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

DETECCIÓN DE CLASES DE GRAFOS EN EL MODELO INTERACTIVO DE VERIFICACIÓN DISTRIBUIDA.

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

DIEGO NICOLÁS RAMÍREZ ROMERO

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

MIEMBROS DE LA COMISIÓN:
MARCOS KIWI KRAUSKOPF
MARTÍN MATAMALA VÁSQUEZ

Este trabajo ha sido parcialmente financiado por Proyecto Fondecyt 1170021 y CMM ANID PIA AFB170001

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA, FECHA: 22/10/2020

DETECCIÓN DE CLASES DE GRAFOS EN EL MODELO INTERACTIVO DE VERIFICACIÓN DISTRIBUIDA.

En este trabajo se estudia el modelo interactivo de verificación distribuida. En este modelo hay dos entidades: un probador con poder ilimitado, identificado como Merlín, y un verificador distribuido identificado como Arturo, que corresponde a una red de comunicación representada por un grafo conexo G. Cada nodo del grafo interactúa con Merlín en una serie de interacciones en las que lanza monedas y recibe una respuesta de éste que puede depender de la topología de la red, así como de las interacciones pasadas. El objetivo de estas interacciones es convencer con buena probabilidad a los nodos de que la topología de la red satisface algún predicado. Luego del último mensaje enviado por Merlín, sigue una ronda de verificación entre cada nodo y sus vecinos. Se dirá que las interacciones son de tipo Arturo-Merlín (dAM) o Merlín-Arturo (dMA) dependiendo de si la ronda de verificación es determinista o aleatorizada, donde en el segundo caso los nodos realizan un nuevo lanzamiento de monedas previo al intercambio de mensajes.

El primer capítulo se centra en definir este modelo y sus variantes, además de presentar el marco teórico de este trabajo. En el segundo capítulo se describe el diseño de protocolos usando sólo una ronda de interacción para distintas clases de grafos. De aquí se desprenden protocolos eficientes para problemas como la detección de grafos de intervalos, grafos cordales, grafos de arcos circulares, entre otros. En el tercer capítulo, se continúa presentando protocolos interactivos, pero permitiendo múltiples rondas de interacción. Aquí se estudian problemas como la presencia de una cierta estructura como subgrafo o menor, para luego ver el caso de la ausencia de estructuras. Se desprenden protocolos para clases tales como cografos, grafos de distancia hereditaria y grafos libres de triángulos.

En el cuarto capítulo se deja de lado el diseño de protocolos para estudiar la construcción de cotas inferiores: mediante distintas técnicas se obtienen cotas inferiores para el cálculo de la degenerancia y la detección de cografos. Luego, se presenta una técnica general para obtener cotas inferiores basada en el pegado de soluciones correctas de forma de engañar a la red. Se obtienen cotas para una ronda y para múltiples rondas de interacción en que la aleatoriedad es compartida entre los nodos. De aquí se desprende que los resultados obtenidos para varios de los problemas anteriores son ajustados. Por último, se extiende un resultado previo sobre el problema de Simetría que consiste en decidir si un grafo admite un automorfismo no trivial. De aquí se obtiene una cota inferior en múltiples rondas de interacción cuando la fuente de aleatoriedad es privada para cada nodo.

Finalmente, en el quinto capítulo, se estudian los efectos sobre el modelo al cambiar la fuente de aleatoriedad, comparando el caso en que ésta es *compartida* entre los nodos de la red y el caso en que es *privada*. Se obtiene que hay diferencias en el poder del modelo dependiendo de si la última interacción le corresponde a Merlín o Arturo. Si ésta es de tipo dAM se tiene que el modelo a moneda privada admite protocolos con un costo asintóticamente más pequeño que el segundo, aún para múltiples rondas de interacción. Por otro lado, si Arturo es el último en interactuar, el uso de moneda compartida permite una mejora sustancial en el largo de los mensajes por sobre aquellos con moneda privada.

GRAPH CLASS DETECTION IN THE DISTRIBUTED INTERACTIVE PROOFS MODEL

In this work we study the distributed interactive proofs model. This model consists of two entities: an all-powerful prover identified as Merlin, and a distributed verifier identified as Arthur, which corresponds to a communication network represented by a connected graph G. Each node in the graph interacts with Merlin by a series of interactions where it draws coins and receives an answer from Merlin. This answer may depend on the network topology, as well as on all previous interactions. The objective of these interactions is to convince the nodes, with good probability, that the network topology satisfies some predicate. After the last message sent by Merlin, a single round of verification between each node and its neighbors follows. We say that the interactions are of type Arthur-Merlin (dAM) or Merlin-Arthur(dMA) depending on whether the verification round is deterministic or randomized where, in the second case, the nodes draw a new set of coins previous to the exchange of messages.

The first chapter centers around defining this model and its variants, as well as presenting the theoretic framework of this work. In the second chapter we describe protocols using a single round of interaction for different graph classes. From here we obtain efficient protocols for problems such as the recognition of interval graphs, chordal graphs, circular-arc graphs, among others. In the third chapter, we continue showing different interactive protocols, but where multiple rounds of interaction are considered. We study problems such as detecting the presence or absence of some structure as a subgraph or minor. From here we obtain protocols for classes such as cographs, distance-hereditary graphs and triangle-free graphs.

In the fourth chapter we leave aside the design of protocols to study the construction of lower bounds. By using different techniques we obtain lower bounds for computing the degeneracy and the recognition of cographs. Then, we present a general technique for obtaining lower bounds based on "glueing" correct solutions in order to fool the network. We obtain bounds for a single and multiple rounds of interaction, where the source of randomness is shared between the nodes. From here we conclude that the results obtained for many of the previous problems are tight. Lastly, we extend a previous result for the problem of Symmetry, which corresponds to recognizing graphs that admit a non trivial automorphism. We obtain a lower bound on multiple rounds of interaction when the source of randomness is private for each node.

Finally, in the fifth chapter, we study the effects of changing the source of randomness in the model, by comparing the case when the source of randomness is shared between the nodes in the network and when it is private for each node. We conclude that there are differences in the power of the model depending on whether the last interaction corresponds to Merlin or Arthur. If the interaction is of type dAM we have that the model using private randomness admits protocols with an asymptotically smaller cost than the latter, even for multiple rounds of interaction. On the other hand, we prove that, if Arthur is the last party to interact, then the use of shared randomness allows for a substantial improvement in message size over those using private randomness.

"Mañana está en duda, pero hoy gané"
Bronko Yotte

Agradecimientos

Este trabajo ha sido producto del esfuerzo y apoyo de muchas personas y, si bien no puedo incluirlos a todos, llevo los nombres de todos ustedes conmigo y les estoy tremendamente agradecido.

En primer lugar quiero dar las gracias a mi familia, a mis tíos, a mis primos y, en especial, a mi mamá Pilar y a mis abuelos Guillermina y Carlos, por darme todo su amor y apoyo y darme todas las herramientas para perseguir mis intereses y convertirme en la persona que soy en este momento.

Quiero darle gracias a mis amigos, los que han estado en distintas etapas de mi vida y que han ayudado a que el paso por la universidad sea mucho más especial. A mis amigos del colegio, a Pedro, Jorge, Billy, Aaron y Aylén por aguantarme 10 años y ser la familia que encontré. A Joaquin, Lucca, Nico Vera y Nico Avilés por las tantas risas y aventuras juntos desde el día 1 en la universidad. A mis amigos del DIM, a Nicolás, Andrés, Pablo, Juan Pedro, Alonso y Julio, por todas nuestras payadas, sesiones de estudio, y muchas conversaciones que me enseñaron mucho más que matemáticas, con especial mención a Tomás y Eduardo por ser tremendos amigos y por convencerme y motivarme a perseguir las matemáticas.

También quiero incluir a los profesores que tuve en el departamento, que fueron un pilar fundamental de mi aprendizaje, en particular a José Soto y Andreas Wiese, quienes despertaron mi interés por las matemáticas discretas. A Pedro Montealegre e Iván Rapaport, por su presencia en cada aspecto de esta tesis, desde las pausas de café a las muchas conversaciones sobre matemáticas, dando su apoyo siempre que surgieron problemas. A Marcos Kiwi y Martín Matamala por ser excelentes profesores y por sus aportes en la revisión de este trabajo. A Natacha, Karen, Eterin, Kuky, Óscar y los demás funcionarios del departamento, por siempre prestar su ayuda y hacer del DIM el departamento que es.

Por último a Constanza, por ser la mejor compañera que podría desear y darme fuerzas en todo momento, incluso cuando creía no poder terminar este trabajo.

Contents

Introduction

Distributed computing concerns situations in which many entities, located at certain points in a network, must operate in a noninterfering and cooperative manner by communicating with each other through a set of channels in the network, in order to perform a set of individual tasks. These tasks can be related to each other, such that the entities (while executing their own protocol) may need to exchange information or share resources in order to fulfill them. Distributed computing deals with many activities occurring today, from their applications in telecommunication networks such as the Internet, the management of distributed databases such as Google's Cloud Spanner [CDE+13], to parallelizing computationally large tasks using multiple processors as in NLHPC's Guacolda-Leftraru [Lef], the coordination of autonomous mobile sensors [ERSR12], as well as many others.

The study of problems in this setting may present multiple challenges that do not appear in its centralized counterpart. First, each entity only has knowledge of its own local input, which means that it is completely unaware of the information held by other participants in the network. Second, each entity is allowed to interact locally, with its neighbors in the network. Last, but not less important, it might be unfeasible for the entities to send extremely large messages due to resource restrictions, given that there is an associated energy cost, a network of small bandwidth, etc. There can be other issues (not addressed in this work) such as the network recovering from an incorrect state (Self-stabilization) or the ability of maintaining functionality when some nodes in the network fail (Fault-tolerance).

Distributed decision refers to the problem of checking that the actual input graph (i.e., the network itself) satisfies a given predicate. Here, the nodes from the network interact with each other by exchanging messages following a set of rules through a series of interactions, after which they must output a decision (either accept or reject). The decision rule typically specifies that (once the protocol terminates), if the predicate is satisfied, then all nodes must accept and, otherwise, at least one node must reject. A single rejecting node can indeed trigger an alarm (in, e.g., hardwired networks), or launch a recovery procedure (in, e.g., virtual networks such as overlay networks). This is an important subject of study because, as for centralized computing, distributed algorithms often assume promises on their inputs, and many algorithms are designed for specific families of graphs, including regular graphs, planar graphs, graphs with bounded arboricity, bipartite graphs, graphs of bounded treewidth, etc.

There are some properties which can be decided locally easily. For example, deciding

whether a graph is regular or if a given coloring is proper can be done by exchanging messages between neighbors. However, other properties such as deciding whether the network is a tree or if the network is bipartite may require long-distance communication for detecting the presence of an (odd) cycle.

The proof-labeling schemes model (PLS) provides a remedy to this issue [KKP10]. These mechanisms have a flavor of NP-computation, but in the distributed setting. That is, an all-powerful, but untrustable prover provides each node with a certificate, and the collection of certificates is supposed to be a distributed proof that the graph satisfies the given predicate. The nodes check locally the correctness of the proof. The specification of a proof-labeling scheme for a given predicate is that, if the predicate is satisfied, then there must exist a certificate assignment leading all nodes to accept, and, otherwise, for every certificate assignment, at least one node rejects. As an example, for the case of the bipartiteness predicate, if the graph is bipartite, then a prover can provide a partition by coloring the nodes either red or blue. It is then sufficient for each node to locally check that all its neighbors have the same color, different from its own color, and to accept or reject accordingly. If the graph is not bipartite, then there is no way that a dishonest prover can fool the nodes, and make them all accept the graph.

Another example could be the verification of a spanning tree construction previously computed in the graph. Such a proof may be encoded distributedly by providing each node with a certificate containing the id of the root of T, and the distance $d(v)$ from v to the root (see, e.g., [KK07]). Indeed, every node v can simply check that $d(p(v)) = d(v) - 1$ (to guarantee the absence of cycles), and that it was given the same root id as all its neighbors in the network (for guaranteeing the unicity of the tree). And thus, with such elements, no dishonest prover can fool the network into accepting an incorrect construction. Similar variants were also introduced: non-deterministic local decisions [FKP13], locally checkable proofs [GS16], and others which carry the same spirit of verification, with slight differences on the initial information or the verification procedure. Through several results in this regard, it has been shown that the inclusion of nondeterminism can greatly increase the power of distributed algorithms.

A positive result in this line is that all (Turing-decidable) predicates on graphs admit a proof-labeling scheme. The difficulty is that there are simple graphs properties (e.g., existence of a non-trivial automorphism [KKP10], non 3-colorability [GS16], bounded diameter [CHPP20], etc.) which require certificates even up to $\Omega(n^2)$ bits in n-node graphs ($\Omega(n)$ in the case of graphs with bounded diameter). Such large certificates do not fit with the requirement that checking algorithms must not only be local, but they must also consume little bandwidth.

In this work we study a recent model which extends the proof labeling scheme setting by allowing interaction with the prover. In distributed interactive protocols, a notion initially introduced by Kol, Oshman and Saxena [KOS18] and further studied in [CFP19, FMO+19, NPY20], a centralized all powerful yet untrustable prover (whom we call Merlin) exchanges messages with a randomized distributed algorithm (to which we refer by Arthur). Specifically, Arthur and Merlin perform a sequence of exchanges during which every node queries the prover by sending a random string, to which Merlin

replies to each node by sending a string called proof. Neither the random strings nor the proofs need to be the same for each node. After a certain number of rounds, every node exchanges information with its neighbors in the network, and decides (i.e., it outputs accept or reject).

From an applied perspective, this type of "assisted computation" is specially relevant today due to the rise of protocols on mobile networks, services based on cloud computing or tasks that require a large amount of data to be processed which proves to be difficult in a more traditional setting. In the event that some entity were to present some failure or may be intervened by a malicious source we need to identify such an issue and respond accordingly.

Two model variants arise in this new randomized scenario, regarding the order of the phases. Let us assume first that we have two phases. When the random phase precedes the non-deterministic phase, we refer to *distributed Arthur-Merlin protocols*, and we denote them by dAM (following the terminology and notation of [KOS18]). Conversely, when nodes access randomness only after receiving the certificates, we refer to *distributed Merlin-Arthur protocol*s, and we denote them by dMA. Note that Merlin is the powerful but untrustable prover of the PLS model, while Arthur represents the nodes, which are simple and limited verifiers that can flip coins. While in a dAM protocol all randomness is seen by Merlin, as it is sent to it before its last interaction, in a dMA protocol, the prover does not see the nodes' randomness when choosing the certificates. Instead, only once the prover assigns certificates to the nodes, each node randomly selects a message that broadcasts to its neighbors. Then, (in both cases) each node decides whether to accept or reject, based on its randomness, input, certificate, and the messages it received from its neighbors.

In distributed interactive proofs, Merlin tries to convince the nodes that G satisfies some property in a small number of rounds and through short messages. We say that an algorithm uses k rounds and $\mathcal{O}(f(n))$ bits if k exchanges of information occur between both parts where the messages exchanged between the nodes (in the verification round) and also the messages exchanged between the nodes and the prover are upper bounded by $\mathcal{O}(f(n))$. The case when k equals one is treated differently depending if the protocol is of type dAM or dMA. While the former uses non-determinism and no randomness (coinciding with protocols in the PLS model), the latter omits all interaction with the prover and relies only on randomized verification. For a problem admitting a protocol with k rounds of interaction and *bandwidth bound* $\mathcal{O}(f(n))$, such a problem is said to be in the class $\mathsf{dAM}[k, f(n)]$ and $\mathsf{dMA}[k, f(n)]$ for the corresponding protocols.

Throughout this work, we study the power of interactive proofs both through the design of protocols for the recognition of different graph classes (and the construction of lower bounds for them) and through its variations on the use of randomness.

The design of protocols gives some insight on the extent to which interaction can be used to improve the communication cost for distributed decision problems in comparison to other models, and how techniques in these models can be extended to obtain lower bounds in our setting. In this regard, we show protocols using one or multiple rounds of

interaction on the detection of problems which can be local in nature (such as triangle-free graphs), as well as others that depend on global information (such as the presence of a fixed subgraph or a minor), or even some graph classes with a well known structure (such as chordal graphs, cographs, distance-hereditary graphs, among others). Then, we study lower bounds for the previous problems, showing that our protocols for many of them are tight. This is achieved by combining results from extremal graph theory, as well as other techniques from the communication complexity setting.

As for the use of randomness, an issue well-studied in the context of communication complexity, but much less considered in distributed computing, provides separation results for the different flavors of interaction as the impact of the two forms of randomness is very different depending on whether we are considering Arthur-Merlin protocols or Merlin-Arthur protocols. By studying protocols (and lower bounds) for two different problems we show that: While private randomness gives more power to the first type of protocols, shared randomness provides more power to the second. These results provide a wider perspective to the study of distributed interactive proofs.

Organization

In the first chapter we start by describing the different phases in a distributed interactive protocol, and then we define the distributed Arthur-Merlin model in both its shared randomness, and private randomness versions. From here we obtain a full description of both the dAM and dMA classes. Then, in Section 1.3 we go over some important models in the field of communication complexity that will be relevant for this work. Next, in Section 1.4, we describe some important results from the literature which will be used. Finally, in Section , we give a brief summary of the results obtained with regards to the complexity of different graph classes.

In the second and third chapter, we study the design of distributed interactive protocols.

In the second chapter we show many problems that admit protocols with small messages and a single round of interaction: In Sections 2.1 and 2.2 we define the problems d-DEGENERATE and TWINS and obtain single round dAM protocols(dM) for each of them (as well as their complementary classes. Next, we focus on problems based on the intersection of different objects: In Sections 2.3 and 2.4, we study the problems PROPER INTERVAL, INTERVAL and CHORDAL, for which we obtain single-round dAM protocols (simply written asdM) using small messages for each of them (which we later show to be tight). Then, following the same ideas, we look at the problems PROPER CIRC-ARC and CIRCULAR-ARC, obtaining an efficient dM protocol for both problems.

In the third chapter, we describe protocols using many rounds of interaction for different graph problems: First we study problems which involve detecting the presence (or lack) of graph structures. Here, we obtain simple, constant size, three round protocols for the problems H-SUBGRAPH and H-MINOR, which are the problems of detecting the presence of a fixed subgraph, or minor H, respectively. Next, we show two round protocols

for the classes CLIQUE and COGRAPH. While the former is a simple use of randomization that can be implemented in the Congest model (where the network nodes interact with each other through rounds of $\mathcal{O}(\log n)$-size messages), the latter corresponds to an adaptation of a protocol in the Broadcast Congested Clique model [KMRS15] (where the network nodes write on a shared blackboard) by exploiting the high connectivity in this class. By use of the tools from this last result we show a three round protocol for the class DIST-HEREDITARY as well. Finally, in Section 3.5, we obtain a $2k$ round protocol in the dMA model using shared randomness with bandwidth cost $n^{\frac{1}{k+1}} \log n$ for verifying the absence of triangles in a graph, which improves on the result by Crescenzi et al. [CFP19].

Next, in the fourth chapter, we focus on the construction of lower bounds. In Section 4.1 we show lower bounds in dM and dAM (using shared randomness) for the problem d-DEGENERATE, by adapting a technique from [FH18]. Next, in Section 4.2 we obtain a dM lower bound for the problem COGRAPH, by use of a technique described in [FPSP19] which constructs a No-instance by crossing edges in a correct instance. Later, in Section 4.3 we adapt and extend a lower bound construction initially described in [GS16] which allows us to "glue" correct solutions in a dAM protocol using shared randomness and k rounds of interaction and provide an accepting *bad instance* given sufficiently small messages and any fixed k. We also adapt this technique to provide lower bounds in the dMA model. Finally, in Section 4.4, we extend a lower bound construction for the class SYMMETRY initially shown in [KOS18] which proves a $\Omega(\log \log n)$ barrier in the private randomness variant even if we increase the amount of interactions

Finally, in the fifth chapter, we study the impact of the use of randomness depending on the type of interaction. First, we show in Section 5.1 that any interactive protocol that uses shared randomness can be derandomized into a non-deterministic proof with an exponential factor overhead in the bandwidth, which induces several lower bounds on other problems described in the literature. Next, in Section 5.2, we separate the two variants of the dAM model through the language AMOS, which is the language of n-node graphs having at most one selected node. It is known that it can be decided with two rounds of interaction and cost $\mathcal{O}(1)$ when private randomness is used, whereas for any constant number of rounds we show that the use of shared randomness can not achieve a bandwidth cost smaller than $\Omega(\log \log n)$. This shows there is an unbounded gap between the two models. Finally, in Section 5.3, we show that roles are reversed if we consider dMA protocols instead. More precisely, first we get an analogous result to that of [CFP19] proving that dMA protocols with shared randomness are more powerful than their private counterpart, albeit with a small additional bandwidth cost of $\mathcal{O}(\log n)$. Then we separate both classes through the language 2-COL-EQ which consists of graphs with n-bit labels corresponding to proper 2-colorings. We show that while this language requires $\Theta(\log n)$ bits of communication when shared randomness is used, the use of private randomness implies a bandwidth cost of $\Theta(\sqrt{n})$.

