they have with their neighbors and each node locally decides with this information whether to accept or reject. It must be true that if the graph satisfies the predicate, then there is a Merlin assignment of messages to nodes that makes them all accept. If it does not satisfy the predicate, for any information delivered by Merlin, at least one node must reject (that is, Merlin cannot fool the nodes). This model, where immediately after receiving the information given by Merlin the nodes enter the verification stage, is the *proof labeling schemes* model. If we allow more rounds of interaction between the nodes and Merlin, where the information that nodes send to Merlin are random bits, the model is called *distributed interactive protocols.*

In the first chapter of the present thesis we recall some well-known notions of Graph Theory and some results of Theory of Numbers, together with some distributed protocols (interactive and non-interactive) that solve particular problems that are used as subroutines in our distributed protocols. The classes of geometric intersection graphs that are studied in this work are also introduced: permutation, trapezoid, circle, k-polygon circle and unit-square graphs.

In the second chapter, permutation and trapezoid graphs are studied, and we obtain a characterization that allows us to give an algorithm with only one round of interaction (*proof labeling scheme*) to recognize (separately) both classes of graphs, with messages of maximum length $\mathcal{O}(\log n)$.

In the third chapter, two independent problems, called Corresponding Order and Linear Assignation, are introduced and solved in a distributed way. These algorithms have a relevance beyond the application given in this work, since they allow messages to be shared between pairs of nodes that are not neighbors, being able to verify that both received the correct messages.

In the fourth chapter, the classes of circle, k-polygon circle and unit-square graphs are studied. For these three classes the interactive distributed algorithms which are introduced use 3 rounds of interaction between Merlin and the nodes to recognize them, with a maximum message length of $\mathcal{O}(\log n)$ bits.

Finally, in the fifth chapter, $\Omega(\log n)$ lower bounds are presented in the number of bits necessary to recognize any of the classes studied in this work. This implies that the algorithms presented in Chapter 2 are optimal in terms of shared information, and the protocols in Chapter 4 are optimal in terms of shared information, and they could only be improved by reducing the number of interaction rounds.

Para mi mamá, papá y hermana.

Acknowledgments

Primero que todo, agraceder a mis profesores guías Ivan Rapaport y Pedro Montealegre por su gran ayuda durante todo el proceso. Gracias por la ayuda para avanzar en el desarrollo de mi investigación, las gratas conversaciones y relación que se tuvo durante el proceso. También agradecer al profesor Martin Matamala por su dedicación en la lectura de mi tesis, que produjo que los que leerán a continuación (espero) este mejor redactado.

Agradecer también a los profesores y funcionarios del DIM. Especialmente al profesor Marcos Kiwi con quien trabaje como a en diversos ramos, y a Karen y Silvia, por ayudarme con mis dudas y los (no pocos) errores burocraticos que cometi en el camino.

También, agradecer por su enorme apoyo y cariño a mi familia. Gracias a mi madre por el amor y preocupación eterna, por apoyarme siempre, levantarme en momentos dificiles, y ser ejemplo de superación y resiliencia. Gracias a mi padre por su amor y dispocisión infinita a ayudarme, por irme a buscar a la hora que sea, y emocionarse tanto con cada logro que consigo. A mi hermana por su amor, ser una segunda madre y amiga. Por apoyarme, guiarme y ser un ejemplo. Gracias a los tres por creer en mi cuando a veces ni yo lo hago, y por apoyarme en cada decisión que tomo. También agradecer a todas mis mascotas por alegrarme la vida. Especialmente al Boby, Chiqui y Donald por acompañarme en tantas noches de estudio.

Gracias a mis amigos y grandes personas que conocí a lo largo de mi paso por la Universidad. Mis primeros en la Universidad, Caro, Javier, Jota, Carlos y Pipin. A mis amigos del DIM Gonzalo, Manu, Mariano, Pedro, Fabian, la oficina 435 y tantos otros por el aguante, los momentos de estudio, y sobre todo los momentos de distensión a lo largo de la ardua travesía del DIM. También agradecer a Yorsh por las siempre gratas (y duras) conversaciones de la vida.

A mis amigos de toda vida Pelao, Andy, Chino, Nicky y Chelbi, por la gran amistad que hemos forjado a lo largo de los años, y las risas interminables en cada junta. Cada momento dificil fue mas ameno después de juntarme con ustedes.

Finalmente, agradecer a mi querido Colo Colo, por las alegrías y las penas en todos estos años. Gracias por los buenos momentos que tanta felicidad me dan, y por los malos momentos que me permitían descargar la frustración en el estadio (o frente a un televisor) durante 90 minutos.

Table Of Content

1.1 Distributed Model 5

1.2 Geometric Intersection Graph Classes 6

1.3 Toolbox 10

1.3.1 Spanning Tree and Related Problems 10

1.3.2 Problems Equality and Permutation 12

2 Proof Labeling Schemes **15**

2.1 Permutation Graphs 15

2.2 Trapezoid Graphs 20

3 Distributed Interactive Protocols **25**

3.1 Corresponding Order Problem 25

3.2 Linear Assignation Problem 28

4 Three-round Distributed Interactive Protocols **30**

4.1 Circle Graphs 30

4.2 Polygon Circle Graphs 33

4.3 Unit-Square Graphs 37

5 Lower Bounds **49**

5.1 Permutation and Trapezoid Graphs 49

5.2 Circle And k-polygon-circle Graphs 51

5.3 Unit Square Graph 53

Bibliography **66**

List of Figures

1.1 An example of a permutation graph with its corresponding intersection model. 7

1.2 An example of a circle graph with its corresponding intersection model. . . . 7

1.3 An example of a permutation graph with its corresponding intersection model. 8

1.4 An example of a 3-polygon circle graph with its corresponding intersection model. 8

1.5 An example of a unit square graph with its corresponding intersection model. 9

2.1 Each trapezoid T is defined by the set $T = \{b_1, b_2, t_1, t_2\}$. 20

2.2 Representation of the two possible intersection cases. 21

4.1 Example of a graph G in the left hand, and a proper unit-square model in the right hand. 37

4.2 Left: A unit square graph G. Right: A nice unit square model of G 39

4.3 40

4.4 Unit square model of K_4 41

4.6 Four orders of Figure 4.4. 41

4.7 To the left, a path with 5 nodes and to the right an invalid unit square model 42

4.8 Invalid model for the path of 5 nodes in Figure 4.7 43

5.1 Graph Q_3 and a permutation model for Q_3. 50

5.2 Graph $\sigma_{\bowtie}^{12}(Q_3)$, 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 4. 50

5.3 width=0.2 51

5.4 Graph $\sigma_{\bowtie}^{i,i+1}(M_4)$, where in red are represented the crossing edges. 51

5.5 Graph S_1 53

5.8 Sets H_1 and H_2 in S_2 54

Introduction

Distributed Computing seeks to study scenarios in which the participants of a network, characterized as nodes of a graph, must carry out a certain task, or make a decision regarding a global property of the network (for example, if there is a certain number of participants all connected to each other). The key point is that, for these tasks, the nodes only have local information that they can share with their network neighbors (those with whom they are connected) to carry out, each of them, the specified task.

Specifically, in the tasks that correspond to decision algorithms, the nodes must decide whether or not the network satisfies a certain predicate, and for this, as each node only has local information (for example, the number of neighbors it has) together with information delivered by its neighbors in a certain number of rounds, it must decide locally whether to ACCEPT or REJECT. To be correct, distributed decision algorithms must satisfy that, if the graph satisfies the property being studied, then all the nodes of the graph must accept; and, if it does not, then at least one node must reject.

The local nature of the information known by each participant brings an inherent problem: For some cases it is necessary for two nodes whose distance in the network is very large to share information with each other. For instance, in order to determine if the network (seen as a graph) has a cycle of odd size. This means that, despite the fact that the number of messages exchanged in each round is small, many interaction rounds are needed in order to verify whether or not the network meets the proposed predicate.

One way to measure the efficiency of distributed algorithms is to study the maximum amount of bits that needs to be exchanged in each round by the nodes, called the *proof-size* of the algorithm. In this sense, in recent years a distributed algorithm model called *proof labeling schemes* (PLS) has been proposed, which correspond to distributed decision algorithms in which the nodes first receive messages delivered by an all-powerful but untrusted prover called *Merlin*, and then perform a round where they exchange messages with their neighbors. Finally, each node makes a decision of where to ACCEPT or REJECT . The *proof-size* of the algorithms in this model corresponds to the maximum number of bits that needs to be exchanged (by Merlin or between the nodes) to carry out the decision. An example of this model may be to form a valid spanning tree. To do this, Merlin communicates to each node the root $r \in V$ of the spanning tree, as well as who is its parent and the distance to the root. With this information, the nodes exchange messages between themselves and verify that they all were told that the same node r was the root, in addition to verifying that if its distance from the root is d, then the distance from its parent to the root is $d - 1$. If all nodes check this information, then the spanning tree proposed by Merlin is valid and all nodes accept; if not, the model is invalid and at least one node rejects.

The fact that there is an entity with unlimited computing power is not a problem for the model, since these models are inherently algorithms of verification, so the aim is that, regardless of the messages delivered by Merlin, he is not able to fool the nodes if the graph does not satisfy the property. In this sense, this round between the nodes can be understood as the distributed version of the NP class, because it is known that the NP class is equal to the class of languages which have a deterministic interactive proof (dIP class).

There are some problems for which lower bounds on the proof-size of proof labeling schemes have been found. For example, any PLS solving the symmetry problem, 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) requires a proof-size of $\Omega(n^2)$ [76]. To study how many rounds of interaction affects the proof-size necessary to solve certain problems, the notion of distributed interactive protocols was introduced in [99] and subsequently studied in [35, 62, 126, 128]. Specifically, this model seeks to study protocols where, by allowing the interaction between the nodes and Merlin, the proof-size is significantly reduced . In the case of symmetry problem, if three rounds of interaction between the nodes and Merlin are allowed, the proof-size is reduced to $\mathcal{O}(\log n)$ bits [99], which represents an exponential optimization compared with the proof-size needed in the PLS model.

In this work we are interested in developing distributed decision algorithms to decide if a graph is in some geometric intersection graph class. 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. The geometric intersection class have all the graphs defined by the same geometric object. 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. 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. [52, 53, 19]

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, polygon-circle and unit square. More precisely, for the two first classes proof labeling schemes are given to recognizing them with logarithmic proof-size. For the other two classes, three-round distributed interactive protocols with $\mathcal{O}(\log n)$ proof-size are presented. Finally, logarithmic lower-bounds on the proof-size of any proof labeling schemes recognizing any of the aforementioned geometric intersection graph classes are given.

Organization

In Chapter 1 we present the basic definitions related to graph theory needed in this work as well as the formal definition of the distributed models, along with some known and important distributed result used as sub-routines in some of the results presented later.

In Chapter 2 the proof-labeling schemes to recognize the classes of permutation and trapezoid graphs are presented. For this, first we present some structural results related to this classes that are used later to present the distributed protocols. Finally, the correctness of algorithms are proved.

In Chapter 3, two important distributed interactive protocols are presented, the Corresponding Order Problem and Linear Assignation Problem. This two problem are used later in the recognition of the remainder geometric intersection classes, but they have have their own value because they represent two specific problems related to the order and how information is transmitted in a distributed network that appear inherently in many problems when they seek to be solved in a distributed model.

In Chapter 4, we present distributed interactive protocols to recognize the class of circle, polygon-circle and unit square graphs. Each of them needs three rounds of interaction between the nodes and *Merlin*. For the first two of them, we first present some structural characterization to be used later in the protocols. For the recognition of unit square graphs, it is necessary to first define some characterizations of the square model used in the definition of the class, with the idea to define a more appropriate language to later present the protocol.

Finally, in Chapter 5 we present lower bounds to the size proof of each class studied in this work in the proof-labeling scheme model, which give us that the PLS presented in Chapter 2 are optimal respect the proof-size of them. Also, this gives a clue that the only improvement in the algorithms presented in Chapter 4 are respect the number of round needed, but no regarding the proof-size. The technique used to achieve this lower bounds relieve in a useful theorem to get lower bounds, where the main work is in create an appropriate graph.

Results

The results obtained in the present work can be summarized in the following table.

Class	Rounds	Proof-size
PERMUTATION	1	$\mathcal{O}(\log n)$
TRAPEZOID	1	$\mathcal{O}(\log n)$
CIRCLE	3	$\mathcal{O}(\log n)$
POLYGON-CIRCLE	3	$\mathcal{O}(\log n)$
UNIT-SQUARE	3	$\mathcal{O}(\log n)$

Also, regarding lower bounds, we get, for the proof labeling scheme model, a lower bound on the proof-size of $\mathcal{O}(\log n)$ bits.

Preliminaries

A simple and undirected *graph* is a pair $G = (V_G, E_G)$ of sets such that $E \subseteq \{\{u,v\}\}_{u,v\in V}$, where the elements of V are called the *nodes* of the *graph* and the elements of E the *edges*. To avoid heavy notation, the edges $e \in E$ with $e = \{u,v\}$ will be denoted simply as $e = uv$[1]. If the graph G is obvious by context we omit the subindex and we simply denote $G = (V,E)$. A graph $G = (V,E)$ is complete if for all $u, v \in V$, $uv \in E$.

Two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are *isomorphic* if there exists a one-to-one function $f\colon V_G \to V_H$ such that for all $u, v \in V_G$, $uv \in E_G$ iff $f(u)f(v) \in E_H$.

Given a graph $G = (V,E)$, we define the open neighbor of a node v as $N(v) = \{u \in V\colon uv \in E\}$. The closed neighbor is denoted as $N[v]$ and correspond to $N(v) \cup \{v\}$. The degree of v is the size of $N(v)$ and it is denoted as $d(v)$.

Given two graphs $H = (V_H, E_H)$ and $G = (V_G, E_G)$, we say H is *subgraph* of G when $V_H \subseteq V_G$ and $E_G \subseteq E_G$. In the same way, given a subset of nodes $U \subseteq V$, the induced subgraph given by U is $(U, E(U))$, where $E(U)$ are the edges with both nodes in U.

Given a graph $G = (V,E)$, a *path* $P = (V_P, E_P)$ in G is a subgraph of G such that there exists an enumeration of the nodes $V_P = \{v_1, ..., v_k\}$ and $E_P = \{\{v_i, v_{i+1}\}\colon i \in [k-1]\}$. The nodes v_1 and v_k are called the *endpoints* of the *path*. The graph G is said to be connected if there exists a path between every pair of nodes.

A cycle $C = (V,E)$ is a graph such that C is a path and the two endpoints are connected, i.e., $v_1 v_k \in E$. Given a graph T, we say T is a forest iff T does not have any cycle as subgraph. If T is also connected, then we say that T is a *tree*. A clique of a graph $G = (V,E)$ is a subset of vertices $C \subseteq V$ such that every two distinct vertices in the clique are adjacent. That is, the induced subgraph by C is complete.

In the rest of the work, we always assume that the input graph is connected. For more details about graph definitions we refer to the textbook of Diestel [43].

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 $G = ([n], E)$.

On other side, given $n, m \in \mathbb{N}$ with $n < m$, we denote as $\mathbb{Z}_n$ the set $\{0, ..., n-1\}$, $[n]$ the set of integers $\{1, ..., n\}$, $[n,m]_{\mathbb{N}} = [n,m] \cap \mathbb{N}$ (where $[n,m]$ is the standard real interval) and S_n the set of permutations of $[n]$.

[1]Notice that as the graph is undirected, then $e = uv = vu$

1.1 Distributed Model

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 are 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).

A *proof labeling scheme* (or just PLS) is a distributed algorithm in which there is an initial round used by the prover Merlin to give a certificate of information $c(v)$ to each node $v \in V$. After this, there is a *verification round* where the neighbors u, v can interchange information $c_{v,u}$ (that can be their id, label (v) and/or certificate $c(v)$) between neighbors, and then each of them decides if accepts or rejects.

The *proof-size* of a PLS is $\mathcal{O}(f(n))$ if the size of the largest message interchanged (including the certificates given by Merlin and the messages shared between neighbors) in a n-node graph G is bounded by a constant factor of $f(n)$.

A correct distributed algorithm has to satisfied two conditions.

Completeness. If the configuration (G, id, I) is in the distributed language $\mathcal{L}$, then there exist certificates $\{c(v)\}_{v \in V}$´that make all the nodes accept.

Soundness. If the configuration (G, id, I) is not in the distributed language $\mathcal{L}$, then for any collection of certificates $\{c(v)\}_{v \in V}$ given by Merlin, at least one node rejects.

Distributed Interactive protocols are an extension of PLS and have two phases: an interactive phase and a verification phase. If Arthur is the one which starts the interactive phase, he picks a random string $r_1(v)$ at each node v of G 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 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 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 that each node broadcasts to its neighbors because there is no randomness in this last verification round.

Definition 1.1 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, with proof-size $\mathcal{O}(f(n))$, we write $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$.

Definition 1.2 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 $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$, and

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

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.

Note that the definition of Completeness and Soundness changes in the interactive model because in this case the algorithms are probabilistic, and then one is looking for a very small probability of fail the original conditions.

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).

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 [6, 58, 59, 90, 116, 105, 146].

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 1.1 we show an example of a permutation graph.

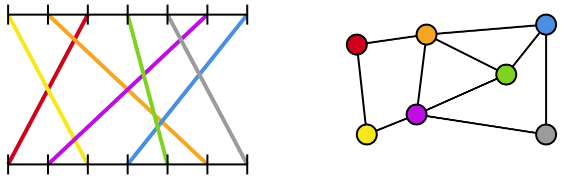


Figure 1.1: An example of a permutation graph with its corresponding intersection model.

Although the class of interval graphs is quite restrictive, there are a number of practical applications and specialized algorithms for interval graphs [74, 80, 101]. 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 [16, 92].

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 [105] and, on the other hand, many NP-complete problems can be solved efficiently when the input is restricted to permutation graphs [28, 110]. A graph G is a *circle graph* if G is the intersection model of a collection of chords in a circle (see Figure 1.2). 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 intersects every other chord. Circle graphs can be recognized in time $\mathcal{O}(n^2)$ [142]. Many NP-complete problems can be solve in polynomial time when restricted to circle graphs [97, 144].