A list of all problems considered in this work is presented in the appendix at the end of this document.

Results

We go over the positive results obtained in Chapters 2 and 3, as well as the lower bound results obtained in Chapter 4.

First, we include the positive results obtained in Chapter 2. That is, all problems for which we obtained a protocol using a single round of interaction. As will be discussed in Chapter 1, these values hold up to a constant factor. We use asymptotic notation to emphasize on those problems whose results (as is shown in Chapter 4) are tight.

Property	**Cost**	**Reference.**
TWINS	$\Theta(\log n)$	Prop. 2.3
TWIN-FREE	$\mathcal{O}(\log n)$	Prop. 2.5
d-DEGENERATE	$\Theta(\log n)$	Prop. 2.1
$\overline{\text{d-DEGENERATE}}$	$\mathcal{O}(\log n)$	Prop. 2.2
PROPER INTERVAL	$\Theta(\log n)$	Prop. 2.9
INTERVAL	$\Theta(\log n)$	Cor. 2.15
PROPER CIRC-ARC	$\Theta(\log n)$	Prop. 2.18
CIRCULAR-ARC	$\Theta(\log n)$	Prop. 2.20
CHORDAL	$\Theta(\log n)$	Thm. 2.14

Now, we include the positive results obtained in Chapter 3. These are all problems that admit efficient protocols using multiple rounds of interaction: We separate them according to the number of interactions and whether the source of randomness is shared or private to each node.

Property	**Class**	**Random**	**Cost**	**Reference**
CLIQUE	dAM	private	$\mathcal{O}(1)$	Prop. 3.3
COGRAPH		shared	$\mathcal{O}(\log n)$	Thm. 3.10
AMOS		shared	$\Theta(\log \log n)$	Lem. 5.11
H-SUBGRAPH	dMAM	private	$\mathcal{O}(\lvert H\rvert)$	Prop. 3.1
H-MINOR		private	$\mathcal{O}(\lvert H\rvert)$	Prop. 3.2
DIST-HEREDITARY		shared	$\mathcal{O}(\log n)$	Thm. 3.14
Δ-FREE	dMA[$2k$]	shared	$\mathcal{O}(n^{\frac{1}{k+1}} \log n)$	Thm. 3.16
2-COL-EQ	dMA	private	$\Theta(\sqrt{n})$	Lem. 5.15

We also include a compilation of the lower bounds obtained in Chapter 4, as well as those appearing in the literature, which we extend following the results in Section 5.1. Notice, again, that these values hold save for a constant factor. Those entries labeled with $*$ indicate that the lower bound is obtained by the same result as the entry to the left.

Class	dM	$\mathsf{dMA}^{\text{pub}}$	$\mathsf{dMA}^{\text{priv}}$	$\mathsf{dAM}^{\text{pub}}[k]$	$\mathsf{dAM}^{\text{priv}}[k]$
DIAMETER	$n/\log n$ [CHPP20]	$n/\log n$	$n/\log n$	$\log n$	
SYMMETRY	n^2 [GS16]	n^2	n^2	$\log n$	$\log\log n$
$\overline{3}$-COL	$n^2/\log n$ [GS16]	$n^2/\log n$	$n^2/\log n$	$\log n$	
MST	$\log^2 n$ [KK07]	$\log\log n$		$\log\log n$	
2-COL-EQ	n	$\log n$	$\sqrt{n}$	$\log n$	
AMOS	$\log n$ [FMO+19]	$\log n$	$*$	$\log\log n$	
INTERVAL	$\log n$	$\log n$	$*$	$\log\log n$	
CHORDAL	$\log n$	$\log n$	$*$	$\log\log n$	
PLANAR	$\log n$ [FFR+20]	$\log n$	$*$	$\log\log n$	
Δ-FREE	$n/\mathrm{e}^{\sqrt{\log n}}$ [DKO14]	$\log n$		$\log n$	
CIRCULAR-ARC	$\log n$	$\log n$	$*$	$\log\log n$	

Chapter 1

Preliminaries and Model Definition

1.1 Preliminaries

A simple undirected graph, from here on simply referenced as graph, is a pair of sets $G = (V, E)$ where V is a finite set, called node set, and E is a subset of the 2-sets of V called edge set, that is, $E \subseteq \{\mathrm{e} \in 2^V \text{ such that } |\mathrm{e}| = 2\}$. Whenever we consider an edge $\mathrm{e} = \{u, v\}$ we shall refer to it simply as $\mathrm{e} = uv$, and say that u and v are *adjacent*. If several graphs are being considered, we will denote each graph's node set and edge set as $V(G)$ and $E(G)$ respectively.

We write $n(G)$ to denote the number of nodes of G and $m(G)$ for the number of edges of G, where we will simplify this notation to simply n and m whenever we are safe from any type of confusion.

For a set U contained in the node set V, we define the induced subgraph of $G = (V, E)$ according to U as the pair $(U, E(U))$, where $E(U) = \{vw \in 2^U \text{ such that } vw \in E\}$. Whenever such a graph exists we say that H is an induced subgraph of G and denote it by $H \subseteq G$. If, instead, we have a graph with node set U such that its edges are only contained in $E(U)$ we simply call it a subgraph of G.

We say that a node u belongs to a graph G, which we write as $u \in G$ whenever u belongs to its node set $V(G)$. For such a node, we define its (open) neighborhood $N(u)$ as the nodes $v \in G$ which are adjacent to u, that is, the pair $\mathrm{e} = uv$ belongs to G's edge set. Similarly, we define its closet neighborhood $N[u]$ to be the union of $N(u)$ and $\{u\}$. We define the degree of a node v in G to be the size of $N(v)$. Given a graph G with node set V we define the minimum degree of G to be the smallest value for $d(v)$ among all nodes in V and denote its value as $\delta(G)$.

A path P in a graph G is an ordered collection of distinct nodes $v_1, \ldots v_k$ such that for all $i \in 1, \ldots k-1$ the pair v_i and v_{i+1} are adjacent. The nodes x_1 and x_k are said to be *linked* by P, with x_1 and x_k its *ends*. Similarly, a cycle can be defined as a sequence of nodes $v_1, \ldots v_k$, with $k \geq 3$, which form a path where the nodes v_k and v_1 are also

adjacent. We say that a graph G is *connected* if for any pair of nodes $u, v \in G$ there exists a path $\{w_1, \dots w_k\}$ where $w_1 = u$ and $w_k = v$ for some integer k. A clique of a graph G is a subset C of its node set V such that all pairs of nodes in C are adjacent.

A tree T is an undirected graph such that it is connected and does not have any cycle. A spanning tree T of some graph G is a subgraph of G that is a tree such that $V(T) = V(G)$.

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

Throughout this work several graph properties and classes will be defined or mentioned. For further details on these concepts please refer to the book "Graph Theory" by Diestel [Die06], or the survey on graph classes by Brandstadt [BS+99]. For a list of all problem definitions, please refer to the annex at the end.

Finally, for simplifying the notation, for a predicate $\mathrm{p}(x)$, we denote $[\![\mathrm{p}(x)]\!]$ to be the function that equals one if and only $\mathrm{p}(x)$ is true.

1.2 The Distributed Interactive Proofs 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 is polinomially bounded on n. Let $\mathsf{id} : V(G) \to \{1, \dots, \mathrm{poly}(n)\}$ 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 can not 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 its $\mathsf{id}(v)$ to its neighbors, then the message is of size $\mathcal{O}(\log n)$.

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

There are two types of interactive protocols: Arthur-Merlin and Merlin-Arthur. Both types of protocols have two phases: an interactive phase and a verification phase. Let us define first *Arthur-Merlin interactive protocols*. If Arthur is the party that starts the interactive phase, he picks a random string $r_1(v)$ at each node v of G (this string could

be either private or shared) and sends 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,c_1, r_2 and (G, id, I). An example of this interaction is shown in Figure 1.1.

This process continues for a fixed number of rounds. If Merlin is the party that starts the interactive phase, then he provides at the beginning every node v with a certificate $c_0(v)$ that is a function of v and (G, id, I), and the interactive process continues as explained before. In Arthur-Merlin protocols, the process ends with Merlin. More precisely, in the last, k-th round, Merlin provides every node v with a certificate $c_{\lceil k/2 \rceil}(v)$. Then, the verification phase begins. This phase is a one-round deterministic algorithm executed at each node. More precisely, every node v broadcasts a message M_v to its neighbors. This message may depend on $\mathsf{id}(v)$, $I(v)$, all random strings generated by Arthur at v, and all certificates received by v from Merlin. Finally, based on all the knowledge accumulated by v (i.e., its identity, its input label, the generated random strings, the certificates received from Merlin, and all the messages received from its neighbors), the protocol either accepts or rejects at node v. Note that Merlin knows the messages each node broadcasts to its neighbors because there is no randomness in this last verification round.

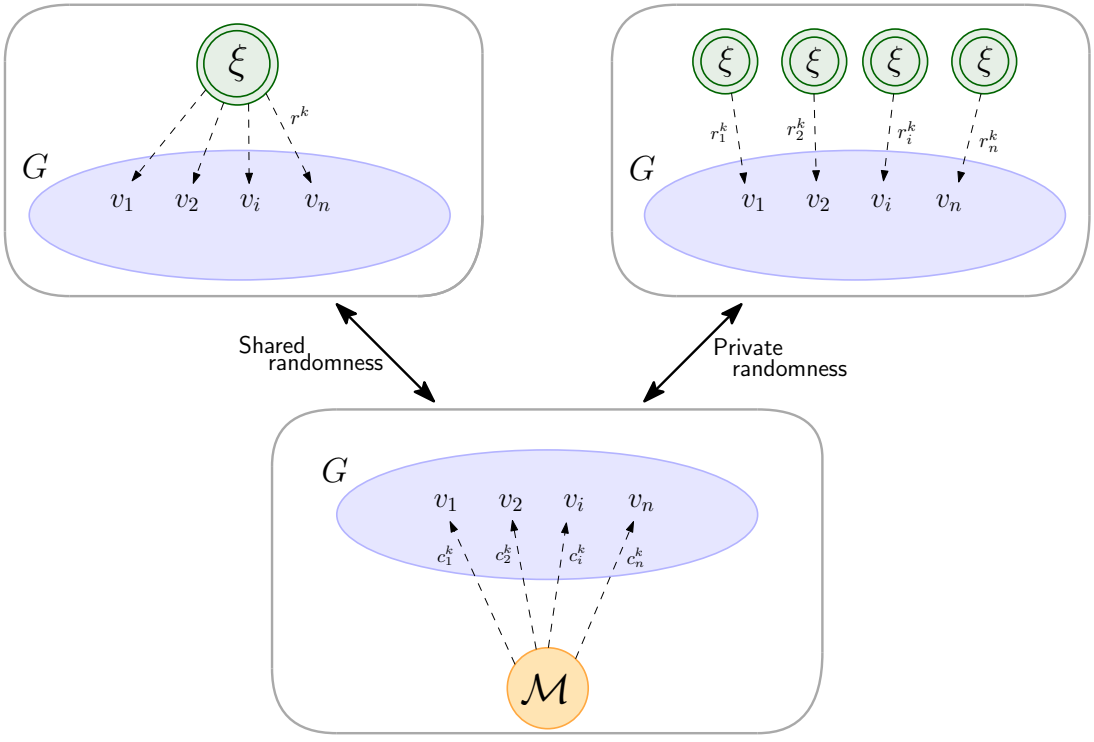

Figure 1.1: On any fixed round k, each v generates a seed r_v from a source of randomness ξ. Then, it receives a message from Merlin $\mathcal{M}$ which depends on each r_v and all previous interaction. If the graph uses shared randomness (left) all nodes access the same source of randomness, while in the private randomness version (right) all nodes have their own source.

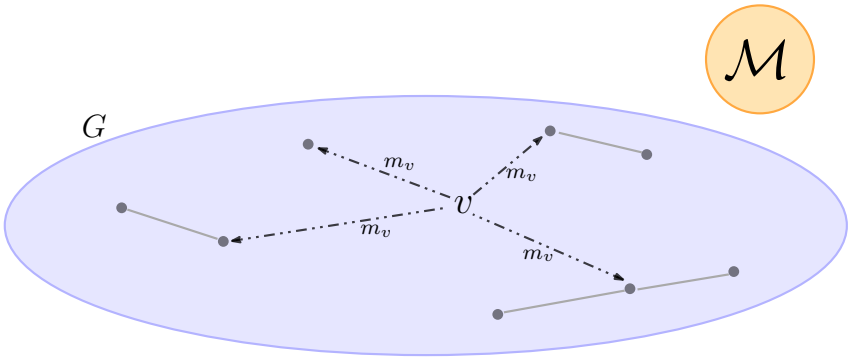

Figure 1.2: During the verification round, a node v in the network broadcasts a message m_v which contains all certificates received as well as all coins drawn during the protocol. In case of interactions of type dMA, these messages may depend on a fresh set of random bits.

A *Merlin-Arthur interactive protocol* of k interactions is an Arthur-Merlin protocol with $k-1$ interactions, but where the verification round is randomized. More precisely, Arthur is in charge of the k-th interaction, which includes the verification algorithm. The protocol ends when Arthur picks a random string $r(v)$ at every node v and uses it to perform a (randomized) verification algorithm. In other words, each node v randomly chooses a message M_v from a distribution specified by the protocol, and broadcast M_v to its neighbors as shown in Figure 1.2. Finally, as explained before, the protocol either accepts or rejects at node v. Note that, in this case, Merlin does not know the messages each node broadcasts to its neighbors (because they are randomly generated). If $k = 1$, a distributed Merlin-Arthur protocol is a (1-round) randomized decision algorithm; if $k = 2$, it can be viewed as a non-deterministic version of randomized decision, etc.

Given a distributed interactive protocol, its (bandwidth) cost is defined as the largest size of any message sent by Merlin or any node v during either the interactive phase or the verification phase.

Having described distributed interactive protocols, we are ready to define the classes dAM and dMA, as studied in [KOS18, NPY20, CFP19].

Definition 1.1 *Let $\mathcal{V}$ be a verifier and $\mathcal{M}$ a prover of a distributed interactive proof protocol for languages over graphs. If $(\mathcal{V}, \mathcal{M})$ corresponds to an Arthur-Merlin (resp. Merlin-Arthur) k-round, $\mathcal{O}(f(n))$ bandwidth protocol, we write $(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ (resp. $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$).*

Definition 1.2 *Let $\varepsilon \leq 1/3$. The class $\mathsf{dAM}_{\varepsilon}[k, f(n)]$ (resp. $\mathsf{dMA}_{\varepsilon}[k, f(n)]$) is the class of languages $\mathcal{L}$ over graphs for which there exists a verifier $\mathcal{V}$ such that, for every configuration (G, id, I) of size n, the two following conditions are satisfied.*

- **Completeness.** *If $(G, \mathsf{id}, I) \in \mathcal{L}$ then, there exists a prover $\mathcal{M}$ such that*

$(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ *(resp.* $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$*) and*

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

- **Soundness.** *If* $(G, \mathsf{id}, I) \notin \mathcal{L}$ *then, for every prover* $\mathcal{M}$ *such that*

$(\mathcal{V}, \mathcal{M}) \in \mathsf{dAM}_{\mathsf{prot}}[k, f(n)]$ *(resp.* $(\mathcal{V}, \mathcal{M}) \in \mathsf{dMA}_{\mathsf{prot}}[k, f(n)]$*),*

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

We also denote $\mathsf{dAM}[k, f(n)] = \mathsf{dAM}_{1/3}[k, f(n)]$ *and* $\mathsf{dMA} = \mathsf{dMA}_{1/3}[k, f(n)]$.

We omit the subindex ε when its value is obvious from the context. For small values of k, instead of writing $\mathsf{dAM}[k, f(n)]$ and $\mathsf{dMA}[k, f(n)]$, we alternate Ms and As. For instance: $\mathsf{dMAM}[f(n)] = \mathsf{dAM}[3, f(n)]$, $\mathsf{dAMA}[f(n)] = \mathsf{dMA}[3, f(n)]$, etc. In particular $\mathsf{dAM}[f(n)] = \mathsf{dAM}[2, f(n)]$, $\mathsf{dMA}[f(n)] = \mathsf{dMA}[2, f(n)]$.

Definition 1.3 *The shared randomness setting may be seen as if all the nodes, in any given round, sent the same random string to Merlin. In order to distinguish between the settings of private randomness and shared randomness, we denote them by* $\mathsf{dAM}^{\mathrm{priv}}[k, f(n)]$ *and* $\mathsf{dAM}^{\mathrm{pub}}[k, f(n)]$*, respectively.*

Finally, to simplify notation, we may write $\mathrm{P} \in \mathsf{dAM}[k, \Omega(f(n))]$ to denote that, if some problem P is in $\mathsf{dAM}[k, g(n)]$ it implies that $g(n) = \Omega(f(n))$. In the same way, we may write $\mathrm{P} \in \mathsf{dAM}[k, \Theta(f(n))]$ to denote that the cost $f(n)$ is tight.

1.3 Other Communication Models.

In this section we mention other computational models related to communication complexity and distributed computing that may be of interest for this work. Specifically, we go over the classic model of 2-party communication complexity, where two entities, Alice and Bob, must jointly compute a 2-input function $f(\cdot, \cdot)$ while communicating the least number of bits possible.

1.3.1 Alice and Bob, 2-Party Communication.

In the 2-party communication model two entities, namely Alice and Bob, own an n bit input each, denoted by x and y respectively, and their goal is to jointly compute the value of a Boolean function that depends on both inputs $f(x, y)$ with $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

A protocol between Alice and Bob may be seen as a sequence of interactions, where they exchange bits until one of them is able to compute an answer. For any given problem P, its (deterministic) complexity in the 2-party communication model corresponds to the minimum number of bits needed to be sent by Alice and Bob in order to decide

the problem. On the other hand, we include the help of a Prover to the model, which sends a certificate to both Alice and Bob during the communication process, the non-deterministic complexity of a problem is defined as the minimum number of bits on the certificate provided by the Prover, combined with the size of the messages sent by both parties.

One of the reasons for this model's importance is that it's sufficiently broad to allow the use of both its protocols and lower bounds in a diverse range of subjects beyond distributed computing, allowing new results in areas such as Computational Complexity, Game Theory, Data Structures, etc.

This work will make use of the lower bounds for the problem EQUALITY, where Alice and Bob must determine whether their inputs x and y are equal. An important result for this problem is that in order to obtain the solution, the best possible method is for one of the entities to simply send its whole input, even if we allow the use of non determinism.

For a formal proof of this result, please check [Kus97].

Theorem 1.4 *The non deterministic complexity of* EQUALITY *in the 2-party communication model is* $\Omega(n)$.

1.3.2 The Simultaneous Messages Model

In the simultaneous messages model (SM) there is a third player, the referee, who wishes, together with *Alice* and *Bob*, to compute a function $f(x, y)$. Alice and Bob are given inputs x and y, respectively. The referee has no input. Alice and Bob are unable to communicate with each other, but, instead, are able to send a single message to the referee. Their messages depend on their inputs and a number of random bits. Then, using only the messages of Alice and Bob and eventually another random string, the referee has to output $f(x, y)$ (up to some error probability ε, given by the coins of Alice, Bob, and the referee). A randomized protocol with error ε is *correct* in the SM model if the answer is correct with probability at least $1 - \varepsilon$.

We are only interested in the SM model with *private coins*, i.e. when the random strings generated by Alice, Bob, and the referee are not shared with each other. Interestingly, in this model, the power of randomness is very restricted. Indeed, in [BK97], Babai and Kimmel show that any randomized protocol computing a function f in the SM model using private coins requires messages of size at least the square root of its deterministic complexity. More precisely, if we define the deterministic complexity of f, $\mathsf{D}(f)$, as the size of the messages of an optimal SM deterministic protocol for f, the following proposition holds.

Proposition 1.5 ([BK97], Theorem 1.4) *Let $f : X \times Y \to \{0, 1\}$ be any Boolean function. Let $0 \leq \varepsilon < \frac{1}{2}$. Any ε-error* SM *protocol for solving f using private coins needs the messages to be of size at least $\Omega\left(\sqrt{\mathsf{D}(f)}\right)$.*

Aside from a negative result, they show a protocol for the problem EQUALITY, which makes the previous result tight.

Proposition 1.6 ([BK97]) *There exists a randomized protocol in the* SM *model for* EQUALITY *with cost* $\mathcal{O}(\sqrt{n})$

We proceed to describe the protocol for the sake of completeness.

Proof. Let t be the least integer such that $n \leq t^2/4$. Let $\Omega = \mathbb{F}_2^t$ be a vector space with the usual product $u \cdot v = \sum_{i=1}^{t} u_i v_i$. Set $S \subseteq \Omega$ and define $S^\perp = \{u \in \Omega : \forall v \in S, u \cdot v = 0\}$ to be the orthogonal subspace to the set S. Also, in order to simplify notation, we denote $\{u\}^\perp$ simply as $u^\perp$.

Lemma 1.7 ([BK97]) *There are at least* $2^{t^2/4}$ *subspaces of* Ω *with dimension* $t/2$.

Now, by using Lemma 1.7 it is possible to assign, for each $z \in \{0,1\}^n$ a subspace of Ω, which we denote by U_z with dimension $t/2$.

With this tool we can proceed to describe the protocol. Given input $x \in \{0,1\}^n$, Alice chooses a vector u at random from U_x and sends it to the referee, while Bob chooses, given $y \in \{0,1\}^n$ an element $v \in U_y^\perp$ uniformly at random and sends it to the referee as well. Finally, the referee accepts if and only if $u \perp v$.

As both u, v are elements of $\Omega = \mathbb{F}_2^t$, we have that Alice and Bob send $t = \mathcal{O}(\sqrt{n})$ bits each to the referee.

Remark If U is a subspace of Ω and $u \notin U$, then $u^\perp \not\supseteq U^\perp$.

Indeed, as Ω has finite dimension we have that $u^\perp \supseteq U^\perp \leftrightarrow \{u\} = u^{\perp\perp} \subseteq U^{\perp\perp} = U$.

- It follows that, if $x = y$, then the referee always accepts, as $u \in U_x$ and $v \in U_x^\perp$.
- In case that $x \neq y$, it follows that $U_x \neq U_y$ and therefore $|U_x \cap U_y| \leq |U_x|/2$: indeed, let B be a base of the subspace $U_x \cap U_y$ and F be its extension to U_x. As U_x is not contained in U_y, we have that $|B| + 1 \leq |F|$, and so

  $$\frac{|U_x|}{2} = \frac{2^{|F|}}{2} \geq \frac{2^{|B|+1}}{2} = \frac{|U_x \cap U_y|}{2}.$$

  Therefore, we have that the probability that u is chosen at random in $U_x \setminus U_y$ is at least $\frac{1}{2}$.

  If Alice picks $u \in U_x \setminus U_y$, we have by the previous remark that, $u^\perp \not\supseteq U_y^\perp$ and, once again, $|U_y^\perp \cap u^\perp| \leq |U_y^\perp|/2$. Then, the probability, given that $u \notin U_y$, of $v \notin u^\perp$ is

at least 1/2. By combining both results

$$\mathbf{Pr}(u \not\perp v) \;\geq\; \mathbf{Pr}(v \not\perp u \mid u \notin U_y) \;\cdot\; \mathbf{Pr}(u \notin U_y) \geq \frac{1}{4}.$$

It then follows that, by running parallel (independent) repetitions of this protocol, it is possible to make this error arbitrarily small.

□

By incorporating a prover, we can define interactive proofs in the SM model. More precisely, we define $\mathsf{MA}^{\mathrm{sym}}$ as follows.

Definition 1.8 *Let $f : X \times Y \to \{0,1\}$ be a Boolean function. We say that $f \in \mathsf{MA}^{\mathrm{sym}}_{\varepsilon}$ if there exists a protocol for Alice and Bob, where:*

- *A fourth player, the prover, provides Alice and Bob with a proof m (which he builds as a function of the input of Alice $x \in X$ and the input of Bob $y \in Y$).*
- *Alice and Bob simultaneously send a message to the referee, that depends on their inputs, their own randomness, and the certificate m provided by the prover. Let $\omega = \omega_{x,m}(r)$ be the message sent by Alice given the input x and the seed r and let $\varphi = \varphi_{y,m}(s)$ be the message sent by Bob, given y and the seed s.*
- *Finally, let $\rho(\omega, \varphi)$ be the random variable indicating the referee's decision given the messages ω, φ and its random bits.*

For all $x \in X, y \in Y$, the protocol must satisfy the following:

- **Completeness.**

  If $f(x,y) = 1$, there exists a proof m such that $\mathbf{Pr}\left(\rho(\omega_{x,m}, \varphi_{y,m} = 1) \geq 1 - \varepsilon\right.$.

- **Soundness.** *If $f(x,y) = 0$ then, for any proof m, $\mathbf{Pr}(\rho(\omega_{x,m}, \varphi_{y,m}) = 1) \leq \varepsilon$.*

Let $f : X \times Y \to \{0,1\}$ be a Boolean function. The cost of an $\mathsf{MA}^{\mathrm{sym}}$ protocol that solves f is the sum of the proof size, along with the maximum size of a message considering all possible random realizations. When there is no randomness we recover the classical definition of non-deterministic complexity in the SM model, which we denote by $\mathsf{M}^{\mathrm{sym}}(f)$.

Remark The assumption that both Alice and Bob receive the same proof does not affect the definition of the class: in case that Alice receives m_a and Bob receives m_b as proofs, then Merlin may concatenate $m_a m_b$ and then Alice and Bob just consider their part of the message (the referee verifies that Alice and Bob received, indeed, the same message).

1.3.3 The Congest Model

The Congest model can be thought of as the classic model in distributed computing, where the communication network is an undirected graph G and the nodes are only allowed to

exchange messages through these channels in a series of synchronous rounds, where at most $\mathcal{O}(\log n)$ bits can be sent through each channel.

In this model we can pay attention to two different types of protocols. Those where the nodes send the same message to all their neighbors at each round (broadcast) and those where the nodes have the freedom to choose a specific message for each of their neighbors (unicast).

Korman et al. show that any broadcast protocol in this model can be naturally simulated in the *Proof Labeling Scheme*(PLS) model [KKP10], which is essentially a variant of the dM model where the nodes know their neighborhood. It suffices for Merlin to send to each node the transcript of the messages exchanged at each round between nodes in the original Congest protocol. Then, by exchanging these messages sequentially check the transcript's correctness.

Theorem 1.9 *Let A be a synchronous protocol for some problem* P *in the broadcast,* Congest *with τ rounds. Then, there exists a* dM *protocol for* P *with cost $\mathcal{O}(\tau \cdot \log n)$ bits.*

One of the goals of this work is to find protocols whose total communication is smaller than the one obtained by applying Theorem 1.9 (even if more rounds of interaction are necessary).

1.3.4 Broadcast Congested Clique

The Broadcast Congested Clique model assumes a similar setting as the former model. The BCC model is a message passing model where n nodes communicate with each other in synchronous rounds over a complete network [BMRT20]. The joint input to the n nodes is an undirected graph G on the same set of nodes, with node u receiving the list of its neighbors in G. Nodes have pairwise distinct identities between 1 and n. All nodes know n, the size of the network. Each node broadcasts, in each round of the algorithm, a single $f(n)$-bit message along each of its $n-1$ communication links. The size of the messages is known as the bandwidth of the system, which is a function of n. Broadcasting is equivalent to writing the messages on a whiteboard, visible to every node. In each round every node produces its message using its input, the contents of the whiteboard, and, possibly, a sequence of random bits. An algorithm is correct if it terminates with every node knowing the correct answer with high probability

As it is a strong model (yet not as strong as its *unicast* variant) it is interesting to study whether either we can simulate its protocols or compete with their costs. Particularly in regards to its reconstruction results [MPSRT18] and some useful protocols such as the one for the COGRAPH problem, which will be mentioned again in Subsection 3.3.

1.4 Some Important Results

In this section we include some results from the literature that will either be useful for our constructions or important benchmarks. First, we include some classic tools from computational complexity and graph theory that will be used repeatedly in this work.

We consider a simple result related to the density of prime numbers. This theorem is useful when we need to use algebraic techniques in randomization that require us to choose a polinomially bounded prime number.

Lemma 1.10 (Bertrand's Postulate) *For any integer n bigger than 1, it is possible to find a prime number p such that $n \leq p \leq 2n$.*

An example on how to apply this result is the case we need a *compact* encoding of a set of objects distributed across the network. If we represent these objects by use of a polynomial defined over a finite field of p elements, for some prime p (which we can represent through $\mathcal{O}(\log n)$ bits), then we can test the equality of different objects by choosing an element at random in this field. If two objects are different, we have that the probability of *collision* between two different encodings based on this random element can be made polinomially small.

For a formal proof of this result, the reader can refer to [AB09].

Lemma 1.11 (Schwartz-Zippel [Sch80, Zip79]) *Let $\mathsf{p} \in \mathbb{F}[x]$ be a non-zero polynomial with coefficients in a field $\mathbb{F}$ and degree d. Consider r to be an element $\mathbb{F}$ taken uniformly at random. Then, the probability that r is a root of p is at most $d/|\mathbb{F}|$*

In probability theory it is useful to have an estimation of how much can a random variable deviate from its expected value. Classical tools such as the Markov or Chebyshev inequalities provide bounds on the probability of such events on arbitrary random variables. Next, we include a classic result used in complexity theory that provides exponentially decreasing lower bounds on the tails of a sum of independent Bernoulli random variables.

Lemma 1.12 (Chernoff bounds [Hoe48, C+52]) *Let $X_1, \ldots X_k$ be a collection of independent random variables, such that $\mathbf{E}(X_i) = p_i$. Let $X = \sum_{i=1}^{k} X_i$ with $\mu = \mathbf{E}(X) = \sum_{i=1}^{k} p_i$. Then, the following inequalities hold.*

$$\mathbf{Pr}[x \geq (1+\varepsilon)\mu] \leq \exp\left(-\frac{\varepsilon^2}{2+\varepsilon}\mu\right).$$

$$\mathbf{Pr}[x \leq (1-\varepsilon)\mu] \leq \exp\left(-\frac{\varepsilon^2}{2}\mu\right).$$

Now we present a result from extremal graph theory which we use extensively in Section 4.3 for the construction of lower bounds. Proved in the year 1974 and extending a

conjecture by Erdős, it shows that whenever we have a graph which is sufficiently dense, we can find large even cycles contained in it [BS74].

Lemma 1.13 (Bondy-Simonovits) *Let G be an n-node graph with m edges. Let k be an integer.*

$$\textit{Then, if } m > 100k \cdot n^{1+\frac{1}{k}} \textit{ it follows that } C^{2\ell} \subset G, \textit{ for all } \ell \in [k, k \cdot n^{\frac{1}{k}}]$$

Now we mention a result that follows a similar spirit as the previous one and which we use in the same context, on the study of structures in hyper-graphs. Initially shown by Erdős in the year 1964, it says that, if we have an r-uniform hyper-graph with a large number of hyper-edges, then we can find a structure that allows us to make "swaps" on our hyper-edges while staying inside this structure [Erd64]. More specifically, if we set $K^{(r)}(\ell)$ to be a r-uniform hyper-graph, where we can split the set of nodes into r parts, with size ℓ each, such that we have a hyper-edge for any selection of elements from each of the r different parts, then we have the following result.

Lemma 1.14 (Erdős) *Let G be an r-uniform hyper-graph. If G does not contain a $K^{(r)}(\ell)$ as a subgraph, then $|E(G)| \leq n^{1-1/\ell^{r-1}}$.*

Where we say that an r-uniform hyper-graph H is a subgraph of an r-uniform hyper-graph G if the node set of H is contained in that of G, and each hyper-edge in H is a hyper-edge in G.

Next, we mention some recent results in the subject of distributed interactive proofs which are repeatedly used throughout this work.

First we start with the IP compiler, constructed by Naor, Parter and Yogev. It states that if we are given a centralized interactive proof protocol, we can translate it into a distributed interactive proof by splitting the set of coins to be sent (and the proofs answered by the prover) among the nodes in the network and then, in the verification round, we simply need to check the consistency of a computation transcript, which can be easily done by use of a set equality protocol [NPY20].

Theorem 1.15 *Let $\mathcal{P} \in \mathsf{AM}$ be a public coin interactive protocol that uses k rounds (with $k > 1$) for deciding some language $\mathcal{L}$ such that the verification round requires time $\tau(n)m$ with $\tau(n) = \Omega(n)$. Then, $\mathcal{L} \in \mathsf{dAM}^{\text{priv}}[k+2, \tau(n)\log(n)/n]$. In particular, if $\mathcal{L} \in \mathsf{NP}$ and the verifier runs in time $\mathcal{O}(n)$, then $\mathcal{L} \in \mathsf{dMAM}^{\text{priv}}[(\log n)]$.*

In essence, this compiler provides a black-box for the design of protocols for different graph problems by simply paying the running time of the verifier. This is specially useful on properties which can be decided in time linear on the size of the graph, examples of this are hereditary graph classes such as Interval Graphs, Chordal Graphs [HMPV00], Cographs [GP12], Circular arc graphs [McC03], among others. This may also prove spe-

cially useful if the classes to be considered are sparse such as planar graphs [HT74],which, by this result, admits a $\mathsf{dMAM}[\mathcal{O}(\log n)]$ protocol.

Now, if we were to consider other important compiler-like results on interactive proofs, we can combine them with the previous theorem and obtain a wider variety of distributed interactive proofs. Starting with verifiers that use small space, by adapting a compiler described in [RRR16], if some problem can be solved using logarithmic space, then there exists a (centralized) interactive proof that requires $\mathcal{O}(1)$ many rounds and requires $\operatorname{poly}\log(n)$ bandwidth.

Theorem 1.16 (by [RRR16]) *Let L be a language that can be decided using $\mathcal{O}(\log n)$ space. Then, there exists an* AM *protocol for L with perfect completeness and soundness error $1/2$ such that, with access to an oracle, has the following properties*

- *The number of rounds is $\mathcal{O}(1)$.*
- *The bandwidth at each round is $\operatorname{poly}\log(n)$.*
- *The prover runs in time $\operatorname{poly}(n)$ and the verifier runs in time $\operatorname{poly}\log(n)$.*

Where we say that an interactive protocol has perfect completeness if it always accepts on a correct instance.

If we combine these results, and simulating the access to the aforementioned oracle, we then have the following: for any language which can be decided in logarithmic space there exists a dAM protocol using $\mathcal{O}(1)$ rounds of interaction with the prover and cost $\mathcal{O}(\log n)$. The RRR compiler is useful for problems such as determining if a graph is free of triangles, or the fact that (for any fixed constant t) a graph has treewidth at most t [EJT10]. We do not know whether these problems admit a centralized protocol with small verification time, yet we know that both can be solved with an efficient use of space.

Finally, the GKR compiler, which shares similar properties to the RRR compiler, allows us to obtain protocols on languages that can be decided by circuits of low depth. More precisely, let *uniform* NC, be the class of languages that can be decided by a family of circuits constructible in $\mathcal{O}(\log n)$ space, which have size $\operatorname{poly}(n)$ and whose depth is $\mathcal{O}(\operatorname{poly}\log n)$.

Theorem 1.17 (by [GKR15]) *Let $\mathcal{L}$ be a language in uniform* NC*. Then, there exists an* AM *protocol where the verifier runs in time $n \cdot \operatorname{poly}\log(n)$ with bandwidth $\operatorname{poly}\log(n)$ with oracle access.*

By combining this result with Theorem 1.15 we have that this language admits a dAM *protocol with $\operatorname{poly}\log(n)$ rounds and bandwidth $\operatorname{poly}\log(n)$.*

This last result gives another way for obtaining protocols. A specially interesting one is the problem of deciding whether the diameter of a graph has a given value, which we know to be in NC [ABCP92] (and therefore has a $\operatorname{poly}\log(n)$ round protocol with

bandwidth poly $\log(n)$). On the other hand, we know that any node in the network needs to send $\Omega(n)$ bits in a dM protocol [CHPP20] (which coincides with non-deterministic verification) .

Now, we describe the protocol that is used the most in this work, which is the one for constructing a *spanning tree*. This is very useful as it allows us to locally verify the presence of global properties of the graph, like the existence of a unique marked node, as well as computing and verifying functions that can be aggregated across the tree, such as partially adding the total number of nodes or edges "up" the tree, or computing a sum of values spread across the graph.

Protocol 1.18 (Spanning tree) *For this problem, each node $v \in G$ has a single marked edge e_v and the nodes need to collectively verify that $\{\mathrm{e}_v\}_{v \in V}$ represents a tree.*

Notice that a protocol for this problem using $\mathcal{O}(\log n)$ bits is equivalent to *constructing* the tree entirely as a tool in other protocols, instead of simply verifying its correctness. Indeed, Merlin can also provide the identifier of the parent of a node v along with the proof and thus, verify that this new set of edges forms a tree. Now, we describe dM protocol using $\mathcal{O}(\log n)$ bits.

Merlin provides each node v with a pair $\langle \mathsf{id}(\rho), d_v \rangle$ where the first message represents the identifier of a root ρ for the tree and d_v represents the distance between v and ρ along the unique path between them in T. During the verification round, all nodes exchange these messages along with their identifier $\mathsf{id}(v)$ and check that:

1. If t_v is the other endpoint of e_v, which we consider to be its parent over T, it holds that $d_v = d_{t_v} + 1$.

2. All nodes check that they received the same identifier $\mathsf{id}(\rho)$ and those with $d_v = 1$ check that ρ is the other endpoint of e_v.

Notice that both parts of the message require $\mathcal{O}(\log n)$ bits. Now we check the correctness of the protocol.

- **Completeness.** An honest prover will provide a unique valid identifier $\mathsf{id}(\rho)$, along with the correct distances to ρ, therefore all nodes accept.

- **Soundness.** Merlin must provide a unique valid identifier for ρ as otherwise the nodes would notice when they exchange messages that there is more than one identifier and those nodes v with $d_v = 1$ must have ρ as a neighbor. Also, if there exists a set $C \subseteq V$ such that $\{\mathrm{e}_v\}_{v \in C}$ forms a cycle, the node with lowest value of d_v among those in C would reject as its parent in the tree must have a larger distance value.

As for the chance of obtaining a better protocol for this problem, it was shown in [GS16] that, any proof for the SPANNING TREE problem in the dM model, requires messages of size $\Omega(\log n)$.

Now we include the (multi)set equality protocol described in [NPY20], which is the basis for the compiler in Theorem 1.15, where the use of interaction proves really useful and, by its generality, it can be used in many different essential protocols. Here, we have a pair of (multi)sets $\mathcal{A}$ and $\mathcal{B}$ whose elements are spread across the graph and we wish to check that they match on each element's multiplicity.

Protocol 1.19 (Equality) *For this problem, each node v in a graph G has a pair of lists $\{a_v^i\}_{i=1}^{\ell}$ and $\{b_v^i\}_{i=1}^{\ell}$ as input , with $\ell \leq n$ where each element in the list can be described with $c \log n$ bits and the goal is to decide if the following two multi-sets are equal:*

$$\mathcal{A} = \{a_v^i : v \in V, i \in [l]\} \quad \textit{and} \quad \mathcal{B} = \{b_v^i : v \in V, i \in [\ell]\}.$$

We show that this can be decided using two rounds of interaction and $\mathcal{O}(\log n)$ bits:

First, consider that any element a of $c \log n$ bits is can be thought of as part of a finite field $\mathbb{F}_q$ with $n^{c+3} \leq q \leq 2n^{c+3}$ so that any multiset $\mathcal{A}$ is represented by a polynomial $p_{\mathcal{A}}(\cdot)$ as follows:

$$\mathcal{A} \rightsquigarrow p_{\mathcal{A}}(x) = \prod_{a \in \mathcal{A}} (a - x).$$

As each multiset has size at most n, each polynomial has degree at most $n \cdot \ell \leq n^2$, and so for any pair of multisets $\mathcal{A}$ and $\mathcal{B}$, the functions $p_{\mathcal{A}}, p_{\mathcal{B}}$ can match in at most n^2 different points. And so, if $\mathcal{A}$ does not match $\mathcal{B}$, by taking an element $s \in \mathbb{F}_q$ uniformly at random, the probability that their evaluations $p_{\mathcal{A}}(s)$ and $p_{\mathcal{B}}(s)$ match is at most $\frac{1}{n^c}$. With this we may proceed to describe the protocol.