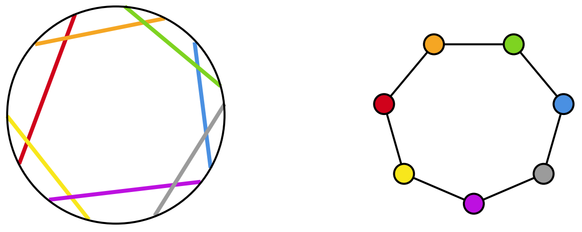


Figure 1.2: An example of a circle graph with its corresponding intersection model.

The 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 1.3). Ma and Spinrad [115] showed that trapezoid graphs can be recognized in $\mathcal{O}(n^2)$ time. Trapezoid graphs were applied in various contexts such as VLSI design [37] and bioinformatics [3]. Also, trapezoid and circle graphs are incomparable: the trapezoid graph of Figure 1.3 is not a circle graph, while the circle graph of Figure 1.2 is not a trapezoid graph.

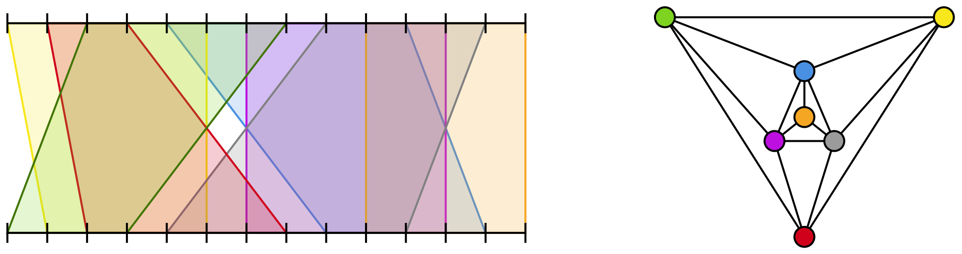


Figure 1.3: An example of a permutation graph with its corresponding intersection model.

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 1.4 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 1.4 is neither a trapezoid graph nor a circle graph.

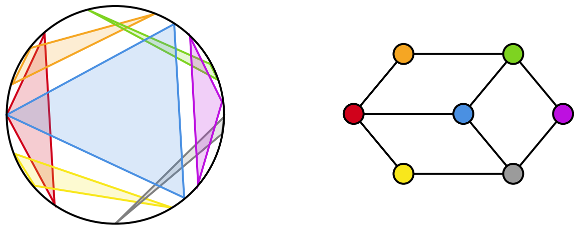


Figure 1.4: An example of a 3-polygon circle graph with its corresponding intersection model.

The problem of recognizing whether a graph is a k-polygon circle graphs, for any $k \geq 3$, is NP-complete [133]. Nevertheless, many NP-complete problems have polynomial time algorithms when restricted to polygon circle graphs [66, 67]. In an unpublished result, M. Fellows proved that the class of polygon circle graphs is closed under taking induced minors.

There is another natural generalization of interval graphs, which is based on the dimension. More precisely, since interval graphs are defined in one dimension (in the real line), they can be generalized to two (or more) dimensions. But this generalization is neither unique nor straightforward. For instance, intervals may be generalized to rectangles or they may be generalized to disks or squares.

In the particular case of unit interval graphs (intervals graphs where all the segments have the same size), these fixed-size intervals are typically generalized either to unit disks (if we consider the Euclidean norm), or to unit squares (if we consider the Manhattan norm). Unit disks and unit square are particularly relevant in the context of wireless networking, where servers are modeled as points in the plane and the range of wireless transmission is assumed to be constant [31, 123].

Many graph problems which are NP-complete become tractable when the input is restricted to unit disk graphs. For example, finding a maximum clique can be solved in polynomial time for unit disk graphs [30]. On the other hand, Breu and Kirkpatrick showed that the recognition of unit disk graphs is NP-hard [22]. Regrettably, is not known whether the recognition of unit disk graphs is in NP. A naive certificate for the recognition problem would be the coordinates of the centers of the corresponding disks. Nevertheless, in some cases, the number of bits required to represent these coordinates is far too big. In fact, McDiramid and Müller showed in [120] the existence of a family of unit disk graphs on n vertices for which, in any realization, the center of at least one disk must be encoded with $2^{2^{\mathcal{O}(n)}}$ bits.

Going from unit disks graphs to *unit square graphs* removes geometric intricacies and tends to simplify the structure of graphs and its representation maintaining several key aspects.A unit-square graph corresponds to the intersection model of a set of squares in the plane, all of the same size. See Figure 4.1 for an example.

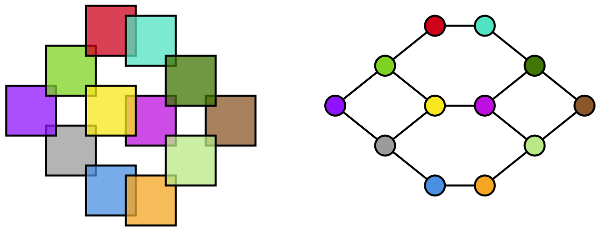


Figure 1.5: An example of a unit square graph with its corresponding intersection model.

The recognition of unit square graphs is also NP-hard. The proof is a simple adaptation of the one given by Breu and Kirkpatrick for unit disk graphs [22]. Nevertheless, in the case of unit square graphs, in contrast with unit disk graphs, the recognition problem is known to be in NP. In fact, in [127] the authors prove that any unit square graph can be represented with all the centers of the squares specified by rational coordinates of at most $\mathcal{O}(\log(n))$ bits. Hence, the recognition of unit square graphs is indeed NP-complete.

Just as it happens with unit disk graphs, restricting the input to unit square graphs often simplifies many graph problems, allowing the existence of efficient algorithms. For instance, Neuen proved that the graph isomorphism problem for unit square graphs can be solved in polynomial time [130].

This motivation takes us in this work to study and design distributed algorithms to verify if a given graph $G = (V, E)$ is some of these geometric graphs. Formally, we are interested in recognize the following distributed algorithms

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 cicle graph$\}$.
k-Polygon-Circle-Recognition $= \{\langle G, \mathsf{id}\rangle$ s.t. G is a k-polygon-circle graph$\}$.
Unit-Square-Recognition $= \{\langle G, \mathsf{id}\rangle$ s.t. G is a unit-square graph$\}$.

The five graph classes studied in this work are *hereditary*. A graph class is hereditary it is closed under induced subgraphs.

1.3 Toolbox

In our results we use some previously defined and studied 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).

But before describing these distributed problems, we present a result regarding density of prime numbers, which is necessary to allow us describe more compact protocols in the distributed interactive setting.

Theorem 1.3 (Bertrand's Postulate) For any integer n bigger than 1, there exists a prime number p such that $n \leq p \leq 2n$.

Also, in some protocols we want to shrink the proof-size, and in order to do that, we use hash functions to compact the messages interchanges in a way collision occurs with very small probability.

Theorem 1.4 Let $m, p \in \mathbb{N}$, where p is prime. There exists a family of functions $h\colon \{0,1\}^m \to \mathbb{Z}_p$ of size $|\mathcal{H}| = p$ such that for $x, y \in \{0,1\}^m$, we have the following properties.

(1) *Linearity:* $h(x+y) = h(x) + h(y) \mod p$.

(2) *Small collision probability:* If $x \neq y$, then

$$\mathbb{P}_{h\in\mathcal{H}}\left(h(x) = h(y)\right) \leq \frac{m}{p}$$

1.3.1 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 1.5 There is a 1-round protocol for Spanning-Tree with certificates of size $\mathcal{O}(\log n)$.

The following Protocol allows the nodes, helped by Merlin, to construct a valid spanning tree.

Protocol 1.6 First, Merlin gives to each node $v \in V$ the following information

- The root's identifier $r \in V$ of the spanning tree.
- The identifier p_v of it's father in the tree.

- Its distance $d(v)$ and the distance of its father $d(p_v)$ to the root r.

Then, in the verification round, each node $v \in V$ verifies

- All of them receives the same root $r \in V$.
- Its father $p(v)$ is its neighbor.
- If $d(p_v) = k$, then $d(v) = k + 1$.

Each node accepts only if the three conditions are satisfied. If not, rejects.

Now we analyze the Correctness and Soundness of the Protocol.

Correctness. An honest prover provides an unique root and the correct distances in the three so all nodes accept.

Soundness. If Merlin gives two or more different roots, then the nodes reject because there are two neighbors u, v with different root node. If the three given by Merlin forms a cycle, then there exist two nodes such that u is father of v but $d(v) < d(v)$, then v rejects. Then the three constructed by Merlin has to be correct and thus the distance too.

Proof-size analysis. The identifiers of the node can be encoded in $\mathcal{O}(\log n)$ and the maximum distance in a n-node graph between two nodes is $n - 1$, thus the distances $d(v)$ can also be encoded in $\mathcal{O}(\log n)$.

From the protocol of Proposition 1.5 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 we are assuming the nodes are only aware of a polynomial upper bound on $n = |V|$).

Proposition 1.7 [104] There is a 1-round protocol for SIZE with certificates of size $\mathcal{O}(\log n)$.

Protocol 1.8 In the first round, Merlin gives to each node $v \in V$ a certificate with the following information

- The information needed according to Protocol 1.6 to construct a valid spanning tree T
- The number of nodes c_v in T_v, the T-subtree rooted in v.

Then, in the verification round each node $v \in V$ validate that the spanning tree constructed is correct according to verification round of Protocol 1.6 and that

$$c_v = 1 + \sum_{\substack{\omega \text{ children} \\ \text{of } v}} c_\omega$$

. The root r of the tree also checks that $c_r = n$. If the verification goes well then the node accepts and if one of them fails, reject.

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 or not; and, if it belongs to the path, it has to know which of its neighbors are its predecessor and successor in P.

Proposition 1.9 [104] There is a 1-round protocol for $s, t -$ Path with certificates of size $\mathcal{O}(\log n)$.

1.3.2 Problems Equality and Permutation

A second important building block to be used in interactive protocols, is a protocol to solve problem Equality, which it is defined as follows. Given G a connected n-node graph, each node v receives a pair of lists $\{a_v^i\}_{i\in[\ell]}$ and $\{b_v^i\}_{i\in[\ell]}$, with $\ell < n$ and each value a_v^i, b_v^i can be encoded with $\mathcal{O}(log(n))$ bits. The problem Equality consists of verifying whether the multi-sets $\mathcal{A} = \{a_v^i \colon v \in V, i \in [[\ell]]\}$ and $\mathcal{B} = \{b_v^i \colon v \in V, \imath \in [\ell]\}$ are equal.

Proposition 1.10 [128] Problem Equality belongs to $\mathsf{dAM}_{1/3}[\log n]$.

Protocol 1.11 The following two-round protocol resolves Equality using $\mathcal{O}(\log n)$ bits.

Given a value a which representation has $c \log n$ bits can thought as a member of a finite field $\mathbb{F}_q$ with $n^{c+3} \leq q \leq 2n^{c+3}$. Now, given the multi-sets $\mathcal{A}$ and $\mathcal{B}$, we define the polynomials $p_{\mathcal{A}}(x)$ and $p_{\mathcal{B}}(x)$ as follow

$$p_{\mathcal{A}}(x) = \prod_{a\in\mathcal{A}} (a - x), \qquad p_{\mathcal{B}}(x) = \prod_{b\in\mathcal{B}} (b - x)$$

Since $\mathcal{A}$ and $\mathcal{B}$ have at most $\ell \cdot n$ elements, $p_{\mathcal{A}}(x)$ and $p_{\mathcal{B}}(x)$ are polynomial of degree at most $\ell \cdot n \leq n^2$. Whit this, $p_{\mathcal{A}}$ and $p_{\mathcal{B}}$ can coincide in at most n^2 points (unless they are equal).

Then, if $\mathcal{A} \neq \mathcal{B}$, by taking an element $s \in \mathbb{F}_q$ uniformly at random, the probability that $p_{\mathcal{A}}(s) = p_{\mathcal{B}}(s)$ is at most $\dfrac{1}{n}$.

As the nodes don't have shared randomness, we simulate in the following way.

In the first round, each node u sample s_u at random, along with a random number $\alpha_u \in [n^2]$ and sends them to Merlin.

Now, in the second round, Merlin first answer a pair (s, α) supposed to be a pair (s_u, α_u) send it by a node with minimal α_u. Also, sends the information needed according to protocol 1.6 to construct a valid spanning tree with root u^*. Finally, the prover sends each to node u the evaluation in the value $x = s$ of the polynomials on the subtree T_u,

$$A_u^s = \prod_{\omega\in T_u, i\in[\ell]} (s - a_{\omega,i}), \qquad B_u^s = \prod_{\omega\in T_u, i\in[\ell]} (s - b_{\omega,i})$$

Finally, in the verification round the nodes verify

1. The correctness of the spanning tree given by Merlin according to Protocol 1.6.
2. The existence of a node v that sent (s, α_s) and the correctness of α_s.
3. Considering C_v to be the set of children of v at T, it holds that
$$A_u^s = \prod_{\omega \in C_v} A_\omega^s \times \prod_{i \in [\ell]} (a_v^i - s), \qquad B_u^s = \prod_{\omega \in C_v} B_\omega^s \times \prod_{i \in [\ell]} (b_v^i - s)$$
If $C_v = \emptyset$ (v is a leaf) then simply verifies $A_u^s = \prod_{i \in [\ell]}(a_v^i - s)$ and $B_u^s = \prod_{i \in [\ell]}(b_v^i - s)$.
4. If $v = u*$ is the root of the spanning tree, then also check that $A_{u^*}^s = B_{u^*}^s$.

If the above is verify, then the node accepts. Else, rejects.

- **Correctness.** The protocol succeeds as long as α_s is uniquely minimal value. However, it is easy to see that $\mathbb{P}\left(\exists i \neq j \colon \alpha_i = \alpha_j\right) \leq \dfrac{1}{n}$. Thus, w.l.o.g. we assume all α_i's are different.

  If $\mathcal{A} = \mathcal{B}$ as multisets, then for any $s \in \mathbb{F}_q$ it holds that $p_{\mathcal{A}}(s) = p_{\mathcal{B}}(s)$. So for any tree T with root u^* and any value $s \in \mathbb{F}_q$ it holds that $A_{u^*}^s = B_{u^*}^s$.

  For this, if $\mathcal{A} = \mathcal{B}$, an honest prover just has to send the correct information which needs to deliver and then all the nodes accept with probability $\geq \dfrac{1}{3}$.

- **Soundness.** Assume that $\mathcal{A} \neq \mathcal{B}$. Suppose that $p_{\mathcal{A}}(s) \neq p_{\mathcal{B}}(s)$. In order for the prover to cheat, it must gives to the root u^* values $\hat{A}_{u^*}^s, \hat{B}_{u^*}^s$ such that $\hat{A}_{u^*}^s \neq p_{\mathcal{A}}(s)$ or $\hat{B}_{u^*}^s \neq p_{\mathcal{B}}(s)$, since otherwise u^* verifies that $p_{\mathcal{A}}(s) \neq p_{\mathcal{B}}(s)$ and it rejects. However, since the root r performs a local verification of that values with its neighbors in the tree, it must hold that the prover gaves wrong values to one of its children as well, lets say this child is u. This argument continues with one of the children of u and then, iterating the argument, it must hold that a leaf of the tree receives a wrong value by Merlin, and this leaf can verify locally and output rejects because it checks that one equality of 3. of the verification round doesn't hold.

  Then, we bound the probability that two products collide. Consider the polynomial $f(x) = p_{\mathcal{A}}(s) - p_{\mathcal{B}}(s)$, which has degree at most n over the field $\mathbb{F}$.

  Claim 1.12 f is not the zero polynomial.

  Proof. Remember we know that $\mathcal{A} \neq \mathcal{B}$. Suppose there exists an element $z \in \mathcal{A} \backslash \mathcal{B}$. Then, we get that $p_{\mathcal{A}}(z) = 0$ and $p_{\lfloor}(z) \neq 0$, therefore $f(z) \neq 0$ and thus f is not the zero polynomial. The same argument holds if $z \in \mathcal{B} \backslash \mathcal{A}$. Since $\mathcal{A}$ and $\mathcal{B}$ are multisets, there is a third possibility that the multisets share the same elements only with different multiplicities. Let $C = \mathcal{A} \cap \mathcal{B}$ and define
$$g(x) = f(x) / \prod_{c \in \mathcal{C}} (c - x)$$

It is enough to show that g is not the zero function. Define $\mathcal{A}' = \mathcal{A} \backslash \mathcal{C}$ and $\mathcal{B}' = \mathcal{B} \backslash \mathcal{C}$. We know for these subsets that there must be an element that is in one set and not in other. Assume without loss of generality that there exists an element $z \in \mathcal{A}' \backslash \mathcal{B}'$. Since $z \notin \mathcal{C}$ we get that $g(z) \neq 0$ and therefore f is not the zero polynomial. $\square$

The polynomial f has at most $n\ell \leq n^2$ roots and since the field $\mathcal{F}$ is of size n^{c+3} we get that

$$\mathbb{P}_r \left[p_{\mathcal{A}}(s) = p_{\mathcal{B}}(s) \right] = \mathbb{P}_r \left[f(s) = 0 \right] \leq \frac{n\ell}{n^{c+3}} \leq \frac{1}{n}$$

And then, for $n \geq 3$ we have that the probability of a collision between $p_{\mathcal{A}}(s)$ and $p_{\mathcal{B}}(s)$, and thus the probability of have all the nodes answering accept , is less than $\frac{1}{3}$.

Proof-size analysis. Construct a valid spanning three can be done with a proof-size of $\mathcal{O}(\log n)$ bits. The computation of A_u^s and B_u^s can be done with the local messages $\{a_{\omega,i}\}_{\omega \in N(v), i \in [\ell]}$, and as ℓ is fixed, every node just share its values $\{a_u^i\}_{i \in \ell}$ with its neighbors, which can be done with $\mathcal{O}(\log n)$ bits. Then, the overall messages shared have a size of $\mathcal{O}(\log n)$.

A closely related problem is PERMUTATION, where a graph $G = ([n], E)$ receives some function π by an external entity, and the nodes must verify whether π is indeed a permutation (a bijective function from $[n]$ 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 proof-size $\mathcal{O}(\log n)$.

Proposition 1.13 [128] Problem PERMUTATION belongs to $\mathsf{dMAM}_{1/3}[\log n]$.

Protocol 1.14 The following three-round protocol resolves PERMUTATION.

In the first round, Merlin sends to each node i its image $\pi(i)$ and a value y_i supposed to be equal to $\pi(i + 1 \mod p)$. Also gives them the number of nodes n.