Each node v generates a seed s_v and some number α_v with $\alpha_v \in [n^2]$ (which requires $\mathcal{O}(\log n)$ bits) and sends them to Merlin. Then, Merlin answers by broadcasting the pair $\langle \bar{s}, \bar{\alpha} \rangle$ where $\bar{\alpha}$ is the smallest value for α_v drawn by a node $\bar{v}$ along with $\bar{s}$. Also, he sends the triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ where ρ is the root of a spanning tree T of G, t_v corresponds to the parent of v at T and d_v is the distance between v and ρ through T.

Finally, to each node v he sends the pair $\langle p_{\mathcal{A}}^v(\bar{s}), p_{\mathcal{B}}^v(\bar{s}) \rangle$ which corresponds to the partial product of $p_{\mathcal{A}}$ and $p_{\mathcal{B}}$ at $\bar{s}$ up to the sub-tree T_v.

Now, during the verification round, the nodes exchange the messages previously described and locally check that

1. The triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ is consistent according to Protocol 1.18.
2. There exists a single node $\bar{v}$ that sent $\alpha_{\bar{v}}$ and $s_{\bar{v}}$.
3. Considering C_v to be the set of children of v at T it holds that

$$p_{\mathcal{A}}^v(\bar{s}) = \prod_{u \in C_v} p_{\mathcal{A}}^u(\bar{s}) \times \prod_{i=1}^{\ell} (a_v^i - \bar{s}), \quad p_{\mathcal{B}}^v(\bar{s}) = \prod_{u \in C_v} p_{\mathcal{B}}^u(\bar{s}) \times \prod_{i=1}^{\ell} (b_v^i - \bar{s}).$$

Accepting if these conditions occur and rejecting otherwise. As for checking that $\mathcal{A}$ and $\mathcal{B}$ are equal, the node ρ owns the correct values for $p_{\mathcal{A}}(\bar{s})$ and $p_{\mathcal{B}}(\bar{s})$. Therefore, ρ compares these values and accepts if they are equal and reject otherwise. As stated above, the probability that the protocol fails is the probability that there exists a node v where Merlin provided the wrong function $p_{\mathcal{A}}(\bar{s})$. And we have that these values match with arbitrarily low probability.

Finally, if all previous messages in the tree are consistent, then the root ρ has the correct representations of $\mathcal{A}$ and $\mathcal{B}$ and, if they are different, these values will differ with high probability.

One variation of the previous protocol which is used several times in this work, is that of checking if a given set of labels in the graph corresponds to a permutation in $[n(G)]$. As an honest prover provides an ordering of the nodes in the graph satisfying some property, we need to verify that this set of labels in the graph are all distinct and lie in the interval $[n(G)]$.

Protocol 1.20 (Permutation) *For this problem, every node v in an n-node graph G owns a number $\{a_v\}_{v\in G} \in [n]$ and they wish to verify that all numbers are distinct.*

Protocol for PERMUTATION

1. Local: Each node v, given its input a_v defines y_v to be $a_v + 1 \mod n$.
2. $P \leftrightarrow v$ (messages 1-2): Both Merlin and the verifier interact through the Equality protocol in order to check that the sets $\{a_v\}_{v\in G}$ and $\{y_v\}_{v\in G}$ are equal.

Most protocols have cost $\Omega(\log n)$ as a natural barrier, as essentially any action below this message size impedes a node from sharing its identifier, as well as counting the size of the graph, or choosing a leader [GS16]. Here we mention a three round protocol by [NPY20], that requires $\mathcal{O}(1)$ bits for constructing a spanning tree which, as described above, is specially useful for spreading information through the graph. We describe the protocol along with a proof of its correctness for the sake of completeness.

Protocol 1.21 (Spanning Tree with $\mathcal{O}(1)$ bits)

Merlin picks an element ρ and sends to each node v its distance to ρ in a BFS tree rooted at ρ modulo 3, that is, $d_v \mod 3$. Then, each node partitions its neighborhood into three sets. Those who have distance $0, 1$ and 2 modulo 3 respectively.

$$L_v = \{u \in N(v) : d_u \equiv_3 d_v + 1\} \quad M_v = \{u \in N(v) : d_u \equiv_3 d_v\}$$

$$U_v = \{u \in N(v) : d_u \equiv_3 d_v - 1\}$$

If a node v is such that $U_v = \emptyset$ it declares itself to be the root of the tree.

This is important as any BFS tree in a graph G has the following property: for any node v all its neighbors are either in the same layer as v, one layer above or one layer below.

Then, each node defines its parent in the tree to be the node u with the smallest port ordering (that is, the anonymous order in its neighborhood input) such that $u \in U_v$. Thus, for the case of a Yes-instance, an honest prover will originally provide a BFS tree which after this modification remains a tree rooted at ρ even though it may loose its original structure.

In the next round, each v generates a random bit $b_v \in \{0, 1\}$ which it sends to Merlin. He then responds to each v with the parity of the sum of random bits sent by all nodes in the unique path P_v between v and ρ, that is, $s(v) = \sum_{u \in P_v} b_v \mod 2$. He also sends b_ρ, which is the random bit originally sent by the root.

Finally, if we set t_v to be the parent selected by v after the first round, v locally checks that $s(v) = s(t_v) + b_v \mod 2$ and that all its neighbors received the same bit b_ρ supposedly sent by the root.

- **Completeness.** It follows directly that an honest prover will choose only one root and constructs a correct BFS tree and its distances.

- **Soundness.** As all nodes choose a unique parent before the random bits are drawn, two problems could arise:

  1. The tree has no root. If T has no root, then it must contain a cycle C. Then, with probability $1/2$ we have that $\sum_{v \in C} b_v \mod 2 = 1$. Therefore, if C can be written as $v_1, v_2, ..v_k$, such that $s(v_i) = s(v_{i+1}) + b_{v_i} \mod 2$ for each i, we can show inductively that for any value of $s(v_i)$ this leads to a contradiction.

  2. The tree T has more than one root. If this is the case, let ρ, ρ' be two different roots, then with probability $1/2$ we have that $b_\rho \neq b_{\rho'}$.

  As described above, the protocol has one-sided error. Therefore, by amplifying the error in the standard way we can reduce the acceptance error to any ε using a constant number of bits.

Protocol for SPANNING TREE

1. $\mathcal{M} \to v$: The prover chooses a root ρ, constructs a BFS tree T and sends each node $d_v \mod 3$.
2. $v \to \mathcal{M}$: Each v generates a random bit b_v and sends it to the prover.
3. $\mathcal{M} \to v$: The prover answers to each node with the bit b_ρ and the value of $s(v) = \sum_{u \in P_v} b_v \mod 2$, where P_v is the unique path between ρ and v.
4. Local: All nodes exchange their values of b_v, d_v and s_v and then check that $s(v)$ is consistent with its parent's and that they received the same b_ρ.

Chapter 2

Protocols Using a Single Interaction

In this chapter we study problems that can be solved by one round of interaction. First, in Sections 2.1 and 2.2, we study simple protocols for the problems TWINS and d-DEGENERATE respectively, as well as their complementary classes. These are simple problems that act as benchmarks for different models in the literature [FOZ16, BMRT14] and their protocols are sufficiently simple to give a first approach to the study of distributed interactive proofs. Then, we study two problems related to graphs that admit a global structure by way of an intersection model: The graphs in each of these problems can be represented by a set of objects, where the adjacencies of each node are determined by the intersections of these objects. In this regard, in Sections 2.3, 2.4 and 2.5 ,we study the problems of INTERVAL, CHORDAL and CIRCULAR-ARC respectively, along with the variants PROPER INTERVAL and PROPER CIRC-ARC . We show that all these problems are in $\mathsf{dM}[\log n]$. Later, in Chapter 4, we show that the bounds obtained for these problems are tight.

2.1 Degeneracy

The degeneracy of a graph G , written as $\mathsf{deg}(G)$, can be defined as the maximum value of the minimum degree over all induced subgraphs, or as the smallest integer d such that we may sequentially *prune* the graph deleting at most d nodes at each step. From this we obtain the following result:

$$\mathsf{deg}(G) = \max_{H \subseteq G} \delta_H = \min_{\pi \in S_n} \max_j d^+_\pi(v_j),$$

where S_n is the set of all possible permutations of $[n]$, and d^+_π is the *out-degree* of the directed graph induced by π, where $\vec{uv} \in E \Leftrightarrow \pi(u) < \pi(v)$. This represents the order in which nodes are trimmed.

Observe that, for any fixed d, d-degenerate graphs are *sparse*. Indeed, we know that if a graph is d-degenerate, then it can be encoded using $d \cdot n \log(n)$ bits following the ordering described above. For example, it is known that the classes of trees and planar graphs are 1 and 5-degenerate respectively. Any graph G with no copy of H as an induced subgraph

satisfies that $\mathsf{deg}(G) \leq 4\mathsf{ex}(H,n)/n$, where $\mathsf{ex}(H,n)$ denotes the *extremal number* for H [DKO14], which is the maximum amount of edges an n-node graph can have without having H as a subgraph. Also, if a graph is H-minor free, there exists a constant c_H such that G is c_H-degenerate. By bounding the degeneracy, we may approximate the arboricity of a graph up to a factor of 2, from which several tools used in distributed computing can be derived, such as partitioning the graph into layers where each node has a bounded number of neighbors in upper layers [BE10].

Getting an upper bound for computing the degeneracy

In the problem d-DEGENERATE we wish to verify that, given an undirected graph G, the value of $\mathsf{deg}(G)$ is upper bounded by d.

Proposition 2.1 d-DEGENERATE $\in \mathsf{dM}[\log n]$

Proof. Given the above definition, it is possible to identify the elimination ordering π for G by directing the edges of G such that the resulting graph is acyclic and its *out-degree* at every node is at most d. Indeed, it suffices for Merlin to assign to each node v its position π_v in the elimination ordering and, during the verification round, all nodes exchange these values and locally direct their edges to their neighbors with a smaller label. A node rejects if it has more than d neighbors with a label lower than π_v

- **Completeness.** An honest prover will provide the right elimination ordering π such that all nodes have at most d neighbors with a lower label and, therefore, all nodes accept.
- **Soundness.** If we consider $\vec{G}$ to be the graph obtained by orienting the edges according to π, then it cannot have any cycles as, otherwise, there would be a sequence of nodes in the cycle $v_1, ..v_k, v_1$ such that their positions according to π would decrease at some point, which is not possible. Then, if the graph has degeneracy bigger than d, then some node has out-degree bigger than d and, therefore, it rejects.

□

Protocol for d-DEGENERATE

1. $\mathcal{M} \to v$: The prover sends to each v its position in an elimination ordering π_v.
2. Local: Each v checks that it has at most d neighboring nodes at a higher position according to π, accepting accordingly.

The simplicity of this protocol turns out to be an interesting result when we relate it to the compiler described in Section 1.4. As we know from Theorem 1.17, any problem computable with small depth circuits admits a dAM protocol with bandwidth $\mathrm{poly}\log(n)$. Yet, we know that computing the degeneracy of a graph is P-complete [KT96], and therefore very unlikely to be in NC. This may tell us that problems in NC may admit more

concise protocols and that it is unlikely for the hardness of a problem according to its circuit complexity to relate to its distributed interactive proof complexity. On a side note, the fact that computing the degeneracy can be done in linear time by sequentially removing minimum degree nodes, by using the other variant of the compiler we obtain a three round protocol with cost $\mathcal{O}(\log n)$, which we still improve upon by use of a single round.

Verifying high degeneracy

To determine if the degeneracy of a graph G is strictly bigger than d, it suffices to mark a subgraph F such that the minimum degree in such subgraph is strictly bigger than d. In one round of interaction the prover can send to each node v the triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ that defines a spanning tree, a single bit $b_F(v)$ such that $b_F(v) = 1 \Leftrightarrow v \in F$, and the minimum degrees of the nodes in F intersected with the nodes in the subtree T_v associated to v (from here on referred to as a *sum protocol*), which we denote by $\delta_F(T_v)$ (assigning $+\infty$ if there are none).

Then, the nodes locally check the correctness of the tree according to Protocol 1.18, and the sum protocol for computing the minimum degree of nodes in F. This means that, each node v must compare its value for $\delta_F(T_v)$ with that of its children in the tree T, and check that its value corresponds to the minimum among all values provided by its children, rejecting if this is not the case. Then, as the values received by the leaves of the tree T are correct, by an inductive argument it follows that the root of the tree, denoted by ρ, must hold the correct value for $\delta_F(T_\rho) = \delta(F)$. Finally, the root ρ accepts if and only if the value obtained for the minimum degree is strictly bigger than d.

Proposition 2.2 $\overline{\text{d-DEGENERATE}} \in \mathsf{dM}[\log n]$.

Protocol for $\overline{\text{d-DEGENERATE}}$

1. $\mathcal{M} \to v$: The prover sends to each v the triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ and a single bit $b_F(v)$.
2. $\mathcal{M} \to v$: The prover sends to each v: the degree $d_F(v)$ and $\delta_F(T_v)$, the minimum degree of F intersected with the nodes in T_v.
3. Local: Each node v checks the spanning tree, the value for $d_F(v)$ and checks that $\delta_F(T_v) = \min\{\min_{u:t_u=v} \delta_F(T_u), d_F(v)\}$, accepting iff these values are correct. The root ρ accepts iff $\delta_F(T_\rho) = \delta_F > d$.

2.2 Twins

We now focus on the problem TWINS. For a graph G, the problem TWINS consists of determining if there exists a pair of nodes $u, v \in V$ such that their neighborhoods are equal.

That is, $N(u) = N(v)$ or $N[u] = N[v]$. In the case that u, v are adjacent ($N[u] = N[v]$) we refer to them as *true* twins and, otherwise, we refer to them as *false* twins. This problem, despite being a simple one to describe, has been used as a benchmark in different distributed models as it allows to encode problems such as ExistsEq in the context of multi-party simultaneous messages [FOZ16], where k players wish to check if at least two of them own the same input.

We show how to construct a dM protocol for this problem that relies in connectivity and has cost $\mathcal{O}(\log n)$, which we then get down to constant cost using interaction.

Proposition 2.3 twins $\in$ dM$[\log n]$.

Proof. Suppose that G has a pair of twins u and v, then Merlin simply marks both u and v with $\bullet$, so that they and their neighbors know they are twins. Then, Merlin marks each neighbor of u and v with $\boxplus$ to mark these as neighbors of u and v. Finally, the prover sends a spanning tree triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ for a tree T and a sum protocol to verify there are at most nodes marked with $\bullet$: Each node v receives a value $m(T_v)$ corresponding to the number of $\bullet$-marked nodes in the subtree T_v induced by v, then v compares this message with those received by its children, rejecting if the sum of the messages received by its children do not match $m(T_v)$. By an inductive proof, it follows that ρ has the total amount of $\bullet$-marked nodes, rejecting if $m(T_\rho) \neq 2$.

During the verification round, all nodes check the tree structure following Protocol 1.18, and the sum protocol as described above. Finally, all $\bullet$-marked nodes check that they have at most one $\bullet$-marked neighbor and every other neighbor is marked with $\boxplus$. Also, all $\boxplus$-marked nodes check that they have exactly two $\bullet$-marked neighbors, rejecting if any of these conditions do not occur.

- **Completeness.** If G has a pair of twins u, v, an honest prover marks u and v with $\bullet$ and all their neighbors with $\boxplus$, marking no other node. Then, all nodes correctly check that there are two $\bullet$-marked nodes and that the marks of their neighbors are correct, making all nodes accept.

- **Soundness.** If G has no pair of twins, by the correctness of the sum protocol, there must be exactly two marked nodes u and v, which must check the correctness of the marks received by their neighbors. Hence, either u, v or their neighbors notice that the marks received by them are not consistent, as $N(u) - v \neq N(v) - u$ and, therefore, one of these nodes reject.

□

Protocol for TWINS

1. $\mathcal{M} \to v$: The prover chooses a fixed pair of twins and sends the mark $\bullet$ if the node is a *twin* or $\boxplus$ if it is the neighbor of a *twin*, besides sending a triple for verifying a spanning tree $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ and a sum protocol for computing the number of nodes marked with $\bullet$.
2. Local: The nodes check that the spanning tree is correct. Nodes marked with $\bullet$ check that all neighbors are marked with $\boxplus$, while those marked with $\boxplus$ check that they received exactly two $\bullet$ and, through the tree T, they check that there are at most two $\bullet$.

Note that it is important to use a spanning tree in order to check that there are at most two $\bullet$ marked nodes, as otherwise the graph could be tricked into accepting a set of nodes that share only a portion of their neighborhood (we show this in detail in Section 4.3). Finally, we observe that by using Protocol 1.21 for computing a spanning tree, we easily obtain that TWINS $\in \mathsf{dAM}[3, 1]$.

Corollary 2.4 TWINS $\in \mathsf{dMAM}[1]$.

Graphs with no pair of twins

Now we study the complementary class for our original problem. For the problem TWIN-FREE given a graph G, the nodes wish to check if G does not have any pair of nodes with the same open neighborhood. Such problem can be directly solved using two rounds and $\mathcal{O}(\log n)$ bits.

Proposition 2.5 TWIN-FREE $\in \mathsf{dAM}[\log n]$.

Proof. Each node v interprets its identifier $\mathsf{id}(v)$ as an element in a field $\mathbb{F}_q$ with $n^{c+3} \leq q \leq 2n^{c+3}$. Then v defines a function to represent its open neighborhood as $p_v = \prod_{u \in N(v)} (\mathsf{id}(u) - x) \mod q$. Notice that such a polynomial has degree at most n.

Next, each v sends to Merlin a pair $\langle s_v, \alpha_v \rangle$ where s_v is a seed in $\mathbb{F}_q$ and, therefore, can be represented by $\mathcal{O}(\log n)$ bits and α_v is a number in $[n^2]$. The prover sends to each node v the seed $\bar{s} = s_{v^*}$, where $\alpha_{v^*} = \min_v \alpha_v$, besides a spanning tree triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ and their function evaluation $p_v(\bar{s})$.

Finally, all nodes check the correctness of the tree and the seed. While the tree is verified according to Protocol 1.18, the seed is verified by checking that there is a unique element providing the smallest value for α_v (which occurs with high probability) and that its value for s_v matches the seed received by the network. Afterwards, the nodes reconstruct their function $p_v(\bar{s})$. Here, node v rejects if and only if there exists a pair of nodes in its closed neighborhood $u \neq w \in N[v]$ such that $p_u(\bar{s}) = p_w(\bar{s})$.

- **Completeness.** If the graph has no pair of twins, the probability that some node

rejects is the probability that there exists a pair such that their function evaluations match for the seed $\bar{s}$. Then, if $\bar{s}$ was chosen uniformly at random, the probability that this occurs is at most $\frac{1}{n^c}$.

- **Soundness.** If G has a pair of twins, namely u and v, then the graph will always reject, as there will be a node in their neighborhoods that finds the pair $p_u(\bar{s})$, $p_v(\bar{s})$ and reject.

□

Protocol for TWIN-FREE

1. $v \to \mathcal{M}$: Each v sends the pair $\langle s_v, \alpha_v \rangle$, corresponding to a candidate for the seed, and a number in $[n^2]$. to choose it.
2. $\mathcal{M} \to v$: The prover sends to each v the triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$, along with the pair $\langle \bar{s}, \alpha_{\bar{v}} \rangle$. Last, he sends the evaluation $p_v(\bar{s})$.
3. Local: All nodes check that the tree is correct and that their evaluations are consistent. A node rejects if there exists a pair in its neighborhood whose valuations $p_u(\bar{s}) = p_w(\bar{s})$ match.

As this corresponds to a protocol with perfect soundness, no prover can fool the graph and make it accept. Yet, as the chance of error lies in choosing a bad seed s, we simply need to ask Merlin to choose a seed s that will make us accept, provided we can assure that one exists.

If a graph G has no twins, then for each pair of nodes u, v their neighborhoods (and, therefore, their encodings) do not match. Then, if we consider the set of elements s in $\mathbb{F}_q$ such that there exists a pair of nodes u, v with $p_u(s) = p_v(s)$, then there can be at most n^3 such elements, as each encoding $p_u(\cdot)$ has degree n and therefore $(p_u - p_v)(\cdot)$ has at most n roots.

If we simply consider $q = \Omega(n^4)$ then there must exist an element for which no pairs match. And we can replicate the previous protocol by asking Merlin to provide such an element and proceed as before. This shows that TWIN-FREE can be solved in a single round of interaction and $\mathcal{O}(\log n)$ bits.

Corollary 2.6 TWIN-FREE $\in \mathsf{dM}[\log n]$.

2.3 Proper Interval Graphs

We start with the class of interval graphs, that is, the class of graphs that admit a representation through the intersection of intervals on the real line. These graphs have several applications related to optimization theory, mostly concerning resource allocation through scheduling problems [BNBYF+01]. The class of interval graphs has not been studied in the distributed setting, other than reconstruction results on hereditary graph

classes [MPSRT18]. Yet, we consider it to be an interesting first step as its structure allows to make good use of the assistance given by a prover.

By defining our problem and studying a simpler variant when the intervals to be considered are not contained in any other. The general case (which allows for inclusions) is studied alongside chordal graphs in Section 2.4, an example of an interval graph and its representation by intervals can be seen in Figure 2.1.

Definition 2.7 *We say that a graph $G = (V, E)$ is an interval graph if there exists a universe $\mathcal{U} = (0, \text{poly}(n(G))) \subseteq \mathbb{Z}$ and a family of interval configurations $\{I_v\}_{v \in V}$ such that the adjacency of G is uniquely determined by the intersections between each pair of nodes' intervals, that is:*

$$\forall v \in V, \exists I_v \subseteq \mathcal{U} : \quad \forall u, w \in V, \quad uw \in E \iff I_u \cap I_w \neq \varnothing$$

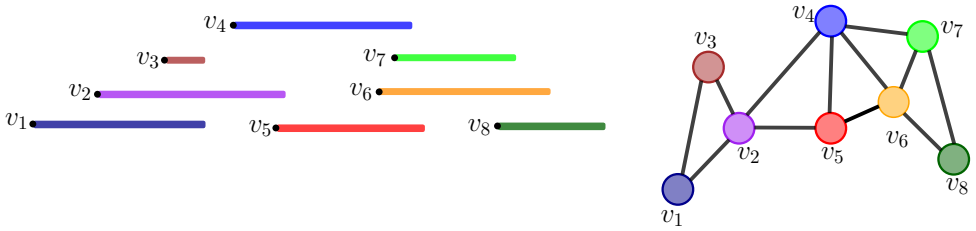

Figure 2.1: An example of an interval graph. On the left side a representation of the graph by overlapping intervals. On the right side its associated graph given by the intersection of these intervals.

We say that G is also *proper* if for no pair of intervals one is properly contained in the other. A graph G will be called *unitary* if all intervals have length one. It is known that these two definitions coincide: any proper interval graph admits a representation via unit length intervals [Rob69].

Proper interval graphs

We begin by studying the class of proper interval graphs, which admit a characterization which is suitable for a distributed algorithm [Gar07]:

Proposition 2.8 *A graph G admits a representation by proper intervals if and only if there exists an ordering $\{v_i\}_{i=1}^n$ such that:*

$$\forall i, j : i \leq j, \quad v_i v_j \in E \implies v_i v_k \in E \text{ and } v_k v_j \in E, \quad \forall i \leq k \leq j$$

Having this characterization, it is immediate to construct a protocol.

Proposition 2.9 PROPER INTERVAL $\in$ dM$[\log n]$.

Proof. The proof that Merlin sends to a node v is comprised of three different messages:

1. Its interval's center, given by $c(v)$, as well as its position when ordering each center from left to right, given by π_v.

2. The id of the node whose center is farthest to the left, which we call v_1, along with the corresponding value $c(v_1)$, and a spanning tree triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ to verify its correctness.

3. A range, given by $[\pi_{v_{\min}}, \pi_{v_{\max}}]$ that indicates the neighbors of v according to their positions.

Then, the nodes locally check that the tree is correct according to Protocol 1.18. They also check that the value for the size of the graph $n(G)$ is correct and that there is a unique node with the smallest value for its intervals' center, given by $c(v_1)$. The first condition can be checked by providing each node with the number of nodes up to the subtree T_v induced by each node v; in this way, it suffices for each node to compare its value with the sum of those received by its children and rejecting if these are not equal. By an inductive argument, this leaves the root ρ with the correct value for $n(G)$ and it rejects if it does not match with the number provided by the prover. The argument for checking that v_1 is unique follows in a similar manner.

From here on out, all nodes check that their neighbors intervals intersect theirs, and that each of their neighbors have an assigned $[\pi_{v_{\min}}, \pi_{v_{\max}}]$ such that all positions are covered. Finally, each node v from v_1 onward such that $\pi_v = i$ checks that it has a unique neighbor with position $i + 1$ whose center is immediately right of his. Node v rejects if any of these conditions does not hold.

- **Completeness.** We can check directly that an honest prover will send the identifier of the node whose center is furthest to the left in the real line and send each node their correct position in the ordering, where each node has a neighborhood matching the range it received.

- **Soundness.** We have that each node knows the size of the graph and there is a unique node positioned first. As all nodes positioned at $i \in [n-1]$ have a unique neighbor positioned at $i+1$, it follows that all nodes must have different positions. Now, if G is not a proper interval graph, by Proposition 2.8 there must exists a node such that its neighborhood does not match its range and therefore such node must reject.

□

Protocol for PROPER INTERVAL

1. $\mathcal{M} \to v$: Merlin sends to each v the identifier of the node whose center is farthest to the left, along with its center $c(v_1)$, and the range of positions that v must be neighbors with, given by $[\pi_{v_{\min}}, \pi_{v_{\max}}]$.
2. Local: The nodes check that there is a unique neighbor v_1 with the farthest left center. Also, if v is positioned at π_v it checks that it has a unique neighbor positioned at $\pi_v + 1$ and that its range matches its neighborhood's positions.

It is known that proper interval graphs can be recognized in the centralized setting (that is, by a single computation unit) in linear time in the size of the graph [HMPV00], which implies, by use of the NPY compiler, a protocol in $\mathsf{dMAM}^{\text{priv}}$ with cost $\mathcal{O}(\frac{n+m}{n} \log n)$. As this class contains the class CLIQUE among other classes with a large number of edges, this implies that the protocol can be of cost up to $\mathcal{O}(n \log n)$, showing that our protocol is a great improvement both in the number of rounds and the bandwidth.

2.4 Chordal Graphs and Interval Graphs

A graph G is said to be *chordal* if any cycle of length at least four has a chord. That is, for any integer $k \geq 4$, G does not have a C_k as an induced subgraph. These graphs are specially relevant from an algorithmic perspective as several graph properties (finding the largest independent set or clique, computing the chromatic number, etc) can be efficiently computed when the input graph is restricted to this class [Hoà94, RTL76].

As for interval graphs, the structure of a chordal graph is determined by its maximal cliques: while an interval graph can be represented as a path formed by its maximal cliques, in the case of chordal graphs these can be seen as a tree.

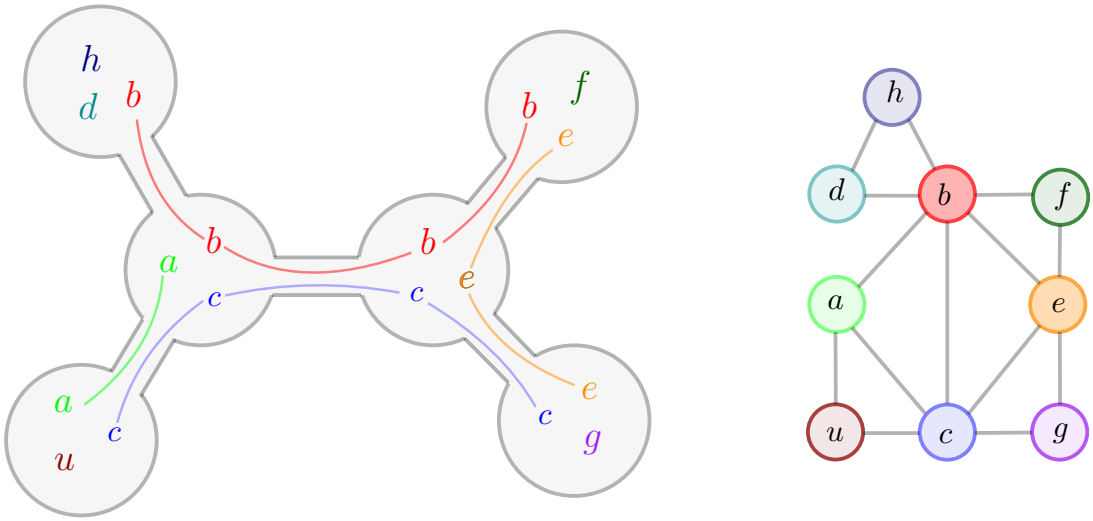

Figure 2.2: An example of a chordal graph, represented in the right by its clique-tree, and in the left by its corresponding drawing. Notice that all bags containing a fixed node form a connected subgraph.

A tree decomposition of some graph G is a tree T_G where each node $b \in T_G$ (referred to as *bags*) represents a set of nodes $b \subseteq V$ in the original graph with the following properties: **(1)** each node $v \in G$ is present in at least one bag, **(2)** for every edge $e = uv \in E(G)$ there exists a bag b that contains both u and v and **(3)** if we define T_v as the set of bags in T_G to which v belongs, they form a connected subgraph of T_G. From this we can define a *clique-tree* of a graph G as the special case of a tree decomposition for G where each bag represents a maximal clique of G.

Similarly, we define a path decomposition of a graph G as a tree decomposition when the tree T_G in question is a path [Bod98]. Following the previous notation, we define a *clique-path* of a graph G as the special case of a path decomposition for G where each bag represents a maximal clique of G.

Proposition 2.10 ([BS+99]) *A graph G is said to be chordal if and only if it admits a clique-tree.*

Proposition 2.11 ([BS+99]) *A graph G is said of be an interval graph if and only if it admits a clique-path.*

Our goal in this section is to show a proof for recognizing chordal graphs using a single round of interaction. For this, we would like to find a way to simulate the nodes in the clique-tree by choosing a set of leaders for each maximal clique. Then, we would like to label each node with the range of cliques it belongs to. The problem is that, while interval graphs require only two endpoints to encode such a range, in the case of chordal graphs we would need to encode an entire subtree which would require labels of size $\mathcal{O}(n \log n)$. Therefore, we need to find a way to encode a tree.

For this, we show that we can "trim" the graph by sequentially removing nodes from it by using the tree structure. If we consider a clique-tree rooted at some node ρ_T and trim the graph in a series of d steps, with d the depth of the clique-tree, we can partition the set of nodes as follows. At each step i, we assume that the clique-tree T_G has depth i and look at the leaves at the deepest level in the clique-tree, and, for each such leaf b, delete all nodes which belong only to this bag and name F_b such a set. Then, we continue to step $i-1$ (where our new tree has depth $i-1$) and repeat this process. We know this set is non-empty by the maximality of the clique represented by the bag b. As this goes on for d steps, we have that a node v is eliminated from the clique-tree at the step corresponding to the lowest depth of a bag containing the node v, as it can be seen in Figure 2.2,

Lemma 2.12 *For any chordal graph G, where T_G is a clique-tree rooted at some bag ρ_T, consider $\{M_b\}_{b \in T_G}$ to be its set of maximal cliques. Then, it is possible to partition the nodes in V into a family $\{F_b\}_{b \in T_G}$ such that for any pair of bags $b \neq b'$ with $\mathsf{depth}(b) \geq \mathsf{depth}(b')$ it holds that $F_b \subseteq M_b \setminus M_{b'}$.*

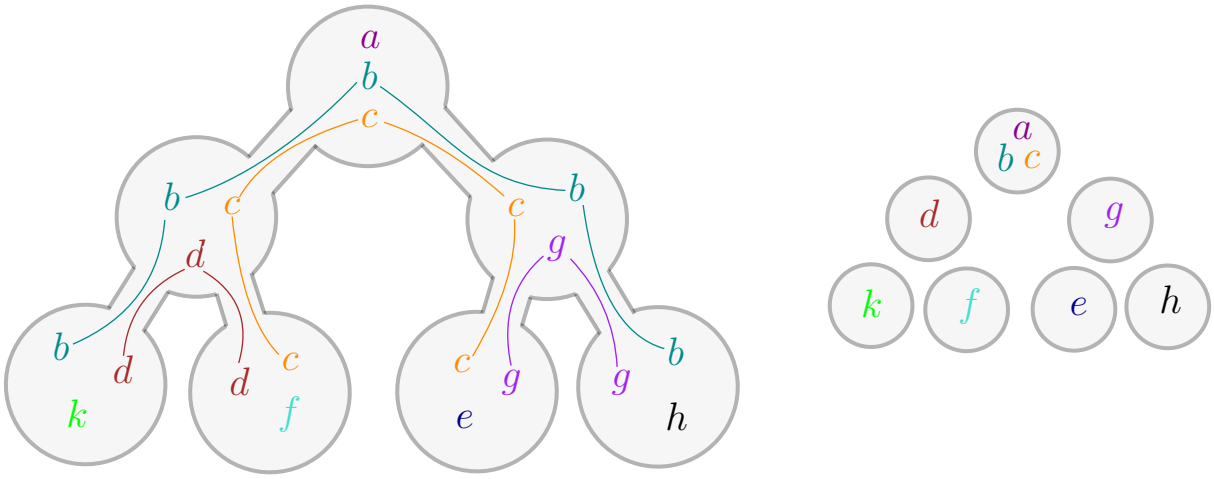

Figure 2.3: Graph partition according to Lemma 2.12. Given a tree decomposition, each node v is positioned at a different set depending on the bag containing v such that its depth is the lowest in the clique-tree.

Proof. We show this by induction on the number of bags in the clique-tree of a graph G, given by $|T_G|$. Indeed, if T_G is composed of only two bags b and b' with T_G rooted at b then, as both M_b and $M_{b'}$ are maximal cliques, we simply consider $F_{b'}$ to be $M_{b'} \setminus M_b$ and $F_b = M_b$. Clearly these sets are disjoint.

Consider now $|T_G| = k$ with $k \geq 3$. Then, as T_G is a tree, there must exists a bag $b \in T_G$ with degree 1, with b' its parent in T_G. We then have that $F_b = M_b \setminus M_{b'}$ is disjoint with all other bags in the tree T_G as, otherwise, if there exists a node in F_b that is also in a bag at a lower depth then, by the definition of a clique-tree, it must belong to $M_{b'}$. From there, we have that the tree $T' = T'_G - b$ is a clique-tree for the graph $G' = G - F_b$. It follows, by induction, that there exists a disjoint collection $\{F_{\bar{b}}\}_{\bar{b} \in T_G - b}$ with the above properties. Hence, the family $\{F_{\bar{b}}\}_{\bar{b} \in T_G - b} \cup \{F_b\}$ is as desired. □

Now, we would like to select a collection of leaders from each bag in T_G and provide a proof to these leaders in order to simulate the overlaying clique-tree. Two difficulties arise:

1. The leaders in each pair of adjacent bags in G, may not be connected, as by construction the leader in some bag b does not belong to b's parent. Therefore, we would like to consider another collection of leaders (in the intersection of adjacent bags) in order to simulate T_G's edges.

2. If we could solve the first problem, we have no guarantees that we are able to choose a leader for each edge of T_G in an injective manner: it could be the case that a leader belongs to the $\Omega(n)$ maximal cliques adjacent to the same bag, and should therefore handle too many messages.

To solve the first issue, we show how to choose a root for T_G and a collection of nodes that belong to the intersection of adjacent bags in such a way that we are able to verify

the correctness of the tree structure.

Lemma 2.13 *Given a chordal graph G, there exists a rooted clique-tree T such that it is possible to choose a collection of leaders for each bag $\{v_b\}_{b\in T}$ and auxiliary nodes $\{w_\ell\}_{\ell\in T}$ for each leaf in T such that, if $\mathsf{depth}(b)$ is the depth of a bag b in the tree and $t(b)$ is the parent of the bag b in T, then:*

- *For each $b \in T$, $\quad v_b \in b$.*
- *For each $b \in T$, $v_b v_{t(b)} \in E(G)$.*
- *If $\mathsf{depth}(b) \neq \mathsf{depth}(b')$, then $v_b \neq v_{b'}$.*
- *If $b \in T$ is a leaf, then $w_b v_b \in E(G)$.*
- *$\{v_b\}_{b\in T} \cap \{w_\ell\}_{\ell\in T} = \emptyset$.*

Proof. Let T be a rooted clique-tree in G such that its set of leaves is the largest and the sum of their depths is as small as possible. Consider now $b_\rho \in T$ to be the root of T, and let some arbitrary node $v_r \in b_\rho = b^0$ be its leader. If we define $\{b_i^1\}_{i=1}^{\ell}$ to be the children of b_ρ in T, for each i we choose an arbitrary leader v_i^1 in $b_r \cap b_i^1$, which is non empty as G is connected.

Consider b^j to be any node at level $j \geq 1$ of the tree with b^{j-1} its parent, with v^j its leader and $v^j \in b^j \cap b^{j-1}$.

- If b^j is a leaf, we choose an auxiliary node $w_j \in b^j$ with $w_j \notin b^{j-1}$. We can pick such a node because b^j represents a maximal clique in G and, otherwise, it would be contained in b^{j-1}.

- If b^j is not a leaf, let $\{b_i^j\}_{i=1}^{k}$ be the set of b^j's children. For each i we choose a leader for b_i^j in $b^j \cap b_i^j - b^{j-1}$ as, otherwise, we would have that $b^j \cap b_i^j \subseteq b^{j-1}$ for some i. This implies that $b_i^j \cap b^j \subseteq b^{j-1} \cap b_i^j$ and it would be possible to define a new tree T' where b_i^j is a child of b^{j-1} instead of b^j. Now, if b_i^j was not a leaf, we would have a tree with more leaves than T. Otherwise, in case that b_i^j were a leaf, the sum of the depths of each of its leaves decreases by one. This contradicts our choice for T.

  Hence, we can choose a leader v_i^j in $b_i^j \cap b^j$ for each value of i and then go for the next level.

From the construction we have that for any pair of adjacent bags in T, either their leaders match or are adjacent, as both belong to the intersection of their respective bags. Also, by the way the leaders were chosen in the latter point, by choosing bags at different depths, it follows directly that their leaders must be different. □

Now we have a set of leaders which belong to the intersection of bags at different levels of a rooted clique-tree. Yet, these leaders may have several bags from the same level

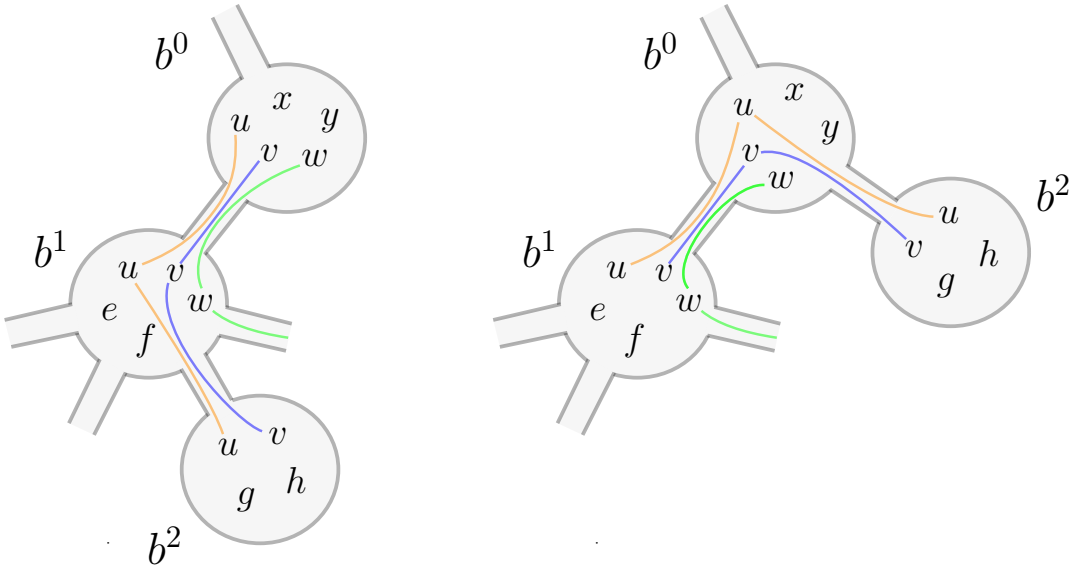

Figure 2.4: Transition from a tree decomposition to another one by reassigning a leaf at an upper level, in case that a bag intersection ($b^1 \cap b^2 = \{u, v\}$) is contained in an intersection at an upper level ($b^0 \cap b^1 = \{u, v, w\}$). We can do this while keeping a feasible decomposition.

assigned to it, as a multiple bags may share a unique element at the intersection with the previous level.