In the second and third round the nodes and Merlin interacts according to Protocol 1.11 to verify if $\mathcal{A} = \{\pi(i) \colon i \in [n]\}$ and $\mathcal{B} = \{y_i \colon i \in [n]\}$ are equal as multi-sets.

In the verification round the nodes interact according to Protocol 1.11 and verifies the number node n is correct according to verification round of Protocol 1.8.

The Completeness and Soundness of Protocol 1.14 is inherited by Protocol 1.8 and Protocol 1.11.

Chapter 2

Proof Labeling Schemes

In this chapter we give PLS to recognize the class of permutation and trapezoid graphs. Both classes are studied in similar way. First, we give some characterization of the class nice to work in a distributed setting, and then, using as main tool this characterizations, we give a PLS to recognize each class.

2.1 Permutation Graphs

We begin this section with a characterization of permutation graphs. Then, we use this characterization to prove the existence of a PLS solving the problem of recognize Permutation Graphs.

Given $\pi \in S_n$, we say that a pair $i, j \in [n]$ is an *inversion under* π if

$$(i - j)(\pi(i) - \pi(j)) < 0.$$

The definition of a permutation graph can be restated as follows (see [20]).

Definition 2.1 A graph $G = ([n], E)$ is a permutation graph if there exists a permutation $\pi \in S_n$ such that for every $i, j \in [n]$, the pair $\{i, j\}$ is an edge of G, if and only i, j is an inversion under π.
In such a case we say that π is a *proper permutation model* of G.

Let us fix a graph $G = ([n], E)$. In order to verify if G is a permutation graph, the PLS proposed later has to be able of recognize if, given a permutation $\pi \in S_n$, it is satisfied Definition 2.1. This means that, (c1) if two nodes are neighbors, they have to form an inversion, and (c2) no-adjacent nodes don't have to form an inversion under π. Since nodes only receive local information from their neighbors, the most complicated part of the protocol is verify (c2), because two non adjacent nodes i, j can form an inversion but they can be far-away in graph distance.

To avoid heavy notation in order to work with (c2), we define a weaker notion of permutation model for a graph.

Definition 2.2 Given a graph $G = ([n], E)$, we say that a permutation $\pi \in S_n$ is a *semi-proper permutation model* of G if π satisfies that for every pair $i, j \in [n]$, if $\{i, j\}$ is an edge of G then i, j is an inversion under π.

Thus, a semi-proper permutation model only satisfied (c1), but no necessary (c2).

The following proposition and lemma characterize permutation graphs.

Proposition 2.3 Let $G = ([n], E)$ be a permutation graph, π a proper permutation model and let $\{i, j\} \in E$ such that $i < j$. The following two conditions holds

1. All values $s \in [\pi(j) + 1, \pi(i) - 1]_{\mathbb{N}}$ satisfy that $s = \pi(k)$, for some node k that is neighbor of i or j.
2. All values $h \in [i + 1, j - 1]_{\mathbb{N}}$ are neighbors of i or j.

Proof. First, let $s \in [\pi(j) + 1, \pi(i) - 1]_{\mathbb{N}}$. As π is a permutation, exists $k \in [n]$ such that $\pi(k) = s$. If k is no neighbor of i, then $k < i$, but then, as $i < j$ it also holds that $k < j$, so the pair j, k is an inversion under π and as π is a proper permutation model of G, then $\{k, j\} \in E$. This prove 1., and the proof for 2. is analogous. $\square$

We denote $N^+_G(i)$ and $N^-_G(i)$ the sets of neighbors of node i with, respectively, higher and fewer identifiers than i. Formally,

$$N^+_G(i) = \{j \in N_G(i) : j > i\}$$
$$N^-_G(i) = \{j \in N_G(i) : j < i\}$$

We also denote by $d^+(i)$ and $d^-(i)$ the number of neighbors of i with, respectively, higher and lower identifiers than i. Formally, $d^+(i) = |N^+_G(i)|$ and $d^-(i) = |N^-_G(i)|$. From previous definition is direct that $|N_G(i)| = d^+(i) + d^-(i)$.

Remark 2.4 Remember that we are only considering undirected graphs, so the notation $N^+(i)$, $N^-(i)$, $d^+(i)$ and $d^-(i)$ must not be confused with the in/out neighborhood or degrees used for directed graphs.

With this, we are ready to give a characterization of permutation graphs in terms of local conditions satisfied by the nodes.

Lemma 2.5 Let $G = ([n], E)$ be a graph and let π be a semi-proper permutation model of G. Then π is a proper permutation model of G if and only if, for every $i \in [n]$,

$$i + d^+(i) = \pi(i) + d^-(i).$$

Proof. Suppose first that G is a permutation graph and π is a proper permutation model for G. Given a node $i \in [n]$, observe that all neighbors $j \in N^+(i)$ satisfy $0 \leq \pi(j) < \pi(i)$. Otherwise, (i, j) would not be an inversion under π. Analogously, if $j \in N^-(i)$ then $\pi(i) < \pi(j) \leq n$. Hence, the pre-images of $\{1, 2, ..., \pi(i) - 1\}$ under π are the nodes in $N^+(i)$ and nodes in $[i-1] \setminus N^-(i)$. Therefore, $\pi(i)-1 = d^+(i)+(i-1)-d^-(i)$, from which the equality $i + d^+(i) = \pi(i) + d^-(i)$ is deduced.

Let us suppose that G is not a permutation graph. We show the existence of a node $i^* \in [n]$ such that $i^* + d^+(i^*) \neq \pi(i^*) + d^-(i^*)$. Remember that we are assuming that π is a semi-proper permutation model for G. Then, we have that necessarily there exists a pair $\{i,j\} \notin E$ such that i,j is an inversion under π. For a node $i \in [n]$, let us denote by $a^-(i)$ and $a^+(i)$ the number of nodes with, respectively, fewer and larger identifier than i, forming an inversion with i, but that are not adjacent to i. Formally:

$$a^+(i) = |\{j \in \{i+1,\dots,n\} : \{i,j\} \notin E \text{ and } \{i,j\} \text{ form an inversion under } \pi\}|$$

$$a^-(i) = |\{j \in \{1,\dots,i-1\} : \{i,j\} \notin E \text{ and } \{i,j\} \text{ form an inversion under } \pi\}|$$

The pre-images of the set $\{1,\dots,\pi(i)-1\}$ under π are: The $d^+(i)$ neighbors higher than i (because the nodes in $N^+(i)$ form an inversion with i, as π is semi-proper); the $a^+(i)$ nodes that are not neighbors of i but form an inversion with i; and the nodes in set $\{1,\dots,i-1\}$ that do not form a inversion with i, which are exactly $(i-1)-(d^-(i)+n^-(i))$, then we have that $\pi(i)-1 = d^+(i)+a^+(i)+(i-1)-(a^-(i)+d^-(i))$, from which it is concluded that

$$i + d^+(i) \neq \pi(i) + d^-(i) \iff a^-(i) \neq a^+(i).$$

Therefore, we have to show that there exists a node i such that $a^-(i) \neq a^+(i)$. Let $U \subseteq V$ be the set of nodes forming an inversion with another node that is not its neighbor. Formally

$$U = \{j \in [n] \colon \exists k \in \{j,k\} \notin E \text{ and } \{j,k\} \text{ is a inversion under } \pi\}$$

Let $i^* = \min U$. Then, by definition of U there exists a $k \in [n]$ such that $\{i^*, k\}$ is an inversion under π and $\{i^*,k\} \notin E$. Also, since i^* is the minimum node in U, necessarily $k > i^*$. Hence $a^-(i^*) = 0$ and $a^+(i^*) \geq 1$. Therefore i^* satisfies the condition $n^-(i) \neq n^+(i)$ and therefore $i^* + d^+(i^*) \neq \pi(i^*) + d^-(i^*)$. □

We are now ready to define our protocol and main result regarding the recognition of permutation graphs.

Theorem 2.6 There is a PLS to recognize PERMUTATION-RECOGNITION with proof-size of $\mathcal{O}(\log n)$ bits.

Sketch of Proof. As commented before, the main idea of the PLS is the following. Merlin gives to each node v its identifier $\ell_1(v) \in [n]$ and image $\pi(\ell_1(v))$ under a supposed proper permutation model π. The rest of the information given by the prover is to let the nodes verify that effectively conditions (c1) and (c2) are satisfied. In fact, check (c1) for a node v is easy because each neighbor $\omega \in N(v)$ shares with him it values $\ell_1(\omega)$ and $\ell_2(\omega)$, and then v just has to verify they form and inversion. The main work on this protocol is to verify condition (c2), for which it is necessary for the nodes to verify Merlin effectively gave them a semi-proper permutation model (i.e., π is bijective and satisfies Definition 2.2), and then check if equality of Lemma 2.5 is satisfied. If at least one node doesn't satisfy it, then π is not a proper permutation model and the node has to reject. In order to verify that π is at least a semi-proper permutation model it is necessary to check that π is bijective, and to do that, Merlin gives a path between the nodes with identifier 1 and n, and then, if π is not bijective, one node of the path rejects because it notices condition of Proposition 2.3 is not satisfied. In the same way the nodes verify if Merlin gave to each node a unique identifier in $[n]$. Also, there are little details to be check, for example, check that Merlin gave them the correct number nodes n. $\square$

Protocol 2.7 The next is a one round protocol for PERMUTATION-RECOGNITION : The certificate provided by the prover to node v is interpreted as follows:

1. The certification of the number of nodes n, according to a protocol for SIZE.
2. Values $\ell_1(v), \ell_2(v) \in [n]$, where ℓ_1 and ℓ_2 are injective functions from V to $[n]$. The pair $(\ell_1(v), \ell_2(v))$ is interpreted as a value of a permutation π such that $\pi(\ell_1(v)) = \ell_2(v)$.
3. Value p_v corresponding to the minimum value grater than $\ell_1(v)$ that not an image under ℓ_1 of a neighbor of v. Formally, $p_v = \min\{k \in \{\ell_1(v)+1, \dots, n\} : \forall u \in N(v), k \neq \ell_1(u)\}$.
4. Value q_v corresponding to the minimum value greater that $\ell_2(v)$ that is not an image under ℓ_2 of a neighbor of v. Formally, $q_v = \min\{k \in \{\ell_2(v)+1, \dots, n\} : \forall u \in N(v), k \neq \ell_2(u)\}$.
5. The certification of a path P_ℓ between the nodes u and w such that $\ell_1(u) = 1$ and $\ell_1(w) = n$, according to the protocol for $s, t-$ PATH.
6. The certification of a path P_π between the nodes u and w such that $\ell_2(u) = 1$ and $\ell_2(w) = n$, according to the 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 $d^+(v)$ and $d^-(v)$, and check the following conditions:

a. The correctness of the number of nodes n according to the protocol of SIZE.

b. The correctness of the paths P_π and P_ℓ according to the protocol of $s, t-$ PATH.

c. The values $\ell_1(v), \ell_2(v)$ belong to the set $\{1, \dots, n\}$.

d. Forms an inversion with its neighbors, i.e., $\forall \omega \in N(v), (\ell_1(v)-\ell_1(\omega))(\ell_2(v)-\ell_2(\omega)) < 0$.

e. The equality $\ell_1(v) + d^+(v) = \ell_2(v) + d^-(v)$ holds.

f. For each $\omega \in N(v)$ such that $\ell_1(\omega) < p_\omega < \ell_1(v)$, $p_\omega = \ell_1(u)$ for some $u \in N(v)$.

g. For each $\omega \in N(v)$ such that $\ell_2(\omega) < q_\omega < \ell_2(v)$, $q_\omega = \ell_2(u)$ for some $u \in N(v)$.

We now analyze the soundness and completeness of our protocol.

Completeness. Suppose that G is a permutation graph. Then a honest prover chooses ℓ_1 and ℓ_2 such that $\pi : [n] \to [n]$ defined by $\pi(\ell_1(v)) = \ell_2(v)$ is a proper permutation model of G. If the prover sends the real value of n, the nodes verify condition **a** according to the completeness of the protocol for Size. Similarly, if the paths P_ℓ and P_π are valid, condition **b** is verified according to the completeness of the protocol for $s, t-$Path. Once that condition **a** is verified, then **c**, **d** and **e** can be verified looking to the certificates in the neighborhood. Finally, the correctness of p_v and q_v are verified by conditions **f** and **g**, which are satisfied by Proposition 2.3.

Soundness. Suppose G is not a permutation graph. If a dishonest prover provides a false value of n, or false paths P_ℓ or P_π, then at least one node rejects it by soundness of the protocols for Size and $s, t-$ Path, respectively. Then, we can assume that the prover has not cheated on those values.

Suppose that the prover gives a function ℓ_1 such that $\{\ell_1(v)\}_{v \in V} \neq [n]$. If some $\ell_1(v)$ is not in the set $[n]$ then v rejects in the verification of condition **b**. Then, we assume the existence of $j \in [n]$ such that $j \neq \ell_1(v)$ for all $v \in V$. As Merlin cannot send an invalid path P_ℓ, necessarily $1 < j < n$. Also, by correctness of the path, there exists nodes u_1, u_2 in the path such that $\{u, v\} \in E$ and $\ell_1(u) < j < \ell_1(v)$. From all possible choices of u and v, let us choose the pair such that $\ell_1(u)$ is maximum. Now we prove that v fails to check condition **f** and rejects. Indeed, as no vertex has j as image through ℓ_1, then $p_u \leq j$. If $p_u = j$, then v fails to check condition **f** and rejects. Suppose that $p_u < j$ and v verifies condition **f**. Then there must exist $\omega \in N(v)$ such that $p_u = \ell_1(\omega) < j$, contradicting the choice of u. We deduce that if condition **f** is verified by every node, necessarily ℓ_1 is an injective function from V into $[n]$.

By an analogous argument, we deduce that if condition **g** is verified by every node, then necessarily ℓ_2 is an injective function from V into $[n]$.

Suppose then that the dishonest prover provides the right value of n, as well as injective functions ℓ_1 and ℓ_2. If condition **d** is verified, then π is a semi-proper permutation model of G. Then, since G is not a permutation graph, at least one node must fail upon verify condition **e** by Lemma 2.5.

Proof-size Analysis. The certification for Size and $s, t-$Path has proof-size of $\mathcal{O}(\log n)$ bits, given by Proposition 1.7 and Proposition 1.9. On the other hand, for each $v \in V$, the values $\ell_1(v), \ell_2(v), q_v, p_v$ are $\mathcal{O}(\log n)$ as they are numbers in $[n]$. Therefore, overall the total communication is $\mathcal{O}(\log n)$.

2.2 Trapezoid Graphs

The protocol for Trapezoid is a sort of generalization of the protocol for permutation graphs. Indeed, for this class we can give an analogous characterization, that later is used to build a compact one-round PLS to recognize Trapezoid graphs.

In a model of a trapezoid graph, there are two parallel lines $\mathcal{L}_t$ and $\mathcal{L}_b$. We denote these lines as the *top and bottom lines*, respectively. Each trapezoid has sides contained in each line, and then is 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 2.1).

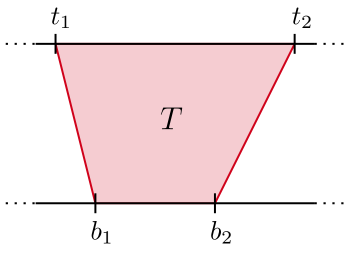


Figure 2.1: Each trapezoid T is defined by the set $T = \{b_1, b_2, t_1, t_2\}$.

The definition of a permutation graph can be restated as follows (see [20])

Definition 2.8 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, all elements in $[2n]$ correspond to a vertex of some trapezoid. The trapezoid model in the example of Figure 1.3 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 these 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 $v \in V$:

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

Intuitively, the set $F_t(v)$ has the positions in the upper line to the left of T_v which are vertices of a trapezoid $T(\omega)$, with $\omega \notin N(v)$. Analogous to $T_b(v)$. We also call $f_t(v) = |F_t(v)|$ and $f_b(v) = |F_b(v)|$.

Analogous to the previous section, the following Lemmas characterize trapezoid graphs by equalities calculable by the nodes with local information given by their neighbors.

Lemma 2.9 Let $G = (V, E)$ an n-connected trapezoid a graph. Then every proper trapezoid model $\{T_v\}_{v \in V}$ of G satisfies for every $v \in V$:

$$f_b(v) = f_t(v)$$

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 $\{1, \ldots, t_1(v)\}$ and their two lower vertices in $\{1, \ldots, b_1(v)\}$, as otherwise T_w and T_v would intersect. Then, the cardinality of the set $F_t(v)$ is equal to the cardinality of the set $F_b(v)$, as every position in $\{1, \ldots, 2n\}$ corresponds to a vertex of some node, so if a position $j < b_1(v)$ is not in $F_b(v)$, then has to be a vertex of some neighbor of v. Analogous for the positions $j < t_1(v)$ in the upper line. □

Lemma 2.10 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. There exists a node $v \in V$ such that some position 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. There exists $v \in V$ such that $f_b(v) \neq f_t(v)$.

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 2.2):

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

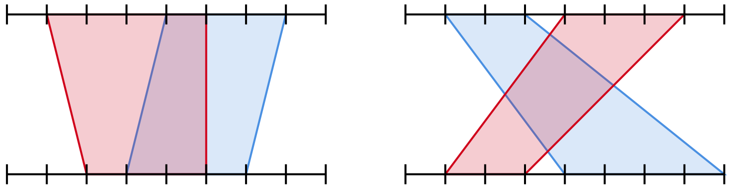


Figure 2.2: Representation of the two possible intersection cases.

In the first case of Figure 2.2, depicted in left, at least one vertex of a trapezoid is contained in the other. In the case of the right hand, the trapezoids intersect, but they not contain each other vertex.

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 there exists a node $\omega \in V \setminus N(u)$ 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)** There 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 $T_u \cap T_\omega \neq \emptyset$. If not, minimality of $b_1(u)$ is contradicted.

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

Theorem 2.11 There is a PLS to recognize TRAPEZOID-RECOGNITION with proof-size of $\mathcal{O}(\log n)$ bits.