To solve this issue we simply need to make use of the tree structure by setting, for each leader of a bag ρ_b, its proof to be provided by one of its children (e.g. the one with the smallest identifier). If a node is leader of several bags, it still receives all corresponding proofs after these are exchanged at the verification round, with auxiliary nodes being chosen in order to cover the case when a node is leader of multiple leaves in the tree. Hence, we have that each bag leader will receive a unique message, no matter the number of bags it represents. Now we are ready to prove the theorem.

Theorem 2.14 CHORDAL $\in$ dM$[\log n]$.

Proof. Assuming that G is chordal, we first select a collection of nodes which represents a bag b in T_G. We do this by selecting a single element ρ_b from each set F_b according to Lemma 2.12, as well as a spanning tree triple for simulating the overlaying clique-tree, which we denote by $\langle \mathsf{id}(\rho_T), d_T(v), t_T(v) \rangle$ indicating the unique leader for the root of the clique tree, as well as the distance to it and ρ_b's parent in this structure.

Also, the prover to provides each node $v \in F_b$ with

- The size of the clique tree $|T_G|$, along with the id of the leader for its root ρ_T.
- A label $F(v)$ corresponding to the identifier $\mathsf{id}(\rho_b)$ of the leader in the set F_b to

which a node v belongs to, as well as the size of F_b.

- The distance from the bag b to the root ρ_T in the clique tree with $v \in F_b$, given by $\mathsf{depth}(v)$.

Also, in order to verify the tree structure, we choose a collection of leaders $\{e_{bb'}\}$ for each edge $bb' \in E(T_G)$ according to Lemma 2.13, as well as the corresponding auxiliary nodes to pass these messages. Then, the nodes exchange their messages and they check the following:

1. The collections $\{\rho_b\}_{b \in T_G}$ and $\{e_{bb'}\}_{bb' \in E(T_G)}$ verify in conjunction the correctness of the tree structure.

2. There is a unique root ρ_T.

3. If $v \in F_b$, then v checks that the nodes with $F(u) = \mathsf{id}(\rho_b)$ form a clique.

4. If v and u are adjacent with $\mathsf{depth}(v) \leq \mathsf{depth}(u)$, then v is adjacent to all the leaders ρ_b (and their sets F_b) in the unique path between $F(v)$ and $F(u)$ which also coincides with the unique path between $F(u)$ and ρ_T. In particular, if $\mathsf{depth}(v) = \mathsf{depth}(u)$, then they must have the same leader.

If all the previous conditions hold, then all nodes accept. Now, we check the correctness of this protocol.

- **Completeness.** Suppose first that the graph G is chordal. An honest prover will provide each node v with its correct set F_b according to the underlying clique tree T_G which all leaders check correctly. By the definition of the clique tree it follows that no node has a neighbor at the same depth from a different bag. As the set of bags that v belongs to correspond to a connected subgraph, it follows that if v is in a bag b and it is connected to a node u (which belongs to a bag b') at a bigger depth, then it is adjacent to all nodes in the bags (and therefore the sets $F_{b'}$) in the path between b and b'.

  With this, each node v recognizes that its sets F_b is a clique, and that the depth for the set of each of his neighbors is consistent with its set. Therefore, all nodes accept.

- **Soundness.** Suppose now that the graph G is not chordal. We have that, by the constructions in Lemmas 2.12 and 2.13, the leaders chosen for both the bags and the edges between them can correctly verify the structure of the clique tree. Now, suppose that G has an induced cycle $\{v_!, \dots v_k\}$ with its nodes arranged such that v_1 is the node of largest depth. It must be that at least one of them has different depth from the rest as otherwise they would reject either because their leaders are different, or because the corresponding set F_b should be a clique and there exists at least two non adjacent nodes. Suppose that $\mathsf{depth}(v_1) = i$, $\mathsf{depth}(v_k) = j$ and $\mathsf{depth}(v_2) = k$ with $i > j \geq k$. Then, it must be that v_k's leader lies in the unique path between ρ_T and v_1's leader as otherwise it would notice an inconsistency with

v_1's proof and it would reject. This must also be true for v_2 for being at a smaller depth. Then, it must lie in the path between v_1's leader and ρ_T and, therefore, adjacent to v_k's leader and subsequently to v_k itself, which contradicts the fact that they are not adjacent as they belong to a large induced cycle.

□

Protocol for CHORDAL

1. $\mathcal{M} \to v$: Merlin selects, a set of leaders ρ_b, $e_{bb'}$ for each bag b and each edge between bags bb', and sends to each of them the proof of a spanning tree structure between bags, given by $\langle \mathsf{id}(\rho_T).d_T(v), t_T(v)\rangle$.
   Also, he sends to each node v, a label $F(v)$ indicating the bag at the lowest depth to which v belongs to, as well as its size, the id of a leader for such a bag and the id of ρ_T, the leader of the root of the clique tree.
2. Local: Each bag leader checks, in conjunction with the leaders from the bag edges, the consistency of the tree structure. Each node v checks that it shares the same id for ρ_T, as well as checking that the nodes labeled by $F(v)$ form a clique and that, for each neighbor u with a leader at a higher depth, v sees both the leader of u and all cliques in the path from $F(u)$ to $F(v)$.

As a corollary, we obtain a dM protocol for the problem INTERVAL by considering the fact that, as described above, interval graphs are a particular subclass of chordal graphs where the clique-tree corresponds to a path [Bod98]. From here it suffices to repeat the same protocol while each leader (with the exception of the root leader) additionally verifies that any bag assigned to it has a unique children in the clique tree T_G.

Corollary 2.15 INTERVAL $\in \mathsf{dM}[\log n]$

Again, these results are an improvement over those derived from the NPY compiler as we showed a way to obtain a protocol using a single round of interaction and cost $\mathcal{O}(\log n)$ while, as chordal graphs (and interval graphs) can be dense graphs, the use of the compiler leads to a three round protocol with cost at most $\mathcal{O}(n \log n)$.

2.5 Circular Arc Graphs

Circular arc graphs are a natural extension of interval graphs. Indeed, they are the graphs that admit a representation by arcs on a circle, and appear in the study of resource allocation problems for periodic tasks [MP06]. We study this class of graphs as we wish to check whether previous results can be extended to this new setting without a large increase in the cost. We start by defining this new class and, again, studying two variations: where no pair of arcs are properly contained and then the general case.

For the sake of simplifying notation, we identify the set of id's $[n]$ with $\{0, \dots n-1\}$. We say that a graph $G = (V, E)$ admits a circular arc representation if there exists a

family of arcs in the unit circle $\{A_v\}_{v \in V}$ such that the adjacency of G can be determined by the intersection of arcs . That is

$$\forall v \in V, \exists A_v : \quad \forall u, w \in V, uw \in E \iff A_u \cap A_w \neq \varnothing$$

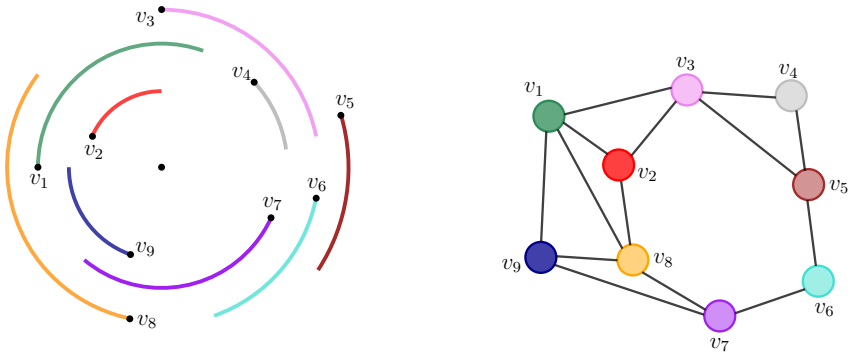

Figure 2.5: An example of a circular arc graph: the left side shows a representation with overlapping arcs in the circle, while the right side shows its associated graph construction.

We say that some graph G is a *proper* circular arc graph, if it admits a representation where no arc is contained in another.

2.5.1 Proper Circular Arc

Here we can try to extend our idea for verifying proper interval graphs by considering an order according to their arcs positions. Nevertheless, two problems arise.

The first is that, in contrast with proper interval graphs, these are a different class than its unit arc variant [DGS14] which is a minor issue as we can, instead of focusing on interval centers consider one of its endpoints and sort them accordingly. The second thing is that, in the case of proper interval graphs, the adjacency of a node could be given as an interval. Hence, it was easier to encode and verify. Now, the natural extension would be for a node adjacency to behave like an "interval in a circle". That is, if we had nodes numbered from 1 to n, we could say that, given $i < j$, the adjacency of a node is given either by (i, j), which we define as $\{i, i+1, \ldots j\}$ or (j, i), which we could set to be $\{j, j+1, ..n-1, 0, ..i\}$. If a graph G satisfies this property we say that its augmented adjacency matrix (the adjacency matrix of G with the addition of 1's in the diagonal), denoted by $M^*(G)$, has the *circular* 1's property [Tuc70].

Now, again, we have graphs that have this property, yet they are not proper circular arc graphs. So there must be a another property that we need in order to pin down this graph class. Fortunately, given a characterization by Tucker [Tuc70], we can show how to recognize this class with a single round of interaction. For this we start by giving some definitions for symmetric matrices.

First, consider π to be a permutation of $[n]$ and some matrix M. The matrix M_π is the resulting matrix when both the rows and columns of M are reordered according to π. Second, consider a symmetric $\{0, 1\}$-matrix M with 1's in the diagonal and the circular

1's property. Then, we define $\mathsf{last}[M, j]$ to be the largest value i such that $M_{i,j} = 1$ and $M_{i+1,j} = 0$. If such an i does not exists (meaning the column $M_{\cdot,j}$ has only 1 entries) we set $\mathsf{last}[M, j] = \perp$.

Last, for the sake of notation, consider $\sigma_{\text{inv}} : [n] \to [n]$ to be the permutation given by $i \to n - i$ if $i \neq n$ and n otherwise and $\sigma_{\text{sh}} : [n] \to [n]$ to be the permutation given by $i \to i + 1$ if $i \neq n$ and 1 otherwise.

Definition 2.16 *Given a symmetric $\{0, 1\}$ matrix M with ones in the diagonal, we say that it has circularly compatible 1's if M has the circular 1's property and, for any reordering π of the rows (and respective columns) of the matrix constructed by a finite composition of σ_{inv} and σ_{sh}, $\mathsf{last}[M_\pi, 0] \leq \mathsf{last}[M_\pi, 1]$, unless one of these values is $\perp$.*

With this we can finally describe the characterization by Tucker for this class of graphs, which we include without a proof:

Proposition 2.17 ([Tuc70]) *A graph G is a proper circular arc graph if and only if its nodes admit an ordering $\{\pi_v\}_{v \in V}$ such that its augmented adjacency matrix $M^*(G)$ has the circularly compatible 1's property.*

This characterization suits us greatly as its condition is highly local. If we were able to find such an ordering, we would only need to verify it by checking the previous and next nodes in the ordering. For this we note two important remarks.

Remark ([Tuc70]) If we sort the nodes according to their right endpoint in counter-clockwise order, we have that the nodes adjacency matrix has the circularly compatible 1's property according to this ordering.

Remark We can rotate the arcs in a graph in such a way that, if each node v is sorted according to the previous order π it follows that v is adjacent to $\pi_v - 1$ and $\pi_v + 1$ modulo n, with the exception of the last node (in position n) which may not be connected to the first.

Now we can start to describe the protocol.

Proposition 2.18 PROPER CIRC-ARC $\in \mathsf{dM}[\log n]$.

Proof. Assume that Merlin assigns to each node a pair $A_v = (r_v, \ell_v)$ which correspond to v's arc coordinates when the arc is visited in a counter-clockwise direction. Here we assume that such coordinates are given as a value in $(0, 360)$ with any pair of values being at distance at least $1/\text{poly}(n)$ from each other (and, as such, we require $\mathcal{O}(\log n)$ bits to encode them). Now, let v_1 be the node such r_v is the smallest and such that, in case the

graph is a proper interval graph, is the first node if we sort them according to their left endpoint as in Proposition 2.3.

Now, first we ask Merlin to provide the identifier $\mathsf{id}(v_1)$ of such a node, as well as a proof that it is the only node with the smallest right endpoint, which can be provided by sending a spanning tree triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ as well as a verification through the spanning tree that v_1 is the unique node with the smallest value for r_v. We also ask Merlin to provide the size of the graph $n(G)$ which can also be verified through the spanning tree.

Next, we ask Merlin to send to each node a position π_v such that $\pi_v = i$ means that r_v is the i-th largest value for a right endpoint in counter-clockwise order. Also, we ask Merlin to provide each node with a range $(v_{\min}, v_{\max})$ which correspond to the positions in π such that v's adjacency equals the set of nodes whose positions are $\{v_{\min}, v_{\min}+1, \dots v_{\max}\}$ given as a circular sequence as described previously.

Then, at the verification round, all nodes exchange these messages, verifying the existence of a node v_1 by using the spanning tree and, starting from v_1, each node v with $\pi_v = i$ checks that there is a unique node labeled by $i+1$ whose arc intersects with its own and whose left endpoint is immediately after his. They also check that their adjacency is circular, meaning that it has a neighbor labeled with each position in the range $(v_{\max}, v_{\min})$, with arcs consistent with such an order.

Finally, in order to check that the matrix has the circularly compatible 1's property, they do the following. In order to check the first two columns in each permutation obtained by a composition of σ_{inv} or σ_{sh} we simply ask each node to adjust its range according to these permutations, such that each node v positioned at π_v with neighbors w and u such that $\pi_w = \pi_v - 1$ and $\pi_u = \pi_v + 1$ must simply consider two cases: **(1)** When v is first and u is second, which occurs when we shift π (by applying σ_{sh}) until v is first in the order or **(2)** when v is first and w is second, which occurs when we invert the order by applying σ_{inv} and then shift π until v is first. In this way, the verification of all $2n$ possible permutation to is distributed among the nodes, with each v in charge of two cases.

We explain how to handle both cases by adjusting the range of v and that of its neighbors as follows:

- If v is first and u is second, we can obtain the corresponding range by setting $k = n - \pi_v + 1$ and translating both ranges by $k \mod n$ as $(\bar{v}_{\min}, \bar{v}_{\max}) = (v_{\min} + k \mod n, v_{\max} + k \mod n)$ and a similar construction for u.
- If v is first and w is second, first we obtain the range after applying σ_{inv} as $(\bar{v}_{\min}, \bar{v}_{\max}) = (n - v_{\max} + 1 \mod n, n - v_{\min} + 1 \mod n)$ and then shifting v to the first position by adding π_v on both sides modulo n, and a similar construction for w.

Given these two different ranges, each node checks that (unless either itself or w or u are universal nodes) the range $\bar{v}_{\max} \leq \bar{u}_{\max}$ (respectively $\bar{v}_{\max} \leq \bar{w}_{\max}$), accepting if this holds and rejecting otherwise.

We have that all these messages are of length $\mathcal{O}(\log n)$ in one round of interaction.

Therefore, it only remains to check the correctness of this protocol.

- **Completeness.** Suppose that G is proper circular arc, an honest prover will provide each node with an ordering π according to each arc's left endpoint, as well a the correct range which all nodes can verify and accept.

- **Soundness.** Now, suppose that G is a No-instance. From what was described above, all nodes correctly compute a starting node v_1 as well as the size of the graph. Also, each node v with $\pi_v = i$ checks that it has a unique neighbor positioned as $i + 1$, which is consistent with its arc. Combining both statements we have that all nodes must have different values in π that match the order of their left endpoints.

  Now, if all nodes check that their adjacency is indeed circular there must exists a node v which, when permuting its order such that it becomes first either $\bar{v}_{\max} > \bar{u}_{\max}$ or $\bar{v}_{\max} > \bar{w}_{\max}$, and then it would immediately reject.

□

The general case

To cope with the general case, it suffices to adapt the characterization found in [Tuc70] for this class of graphs. Given a symmetric $\{0, 1\}$-matrix M, with ones in the diagonal, consider a column i and define U_i as the set of $1's$ starting from the diagonal and going downwards in a circular manner, until a zero appears. Now, define V_i as the set of $1's$ on row i starting from the diagonal and going rightwards in the same manner. M is said to have the *quasi-circular* $1's$ *property* if all $1's$ in the matrix are covered by some U_i or V_i. It is important to mention that, since M is symmetric, we have that U_i and V_i have the same size.

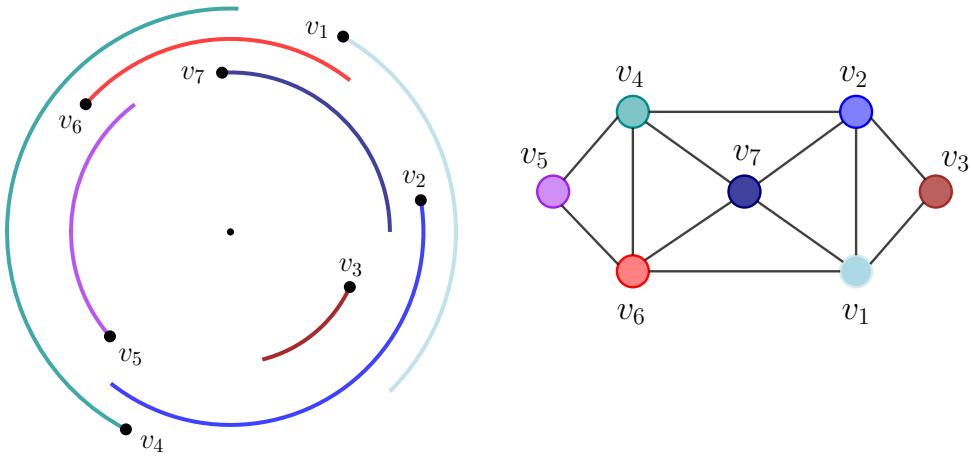

Figure 2.6a: A circular arc representation for a graph, along with its associated drawing.

Proposition 2.19 ([Tuc70]) *Let $M^*(G)$ be the augmented adjacency matrix of G. We have that G is a circular arc graph if and only if there exists an ordering for the nodes such that $M^*(G)$ has quasi-circular $1's$.*

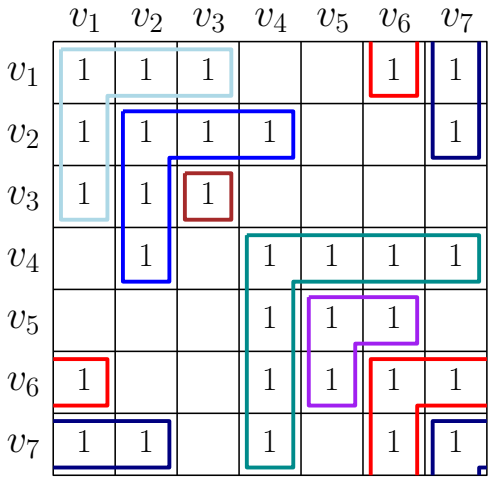

Figure 2.6b: Augmented adjacency matrix for the previous graph with the quasi-circular $1's$ property.

From here, we can describe a three round protocol with cost $\mathcal{O}(\log n)$.

Proposition 2.20 CIRCULAR-ARC $\in$ dM$[\log n]$.

Proof. First, the prover sends to each node v:

1. Its position in the ordering π_v as well as the total number of nodes $n(G)$.
2. A spanning tree given by the triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$.
3. The size of its set U_{π_v} denoted by L_v.

After the nodes exchange their certificates, they check the consistency of the spanning tree and use it in order to verify that the total number of nodes is correct. Then, in order to verify the consistency of $\pi(\cdot)$ as a correct ordering, the nodes proceed as follows.

If we set $N^\pi(v)$ to be the set of nodes in $N(v)$ such that they are positioned between π_v and $\pi_v + L_v - 1$, for $i \in \{0, \dots n-2\}$ each node v in position $\pi_v = i$ must check that it has a unique neighbor u positioned at $\pi_u = j$ for all positions j in $\{\pi_v + h_v\}$, where $w \in N^\pi(v)$ with $\pi_w = h_v$ is the first node such that $\pi_w + L_w - 1 > \pi_v + L_v - 1$. By this process, each node starting from $i = 0$ makes sure that there are nodes labeled with a position in U_{π_v} and that there is a unique node that can continue this process after him. As G is connected, we can assume that this process continues on until all nodes with positions in $\{0, n-1\}$ are verified. As there are n nodes in the graph, all positions are distinct.

Finally, each node v with a neighbor u checks that, either $u \in N^\pi(v)$ or $v \in N^\pi(u)$ and that v is adjacent to all nodes with positions in $N^\pi(v)$. Rejecting if any of these conditions are not satisfied.

- **Completeness.** We have that, if G is a circular arc graph, then it admits an ordering with the previous property. Then, each node v has neighbors whose positions are between π_v and $\pi_v + L_v$ circularly and any other neighbor is such that v verifies that property for them. Therefore, all nodes always accept.

- **Soundness.** If G is not a circular arc graph, then we'll have that, for any ordering, there exists a pair of adjacent nodes u, v such that, as a pair, do not belong to any U_i or V_i. Thus, we have that either u rejects as $\pi_u \leq \pi_v + L_v - 1 \mod n$ or v rejects as one of them notices that fact.

□

.

Protocol for CIRCULAR-ARC

1. $\mathcal{M} \to v$: The prover sends to each v its position π_v, the number of nodes $n = n(G)$, a spanning tree triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ and L_v.
2. Local: The nodes check that the spanning tree is consistent, n is correct, all values for $\pi(\cdot)$ are distinct, and both v and each of its neighbors u satisfy that either $v \in N^\pi(u)$ or $u \in N^\pi(v)$.

These results directly improve in the only previous result in this setting, which is derived from the NPY compiler. As (proper) circular arc graphs can be recognized in linear time in the centralized setting [McC03], this induces a protocol using three rounds and $\mathcal{O}(\frac{n+m}{n} \log n)$ bits, where m is the number of edges in the graph. As each of these classes can contain arbitrarily dense graphs, this can be at most $\mathcal{O}(n \log n)$. Both our protocols for proper circular arc graphs and circular arc graphs improve on this result, as we obtained a one round protocol for both with cost $\mathcal{O}(\log n)$.

Chapter 3

Protocols Using Multiple Rounds of Interaction

In this chapter, we study a different set of problems for which we obtained efficient protocols by use of multiple rounds of interaction. First we study the problem of detecting the presence (or absence) of some graph structure. These are interesting problems from an algorithmic perspective as different graph problems are determined by the absence of some structure [BS+99, RS04] or allow for the efficient computation of different graph parameters [HRSS14, PS15]. We study the problems H-SUBGRAPH and H-MINOR obtaining dMAM protocols for each of them. On the problem of H-FREENESS, first we obtain protocols for the particular cases when a graph is free of a P_3 or a P_4 as an induced subgraph, which correspond to the problems CLIQUE and COGRAPH obtaining dAM protocols for both, deriving a set of tools for a $\mathsf{dMAM}^{\text{pub}}$ protocol for the problem DIST-HEREDITARY as an extension of the latter. Then, for any fixed k we show a $2k$ round protocol for the problem Δ-FREE with a sublinear cost decreasing in k.

3.1 H-subgraph and H-minor

In this section we focus on the problem of detecting the presence of some graph structure, as it is a common question from a distributed perspective. In this regard, we are interested in two different questions: Whether some fixed graph is present as a subgraph, or whether it appears as a minor.

We start with the H-SUBGRAPH problem. Here, the nodes want to check, for some fixed graph H, whether they have an isomorphic copy of H as an induced subgraph.

H-subgraph

For this, the most direct approach would be to mark each node v in the isomorphic copy of H by sending to v a new, extra identifier $\mathsf{id}_H(v)$, and to count (through a spanning tree) the number of nodes which were assigned such a label. Then, the nodes simply need

to exchange these identifiers and see that their adjacencies matches the ones of $\mathsf{id}_H(v)$ in the graph H. At each of H's connected components, starting from a selected node, we can verify that all nodes in H's copy are identified and therefore all nodes accept. This shows that we can solve H-SUBGRAPH in a single round and cost $\mathcal{O}(\log n + \log |H|)$, where the first additive factor comes from the spanning tree construction and the second one from send to each v its identifier in H.

Now, we can improve this protocol by making the verification independent from the size of the graph. We can have the nodes run a SPANNING TREE protocol as described in Protocol 1.21, along with a PERMUTATION protocol, in order to check that all identifiers are distinct and accepting if their adjacencies are consistent.

As the PERMUTATION protocol requires two rounds, a spanning tree and cost $\mathcal{O}(\log k)$, to be implemented, where k is the maximum value of a label, we have that H-SUBGRAPH can be solved in three rounds and cost $\mathcal{O}(\log |H|)$.

Proposition 3.1 H-SUBGRAPH $\in \mathsf{dMAM}(\log |H|)$.

Protocol for H-SUBGRAPH

1. $\mathcal{M} \to v$: The prover sends to each node in the copy of H its identifier given by $\mathsf{id}_H(v)$.
2. $\mathcal{M} \leftrightarrow v$ (messages 2-3) The nodes run a SPANNING TREE and a PERMUTATION protocol in order to check that the identifiers are unique.
3. Local: All nodes exchange id's and check that its adjacency according to H is consistent, rejecting otherwise.

H-Minor

Now we go over the recognition of graph structures as a minor. In the problem H-MINOR we would like to determine if for some fixed graph H, G contains H as a minor, that is, there exists a sequence of edge contractions and either node or edge deletions on G such that the resulting graph after all these operations is H.

The detection of minor graph structures is specially relevant given a result by Robertson & Seymour [RS04] stating that any graph property closed under taking minors can be characterized by a *finite* family of forbidden minor structures. As proving that a graph is H-minor free can be much harder, as a first approach we go over detecting a minor structure.

Since the previous definition might not be useful in the distributed setting, we use the following characterization: We say that H is a minor of G with $|H| = k$ if there exists a function $\mu : V \to [k]$ that partitions the set of nodes into disjoint sets $V = \bigcup_{i=1}^{k} V_i$ such that, for each i, the graph $G(V_i)$ is connected and for any edge $e = uu' \in E(H)$ there exists a pair of nodes adjacent nodes $vw \in E(G)$ such that $\mu(v) = u, \mu(w) = u'$. We call such a function a *model* of H in G.

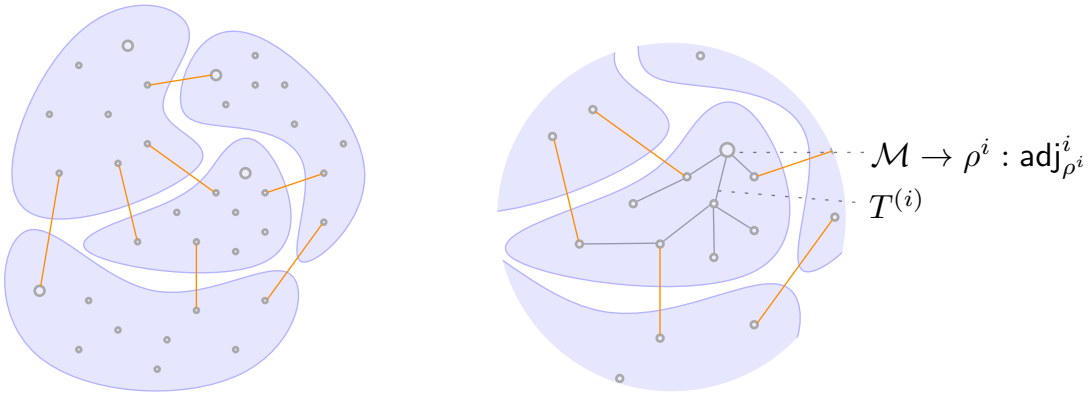

Figure 3.1: An example of the proof given by Merlin in the protocol for H-MINOR. The blue areas show each connected subgraph representing a node in the model for a K_4. The large gray nodes are the roots ρ^i at each component, to which Merlin sends a proof adj^i of the adjacency of the node they represent.

With this characterization in mind we proceed to describe a dM protocol using $\mathcal{O}(|H| + \log n)$ bits, where the $\mathcal{O}(\log n)$ term arises from the construction of a spanning tree. Therefore, by using Protocol 1.21 we obtain a three round private coin protocol using $\mathcal{O}(|H|)$ bits.

Proposition 3.2 *Let H be a fixed graph. Then,* H-MINOR $\in \mathsf{dAM}^{\text{priv}}[3, |H|]$.

Proof. The message of the prover is divided into three parts:

1. An identifier $\mathsf{id}_H(v) \in [k]$ denoting which of H's nodes is represented by v, where $k = |V(H)|$.

2. A pair of triples $\langle \rho^0, t_v^0, d_v^0 \rangle, \langle \rho^i, t_v^i, d_v^i \rangle$ containing information about the root, parent and distance to the root in the spanning trees T of G and $T^{(i)}$ of $G(V_i)$, when $\mathsf{id}_H(v) = i$.

3. The vectors $\mathsf{adj_v}^i$ and $\mathsf{label_v}$ in $\{0,1\}^k$. With the former allowing the nodes to check that the adjacency of $i \in H$ is consistent with that of the neighbors of the i-labeled nodes and the latter being used to check that no labels in $[k]$ are repeated in different components (verification that will be run by the root of each component).

Finally, each node locally checks the correctness of the main spanning tree, as well as the trees of each component. For this, consider a node v in the main tree T, it collects the sets $\mathsf{acc}_u = \{j \in [k] : \mathsf{label}_u[j] = 1\}$ for all u that are children of v in T and checks that they are pairwise disjoint. Now, if v has been assigned an index $i \in [k]$ such that $v = \rho^i$, it then checks that, in position i, the vector label_v has value 1 and the vector label_u has value 0 for all u which are children of v in T.

From here we know that for each set the partition of V, there is a unique node labeled

as the root ρ^i for some i where they all have different labels. In order to check the vectors adj_v^i, consider a tree $T^{(i)}$ and its root ρ^i. By the correctness of the vector label_{ρ^0}, we have that this tree is unique as there is a unique root. Now, it suffices to run another sum protocol across the tree $T^{(i)}$ as follows: for any node v such that $\mathsf{id}_H(v) = i$, v first checks that if there exists a node u among its children in the tree $T^{(i)}$ such that $\mathsf{adj}_u^i[j] = 1$, then its entry for j must also be 1. Then, v considers the set of entries that have value 0 for all its children. For any entry k in this set, it checks that $\mathsf{adj}_v^i[k]$ has value 1 if and only if there exists a node w in v's neighborhood such that $\mathsf{id}_H(w) = k$. If this is correct, it then accepts and, by a sum protocol, we have that $\mathsf{adj}_{\rho^i}^i$ has the correct value among all nodes labeled i, therefore a node accepts if and only if the entries in the vector match the adjacency vector for i in H.

□

Protocol for H-MINOR

1. $\mathcal{M} \to v$: The prover sends to each v its label $i = \mathsf{id}_H(v)$, a spanning tree triple for its label component $\langle \rho^i, t_v^i d_v^i \rangle$ and the vector adj_v^i which lets it check the adjacency of the label i up to the subtree $T^{(i)}$.
2. $\mathcal{M} \to v$: The prover also sends to each v a spanning tree triple $\langle \rho^0, t_v^0, d_v^0 \rangle$ for G and a vector label_v which keeps track of the number of component roots assigned up to T_v for each label.
3. Local: Nodes check the spanning trees correctness, the roots ρ^i check that the vector label is correct, and at each spanning tree, all nodes check that their adj_ρ^i is correct.

3.1.1 The Problem of H-freeness

In the H-FREE problem, our objective is to certify that a graph G has no copy of H as an induced subgraph. This problem is relevant in both the centralized and distributed setting because of the fact that many graph properties are related to the absence of a certain local structure. It has been shown, by reductions to 2-party communication complexity, that it can be "almost maximally hard", in the sense that for any integer k there exists a graph H with k nodes such that, to solve H-freeness, even with the help of non determinism or randomization, it would require $\Omega(n^{2-\frac{1}{k}})$ rounds of interaction in the Congest model [FGKO18], which naturally gives us the exact same lower bound for a proof size in the dM model.

However, several results have appeared in the distributed setting for specific cases such as triangles [ILG17], large cycles [FGKO18], trees [EFF+17] and many others both in the Congest model and its fully connected sibling, the Congested Clique model.

Depending on the graph H to be selected, several structural results arise that may be used to our advantage. We start by studying the class of graphs that have no P_3 as an induced subgraph, which turns out to be a disjoint union of cliques, sometimes called *cluster graphs*. As we start by assuming that our configuration network is connected, this

class reduces to the CLIQUE class.

3.2 Clique

As previously described, for the problem CLIQUE we have an n-node graph G and we wish to check if all nodes are connected with each other. If we are allowed to obtain a leader (and therefore a root for a tree), we can solve this problem directly by allowing all nodes to check that their degree matches the size of the graph and comparing such a value to their neighbors.

Proposition 3.3 CLIQUE $\in \mathsf{dM}[\log n]$ *and* $\mathsf{dAM}^{\text{priv}}[1]$.

A trivial algorithm is a dM protocol with cost $\mathcal{O}(\log n)$ in order to compute $n(G)$ and compare it with the degree of each node. Indeed, the prover sends a spanning tree triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$ to each node, along with the total size of the graph $n(G)$ which the nodes verify by means of a sum check protocol: A node v computes the size of the graph up to a subtree rooted at v, by comparing this number to the one in its children's subtree. From here, they can easily check that the spanning tree is correct, as well as the sum at each level. Then, they accept iff the size of each node's neighborhood is $n(G)$

Protocol for CLIQUE

1. $\mathcal{M} \to v$: The prover sends to each node v a spanning tree triple $\langle \mathsf{id}(\rho), d_v, t_v \rangle$, the size of the graph $n(G)$ and a partial computation of this value up to the subtree T_v.
2. Local: The nodes check that they all have the root of the tree as a neighbor, that they have $n(G)$ neighbors and that the computation of $n(G)$ is correct.

Now, it is easy to show that, using randomness, the cost can be greatly reduced. Indeed, consider the following protocol: each node draws a single bit b_v and sends it to the prover. Then, Merlin answers to each node v with the sum of all bits drawn in its closed neighborhood, that is, $s_v = \sum_{u \in N[v]} b_u$. At the verification round, each node broadcasts to its neighbors its value for s_v. Finally, all nodes accept depending on whether they all received the same value for s_v.

- **Completeness.** If the graph is indeed a clique then naturally all nodes will have the same value for s_v as $s_v = \sum_{v \in G} b_v$ and therefore all nodes will accept.
- **Soundness.** If the graph is not a clique then there exists a pair of adjacent nodes u, v such that $N(u) \neq N(v)$. Then, the probability that the values for s_u and s_v match is exactly $1/2$. Therefore, if we run several repetitions in parallel of this process we can make this value arbitrarily small.

From here we have that from a dM protocol with cost $\mathcal{O}(\log n)$, we can easily go down to a constant cost by an addition of a single round of interaction, which is merely a restriction

of our model as the protocol can be set to work in the (broadcast) Congest model with randomization. Yet we think that CLIQUE is an interesting problem to consider as it is a natural question for the nodes to check that they are all connected, and which we would expect to have a lower bound of $\Omega(\log n)$ for a single round of interaction. However, we have not been able to obtain such a bound, as all lower bound results we will study later in Section 4.3 rely on the fact that the graph admits a small cut. It is therefore an open question to obtain such a bound or make use of the high connectivity of the network to our advantage for achieving a smaller message size.

3.3 Cograph

Following our interest in the detection of forbidden structures we study the COGRAPH class. In this problem we are given an n-node graph G and we wish to check that it has no P_4 as an induced subgraph. Cographs often appear in the literature related to algorithmic graph theory as many different (hard to compute) graph parameters (such as the maximum clique size, chromatic number, treewidth, etc) can be computed efficiently [BS+99] when the input is restricted to cographs. This class has been recently studied in the Congested Clique model, where a public-coin broadcast algorithm was obtained [KMRS15], as well as a private-coin reconstruction result [MPSRT18] in the same model.

An advantage of this graph class is that it admits other characterizations that may be useful for local verification. Another equivalent definition states that cographs are the graphs which can be obtained recursively following three rules: **(1)** A single node is a cograph, **(2)** the disjoint union between two cographs is a cograph and **(3)** the join of two cographs is a cograph. This result, while not easy to verify by itself, proves useful when combining its structural properties with other characterization. We define a *twin ordering* as an ordering of the nodes $V = \{v_i\}_{i=1}^{n}$ such that, for each $j \geq 2$, v_j has a twin in $G(v_1, ..v_j)$.

Proposition 3.4 (by [KMRS15]) *Given a graph G the following are equivalent:*

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

In order to describe a protocol, we first show a way to distribute the proofs received by the network in such a way that we can centralize the verification process, by considering properties of cographs related to their connectivity.

Remark If G is a connected cograph, then it can be written as the join between two cographs G_1, G_2.

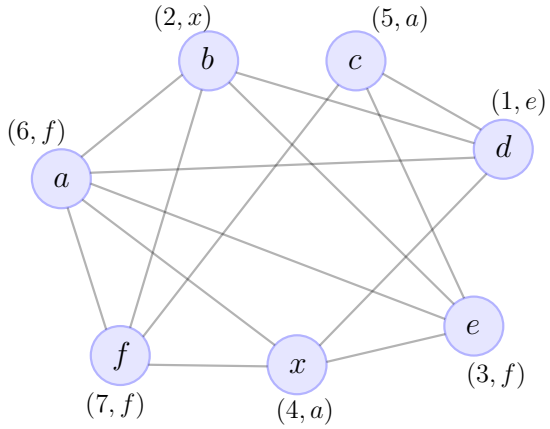

Figure 3.2: A cograph with labels according to a twin ordering. The first entry represents the step at which they are removed, while the second entry indicates the node's twin at such step

Lemma 3.5 *Given a connected n-node cograph G, it is possible to construct a spanning tree T of* depth *two, such that each node at the first level has at most one child.*

Proof. As G is connected, as per the previous Remark, it follows that G can be obtained from the join of two smaller cographs G_1 and G_2. Then, let G_1 be the one with at least $\frac{n}{2}$ nodes. Consider now $\rho \in G_2$, the root of the tree T to be constructed. It follows that ρ has all of G_1 as neighbors. Then, for each node in G_1, we set ρ to be its parent in T.

Finally, we have that, $\hat{G} = G(G_1, G_2 - \rho)$, this is, the bipartite graph obtained by considering only the edges between G_1 and G_2, is a complete bipartite graph, with $n(G_2 - \rho) < \frac{n}{2}$. Therefore, by Hall's theorem it follows that there exists a matching M between both sides of $\hat{G}$ such that all nodes in $G_2 - \rho$ have a match. Thus, for $u \in G_2 - \rho$, we set its parent in T to be its match $m(u)$ in G_1. The lemma follows. □

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

Lemma 3.6 *On any connected cograph, there exists a node ρ which can be assumed to receive all certificates in a* dAM *protocol with cost L by including an additive cost of $\mathcal{O}(\log n)$.*

An advantage of this procedure is that we may simulate any protocol in the Broadcast Congested Clique model (by using the root ρ as referee) by either using one round (if it is deterministic) or two rounds (either with public or private coins). From here it follows that we can use the public coin protocol by [KMRS15] to recognize cographs, therefore constructing a protocol for cograph detection in two rounds of interaction and $\mathcal{O}(\log n)$ bits. That is, COGRAPH $\in \mathsf{dAM}^{\text{pub}}[\log n]$.

We now describe the protocol for the sake of completeness.

Definition 3.7 *Given a cograph $G = (V, E)$, we can define its* canonical order *as follows:*

We start by choosing the smallest pair of twins*, that is, those with the smallest identifiers in lexicographic order (which we know to exist by Proposition 3.4, second property). Next, we choose and remove the smallest node from this pairing. Then, we repeat this process by finding another pair and removing one of its members until we end up with a single node.*

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

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

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

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

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

Definition 3.8 *We say that a vector $m = ((a_w, b_w))_{w\in V} \in (\mathbb{Z}_p)^{2n}$ is* valid for G in $t \in \mathbb{F}_p$ *if there exists a family of linearly independent polynomials $(\phi_w)_{w\in V}$ in $\mathbb{F}_p[X]$ such that $a_w = \phi_w(t)$ and $b_w = q_w(t)$ for each $w \in V$.*

Lemma 3.9 *Let $m = ((a_w, b_w))_{w\in V} \in (\mathbb{Z}_p)^{2n}$ be a valid vector for G in t. Consider u, v to be a pair of twins in G such that $a_u \neq a_v$. Then, the vector $m' = ((a'_w, b'_w))_{w\in V-v} \in (\mathbb{Z}_p)^{2n-2}$ is valid for $G - v$ in t, where its coordinates are given by*

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

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

Proof. Let $(\phi_w)_{w\in V}$ be a family of linearly independent polynomials associated to m. Given that u are v are twins and $a_u \neq a_v$ it follows that $a_u + b_u = a_v + b_v$ if and only if u are v adjacent. Therefore, $\delta_{u,v} = 1$ iff u are v are adjacent.