Sketch of Proof. The main idea to recognize trapezoid graphs involves Lemma 2.9 and Lemma 2.10 as follow. Merlin gives to each node the vertices in $[2n]$ at both, lower and upper line, corresponding it's trapezoid in a supposed valid model $\{T_v\}_{v \in V}$. Next, the nodes has to validate at least Merlin didn't cheat giving the same vertex to two different nodes, for which nodes uses Protocol 1.14, and then, they validate if equality of Lemma 2.9 is satisfied (notice that this can be check by a node v only with local information given by its neighbors). If this equality holds, then the node accepts, and if not reject. Later we prove the Completeness and Soundness of the protocol using mainly the results of Lemmas lemma 2.9 and lemma 2.10 because they give us exactly what we want in a distributed algorithm: If a graph G is a trapezoid graph, then Lemma 2.9 says that there exists a proper trapezoid model $\{T_v\}_{v \in V}$ satisfying the conditions of the protocol below, and if G is not a trapezoid graph, then Lemma 2.10 ensures that any semi-proper model of G would satisfied equality of Lemma 2.9 in all nodes. □

Protocol 2.12 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. The certification of a path P_t between the nodes with vertex 1 and $2n$ in the upper line and a path P_b between the nodes with vertex 1 and n in the lower line according to a protocol for $s, t-$ PATH.

3. 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.
4. 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.
5. 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.

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 $[t_1(v)+1, t_2(v)-1]_{\mathbb{N}}$ and $[b_1(v)+1, b_2(v)-1]_{\mathbb{N}}$ are 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. $f_b(v) = f_t(v)$.

Each node accepts if all conditions are satisfied. If not, the node rejects.

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 to the trapezoid model. Then, the nodes 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 2.9.

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 rejects 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 $\bigcup_{v\in V}\{t_1(v), t_2(v)\} \neq [2n]$. 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 2.10 we deduce that condition **i** cannot be satisfied and some node rejects.

Proof-size Analysis. the certification for SIZE and $s, t-$ PATH have proof-size of $\mathcal{O}(\log n)$ bits, given by proposition 1.7 and proposition 1.9. 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 $\mathcal{O}(\log n)$ as they are numbers in $[2n]$. Overall the total communication is $\mathcal{O}(\log n)$.

Chapter 3

Distributed Interactive Protocols

Before present the main distributed interactive protocols in the next chapter, we present two distributed problems which we prove to have a compact protocol to solve them. This two problems are no just useful to solve our particular recognition graph protocols, but they have their own value because represents two natural problems that arise in more general distributed problems and we present a distributed interactive solution to them.

3.1 Corresponding Order Problem

First we tackle a problem that we call itCORRESPONDING ORDER, which is defined as follow.

Corresponding order problem. Given a graph $G = ([n], E)$ and a number $N \geq n$, each node receives an element $n(i) \in [N]$ and its claimed position $\pi(i)$ in the set $\{n(i)\}_{i \in V}$, how can we recognize the ordering $\{\pi(i)\}_{i \in V}$ with respect to $\{n(i)\}_{i \in V}$ is correct?

Notice that a natural problem in a distributed setting is that nodes are only in interaction with it's neighbors but there are situations when far-away graph nodes need to interact in some way and validate an information. To solve CORRESPONDING ORDER allows us in a way to solve this because notice that the ordering π has no relation with the distance of the nodes in the graph, so if this ordering can be validated, then the nodes would have a natural ordering respect a collection $\{n(i)\}_{i \in V}$ (which may not be permutation) to share and validate information.

Theorem 3.1 CORRESPONDING ORDER has a $\mathsf{dMAM}_{1/3}[\log n + \log N]$ Protocol that solves it. In specific, if $N \in O(n)$, then the Protocol is $\mathsf{dMAM}_{1/3}[\log n]$

Protocol 3.2 In the first round, Merlin send to each node $i \in V$: a pair $(n(i), \pi(i))$ (the number of $[N]$ and the corresponding position $\pi(i)$), a pair $(u(i), \pi(i) + 1 \mod n)$ ($u(i)$ the claimed successor of $n(i)$ in $\{n(i)\}_{i \in V}$), the information needed according to Protocol 1.8 to give the number of nodes n, and the information needed according to Protocol 1.6 to construct a valid spanning tree T_r rooted in $r \in V$.

In the following round (including verification round), besides to the explicit information, the nodes shares the information to verify the number of nodes using Protocol 1.8 and that $\{\pi(i)\}_{i\in V}$ is a permutation using Protocol 1.14.

Then, in the second round, each node $v \in V$ sends to Merlin an index $i_v \in [|\mathcal{H}|]$ (corresponding to a hash function h_{i_v}).

In the Last round, Merlin sends to each node $i \in [n]$.

- The index i_r sent by the root in the previous round.
- The following vectors

$$\overline{h_i} = \sum_{k\in T_r[i]} h_{i_r}(e_k)$$
$$\overline{g_i^a} = \sum_{k\in T_r[i]} h_{i_r}(e'_k)$$

  Where $T_r[i] = T_r(i) \cup \{i\}$ and $T_r(i)$ are the children of i in the tree T_r.

  The vectors $e_k, e'_k \in \{0,1\}^{n \log N}$ have n blocks of sizes $\log N$ each one. Both vectors e_k, e'_k have only one block distinct of $0^{\log N}$, with the value $|n(i)|_2$ in the $\pi(k)$th block and $u(k)$ in the $(\pi(k) +_n 1)$th block, respectively, padding with $0's$ if necessary.

In the Verification Round, first, each node verifies that $\{\pi(i)\}_{i\in V}$ is a permutation in accordance of Protocol 1.14 and the correctness of the spanning tree running the verification round of Protocol 1.6 . If it passes these verification, then it continues. Else, it rejects.

In addition, each node $i \in [n]$ verifies the correctness of the hash values checking that

$$\overline{h_i} = h_{i_r}(e_i) + \sum_{j\in T_r(i)} \overline{h_j} \tag{3.1}$$
$$\overline{h_i^a} = h_{i_r}(e'_i) + \sum_{j\in T_r(i)} \overline{h_j^a} \tag{3.2}$$

Finally, each node $i \neq r$ ACCEPT iff equalities 3.1 and 3.2 hold and the successor $u(i)$ given by Merlin is bigger than $n(i)$. The root, in order to accept has to verify in addition that $h_i = h_i^a$.

We analyze the correctness of the above protocol.

Correctness. If the ordering $\{\pi(i)\}_{i\in[n]}$ is correct respect $\{n(i)\}_{i\in[n]}$, then Merlin just has to send the real successor $u(i)$ of each node and a real information to construct a spanning tree and the hash values $h_i n h_i^a$. Then, with this information the nodes verify $\{\pi(i)\}_{i\in[n]} = [n]$ and then, each node $i \neq r$ verifies the correctness of the hash values, and the following vectors are equal

$$e = \sum_{i\in[n]} e_i$$
$$e' = \sum_{i\in[n]} e'_i$$

So, by linearity of h_{i_r}, we have $\overline{h_r} = \overline{h_r^a}$ and then, the root ACCEPTS.

Soundness. If $\{\pi(i)\}_{i\in[n]}$ is not the correct order of $\{n(i)\}_{i\in[n]}$, then

$$\exists i, j \in [n], n(i) < n(j) \text{ and } \pi(i) > \pi(j) \tag{3.3}$$

If the information sent by Merlin passes the verification of:

- $\{\pi(i)\}_{i\in[n]} = [n]$.
- The correctness of the spanning tree.

Suppose the root r accept, then $h_r = h_r^a$, and by Theorem 1.4, with probability $\dfrac{1}{n^c}$ we have that $e = e_p$.

If $e = e_p$, Merlin gave to each node a valid successor $u(i) \in \{n(i)\}_{i\in[n]}$ such that there exists a node $j \in [n]$ to which Merlin gives the pair $(u(i), \pi(i) + 1 \mod n)$.

Then, by (3.3), there exists a pair of nodes u, v such that $\pi(u) = \pi(v)+1$ and $n(u) < n(v)$ (if not, vector e is increasing and this contradict 3.3), so v checks that the predecessor given by Merlin is higher than $n(v)$ and then v rejects.

That is, if $\{\pi(i)\}_{i\in[n]}$ is not the correct order of $\{n(i)\}_{i\in[n]}$, with probability $\geq 1 - \dfrac{1}{n^c}$ a node REJECTS to every input given by Merlin.

Proof-size Analysis. The proof-size needed to construct a valid spanning tree and check the number-node n can be done with $\mathcal{O}(\log n)$ bits by Protocols 1.6 and 1.8, respectively. Also, by Theorem 1.4, $|\mathcal{H}|$ has size p and we take p to be polynomial on n, so the size of $|\mathcal{H}|$ is polynomial in n and then encode $j \in [|\mathcal{H}|]$ can be done in $\mathcal{O}(\log n)$. Finally, h_i and $h_i^a \in \{0, ..., p-1\}$ also are encoded in $\mathcal{O}(]\log n)$. Overall, the proof-size is of $\mathcal{O}(\log n)$ bits.

Notice that CORRESPONDING ORDER protocol can be extended to verify any total order $\preccurlyeq$ in $\mathbb{R}^k$ (k fixed) which defines a order $\pi\colon V \to [n]$ to the nodes, according to some collection of labels $\{\ell(v)\}_{v\in V}$ with $\ell(v) \in \mathbb{R}^k$ (i.e., $\pi(v) < \pi(\omega)$ iff $\ell(v) \preccurlyeq \ell(\omega)$). Just it is necessary to replace $c(v)$ by $\ell(v)$ and use a hash function in $\mathcal{H}_{kn^2,p}$[1], and interact the same way in Protocol 3.2, hashing the vectors $\ell(v)$ in this new hash function. As k is fixed, interchanging $\ell(v)$ still is $\mathcal{O}(\log n)$. We call this more general problem as and it is define as follow in the next section.

[1]p chosen in a way still is satisfied that $\dfrac{kn^2}{p} \leq \dfrac{1}{3}$

3.2 Linear Assignation Problem

Consider the next problem called LINEAR ASSIGNATION: Given a graph $G = (V, E)$, each node receive its position $\pi(v) \in [n]$ according to a total order $\preccurlyeq$ (known by the nodes) and two values $n(v), s(v) \in \{0,1\}^q$ for some q constant known ($q \in O(n)$), such that, supposedly, the predecessor ω of v in the total order receive the value $s(\omega) = n(v)$ and $s(v) = n(z)$, with z the successor of v in the total order. Can the nodes verify correctness of values $n(v)$ and $s(v)$ given to each node? Notice the predecessor/successor in the total order isn't neighbors necessarily.

Theorem 3.3 LINEAR ASSIGNATION problem has a $\mathsf{dAM}_{1/3}[\log n]$ algorithm that solves it.

Protocol 3.4 (LAP) In the first round, each node v sends to Merlin a random bit $b_v \in [m]$, with $m = |\mathcal{H}_{nq,p}|$.

In the second round, Merlin sends to each node $v \in V$

- The position $\pi(v)$ according to the total order $\preccurlyeq$.
- The values $n(v)$ and $s(v)$ in $\{0,1\}^q$.
- Information to construct a valid spanning tree rooted in $r \in V$.
- The random bit b_r chosen by the root r of the spanning tree.
- The following vectors

$$\overline{h_v^c} = \sum_{\omega \in T_r[v]} h_r(e_\omega)$$
$$\overline{h_v^s} = \sum_{\omega \in T_r[v]} h_r(e'_\omega)$$

  Where the vectors $e_\omega, e'_\omega \in \{0,1\}^{n\times q}$ have n blocks of sizes q each one. Both vectors e_k, e'_k have only one block distinct of 0^q, with the value $n(\omega)$ in the $\pi(\omega)$th block and $s(\omega)$ in the $(\pi(\omega) +_n 1)$th block, respectively, padding with $0's$ if necessary.

Finally, in the Verification round each node $v \in V$ verifies

- The correctness of the spanning tree according to the verification round of Protocol 1.6.
- the following equalities holds

$$\overline{h_v^c} = h_r(e_v) + \sum_{\omega \in T_r(v)} \overline{h_\omega^c}$$

$$h_v^s = h_r(e'_v) + \sum_{\omega \in T_r(v)} \overline{h_\omega^s}$$

- If $v = r$, it verifies also that $\overline{h_r^c} = \overline{h_r^s}$.

If the above conditions are verified by the node, then it accepts. If not, it rejects.

- **Completeness.** If Merlin send honest information, then by Theorem 1.4, the hashed values $\overline{h_r^c}$ and $\overline{h_r^s}$ with probability $\geq \frac{2}{3}$, and with the same probability the rest of nodes verifies the correctness of hashed values on the spanning tree.

- **Soundness.** The Protocol should reject if

  1. The order $\{\pi(v)\}_{v\in V}$ is not right. Equivalently[2], it should rekect if there exists a value $k \in [n]$ such that $\pi(v) \neq k$ for all $v \in V$.
  2. The equalities $s(\omega) = n(v)$ or $s(v) = n(u)$ do not hold for ω and u the predecessor of v, respectively.

  In case 1., the k-block in $\hat{e} = \sum_{v\in V} e_v$ is equal to 0^q, the same k-block in $\hat{e'} = \sum_{v\in V} e'_v$ is 0^q iff there is not a node with position $\pi(\omega) = k+1$. Then block $k+1$ in $\hat{e}$ is 0^q. Iteratively, there is not a node s with position $\pi(s) = n$. But Merlin gives to some node this position. Then, it has to exists a position $\hat{k}$ such that the $\hat{k}$-block equal to 0^q in $\hat{e}$ but not equal to 0^q in $\hat{e'}$, so $\hat{e} \neq \hat{e'}$ and then by Theorem 1.4 the root rejects with probability $\geq \frac{2}{3}$.

  In case 2., again denoting $\hat{e} = \sum_{v\in V} e_v$ and $\hat{e'} = \sum_{v\in V} e'_v$. It is true that $\overline{h_r^c} \neq \overline{h_r^s}$ with probability $\geq \frac{2}{3}$, and then, by construction of the vectors, it is satisfied that $\hat{e} \neq \hat{e'}$ because there exists a node v such that $s(\omega) \neq n(v)$ or $s(v) \neq n(u)$ with ω and u the predecessor/successor of v, respectively. For this, $\hat{e}$ and $\hat{e'}$ differs in the position $\pi(v)$ or the position $\pi(v)+1$, and then, again by Theorem 1.4, the root rejects with probability $\geq \frac{2}{3}$.

  In any case, the Protocol rejects with probability $\geq \frac{2}{3}$.

  Proof-size Analysis. All values shared not corresponding to hashed values are $\mathcal{O}(n)$ so can be encoded with $\mathcal{O}(\log n)$ bits. The size p of the hashed values again is chosen such that p is $\mathcal{O}(n^k)$ for some fixed $k \in \mathbb{N}$, so they can also be encoded with $\mathcal{O}(\log n)$ bits.

[2]because all values $\pi(v)$ are in $[n]$

Chapter 4

Three-round Distributed Interactive Protocols

In this chapter, we study the recognition of Circle-Recognition ,k-Polygon-Circle-Recognition and Unit-Square-Recognition in the distributed interactive setting. The need of more rounds is in order to shrink the proof-size needed by the protocols, or to use the protocols described in Chapter 4 in order of validate the correctness of some messages. These messages are related to the same problem presented in Chapter 2, which is that, given the definition of a geometric intersection graph class, it is relatively easy for two adjacent nodes to check if they satisfy the geometric characterization of been neighbors, but it is hard to check if two no-adjacent nodes effectively does not satisfy the condition. The distributed interactive protocols here presented use three round of interaction and have proof-size of $\mathcal{O}(\log n)$ bits.

4.1 Circle Graphs

First, we tackle the recognition of Circle-Recognition. Unlike the cases of permutation and trapezoid graphs, for this class we are unable to provide a one-round protocol. Instead, we give a compact three round interactive protocol, which is based in the following characterization of circle graphs (see [20]):

A graph $G = (V, E)$ is a circle graph if and only each node $v \in V$ can be associated to an interval $I_v = [m_v, M_v] \subseteq \mathbb{R}$ such that:

C1 For each $u, v \in V$, $\{u, v\} \in E$ then $I_u \cap I_v \neq \emptyset$, $I_u \not\subseteq I_v$ and $I_v \not\subseteq I_u$.

C2 For each $u, v \in V$, $I_u \cap I_v \neq \emptyset$, $I_u \not\subseteq I_v$ and $I_v \not\subseteq I_u$ then $\{u, v\} \in E$.

C3 For each $v \in V$, $m_v, M_v \in [2n]$, and $\{m_v\}_{v\in V} \cup \{M_v\}_{v\in V} = [2n]$, i.e., for every pair of different nodes $u, v \in V$, $m_u \neq m_v$ and $M_u \neq M_v$.

In words, for every circle graph of size n there is a collection of n intervals with extremes in $[2n]$ whose extremes do not coincide, and where two nodes are adjacent if and only if their corresponding intervals have nonempty intersection, and one interval is not included in the other. We remark that the later characterization is not an intersection model of a circle graphs, as we ask more than simply the intersection of the objects. Given a graph $G = (V, E)$, a set of intervals $\{I_v\}_{v\in V}$ satisfying conditions **C1**, **C2** and **C3** is called a *proper model* for G. A set of intervals satisfying conditions **C1** and **C3** is called a *semi-proper model* for G.

Let $G = (V, E)$ be a graph, $\{[m_v, M_v]\}_{v\in V}$ be a semi-proper model for G, and let $v \in V$. We denote by $n_m(v)$ the number of nodes $u \in N(v)$ whose lower limit m_u is such that $m_v \leq m_u \leq M_v$ and $M_u \notin [m_v, M_v]$. Similarly, we denote by $n_M(v)$ the number of nodes $u \in N(v)$ such that $m_v \leq M_u \leq M_v$ and $m_u \notin [m_v, M_v]$. We also denote by $\pi_m(v)$ (respectively $\pi_M(v)$) the position of m_v (resp. M_v) respect $\{m_u\}_{u\in V}$ (resp. $\{M_u\}_{u\in V}$).

Analogously, for each $v \in V$ we denote $\overline{n_m(v)}$ (resp. $\overline{n_M(v)}$ the number of nodes $u \notin N(v)$ such that $m_u \in [m_v, M_v]$ (resp. $M_u \in [m_v, M_v]$).

Lemma 4.1 Let $G = (V, E)$ be a graph, $\{[m_v, M_v]\}_{v\in V}$ be a semi-proper model for G and $\{\pi_m(v)\}_{v\in V}, \{\pi_M(v)\}_{v\in V}$ the relative position of m_v and M_v respect the sets $\{m_v\}_{v\in V}$ and $\{M_v\}_{v\in V}$, respectively, for each $v \in V$. Then $\{[m_v, M_v]\}_{v\in V}$ is a proper model for G if and only if, for every $v \in V$,

$$2(\pi_M(v) + \pi_m(v)) = M_v + m_v + n_m(v) - n_M(v)$$

Proof. Let $v \in V$ be an arbitrary node. Observe that there are $n - \pi_M(v)$ nodes u such that $M_u > M_v$. Then, there are $n - M_v + \pi_M(v)$ nodes w such that $m_w > M_v$, because if a position in $[M_v, 2n]$ is either the lower or upper limit of an interval. Similarly, there are $n-\pi_m(v)$ nodes w satisfying $m_w \geq m_v$, and then $n-m_v+\pi_m(v)$ nodes u such that $M_u > m_v$.

Then, in $[m_v, M_v]$ there are $\pi_m(v) + \pi_M(v) - m_v$ nodes u such that $m_u \in [m_v, M_v]$ and $M_v - \pi_m(v) - \pi_M(v)$ nodes w such that $M_w \in [m_v, M_v]$. For each $v \in V$, let us call S_s be the set of nodes z such that $[m_z, M_z] \subset [m_v, M_v]$.

Suppose that $\{[m_v, M_v]\}_{v\in V}$ is a proper model for G, and for each $v \in V$. Then, we have that $\overline{n_m(v)} = \pi_m(v)+\pi_M(v)-m_v-n_m(v)-|S_v|$, $\overline{n_M(v)} = M_v-\pi_m(v)-\pi_M(v)-n_M(v)-|S_v|$ and $\overline{n_M(v)} = \overline{n_m(v)} = 0$, because as $\{[m_v, M_v]\}_{v\in V}$ is a proper model, then if $\overline{n_M(v)} > 0$ or $\overline{n_m(v)} > 0$ for some $v \in V$, exists $u \notin N(v)$ such that $[m_v, M_v] \cap [m_u \cap M_u] \neq \emptyset$, which it is a contradiction. We deduce that $\pi_m(v)+\pi_M(v)-m_v-n_m(v) = M_v-\pi_m(v)-\pi_M(v)-n_M(v)$, from which we deduce $2(\pi_M(v) + \pi_m(v)) = M_v + m_v + n_M(v) - n_m(v)$.

Suppose now that $\{[m_v, M_v]\}_{v\in V}$ is not a proper model for G. Then, there exist a pair of nodes $\{u, v\} \notin E$ such that $[m_u, M_u] \cap [m_v \ \cap M_v] \neq \emptyset$ with the intervals not containing each other. From all such pairs, let us pick one such that m_v is minimum. Then necessarily there exists $m_\omega \in [m_v, M_v]$, $M_\omega > M_v$ and $\{v, \omega\} \notin E$. This implies that $n_m(v) + |S_v| < \pi_m(v) + \pi_M(v) - m_v$. On the other hand, the choice of v implies that all nodes such u such that $M_u \in [m_v, M_v]$ either belong to $N(v)$ or belong to S_v. Then $n_M(v) + |S_v| = M_v-\pi_m(v)-\pi_M(v)$. We deduce that $\overline{n_m(v)} > 0$ and $\overline{n_m(v)} = 0$, and then $2(\pi_M(v)+\pi_m(v)) \neq M_v + m_v + n_M(v) - n_m(v)$. □

Now we are ready to give the main result of this section.

Theorem 4.2 CIRCLE-RECOGNITION belongs to $\mathsf{dMAM}[\mathcal{O}(\log n)]$.

Protocol 4.3 Consider the Protocols 1.8,1.14 and 3.2 for SIZE, PERMUTATION and CORRESPONDING ORDER, respectively. 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 limits of the interval $I_v = [m_v, M_v]$, and the values $\pi_m(v)$ and $\pi_M(v)$.
3. The certification of $\{m_v\}_{v\in V} \cup \{M_v\}_{v\in V} = [2n]$ according to the protocol for PERMUTATION.
4. The certification of the correctness of $\{(m_v, \pi_m(v))\}_{v\in V}$ according to the protocol for CORRESPONDING ORDER.
5. The certification of the correctness of $\{(M_v, \pi_M(v))\}_{v\in V}$ 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 shares all the information given by Merlin and then, each node checks first the correctness of **1-5** according to the verification rounds for SIZE, PERMUTATION and CORRESPONDING ORDER. Then, each node v computes $n_m(v)$ and $n_M(v)$, and checks the following conditions:

a. For every $u \in N(v)$, $I_v \cap I_u \neq \emptyset$, $I_v \not\subseteq I_u$ and $I_u \not\subseteq I_v$.

b. $2(\pi_M(v) + \pi_m(v)) = M_v + m_v + n_M(v) - n_m(v)$.

We now analyze completeness and soundness.

Completeness. Suppose that input graph G is a circle graph. Then Merlin gives a proper model $\{[m_v, M_v]\}_{v\in V}$ for G. Merlin also provides the correct orders $\{\pi_m(v)\}_{v\in V}$ and $\{\pi_M(v)\}_{v\in V}$, the correct number of nodes n, and the certificates required in the sub-routines. Then, the nodes verify correctness of **1-5** 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 4.1. We deduce that every node accepts with probability greater than 2/3.

Soundness Suppose now that G is not a circle graph. By the soundness of the protocols for SIZE, PERMUTATION and CORRESPONDING ORDER, we know that at least one node rejects the certificates not satisfying **1**-**5**, with probability greater than 2/3. Suppose then that conditions **1**-**5** are verified. Observe that each set of intervals satisfying condition **a** form a semi-proper model for G. Since G is not a circle graph, by Lemma 4.1 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.

Proof-size Analysis. the certification for SIZE, PERMUTATION and CORRESPONDING ORDER have $\mathcal{O}(\log n)$ proof-size, given by Protocol 1.8, Protocol 1.14 and Protocol 3.2. Also, for each $v \in V$, the values $m_v, M_v, \pi_m(v), \pi_M(v)$ can be encoded in $\mathcal{O}(\log n)$ as they are numbers in $[2n]$ or $[n]$. Overall the total communication is $\mathcal{O}(\log n)$.

4.2 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 [20]: (1) for each $v \in V$, the polygon P_v is represented as a set of k vertices $\{p_1(v), \dots, p_k(v)\}$ such that, for each $i \in [k-1]$, $0 \le p_i(v) < p_{i+1}(v) \le n \cdot k - 1$, and (2) $\bigcup_{v\in V} \bigcup_{i\in[k]} \{p_i(v)\} = \mathbb{Z}_{nk}$. In other words, each value in $[k \cdot n]$ corresponds to a unique vertex 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 $\{1, \dots, p_1(v)\} \cup \{p_k(v), \dots, kn\}$ that do not correspond to a neighbor of v. For each $i \in [k]$, we also call $\beta_i(v)$ the set of positions $j \notin \alpha(v)$ such that $p_i(w) = j$ for some $w \in V$. Formally,

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

Lemma 4.4 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 such that $p_i(w) \in \beta_i(v)$, the k the vertices $V(P_w)$ are contained in $\alpha(v)$. Otherwise, the polygons P_w and P_v have non-empty intersection, which contradicts the fact that $\{P_v\}_{v\in V}$ is a proper model, because if $p_i(\omega) \in \alpha(v)$, then $w \notin N(v)$. 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 $[0, p_1(v)]_{\mathbb{N}} \cup [p_k(v), kn]_{\mathbb{N}}$. Formally,

$$N_{1,k}(v) = |\{q \in [0, p_1(v)]_{\mathbb{N}} \cup [p_k(v), kn]_{\mathbb{N}} : \exists w \in N(v), q \in P_w\}|$$

And $N_1 = |\{q \in [0, p_1(v)]_{\mathbb{N}} \colon \exists \omega \in N(v), p_1(\omega) = q\}|$

Lemma 4.5 Let $G = (V, E)$ be a graph, and $\{P_v\}_{v\in V}$ be a semi-proper polygon model for G and $\{\pi_{1k}(v)\}_{v\in V}$ the relative positions of $p_1(v)$ and $p_k(v)$ respect the set $\{p_1(v)\}_{v\in V} \cup \{p_k(v)\}_{v\in V}$ for each $v \in V$. Then, $\sigma_k(v) = \pi_{1k}(v) - 1$.

Proof. Direct by definition of $\sigma_k(v)$ and $\{\pi_{1k}(v)\}_{v\in V}$. □

Lemma 4.6 Let $G = (V, E)$ be a graph, and $\{P_v\}_{v\in V}$ be a semi-proper polygon model for G and $\{\pi_1(v)\}_{v\in V}, \{\pi_k(v)\}_{v\in V}$ the relative positions of $p_1(v)$ and $p_k(v)$ respect the sets $\{p_1(v)\}_{v\in V}$ and $\{p_k(v)\}_{v\in V}$, respectively, for each $v \in V$. Then,

$$|\alpha(v)| = kn - p_k(v) + p_1(v) - 1 - N_{1,k}(v)$$
$$|\beta_1(v)| = n - \sigma_k(v) + \pi_k(v) + \pi_1(v) - N_1(v)$$

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 in $[p_k(v)+1, n\cdot k-1]_{\mathbb{N}}$ used by elements of $\bigcup_{u\in V}\{p_1(u), p_k(u)\} \cap [p_k(v)+1, kn-1]_{\mathbb{N}}$. Of These $2n - \sigma_k(v)$ positions, 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 in

$[p_k(v), n \cdot k - 1]$ used by elements of $\bigcup_{u \in V}\{p_1(u)\} \cap [p_k(v)+1, kn-1]_{\mathbb{N}}$. This means that there are $n - \sigma_k(v) + \pi_k(v)$ elements of the set $\{p_1(u)\}_{u \in V}$ in $[p_k(v), kn-1]_{\mathbb{N}}$. Finally, noticing that there are $\pi_1(v)_{N1}(v)$ positions used by $\bigcup_{u \in V}\{p_1(u)\} \cap [0, p_1(v)]_{\mathbb{N}}$, we deduce that $|\beta_1(v)| \ = n - \sigma_k(v) + \pi_k(v) + \pi_1(v) - N_1(v)$. □

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

Theorem 4.7 k-POLYGON-CIRCLE-RECOGNITION belongs to $\mathsf{dMAM}_{1/3}[\log n]$.

Proof. Consider the protocols for SIZE, PERMUTATION and CORRESPONDING ORDER of Protocols 1.8, 1.14 and 3.2. 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 $V(P_v) = \{p_1(v), \dots, p_k(v)\}$.
3. The values of $\pi_1(v)$, $\pi_k(v)$ and $\sigma_k(v)$.
4. The certification to prove that $\bigcup_v V(P_v) = \mathbb{Z}_{kn}$ 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, as appropriate.

Remark: In order to check 7, each node has to play two different roles 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 to v the certificates of v' and v'', and v answers with the random bits as if they were generated by v' and v''. Obviously, this increases the communication cost by a factor of 2.

Then, each node v computes $\sigma_k(v)$ according to Lemma 4.5 and $|\beta(v)|$ and $|\alpha(v)|$ according to the expressions of Lemma 4.6, 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 $\{\pi_1(v)\}_{v\in V}$,$\{\pi_k(v)\}_{v\in V}$, $\{\pi_{1k}(v)\}_{v\in V}$, 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 4.4. 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 4.4 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.

Proof-size Analysis. The certification for Size, Permutation and Corresponding Order have $\mathcal{O}(\log n)$ proof-size, given by Protocols Protocol 1.8, 1.14 and 3.2. On the other hand, for each $v \in V$, the values $\pi_1(v), \pi_k(v), \pi_{1k}(v), V(P_v)$ can be encoded in $\mathcal{O}(\log n)$ as they are numbers in $[n]$, $[2n]$ or $[kn]$ (k fixed). Overall the total communication is $\mathcal{O}(\log n)$. □

4.3 Unit-Square Graphs

Given a graph $G = (V, E)$ of size n, an *unit-square model* of G is a function $p : V \to \mathbb{R}^2$, such that, every vertex $v \in V$ is associated with a unit square S_v in $\mathbb{R}^2$ defined by:

$$S_v = \left\{ z \in \mathbb{R}^2 : \max |p(v) - z| \leq \frac{1}{2} \right\}.$$

In words, a unit square model is an assignation of *centers of closed balls of diameter* 1 defined with the Manhattan norm. A unit square model p is called *proper* if, for every $u, v \in V$,

$$\{u, v\} \in E \iff S_u \cap S_v \neq \emptyset.$$

In other words, a unit square model is proper if the intersection graph of $\{S_v\}_v$ corresponds to G. A graph G is a *unit square graph* if it admits a proper unit square model. See Figure 4.1 for an example.

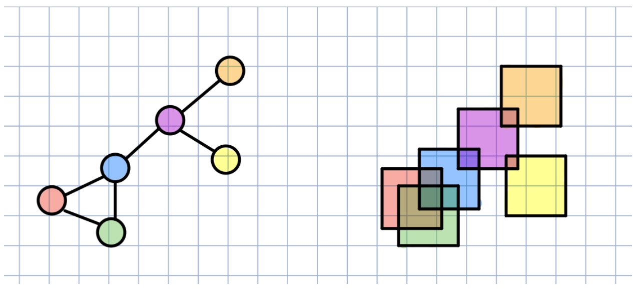


Figure 4.1: Example of a graph G in the left hand, and a proper unit-square model in the right hand.

The certification protocol for the recognition of a unit square graph G includes a unit-square model p for G; this is, the coordinates of the centers of the unit squares associated to the vertices. Remember that in the verification round, the nodes interchange certificates with their neighbors. Then, in the verification round the nodes can check if the squares of their neighbors intersect with their square, and it immediately rejects if this is not satisfied.

Definition 4.8 Given G a unit square graph and a unit square model p for G. The model p is called *semi-proper* if, for every pair of vertices, $u, v \in V$ $\{u, v\} \in E$ implies that $S_u \cap S_v \neq \emptyset$.

In other words, a semi-proper unit square model for a graph G is a model in which all adjacent nodes have intersecting squares. The difference between a semi-proper and a proper unit square model is that in a semi-proper we admit non-adjacent nodes with intersecting squares.

Definition 4.9 Given G a unit square graph and p a semi-proper unit-square model. A pair of vertices $\{u, v\}$ are called *ghosts* between them if $\{u, v\} \notin E$ and $S_u \cap S_v \neq \emptyset$.

Given some model p for a graph G, the local verification has two parts: For one side, the nodes has to check that it's unit squares intersects the unit squares of it's neighbors, and (2) it's unit squares does not have to intersect the unit square of nodes that are not it's neighbors. In resume, it is necessary to difference between semi-proper and proper unit square model for a graph.

Nice unit square models. As we mentioned in the introduction, in [127] it is shown that every unit-square graph $G = (V, E)$ admits a model where, for every $v \in V$, $p(v)$ can be encoded in $\mathcal{O}(\log n)$ bits. Such models are called *efficient*.

Definition 4.10 Given a point $q = (q_1, q_2) \in \mathbb{R}^2$, it's say that q is an *integer point* if either q_1 or q_2 are integers. When q belongs to $\mathbb{Z}^2$, we say that q is a *total integer point*.
A unit square model p of a graph G is called *nice* if $p(v)$ is efficient, and $p(v)$ is not an integer point, for every $v \in V$.

Remark 4.11 The nice unit square models have the following useful property: for every vertex v, the square S_v contains exactly one full-integer point.

Interestingly, every unit square graph admits a nice and proper unit square model, as we show in the following lemma.

Lemma 4.12 Every unit-square graph admits a nice and proper unit-square model.

Proof. Let G be a unit square graph and let p be an efficient and proper unit square model, provided by [127]. Suppose that p is not nice. Let us call X the set of non-integer coordinates of points in $p(V)$. Formally,

$$X = \{x \in \mathbb{R} : x \notin \mathbb{Z} \text{ and } (\exists v \in V : x = p(v)_1 \text{ or } x = p(v)_2)\}.$$

Suppose that X is nonempty. Define ε as the minimum distance between a coordinate of X and a point in $\mathbb{Z}$. Formally, $\varepsilon = \min\{|x - a| : x \in X,\ a \in \mathbb{Z}\}$. Observe that $0 < \varepsilon < 1$ and can be encoded in $\mathcal{O}(\log n)$ bits as well.

Now define the unit-square model p' obtained from p translating each point in $\varepsilon/2 \cdot (1, 1)$. Formally for every $v \in V$, $p'(v) = p(v) + \varepsilon/2 \cdot (1, 1)$. Observe that p' is also an efficient and proper unit-square model, as p' corresponds to a translation of p and $\varepsilon/2$ can be encoded in $\mathcal{O}(\log n)$ bits. Let v be an arbitrary vertex. If some coordinate of $p(v)$ is an integer, then necessarily that coordinate is not going to be an integer in $p'(v)$ (as $\varepsilon \notin \mathbb{Z}$). Also, by definition of ε, if a coordinate of $p(v)$ is not an integer, then neither that coordinate in $p'(v)$ is an integer. It's deduced that p' is a nice and proper unit square model for G. □

Total integer components. Remark 4.11 implies that for every nice unit square model p of a graph $G = (V, E)$, we can define a partition of V consisting in the squares that contain a given total integer point $q \in \mathbb{Z}^2$. More precisely, for $q \in \mathbb{Z}^2$ we define,

$$V_q = \{v \in V : q \in S_v\} \text{ and } V = \bigcup_{q \in \mathbb{Z}^2} V_q.$$

For each $q \in \mathbb{Z}^2$, the set V_q is called the *q-component* of G. Observe that the some sets V_q can be empty. In a proper unit square model, the non-empty q-components induce maximal cliques of G.

Let (G, id) be a unit square graph where the nodes have unique id's given by id. Let p be a nice unit square model. For each non-empty q-component, the *leader* of V_q is defined as the vertex $v \ \in V_q$ with maximum id.

For each $(i,j) \in \mathbb{Z}^2$, we define the *neighbor components* of (i,j)-component as the set of q-components such that the Manhattan distance between p and p is exactly 1.[1]. Observe that in a nice semi-proper unit square model, every pair of adjacent nodes u, v belong to adjacent components.

For $q = (q_1, q_2) \in \mathbb{Z}^2$, we denote by $S_q = \cup_{v \in V_q} S_v$ the *area of the component*, and by ∂S_q the set of square vertices on its perimeter. Relative to S_q, define the *lower-left* stair of the q-component, denoted by ∂S_q^{ll}, as the portion of the perimeter ∂S_q intersecting the set $\{(x,y) : x \leq q_1 \text{ and } y \leq q_2\}$. Similarly, the *upper-right stair* of the q-component, denoted by ∂S_q^{ur}, is the portion of ∂S_q that intersects the set $\{(x,y) : x \geq q_1 \text{ and } y \geq q_2\}$. The *upper-left stair* and *lower-right stair* are defined analogously and denoted as ∂S_q^{ul} and ∂S_q^{lr}, respectively.

The vertices of a stair are defined as the points of the stair that are the star and/or end of a stair's step.

For example, consider the next unit square model for K_4.

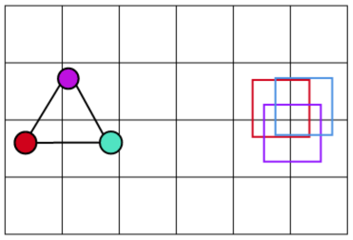
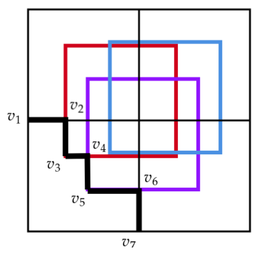


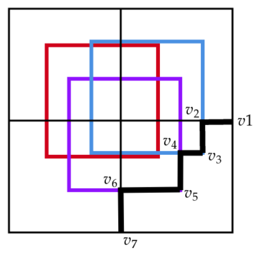
Figure 4.2: Left: A unit square graph G. Right: A nice unit square model of G

Then, the lower-left and lower-right stair of the unit square model of Figure 4.2 are the the white paths marked in Figure 4.3.

[1]If $p = (i,j))$, then $q \in \{(i-1,j-1),(i,j-1),(i+1,j-1),(i-1,j),(i,j),(i+1,j),(i-1,j+1),(i,j+1),(i+1,j+1)\}$



(a) lower-left stair marked with black and its vertices $v_1, ..., v_7$



(b) lower-right stair marked with black and its vertices $v_1, ..., v_7$

Figure 4.3

Now, the last notion needed is the *node steps* of a lower-left or lower-right stair.

Definition 4.13 Given a unit square model of a graph G and the lower-left stair of a component (i,j), a node v in the component are said to be in the lower-left stair if its left and lower sides are part of the stair. The step define by v, denoted by $step(v)$ is the horizontal border of S_v in ∂S_q^{ll}.
The size of a stair is the number of squares participating in the stair, and the lower-left stair of (i,j)-component with is represented by $left_{i,j} := (step(v_1), ..., step(v_k))$, with k the size of the stair and $v_1, ..., v_k$ the nodes participating in the stair (from left to right). The representation of the other stairs is defined analogously.

For example, in the above configuration, there are three nodes in the lower-left stair, with steps (v_3, v_4) (red) and (v_5, v_6) (purple). Notice also that the red node is not part of the lower-right stair.

Total orders in the two dimensional grid. We consider the following four total orders that can be defined over $[n]^2$.

1. **Vertical order.** Given by $(a, b) \preccurlyeq_v (c, d) \iff (a < c) \vee (a = c \wedge b < d)$
2. **Horizontal order**. Given by $(a, b) \preccurlyeq_h (c, d) \iff (b < d) \vee (b = d \wedge a < c)$
3. **Positive diagonal**. Given by $(a, b) \preccurlyeq_{dp} (c, d) \iff n_{a,b} < n_{c,d} \vee (p_{a,b} = p_{c,d} \wedge a < c)$
   Where $p_{a,b}$ is the only integer value such that $b = a + p_{a,b}$
4. **Negative diagonal**. Given by $(a, b) \preccurlyeq_{dp} (c, d) \iff n_{a,b} < n_{c,d} \vee (n_{a,b} = n_{c,d} \wedge a < c)$
   Where $n_{a,b}$ is the only integer value such that $b = -a + n_{a,b}$

To exemplify, consider the following square model of K_4.

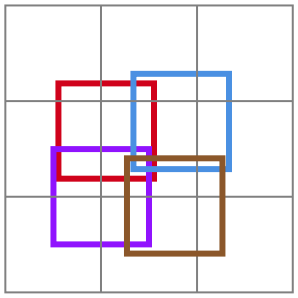
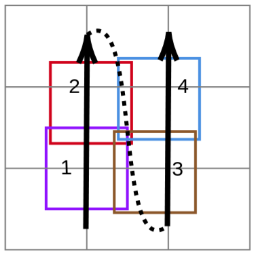


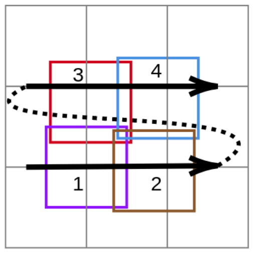
Figure 4.4: Unit square model of K_4

.

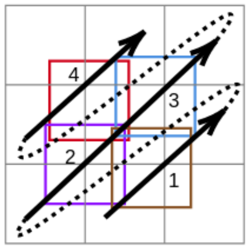
In this setting, as there is only one square intersecting each total integer point, so the only possible representative set is V. Then, the four natural orders are:



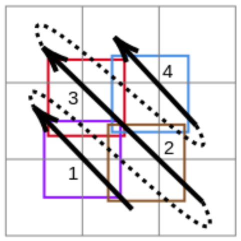
(a) Vertical order.



(b) Horizontal order.



(a) Positive diagonal order.



(b) Negative diagonal order.

Figure 4.6: Four orders of Figure 4.4.

Where the position of each node is the position of the total integer point contained in it.

Notice that it is not well define what happens if we consider two square in the same total integer point, but this case does not happen in our ordering on the Protocol.

We now have all the necessary to present the Protocol.

Theorem 4.14 There is a $\mathsf{dMAM}_{1/3}[\mathcal{O}(\log n)]$ protocol to recognize UNIT-SQUARE-RECOGNITION.

Sketch of Protocol. In the first round, Merlin gives to each node v its respective unit square S_v satisfying Lemma 4.12, defined by its rational center (x_v, y_v). Notice that if each node share its square with their neighbors, then each node v locally can verify if $S_v \cap S_u \neq \emptyset$ for all $u \in N(v)$. Then, all the rest information shared is to verify that if two nodes u, v aren't neighbors, then $S_v \cap S_u = \emptyset$ (which, by only knowing S_v, the nodes cannot verify locally). This is, each nodes needs to verify to no have ghosts.

In order too verify that there is no ghosts in the representation given by Merlin, we separate the verification in three cases. Each node v in component (i,j) needs to verify

1. that v is neighbor of all nodes with square in the same component.
2. There is no node ω in components (i,j+1),(i,j-1),(i-1,j),(i+1) ghost of v.
3. There is no node ω in the diagonal components (i-1,j-1),(i-1,j+1),(i+1,j+1) or (i+1,j-1) ghost of v.

In order to check this, the way of do it is the follow. To verify 1., Merlin chooses a representative of each component with at least one square. Then the nodes needs to verify that Merlin didn't cheat. That is, the nodes verify that effectively Merlin just gave one representative by component. If the nodes verify this, we prove that at least one node rejects if there are ghosts u, v in the same component. With this, if a node v is not neighbor of all the nodes in the same component, the Protocol rejects with high probability.

To verify 2., Merlin gives to each node the closest point $c_{i,j}$ in its component to the successor component, and the closest point from the predecessor[2] component to its component. The nodes needs to verify the correctness of the closest point to the successor component (which nodes can verify locally if they are neighbors of all the other nodes in the component), and that, its successor receive the same value. To do this last verification, we use the Linear Assignation Protocol to verify a correct assignation by Merlin. If the nodes can verify this, if v and u are ghosts, assuming w.l.o.g., v is successor of u in vertical or horizontal order, then v rejects, because Merlin sent the right closest point of u's component, which is at least a point in S_u, and then v rejects because it realizes intersect a square containing the closest point given by Merlin.

For example, consider the following graph and a model given by Merlin

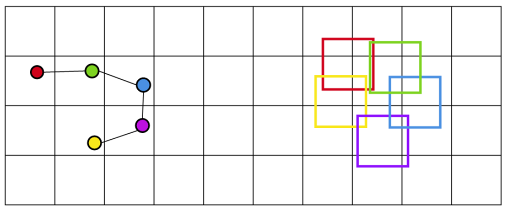


Figure 4.7: To the left, a path with 5 nodes and to the right an invalid unit square model

In the model to the right, red and yellow nodes are ghosts,and then one of them should reject. In this case, the red node is successor of yellow node in vertical order. So, if Merlin

[2]successor and predecessor respect both vertical and horizontal order

gave to the red node the correct closest point to component (i, j) in component $(i, j-1)$, then this point is included in S_v, but then the red node realizes there is a square intersecting it in component $(i, j-1)$, but it doesn't has a neighbor in that component, so reject. If the red node doesn't have this information cannot realizes it has a ghost in component $(i, j-1)$.

Observation: The graph above is a unit square graph and has a valid unit square model, but this model in specific is invalid.

To verify 3., if there exists ghosts u, v that are predecessor/successor in some diagonal order, in this case Merlin give to each component its lower-(right/left) stair, and the lower-(right/left) stair of the next component in both diagonal orders, then if two nodes u, v are ghosts, suppose u is in component $(i-1, j-1)$ and v in component (i, j). If u and v have a common neighbor ω in component $(i-1, j-1)$ or (i, j), then if u, v shares how many neighbors they have in its neighbor components, ω rejects because can verify this value. For example if ω is in component (i, j), is neighbor of all the nodes in that component (if not, is guaranteed that the Protocol rejects by above argument), so is is neighbor of u, ω receives from u its square S_u and can calculate how many nodes in (i, j)-component are its neighbors (if v is ghost of u then the number k of neighbors in (i, j)-component given by u to ω is smaller than the real).

If u and v don't have a common neighbor in one of its components and they are ghosts, then u has to cross the lower-left stair of component (i, j). Assuming that v is part of the lower-left stair (if v is not part of lower-left stair, exists other node z in (i, j)-component ghost of u), then it is enough for the nodes in component (i, j) and $(i-1, j-1)$ to both construct part of the lower-left stair of component (i, j) where S_u crosses the stair, because if its construction is equal, then nodes in $(i-1, j-1)$-component notice S_u crosses the lower-left stair of (i, j)-component and then at least one of them rejects.

For example, consider the following unit square model for the same graph of Figure 4.7

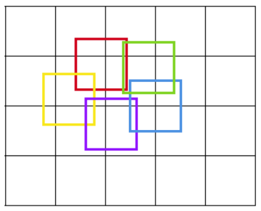


Figure 4.8: Invalid model for the path of 5 nodes in Figure 4.7

In the model of Figure 4.8, the red and yellow nodes are ghosts again, and if red node is in component (i, j), yellow node is in component $(i-1, j-1)$. Notice that u crosses the lower-left stair of component (i, j) (which it only step is given by v), so if Merlin gives to u that portion of the lower-left stair of (i, j), then u notices that intersects a node in component (i, j) and rejects. To prevent Merlin of sending an invalid lower-left stair of component (i, j) to the nodes in component $(i-1, j-1)$, Linear Assignation Protocol is used to verify the lower-left stair constructed by the nodes in component (i, j) (they can construct it locally) is the same that the one given to the nodes in its predecessor component. As the size of a stair is $\mathcal{O}(n)$, compare the entire stair can need $\mathcal{O}(n \log n)$ bits, so in order to shrink the message interchanged, it is necessary to hash the stairs to be compared.

Now, we present the formal protocol to recognize Unit-Square-Recognition .

Protocol 4.15 In the *First round*, Merlin sends to each node $v \in V$:

1. The total number of nodes n, and the corresponding certification of this value according to Protocol of Theorem 1.8;

2. a bit $rb(v)$ indicating if v is representative of its component ($rb(v) = 1$) or not ($rb(v) = 0$) and the number of nodes in that component $g(v)$. If $rb(v) = 0$, also sends the representative's ID of the component, and if $rb(v) = 1$, the position $\pi_h(v), \pi_v(v), \pi_{dp}(v)$ and $\pi_{dn}(v)$ of the (i,j)-component respect horizontal, vertical, diagonal positive and diagonal negative order, respectively, and the corresponding predecessor component in each order.

3. Number of neighbors $a_{k,h}$ in each neighbor's component of S_v, with $k, h \in \{i-1, i, i+1\} \times \{j-1, j, j+1\}$.

4. Values $f_1^h(v)$ and $f_1^v(v)$ representing the maximum values x_0 and y_0, satisfying that there exists z_0, w_0 such that (x_0, z_0) and (w_0, y_0) are in some square of the predecessor component, respect to horizontal and vertical order, respectively.

5. Values $f_2^h(v)$ and $f_2^v(v)$ representing the maximum values x_1 and y_1, satisfying that there exists z_1, w_1 such that (x_1, z_1) and (w_1, y_1) are in some square of the component, respectively.

6. A bit s_{dp} and s_{dn} indicating if the node v is part of the lower-left and lower-right stair, respectively, and the step $step_{ll}(v)$ and $step_{lr}(v)$ defined by S_v in each stair, respectively.

7. If $s_{dp} = 1$ (resp. $s_{dn} = 1$), a bit t_{dp} (resp. t_{dn}) to indicate if there is a node in the predecessor's component respect diagonal positive (resp. negative) order which upper-right vertex (resp. upper-left vertex) is under the step defined by S_v in the lower-left (resp. lower-right) stair, respectively.

8. If S_v is under some step $step(\omega_1)$ of component $(i+1, j+1)$-component, the $step(\omega_1)$. If S_v is under some step $step(\omega_2)$ of component $(i-1, j+1)$-component, the $step(\omega_2)$.

In the *Second Round* each node $v \in V$ sends to Merlin

- A random string $i_1(v)$ according to Corresponding Order protocol to verify each of four orders.

- A random string $i_2(v) \in [X]$ with $X = |\mathcal{H}_{n^2, q_1}|$ denoting a hash function $h_{i_2(v)}$ to verify correctness of lower-left and lower-right stair given by Merlin in next round

- A random string $i_3(v) \in [K]$ with $K = |\mathcal{H}_{nq_1, q_2}|$ indicating a hash function $h_{i_2(v)} \in \mathcal{H}_{nq_1, q_2}$ to verify a correct distribution of the hashed stairs between the nodes and its predecessors, given by Merlin in the next round.

- A random string according to Equality protocol to verify $\{f_1^v(v)\}_{v\in V}= \{f_2^v(v)\}_{v\in V}$ and $\{f_1^h(v)\}_{v\in V} = \{f_2^h(v)\}_{v\in V}$

In the *Third Round*, Merlin sends to each node $v \in V$, which square S_v is in component (i, j):

- The information according to Corresponding Order Protocol to verify the positions respect the four orders given.
- The information needed according to Equality Protocol, to verify $\{f_1^v(v)\}_{v\in V} = \{f_2^v(v)\}_{v\in V}$ and $\{f_1^h(v)\}_{v\in V} = \{f_2^h(v)\}_{v\in V}$.
- Two hashed vectors $st_r^c(v)$ and $st_r^s(v)$ claimed to be

  $$st_p^c(v) = h_{i_2(r)}\left(step_{ll}(v_1), ..., step_{ll}(v_k))\right)$$

  $$st_p^s(v) = h_{i_2(r)}\left(step_{ll}(\omega_1), ..., step_{ll}(\omega_{k_2})\right)$$

  Being $v_1, ..., v_{k_1}$ the nodes in the lower-left stair and $t_{dp} = 1$.

  Analogously, $step(\omega_1), ..., step(\omega_{k_2})$ the lower-left stair steps of component (i+1,j+1) such that there is a node in the (i,j)-component with right upper vertex under that steps. (Merlin gave this steps in point 8. of round 1). ´
- Analogously, give the same hash values $st_n^c(v)$ and $st_n^s(v)$ according to the lower-right stair and negative diagonal order.
- The information needed according to Linear Assignation Protocol, using the hash function $h_{i_3(r)} \in \mathcal{H}_{np,q}$ given by the root.

Then, in the Verification round, each node shares all the messages given by Merlin. Then each node v verifies:

1. Correctness of number-node n according to Protocol 1.8.
2. that $S_v \cap S_u \neq \phi$ for all $u \in N(v)$ and $g(v) = g(z)$ for all neighbors z in the same component.
3. Using the spanning three that $\sum_{v\in R} g(v) = n$ with R set of representatives.
4. Correctness of positions in the four orders using verification round of Corresponding Order Protocol.
5. That $\{f_1^v(v)\}_{v\in V} = \{f_2^v(v)\}_{v\in V}$ and $\{f_1^h(v)\}_{v\in V} = \{f_2^h(v)\}_{v\in V}$ according to Equality Protocol.
6. Every node checks the is a unique representative of its component.

7. If ω is the component's representative, v verifies that $f_1^v(\omega)$ and $f_1^h(\omega)$ are not in S_v if $a_{i-1,j}(v) = 0$ and $a_{i,j-1}(v) = 0$, respectively. (If v has no neighbors in components $(i-1,j)$ and $(i,j-1)$, then the closest points $f_1^v(\omega)$, $f_1^h(\omega)$ of this components to (i,j) has to be outside S_v)

8. Correctness of $st_p^c(v)$ and $st_n^c(v)$ reconstructing the respective hash with the vertex of all squares in the component such that $s_{dp} = 1$ and $s_{dn} = 1$, respectively.

9. Correctness of $\{st_p^c(v)\}_{v\in V}$ and $\{st_p^s(v)\}_{v\in V}$ respect DP order and $\{st_n^c(v)\}_{v\in V}$ and $\{st_n^s(v)\}_{v\in V}$ respect DN order using verification round of Linear Assignation Protocol.

Node v accepts if the above verification are satisfied. If not, rejects.

In order to verify the Correctness and Soundness of the above Protocol, consider the following proposition.

Proposition 4.16 Suppose there is two nodes u and v that are ghosts in the square model given by Merlin in the above Protocol. If v is in the component (i,j), then is satisfied

1. u is in a neighbor component of v

2. If u is in component (i,j), then in order to cheat the nodes, Merlin has to say to two different nodes in some component, that they are the representative. These two representatives are ghosts between them too.

3. If u is in component (i+1,j), (i-1,j), (i,j+1) or (i,j-1) and $a_{k,h}(v) = 0$[a], in order to cheat the nodes, Merlin send wrong value $f_1^v(u)$, $f_1^v(v)$, $f_1^h(u)$ or $f_1^h(v)$, respectively.

4. If u is in component (i-1,j-1), (i+1,j-1), (i-1,j+1) or (i+1,j+1) and $a_{k,h}(u) = 0$ and 6. of Verification round is satisfied, then in order to cheat the nodes, Merlin sent wrong value $st_p^s(u)$, $st_n^s(v)$, $st_n^s(u)$ or $st_p^s(v)$, respectively.

[a]The corresponding k, h depending on which component is u

Proof. Given two nodes u and v ghosts such that v is in component (i,j).

1. Direct because S_u and S_v are unit square.

2. If u, v are ghosts and there is a unique representative vertex ω in its component, then ω if neighbor of v and u (if not u or v doesn't have a representative) and checks that $g(\omega) = g(v) = g(u)$. But $g(v)$ and $g(u)$ are less than the real number of nodes in that component, so if there is only one representative $r_{i,j}$ by component, then $\sum g(r_{i,j}) < n$, because $g(\omega)$ is less that the rial number nodes in that component, and each representative has $g(r_{i,j})$ bounded by the rial number of nodes in the component.

   Then, there have to be two representatives in some component to achieve $\sum g(r_{i,j}) = n$ in the verification round.

3. We prove that if u is in component (i-1,j), then Merlin send wrong value $f_1^v(v)$ (the other cases are analogous).

   If u is in component (i-1,j) and $S_u \cap S_v \neq \emptyset$, then the point (i, a) closest to (i, j) and contained in some square of (i-1,j) component is at least (i, b), the point in the upper boundary of S_u. But is Merlin send the real value $f_1^v(v)$, v can checks that this point is in S_v and then should be neighbor of a node with square in component (i-1,j); but Merlin said that $a_{i-1,j}(v) = 0$, so v rejects. So in order to cheat, Merlin has to send a wrong value $f_1^v(v)$, lower than the real.

4. We prove that if S_u is in component (i-1,j-1), then Merlin send a wrong value $st_p^s(u)$, the other cases are analogous.

   If $a_{h,k} = 0$, then Merlin said to u that his upper-right vertex is under some step $step(u)$ of S_v's lower-left stair. But this step is not a real step of S_v's lower-left stair, because as $S_u \cap S_v$, then the lower-left stair of v 'goes through' S_u, but the step given to u has to be one that makes the suppose lower-left stair doesn't go through S_u.

□

With this, know it can be prove the Completeness and Soundness of the Protocol.

Completeness. Suppose that G is a unit square graph. An honest prover just has to send the real number of nodes n, and correct information according to a valid unit square model $\{S_v\}_{v \in V}$. Then, all the hashed values have to be correct, so with probability $\geq \frac{2}{3}$ all nodes accept.

Soundness. Suppose G is not a unit square graph. If a dishonest prover provides a wrong value of n, or a wrong spanning tree, then the nodes reject by correctness of Protocol given by Theorem 1.8.

If all nodes $v \in V$ check that $S_v \cap S_u \neq \emptyset$ for all $u \in N(v)$ using the unit square model given by Merlin, then, as G is not a unit square graph, there exists two nodes $u, v \in V$ that are ghosts between them.

There are three cases. Lets prove in each case that at least one node rejects with probability $\geq \frac{2}{3}$.

If node v is in component (i, j), then by Proposition 4.16 the node u is in a neighbor component of (i, j).

First, if u is in the same component (i, j), by Proposition 4.16 there is a component (k_1, k_2) with two different representatives ω_1 and ω_2. Merlin said to both they are in positions $\pi_v(\omega_1)$ and $\pi_v(\omega_2)$ about vertical order. By Correctness of Corresponding Order Protocol, w.l.o.g, if $\pi(\omega_1) < \pi(\omega_2)$, then $\pi(\omega_2) = \pi(\omega_1) + 1$ (are consecutive), because if not, exists a node ω_3 such that $\pi(\omega_1) < \pi(\omega_3) < \pi(\omega_2)$, and has ω_3 is in a different component, and its component cannot be in the same time greater and lower than (i, j)-component[3], so by

[3]If ω_3 is in the same component, we can iterate the argument

correctness of CORRESPONDING ORDER Protocol, it rejects with probability $\geq \frac{2}{3}$. Then $\pi(\omega_2) = \pi(\omega_1) + 1$ and Merlin should say to ω_1 that ω_2 is it successor and it is in the same component, so ω_1 rejects because they check there is other representative in the component.

Second, if u is in component (i+1,j), (i-1,j), (i,j+1) or (i,j-1), then Merlin cheats in a value $f_1^v(v), f_2^v(v), f_1^h(v)$ or $f_2^h(v)$.

If Merlin cheats on $f_1^v(v)$ or $f_1^h(v)$, as v with probability $\geq \frac{2}{3}$ is neighbor of all the nodes in its component, with the same probability v can verify the correctness of $f_1^v(v)$ or $f_1^2(v)$ (it can calculate the values $f_1^v(v)$ or $f_1^2(v)$ if it is neighbor of all nodes in the component), then v rejects with probability $\geq \frac{2}{3}$.

If Merlin does not cheat on $f_1^v(v)$ and $f_1^h(v)$, but cheats in some $f_2^v(v)$ or $f_2^h(v)$, then exists nodes ω, v predecessor/successor in the respective order such that $f_1^v(\omega) \neq f_2^v(v)$ or $f_1^h(\omega) \neq f_2^h(v)$, and then LINEAR ASSIGNATION Protocol rejects with probability $\geq \frac{2}{3}$.

Finally, if u is in component (i-1,j-1), (i+1,j-1), (i-1,j+1) or (i+1,j+1). Again two cases

1. If $a_{k,h}(v) > 0$, then v has a neighbor z in component (k, h) which is neighbor of u with probability greater than $2/3$[4]. The value $a_{k,h}(v)$ is smaller than the real and z can count the rial number of square intersecting S_v and then, z rejects because check that the value $a_{k,h}(v)$ is incorrect.
2. If $a_{k,h} = 0$, by Proposition 4.16, Merlin sent a wrong hash value $st_p^s(u)$, $st_n^s(v)$ or $st_n^s(u)$ (depending in which component is u). If u is in $(i + 1, j + 1)$, then Merlin sent wrong value $st_p^s(v)$ to v. If he sent the same wrong hash value to u to satisfy $st_p^c(u) = st_p^s(v)$, then u rejects with probability $\geq \frac{2}{3}$ because the nodes in (i+1,j+1) can evaluate the correct step's stair and validate that hashed value is not equal to $st_p^c(u)$. If Merlin sent correct hashed value $st_p^c(u)$ to u, then $st_p^c(u) \neq st_p^s(v)$ and with probability $\geq \frac{2}{3}$, at least one node rejects by Soundness of LINEAR ASSIGNATION Protocol.

Then, in any case, at least one node rejects with probability $\geq \frac{2}{3}$ □

Proof-size Analysis. The messages interchanged in the Protocol are square centers (we assume are encoded in $\mathcal{O}(\log n)$), ID's, positions in four orders and single bits, being of polynomial size in n the last three, thus the proof-size of Protocol 4.15 is $\mathcal{O}(\log n)$.

[4] v and ω are neighbors, and with probability $\geq \frac{2}{3}$ ω is neighbor of all nodes in component (k, h), and then neighbor of u

Chapter 5

Lower Bounds

In this section logarithmic lower-bounds are given in the certificate sizes of any PLS that recognizes the class of permutation, trapezoid, circle or polygon-circle graphs. In order to so, it is used a technique given by Fraigniaud et al [63], 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 5.1 ([63]) 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 5.2 ([63]) 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 proof-labeling scheme for $\mathcal{P}$ and $\mathcal{F}$ is $\Omega\left(\dfrac{log(r)}{s}\right)$.

5.1 Permutation and Trapezoid Graphs

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. 5.1 is depicted the graph Q_3 and its corresponding model.

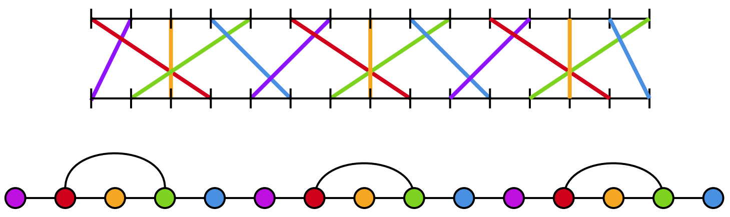


Figure 5.1: Graph Q_3 and a permutation model for Q_3.

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 5.1 *For each $i \neq j$, the graph $\sigma^{ij}_{\bowtie}(Q_n)$ it is neither a permutation graph nor a trapezoid graph with $\sigma_i \colon V(H_1) \to V(H_i)$ such that $\sigma_i(v_3) = v_{5i-2}$ and $\sigma_i(v_4) = v_{5i-1}$.*

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. 5.2 for an example).

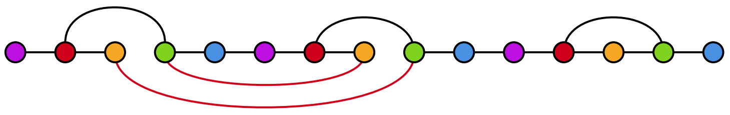


Figure 5.2: Graph $\sigma^{12}_{\bowtie}(Q_3)$, where in red are represented the crossing edges. Observe that this graph is not a trapezoid graph, as it contains an induced cycle of length 4.

As a trapezoid graph have induced cycles of length at most 4, we deduce that $\sigma^{ij}_{\bowtie}(Q_n)$ is not a trapezoid graph.

Finally, as the class of permutation of permutation graph is contained in the class of trapezoid graphs, then Q_n neither it is a permutation graph. □

By Theorem 5.2 and the above result, the lower bound result is direct.

Theorem 5.2 *Any PLS for* Permutation-Recognition *or* Trapezoid-Recognition *has a proof-size of $\Omega(\log n)$ bits.*

Proof. MQn is a permutation graph (and then also a trapezoid graph) for each $n > 0$, and by Lemma 5.1 $\sigma_i(Q_n)$ is neither a permutation graph nor a trapezoid graph. As there are $r = \mathcal{O}(n)$ such isomorphism σ_i and subgraphs H_i, each one with one edge, then by theorem 5.2, any PLS that recognize .r k-Polygon-Circle-Recognition needs a proof-size of $\Omega(\log n)$ bits. □

5.2 Circle And k-polygon-circle Graphs

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 5.3 is depicted the graph M_4 and its corresponding model.

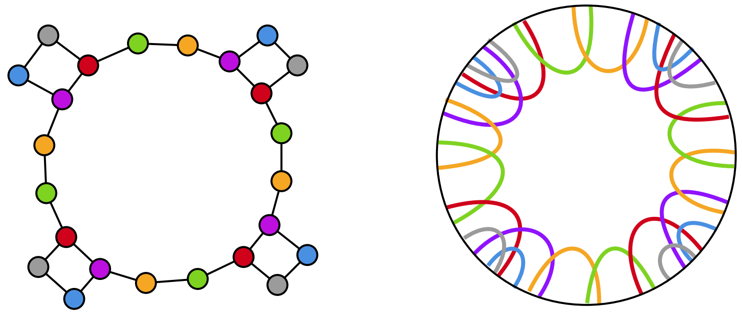


Figure 5.3: Graph M_4 and a permutation model for M_4.

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 $\sigma_i(5n+1) = v_{4n+i}$.

Lemma 5.3 *For every $k > 0$ and each $i \neq j$, the graph $\sigma^{ij}_{\bowtie}(M_n)$ it is neither a k-polygon-circle graph nor a circle graph with $\sigma_i \colon V(H_1) \to V(H_i)$ defined as $\sigma_i(4n+1) = v_{5n+i}$ and $\sigma_i(5n+1) = v_{4n+i}$.*

Proof.

First, observe that in $\sigma^{ij}_{\bowtie}(M_n)$ 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. 5.4 for a representation of $\sigma^{i,i+1}_{\bowtie}(M_4)$.

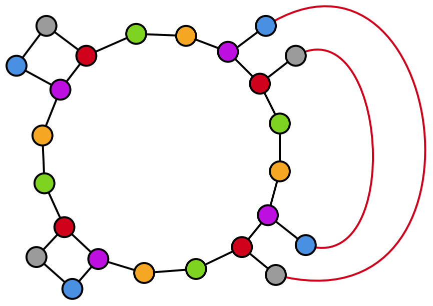


Figure 5.4: Graph $\sigma^{i,i+1}_{\bowtie}(M_4)$, where in red are represented the crossing edges.

Claim 5.3 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 $k > 0$.

Proof. 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 claim 5.3, we deduce that the graph induced by $C_1 \cup C_2$ is not a k-polygon circle graph. Since the class of polygon-circle graphs is hereditary, we deduce that $\sigma^{ij}_{\bowtie}(M_n)$ it is not a k-polygon-circle graph.

Finally, as the class of circle graphs is contained in the class of k-polygon circle graph, then M_n neither it is a circle graph. □

Direct by Theorem 5.2 and Lemma 5.3 we deduce the following result.

Theorem 5.4 *Any PLS for* Circle-Recognition *or k-*Polygon-Circle-Recognition *has a proof-size of* $\Omega(\log n)$ *bits.*

Proof. M_n is a circle graph (and then also a k-polygon circle graph) for each $n > 2$, and by Lemma 5.3 $\sigma_i(M_n)$ is neither a circle graph nor a k-polygon circle graph. As there are $r = \mathcal{O}(n)$ such isomorphism σ_i and subgraphs H_i, each one with 1 edge, then by theorem 5.2, any PLS that recognize .r k-Polygon-Circle-Recognition needs a proof-size of $\Omega(\log n)$ bits. □

5.3 Unit Square Graph

Finally, to show a logarithmic lower bound on any PLS recognizing Unit-Square-Recognition consider the following graphs.

For all $p \in \mathbb{N}$, consider S_p to be a graph defined recursively as:

- If $p = 1$, then G_1 is as follow

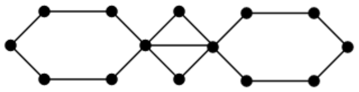
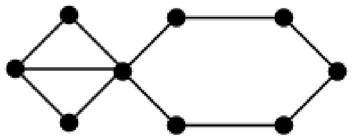
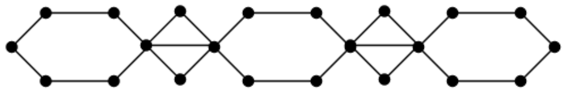


Figure 5.5: Graph S_1

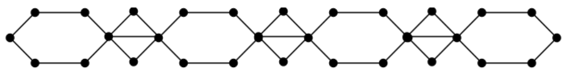
- For $p > 1$, S_p consists in add the following graph, denoted as increase-graph, in one endpoint of S_{p-1}.



Then, for example, S_2 and S_3 are as follow



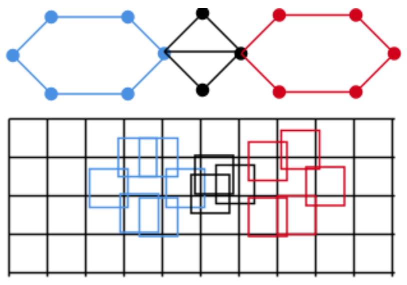
(a) Graph S_2



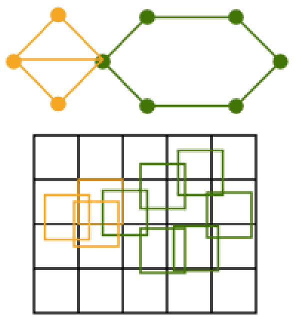
(b) Graph S_3

Proposition 5.4 For all p, S_p is a unit square graph.

Proof. Notice that the following are valid unit square models for S_1 and increase-graph.



(a) Unit Square Model of S_1



(b) Unit Square Model of $increase - graph$

Then, can be built unit square model for each S_p joining the unit square models of S_1 and increase-graph.

□

Now, for all S_p, consider the set of nodes $H_1, ..., H_r \subset V(S_p)$ to be each H_i the two nodes v_1^i, v_2^i which are in C_6 and the diamond.

For example, in S_2 there are two such sets $H_1 = \{v_1, v_2\}$ and $H_2 = \{v_3, v_4\}$, being v_1, v_2, v_3 and v_4 the nodes marked in the following Figure

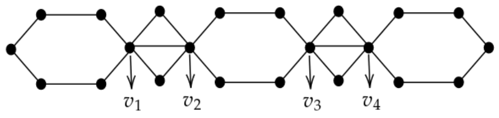
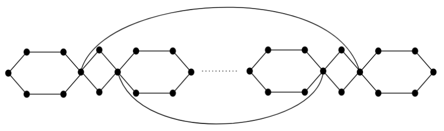


Figure 5.8: Sets H_1 and H_2 in S_2

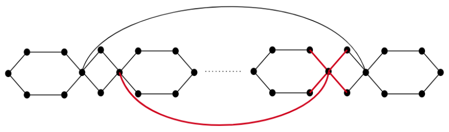
Notice that by construction, the set of nodes $H_1, ..., H_r$ are pairwise independent, isomorphism and $r = \mathcal{O}(n)$.

Lemma 5.5 For each S_p and subsets $H_1, ..., H_r$ as describe it above, the graph $\sigma_{\bowtie}^{ij}(S_p)$ is not a unit square graph. Where the isomorphism σ_i is one of the possible bijection between H_1 and H_i.

Proof. By contradiction, suppose there exists S_p and $i < j$ such that $\sigma_{\bowtie}^{ij}(S_p)$ is a unit square graph. For each $i < j$, the stair of 6 nodes is a subgraph of $\sigma_{\bowtie}^{ij}(S_p)$ as shown in the following Figure



(a) Graph $\sigma_{\bowtie}^{ij}(S_p)$. New edges drawn with orange.



(b) Stair subgraph of 6 nodes marked with red.

As observation, it's easy to notice that the same subgraph is obtained if we use the other possible bijection as isomorphism. Denote by E_6 the stair graph of 6 nodes.

Now, as the class of unit squares are hereditary, E_6 should also be a unit square graph, so in order to conclude, we prove that E_6 is not a unit square graph.

Claim 5.6 E_6 is not a unit square graph.

Proof. Notice that the maximum number of unit squares intersecting just a fixed unit square and no intersecting between them is 4, then it's impossible to have a unit square model in which 5 squares no intersect between them but yes intersect another fixed square. □

And then, $\sigma_{\bowtie}^{ij}(S_p)$ cannot be a unit square graph

$\square$

Theorem 5.7 Any PLS for Unit-Square-Recognition has a proof-size of $\Omega(\log n)$ bits.

Proof.

S_n is a is a unit square graph for all $n \geq 1$, and by Lemma 5.5 $\sigma_i(M_n)$ is not a unit square graph for all possible bijection between H_1 and H_i. As there are $r = \mathcal{O}(n)$ such isomorphism σ_i and $\mathcal{O}(n)$ subgraphs H_i, each one with 1 edge, then by theorem 5.2, any PLS that recognize Unit-Square-Recognition needs a proof-size of $\Omega(\log n)$ bits.

$\square$

Bibliography

[1] NLHPC's Guacolda-Leftraru's infraestructure. `https://www.nlhpc.cl/infraestructura/`. Accessed: 2020-08-14.

[2] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory (TOCT)*, 1(1):1–54, 2009.

[3] Mohamed Ibrahim Abouelhoda and Enno Ohlebusch. Chaining algorithms for multiple genome comparison. *Journal of Discrete Algorithms*, 3(2-4):321–341, 2005.

[4] Ittai Abraham, Danny Dolev, and Dahlia Malkhi. Lls: A locality aware location service for mobile ad hoc networks. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, DIALM-POMC '04, page 75–84, New York, NY, USA, 2004. Association for Computing Machinery.

[5] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking Computing*, MobiHoc '02, page 157–164, New York, NY, USA, 2002. Association for Computing Machinery.

[6] 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.

[7] Sepehr Assadi, Gillat Kol, and Rotem Oshman. Lower bounds for distributed sketching of maximal matchings and maximal independent sets. In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, pages 79–88. ACM, 2020.

[8] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Low-diameter graph decomposition is in nc. In *Scandinavian Workshop on Algorithm Theory*, pages 83–93. Springer, 1992.

[9] László Babai and Peter G Kimmel. Randomized simultaneous messages: Solution of a problem of Yao in communication complexity. In *Computational Complexity. Twelfth Annual IEEE Conference*, pages 239–246. IEEE, 1997.

[10] 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.

[11] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)*, 48(5):1069–1090, 2001.

[12] Mor Baruch, Pierre Fraigniaud, and Boaz Patt-Shamir. Randomized proof-labeling schemes. In *Symposium on Principles of Distributed Computing*, pages 315–324, 2015.

[13] Florent Becker, Martin Matamala, Nicolas Nisse, Ivan Rapaport, Karol Suchan, and Ioan Todinca. Adding a referee to an interconnection network: What can (not) be computed in one round. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 508–514. IEEE, 2011.

[14] Florent Becker, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. The simultaneous number-in-hand communication model for networks: Private coins, public coins and determinism. In *International Colloquium on Structural Information and Communication Complexity*, pages 83–95. Springer, 2014.

[15] Florent Becker, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. The impact of locality in the broadcast congested clique model. *SIAM Journal on Discrete Mathematics*, 34(1):682–700, 2020.

[16] 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.

[17] Hans L Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1-2):1–45, 1998.

[18] John A Bondy and Miklós Simonovits. Cycles of even length in graphs. *Journal of Combinatorial Theory, Series B*, 16(2):97–105, 1974.

[19] Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Local certification of graph decompositions and applications to minor-free classes. *arXiv preprint arXiv:2108.00059*, 2021.

[20] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey.* Society for Industrial and Applied Mathematics, January 1999.

[21] Andreas Brandstadt, Jeremy P Spinrad, et al. *Graph classes: a survey*, volume 3. Siam, 1999.

[22] Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Computational Geometry*, 9(1-2):3–24, January 1998.

[23] H.J. Broersma, E. Dahlhaus, and T. Kloks. A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. *Discrete Applied Mathematics*, 99(1-3):367–400, February 2000.

[24] Keren Censor-Hillel, Petteri Kaski, Janne H Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. *Distributed Computing*, 32(6):461–478, 2019.

[25] Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theoretical Computer Science*, 2018.

[26] Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theoretical Computer Science*, 811:112–124, 2020.

[27] Yaotsu Chang, Chong-Dao Lee, and Keqin Feng. Multivariate interpolation formula over finite fields and its applications in coding theory. *arXiv preprint arXiv:1209.1198*, 2012.

[28] 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.

[29] Serafino Cicerone, Gabriele Di Stefano, and Michele Flammini. Compact-port routing models and applications to distance-hereditary graphs. *Journal of Parallel and Distributed Computing*, 61(10):1472–1488, 2001.

[30] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, December 1990.

[31] Francisco Claude, Gautam K Das, Reza Dorrigiv, Stephane Durocher, Robert Fraser, Alejandro López-Ortiz, Bradford G Nickerson, and Alejandro Salinger. An improved line-separable algorithm for discrete unit disk cover. *Discrete Mathematics, Algorithms and Applications*, 2(01):77–87, 2010.

[32] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. Spanner: Googleâ€™s globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):1–22, 2013.

[33] D.G. Corneil, H. Lerchs, and L.Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, July 1981.

[34] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, March 2000.

[35] 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.

[36] Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, volume 146 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[37] Ido Dagan, Martin Charles Golumbic, and Ron Yair Pinter. Trapezoid graphs and their coloring. *Discrete Applied Mathematics*, 21(1):35–46, 1988.

[38] Elias Dahlhaus. Efficient parallel recognition algorithms of cographs and distance hereditary graphs. *Discrete applied mathematics*, 57(1):29–44, 1995.

[39] Guillaume Damiand, Michel Habib, and Christophe Paul. A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. *Theoretical Computer Science*, 263(1-2):99–111, July 2001.

[40] Guillaume Damiand, Michel Habib, and Christophe Paul. A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. *Theoretical Computer Science*, 263(1-2):99–111, 2001.

[41] Alessandro D'Atri and Marina Moscarini. Distance-hereditary graphs, steiner trees, and connected domination. *SIAM Journal on Computing*, 17(3):521–538, June 1988.

[42] R. Diestel. *Graph Theory.* Electronic library of mathematics. Springer, 2006.

[43] Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173, 2005.

[44] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *ACM symposium on Principles of distributed computing*, pages 367–376, 2014.

[45] Guillermo Durán, Luciano N Grippo, and Martín D Safe. Structural results on circular-arc graphs and circle graphs: A survey and the main open problems. *Discrete Applied Mathematics*, 164:427–443, 2014.

[46] Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. Sublinear-time distributed algorithms for detecting small cliques and even cycles. In *33rd International Symposium on Distributed Computing (DISC 2019).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[47] Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of bodlaender and courcelle. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 143–152. IEEE, 2010.

[48] Milan Erdelj, Tahiry Razafindralambo, and David Simplot-Ryl. Covering points of interest with mobile sensors. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):32–43, 2012.

[49] P Erdös. On extremal problems of graphs and generalized graphs. *Israel Journal of Mathematics*, 2(3):183–190, 1964.

[50] Paul ErdHos and Alfréd Rényi. Asymmetric graphs. *Acta Mathematica Academiae Scientiarum Hungarica*, 14(3-4):295–315, 1963.

[51] Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, et al. Three notes on distributed property testing. In *31st International Symposium on Distributed Computing (DISC 2017).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[52] 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.

[53] Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and Ioan Todinca. Compact distributed certification of planar graphs. *Algorithmica*, pages 1–30, 2021.

[54] Laurent Feuilloley and Juho Hirvonen. Local verification of global proofs. *arXiv preprint arXiv:1803.09553*, 2018.

[55] Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and impossibilities for distributed subgraph detection. In *Symposium on Parallelism in Algorithms and Architectures*, pages 153–162, 2018.

[56] Orr Fischer, Rotem Oshman, and Uri Zwick. Public vs. private randomness in simultaneous multi-party communication complexity. In *Proc. of the International Colloquium on Structural Information and Communication Complexity*, volume 9988 of *Lecture Notes in Computer Science*, pages 60–74, 2016.

[57] Peter C Fishburn. *Interval orders and interval graphs: A study of partially ordered sets*. Wiley New York, 1985.

[58] Florent Foucaud, George B Mertzios, Reza Naserasr, Aline Parreau, and Petru Valicov. Identification, location–domination and metric dimension on interval and permutation graphs. i. bounds. *Theoretical Computer Science*, 668:43–58, 2017.

[59] Florent Foucaud, George B Mertzios, Reza Naserasr, Aline Parreau, and Petru Valicov. Identification, location-domination and metric dimension on interval and permutation graphs. ii. algorithms and complexity. *Algorithmica*, 78(3):914–944, 2017.

[60] Pierre Fraigniaud, Mika Göös, Amos Korman, Merav Parter, and David Peleg. Randomized distributed decision. *Distributed Computing*, 27(6):419–434, 2014.

[61] Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *Journal of the ACM (JACM)*, 60(5):1–26, 2013.

[62] 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.

[63] Pierre Fraigniaud, Boaz Patt-Shamir, and Mor Perry. Randomized proof-labeling schemes. *Distributed Computing*, 32(3):217–234, 2019.

[64] Stefan Funke, Alexander Kesselman, Ulrich Meyer, and Michael Segal. A simple improved distributed algorithm for minimum cds in unit disk graphs. *ACM Transactions on Sensor Networks (TOSN)*, 2(3):444–453, 2006.

[65] Frédéric Gardi. The roberts characterization of proper and unit interval graphs. *Discrete Mathematics*, 307(22):2906–2908, 2007.

[66] 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.

[67] Fanica Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73(5-6):181–188, 2000.

[68] Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks i: Planar embedding. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 29–38, 2016.

[69] Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed mst and routing in almost mixing time. In *ACM Symposium on Principles of Distributed Computing*, pages 131–140, 2017.

[70] Emeric Gioan and Christophe Paul. Split decomposition and graph-labelled trees: characterizations and fully dynamic algorithms for totally decomposable graphs. *Discrete Applied Mathematics*, 160(6):708–733, 2012.

[71] 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.

[72] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):27, 2015.

[73] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[74] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

[75] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., NLD, 2004.

[76] Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory of Computing*, 12(1):1–33, 2016.

[77] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Berlin Heidelberg, 1993.

[78] Michel Habib, Ross McConnell, Christophe Paul, and Laurent Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1-2):59–84, 2000.

[79] Magnús M Halldórsson and Christian Konrad. Distributed algorithms for coloring interval graphs. In *International Symposium on Distributed Computing*, pages 454–468. Springer, 2014.

[80] 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.

[81] Xin He. Parallel algorithm for cograph recognition with applications. *Journal of Algorithms*, 15(2):284–313, 1993.

[82] Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Large cuts with local algorithms on triangle-free graphs. *arXiv preprint arXiv:1402.2543*, 2014.

[83] Chính T Hoàng. Efficient algorithms for minimum weighted colouring of some classes of perfect graphs. *Discrete Applied Mathematics*, 55(2):133–143, 1994.

[84] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974.

[85] Edward Howorka. A characterization of distance-hereditary graphs. *The Quarterly Journal of Mathematics*, 28(4):417–420, 1977.

[86] Sun-Yuan Hsieh, Chin-Wen Ho, Tsan-Sheng Hsu, Ming-Tat Ko, and Gen-Huey Chen. Efficient parallel algorithms on Distance hereditary graphs. *Parallel Processing Letters*, 09(01):43–52, March 1999.

[87] Ruo-Wei Hung and Maw-Shang Chang. Linear-time algorithms for the hamiltonian problems on distance-hereditary graphs. *Theoretical Computer Science*, 341(1-3):411–440, September 2005.

[88] Taisuke Izumi and François Le Gall. Triangle finding and listing in congest networks. In *ACM Symposium on Principles of Distributed Computing*, pages 381–389, 2017.

[89] H.A Jung. On a class of posets and the corresponding comparability graphs. *Journal of Combinatorial Theory, Series B*, 24(2):125–133, April 1978.

[90] 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.

[91] 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.

[92] 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.

[93] Jarkko Kari, Martín Matamala, Ivan Rapaport, and Ville Salo. Solving the induced subgraph problem in the randomized multiparty simultaneous messages model. In *International Colloquium on Structural Information and Communication Complexity*, pages 370–384. Springer, 2015.

[94] David G Kirkpatrick and T Przytycka. Parallel recognition of complement reducible graphs and cotree construction. *Discrete applied mathematics*, 29(1):79–96, 1990.

[95] D.G. Kirkpatrick and T. Przytycka. Parallel recognition of complement reducible graphs and cotree construction. *Discrete Applied Mathematics*, 29(1):79–96, November 1990.

[96] Lefteris M Kirousis and Dimitris M Thilikos. The linkage of a graph. *SIAM Journal on Computing*, 25(3):626–647, 1996.

[97] Ton Kloks. Treewidth of circle graphs. In *International Symposium on Algorithms and Computation*, pages 108–117. Springer, 1993.

[98] Gillat Kol, Rotem Oshman, and Raghuvansh Saxena. Interactive distributed proofs. pages 255–264, 07 2018.

[99] Gillat Kol, Rotem Oshman, and Raghuvansh R Saxena. Interactive distributed proofs. In *ACM Symposium on Principles of Distributed Computing*, pages 255–264. ACM, 2018.

[100] Christian Konrad and Viktor Zamaraev. 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*, pages 159–161, 2018.

[101] 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.

[102] Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20(4):253–266, 2007.

[103] Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22:215–233, 01 2005.

[104] Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.

[105] 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.

[106] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

[107] Ilan Kremer, Noam Nisan, and Dana Ron. Errata for: "On randomized one-round communication complexity". *Computational Complexity*, 10(4):314–315, 2001.

[108] Fabian Kuhn, Rogert Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing*, PODC '03, page 63–72, New York, NY, USA, 2003. Association for Computing Machinery.

[109] Eyal Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44. Elsevier, 1997.

[110] 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.

[111] Emmanuelle Lebhar and Zvi Lotker. Unit disk graph and physical interference model: Putting pieces together. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–8. IEEE, 2009.

[112] Christoph Lenzen, Yvonne Anne Oswald, and Roger Wattenhofer. What can be approximated locally? case study: Dominating sets in planar graphs. In *Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 46–54, 2008.

[113] R. Lin and S. Olariu. Fast parallel algorithms for cographs. In *Lecture Notes in Computer Science*, pages 176–189. Springer Berlin Heidelberg, 1990.

[114] 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.

[115] Tze-Heng Ma and Jeremy P Spinrad. On the 2-chain subgraph cover and related problems. *Journal of Algorithms*, 17(2):251–268, 1994.

[116] MS Madanlal, G Venkatesan, and C Pandu Rangan. Tree 3-spanners on interval, permutation and regular bipartite graphs. *Information Processing Letters*, 59(2):97–102, 1996.

[117] Swagata Mandal and Madhumangal Pal. Maximum weight independent set of circular-arc graph and its application. *Journal of Applied Mathematics and Computing*, 22(3):161–174, 2006.

[118] MV Marathe, HB Hunt II, and SS Ravi. Geometry based approximations for intersection graphs. In *Proceedings of the Fourth Canadian Conference on Computational Geometry*, pages 244–249, 1992.

[119] Ross M McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003.

[120] Colin McDiarmid and Tobias Müller. Integer realizations of disk and segment graphs. *Journal of Combinatorial Theory, Series B*, 103(1):114–143, January 2013.

[121] Terry A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. Society for Industrial and Applied Mathematics, January 1999.

[122] George B Mertzios and Ivona Bezáková. Computing and counting longest paths on circular-arc graphs in polynomial time. *Discrete Applied Mathematics*, 164:383–399, 2014.

[123] Anisur Rahaman Molla, Supantha Pandit, and Sasanka Roy. Optimal deterministic distributed algorithms for maximal independent set in geometric graphs. *Journal of Parallel and Distributed Computing*, 132:36–47, 2019.

[124] Pedro Montealegre, Sebastian Perez-Salazar, Ivan Rapaport, and Ioan Todinca. Two rounds are enough for reconstructing any graph (class) in the congested clique model. In *International Colloquium on Structural Information and Communication Complexity*, pages 134–148. Springer, 2018.

[125] Pedro Montealegre, Sebastian Perez-Salazar, Ivan Rapaport, and Ioan Todinca. Graph reconstruction in the congested clique. *Journal of Computer and System Sciences*, 2020.

[126] 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.

[127] Tobias Muller, Erik Jan van Leeuwen, and Jan van Leeuwen. Integer representations of convex polygon intersection graphs. *SIAM Journal on Discrete Mathematics*, 27(1):205–231, 2013.

[128] 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.

[129] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[130] Daniel Neuen. Graph isomorphism for unit square graphs. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 70:1–70:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[131] Ilan Newman and Mario Szegedy. Public vs. private coin flips in one round communication games. In *Proc. of the 28th ACM Symposium on Theory of Computing*, STOC '09, pages 561–570, 1996.

[132] SebastiÃ¡n Perez-Salazar. El problema de la degenerancia de grafos en congested clique. mathesis, Universidad de Chile, 2016.

[133] 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.

[134] Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Information and Computation*, 243:263–280, 2015.

[135] Diego Ramírez. Detecting graph classes in the distributed interactive proof model. *Mater's Thesis*, 2020.

[136] Omer Reingold, Guy N Rothblum, and Ron D Rothblum. Constant-round interactive proofs for delegating computation. In *ACM symposium on Theory of Computing*, pages 49–62. ACM, 2016.

[137] Fred S Roberts. Indifference graphs. proof techniques in graph theory. In *Proceedings of the Second Ann Arbor Graph Conference, Academic Press, New York*, 1969.

[138] Neil Robertson and Paul D Seymour. Graph minors. xx. wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[139] Donald J Rose, R Endre Tarjan, and George S Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

[140] D Seinsche. On a property of the class of n-colorable graphs. *Journal of Combinatorial Theory, Series B*, 16(2):191–193, April 1974.

[141] Adi Shamir. Ip= pspace. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.

[142] Jeremy Spinrad. Recognition of circle graphs. *Journal of Algorithms*, 16(2):264–282, 1994.

[143] David P. Sumner. Dacey graphs. *Journal of the Australian Mathematical Society*, 18(4):492–502, December 1974.

[144] Alexander Tiskin. Fast distance multiplication of unit-monge matrices. *Algorithmica*, 71(4):859–888, 2015.

[145] Alan Tucker. Characterizing circular-arc graphs. *Bulletin of the American Mathematical Society*, 76(6):1257–1260, 11 1970.

[146] 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.

[147] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *ACM symposium on Theory of computing*, pages 209–213, 1979.