Let now $(\phi'_w)_{w\in V-v}$ where $\phi'_w = \phi_w$ for all $w \neq u$, and $\phi'_u = \phi_u + \phi_v$. It is clear that this family is linearly independent.

For $w \neq u$ we have that $a'_w = a_w = \phi_w(t) = \phi'_w(t)$. Also, $b'_w = b_w$ and $b_w = q_w(t)$. Now, as u v are twins either both nodes are in $N(w)$ or neither u nor v are. In both cases it follows that $b'_w = q'_w(t)$

By definition we have that $a'_u = a_u + a_v = \phi_u(t) + \phi_v(t) = \phi'_u(t)$ and $b'_u = b_u - \delta_{uv}a_v$. As $b_u = q_w(t) = \delta_{uv}\phi_v(t) + q'_w(t)$, we finally have that $b'_u = q'_w(t)$.

□

With this lemma now me can proceed to describe the protocol

Theorem 3.10 *For any $c > 0$ we have that* COGRAPH $\in \mathsf{dAM}_{1/n^c}[\log n]$

Proof. Let $G = (V, E)$ be an n-node graph. Without loss of generality we may assume the graph has identifiers in $[n]$ as, following Lemma 3.5, it is possible to implement a PERMUTATION protocol in a single round: Merlin sends to each node v an identifier $\bar{\mathsf{id}} : V \to [n]$ and the root, by receiving all proofs according to Lemma 3.6, can see that they all received distinct identifiers which are consistent with their original ones.

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

After the nodes exchange messages, once again following Lemma 3.5 the root ρ will own a vector $m \in \mathbb{F}_p^{2n}$. From here, the root repeats the following procedure at most $n-1$ times trying to construct a canonical ordering $\{v_i\}_{i=1}^n$ for G.

At step i, it starts at graph G^i and a vector $m_i \in \mathbb{F}_p^{2(n-i+1)}$ (where $G^1 = G$ and $m^1 = m$) and looks for a pair of nodes u, v in G^i such that $a^i_u \neq a^i_v$ and either $b^i_u = b^i_v$ or $a^i_u + b^i_u = a^i_v + b^i_b$.

Then it chooses, among all pairs it has found, the first of these in lexicographic order. If no such pair exists, then he rejects. On the contrary, he defines $G^{i+1} = G^i - v$, and setting $v_{n-i+1} = v$ (without loss of generality we assume that $\mathsf{id}(v) < \mathsf{id}(u)$). Then the root computes m^{i+1} from the previous vector m^i following Lemma 3.9. If the root reaches step $n-1$ then he accepts.

- **Completeness.** & **Soundness.** It follows then that as the messages depend on the

original identifiers and the root ρ has access to all messages, then both acceptance errors depend solely in the BCC construction. Now, by Lemma 3.9 it follows that the only point at which the protocol might fail is if the chosen t turns out to be a root for any of the polynomials in the canonical family from Definition 3.7. Then, as there are at most $3n^3$ such functions with degree at most n , we have that the acceptance error is at most $3n^4/3n^{c+4} = 1/n^c$ and the theorem follows.

□

This protocol improves on the result on the NPY compiler: Given that cographs can be recognized in linear time [DHP01], its use would provide a protocol using private randomness and three rounds of interaction. As the class of cographs contains graphs with a large set of edges, the protocol obtained by the compiler can have a cost up to $\mathcal{O}(n \log n)$, which is a large gap in comparison to this protocol.

Given the correctness of this proof, and the fact that it uses two rounds in order to simulate the randomness in the BCC model, the next question would be if there exists either a dM protocol, or a one round deterministic protocol in the BCC model, which would imply a protocol for our model following Lemma 3.5. Indeed, we can not provide a sub linear one round protocol for this problem (the proof that COGRAPH $\in$ dM$[n]$ is trivial) and currently there are no known deterministic BCC protocols for the COGRAPH class. Yet, we take note of the following corollary which may be of interest, in case that there exists a positive result:

Corollary 3.11 *Any (non deterministic) protocol for* COGRAPH *in the* BCC *model with cost L would imply a* dM *protocol with bandwidth $L + \mathcal{O}(\log n)$.*

This adds a new perspective either for the search of protocols (we can simulate any non deterministic protocol in the BCC model following Lemma 3.5) or the search for negative results in our model in a future work, implying lower bounds in the BCC model.

A dMAM$^{\text{pub}}$ protocol for COGRAPH

Now, we proceed to describe a way to change the verification routine for our protocol so that it can be done in a distributed manner. This will be done using three rounds of interaction so that this new protocol can be adapted for the class DIST-HEREDITARY which will appear in the following section.

Notice that for the protocol in Theorem 3.10, the verification process is done by the root as it prunes the graph for $n-1$ steps. This leads to an order by which the nodes were selected which we call the *canonical* ordering. For this ordering and a node v, consider the *predecessor* of v, denoted by $\mathsf{ant}(v)$, to be v's neighbor whose value for $\pi(\cdot)$ is immediately after that of v among its neighbors. With this, for distributing the verification process, Merlin provides each node v with an ordering $\pi(v)$ (which the nodes will verify in parallel through rounds 2 and 3 by means of the PERMUTATION protocol), along with:

1. The id of v's twin at step π_v of the computation, denoted by $\mathsf{twin}(v)$.

2. The id of $\mathsf{ant}(v)$ according to π.

3. The id of the predecessor of $\mathsf{twin}(v)$ according to π. That is, $\mathsf{ant}(\mathsf{twin}(v))$.

4. The number of v's neighbors that claim to have v as their twin according to the order π, which we denote by $\#_v$

The nodes (collectively), set a finite field $\mathbb{F}_p$, with $p = \text{poly}(n)$ sufficiently large, and send a seed $t \in \mathbb{F}_p$. Then, Merlin proceeds to answer with the two coordinates (a_v, b_v) belonging to v in the original valid vector m described in Theorem 3.10, as well as the two coordinates belonging to v in the vector that the root would have obtained at step π_v of the computation, which we denote by (a_v^π, b_v^π).

Finally, at the verification round, all nodes start by computing their canonical family of functions, and therefore their values for a_v and b_v, rejecting if these do not match.

Fix a node v. The node $\mathsf{ant}(v)$ has the role of checking that, first, there are exactly $\#_v$ nodes u such that $\mathsf{twin}(u) = v$ at some step of the computation which know $\mathsf{ant}(v)$ to be v's predecessor (otherwise they would reject, as $\mathsf{ant}(v)$ is unique) and, second, the valid vectors (a_u^π, b_u^π) match v's vector at the step u is pruned off, after which v's vector is updated according to Lemma 3.9. This goes on until until arriving at vector a_v^π, b_v^π.

More precisely, the node $\mathsf{ant}(v)$ sorts each of the nodes who have v as a twin according to their value for $\pi(\cdot)$. Let this ordering be $w_1, \dots w_{\#_v}(v)$. Now suppose we are at step i of the computation and let (a_v^i, b_v^i) be the pair of $v's$ coordinates at step i with $(a_v^1, b_v^1) = (a_v, b_v)$. Then, $\mathsf{ant}(v)$ compares the value for $a_v^i + b_v^i$ with $a_{w_i}^\pi + b_{w_i}^\pi$ (if w_i is assigned as a *true* twin) and rejects if they differ. Otherwise, it sets $(a_v^{i+1}, b_v^{i+1}) = (a_v^i + a_{w_i}^\pi,\ b_v^i - a_{w_i}^\pi \delta_{vw_i})$ (where $\delta_{v,w_i} = 1$ if and only if v and w_i are adjacent) and proceeds to step $i + 1$.

Finally, at step $i = \#_v$ it checks that $(a_v^{i+1}, b_v^{i+1}) = (a_v^\pi, b_v^\pi)$. and accepts if these values are equal and reject otherwise. Now, we prove the correctness of this protocol.

- **Completeness.** & **Soundness.** Notice that, under the correctness of the PERMUTATION protocol, we can assume with high probability that $\pi(\cdot)$ values are all distinct as otherwise some node would reject. Now, fix a node v, and consider its predecessor $\mathsf{ant}(v)$, according to the ordering π. Any node that claims to have v as its (*true* or *false*) twin must have $\mathsf{ant}(v)$ as a neighbor. Therefore, he is aware of all nodes that have to be present in the computation of (a_v^π, b_v^π) as otherwise he would reject. Then, as for the computation of the twin ordering by the root in the original protocol, we have that for each node v the values (a_v, b_v) are only updated when it is either deleted from the graph, or when it is twin of a node in some higher layer. Therefore, the computation for each of these values can be done in a distributed manner, as the only information we need are the proofs received by the nodes that have v as their twin. As $\mathsf{ant}(v)$ receives all these messages, it can check the correctness of v's vector. And so we have, by an inductive proof, that for each node v, the value of its coordinates at point π_v are correctly computed with the same

probability as that in the original protocol. Then, the graph either fails to find such an ordering or accepts a false ordering with probability at most $1/n^c$.

3.4 Distance Hereditary

As a consequence of the protocol described for cographs, it is possible to derive an interactive protocol for another graph class which admits a similar construction. A graph G is said to be DIST-HEREDITARY if for any induced subgraph $H \subseteq G$ and any pair $u, v \in H$ we have that $d_H(u, v) = d_G(u, v)$. That is, any induced path between a pair of nodes is a shortest path. This class of graphs has not been previously studied in the distributed setting, yet it is an interesting problem to consider as it appears frequently in Port Routing problems [CDSF01] and may be used for other routing schemes.

A relevant (more related to what has been discussed) characterization for this class is the following.

Proposition 3.12 ([BS+99]) *An n-node graph G is said to be distance hereditary iff there exists an ordering $\{v_i\}_{i=1}^n$ such that, for any $i \in [n]$, either there exists $j < i$ such that v_i and v_j are* twins *in $G_i = G(v_1, ..v_i)$ or v_i is a pending node at G_i.*

That is, any distance-hereditary graph can be constructed by sequentially adding twins or pending nodes.

Using this result, we can construct a $\mathsf{dMAM}^{\text{pub}}$ protocol for this class by adapting the protocol for the COGRAPH class from Theorem 3.10 with some small adjustments. While we can not delegate the verification routine to a single node (as distance-hereditary graphs can have arbitrarily large diameter), we can distribute the verification process by letting different nodes check different steps of the computation. As the rule described in Lemma 3.9 for pruning the graph involves only the pair of twins at each step, we only need to find nodes that, for a fixed node v, can receive all the proofs sent by v, its twins and its pending nodes.

First, in order to prune the graph in this new setting, we need a rule for pruning pending nodes from a graph and updating the vectors of each node accordingly. Once again, we use the definition of a *valid* vector as described in Section 3.3.

Lemma 3.13 *Let $m = ((a_w, b_w))_{w \in V} \in (\mathbb{Z}_p)^{2n}$ be a valid vector for G at some point t. If $u \in G$ has v as a leaf adjacent to it, then, the vector $m' = ((a'_w, b'_w))_{w \in V-v} \in (\mathbb{Z}_p)^{2n-2}$ is valid for $G - v$ in t, where the coordinates of m' are given by*

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

Proof. If $\{\phi_w\}_{w \in V}$ is a family of linearly independent polynomials associated to m, we can use this family and, as v was only connected to u, it follows that $b'_u = b_u - a_v = q_u(t) - \phi_v(t) = \sum_{w \in N(u)-v} \phi_w(t) = q'_u(t)$. □

Combining both rules from Lemmas 3.9 and 3.13, we can establish the following:

Theorem 3.14 DIST-HEREDITARY $\in \mathsf{dMAM}^{\mathrm{pub}}[\log n]$.

Proof. We simply need to check that, given an ordering π (which we can compute by use of the PERMUTATION protocol) we can compute the coordinates for a valid vector at each step of the computation by delegating this information to the correct nodes. Indeed, for each node v, we define $\mathsf{twin}(v)$ to be the node assigned as its twin, $\mathsf{ant}(v)$ to be its predecessor according to the definition of the previous section, $\mathsf{Twins}(v)$ to be the set of nodes u such that $\mathsf{twin}(u) = v$ and $\mathsf{Pending}(v)$ to be the set of nodes that have v as their unique neighbor according to π. First, Merlin sends, to each node v, a set of messages similar to those described in Protocol 3.3, that is

1. Its position in the ordering π given by π_v.
2. The id of the unique neighbor which is immediately after it at π, which we denoted by $\mathsf{ant}(v)$.
3. If v is removed from the graph as a *pending node*
   a. The id of the node to which v is a pending node, denoted by $\mathsf{pending}(v)$.
4. If v is removed from the graph as a *twin*,
   a. The id of v's twin at step π_v of the computation, denoted by $\mathsf{twin}(v)$.
   b. The id of $\mathsf{ant}(\mathsf{twin}(v))$, i.e. the id of the predecessor of the twin of v according to π.
5. The number of neighbors u of v such that $\mathsf{twin}(u) = v$ according to π, denoted by $|\mathsf{Twins}(v)|$.

This set of certificates, while similar to those in the previous section, are not sufficient. By receiving these certificates, $\mathsf{ant}(v)$ can collect all proofs received by each node u with $\mathsf{twin}(u) = v$ yet we cannot follow the same decomposition in order to check that the pair (a^π_v, b^π_v) is correct. Indeed, the nodes in $\mathsf{Pending}(v)$ are not adjacent to $\mathsf{ant}(v)$, so $\mathsf{ant}(v)$ can not see these proofs.

To fix this, Merlin will distribute a set of proofs between v and those in $\mathsf{Twins}(v)$. First, Merlin sends to v:

6. The number of nodes u such that $\mathsf{pending}(u) = v$, that is, $|\mathsf{Pending}(v)|$.
7. The id of the node $u \in \mathsf{Pending}(v)$ with the smallest value for $\pi(\cdot)$, denoted by $\mathsf{m\text{-}leaf}(v)$, as well as π_u.

Now, fix some node u such that $\mathsf{twin}(u) = v$. Then, Merlin sends to u

8. The id of the node in $\mathsf{Twins}(v)$ whose value for π is immediately after $\pi(u)$, denoted

by co-twin(u).

9. The number of nodes in Pending(v) with values for π between those for u and co-twin(u).

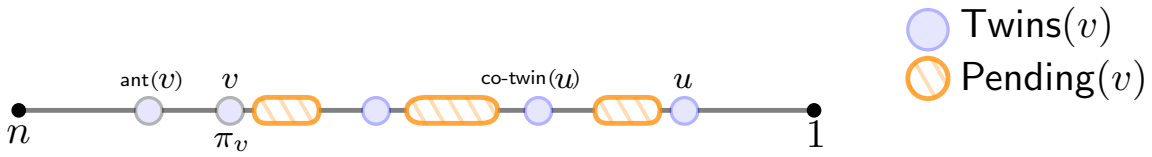

Figure 3.3: Visualization of ant(v) and the sets Pending(v) and (v) for some node v: the neighbor ant(v) sees all nodes in (v), which are the nodes u that are assigned as twins of v. If we order these according to π, there may have nodes in Pending(v) positioned between them.

After this first interaction, all nodes (collectively) send a seed $t \in \mathbb{F}_p$, with $p = \text{poly}(n)$ a prime number to be defined accordingly. From here, Merlin answers to each node v with the following set of messages. First, he sends (in the same way as in the previous procedure) the pair of coordinates (a_v, b_v) which belong to v at the start of the computation, as well as the pair he would have sent at step π_v of the computation, denoted by (a_v^π, b_v^π). Then, he sends the value $P(v,t) = \sum_{u \in \text{Pending}(v)} \phi_u(t)$ which is an encoding for the set of v's pending nodes according to t. Finally, if v is assigned as a twin, with twin(v) $= u$, Merlin sends the message $\sum_{w \in S(v)} a_w$. Where we define the set $S(v)$ as

$$S(v) = \{w \in \text{Pending}(u) : \pi_{\text{co-twin}}(v) > \pi_w > \pi_v\},$$

that is, the set of pending nodes connected to v's twin such that they appear between twin(v) and co-twin(v).

Finally, at the verification round, the nodes exchange their certificates and they try to collectively compute the correct values for (a_v^π, b_v^π) as follows: First, each node checks that its original values for the pair (a_v, b_v) are correct, along with the size of its set of pending nodes Pending(v) and ant(v). Now, if v is assigned as a leaf, it simply checks that it is adjacent to pending(v) and that it is the only neighbor with a larger position at π. If v is assigned as a twin instead, with $u =$ twin(v), it delegates this verification to the node ant(u) in a similar manner as that of Protocol 3.3, with the following modifications.

First, the node ant(u) compares the number of nodes adjacent to it which are assigned as twins of u with the size of the set Twins(u) sent by Merlin. Then, it sorts the nodes in Twins(u) in increasing order according to π , and name them $w_1, \dots w_{|\text{Twins}(v)|}$. For each i, it checks that co-twin(w_i) $= w_{i+1}$. The only issue we are left to determine is the values for $P(i,t) \coloneqq P(w_i, t)$. That is, the encoding for all pending nodes that lie between w_i and w_{i+1} for $i = 1, \dots$ Twins(u), where we set $w_{\text{Twins}(u)+1} = u$. We can assume that we know the encoding for any pending node that occurs before w_1 as they can be checked by u and sent to ant(u) during the verification round. This is because Merlin can send to node u the position of the first node which has u as a twin.

To obtain $P(i,t)$, we simply need to show that ant(u) is able to partition the set Pending(u) (and its encoding) into groups according to their positions in the permutation

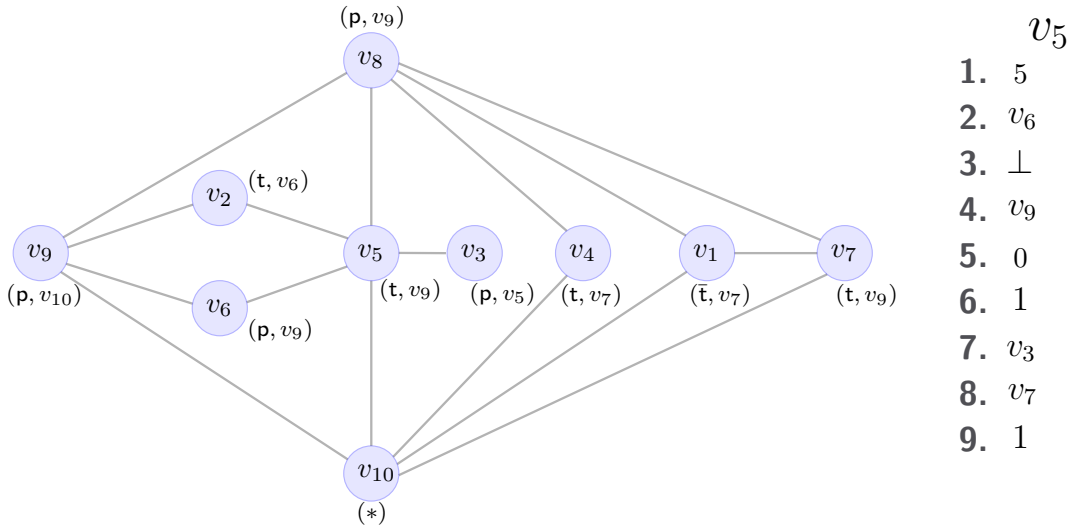

Figure 3.4: An example of a first round of interaction for a fixed node v_5. Given a distance-hereditary graph, each node is identified according to their ordering π and labeled depending on the way they are removed from the graph. The first entry shows whether some node is a *false twin* (t), a *true twin* ($\bar{\mathsf{t}}$) or a *pending node* (p), with the second entry showing its assigned node on each of these cases. A list of all certificates sent by Merlin to v_5 according to the previous list is included.

in-between nodes assigned as twins. Indeed, as $\mathsf{ant}(u)$ knows the size of $\mathsf{Pending}(u)$, it simply needs to sum the number of pending nodes that appear between w_i and w_{i+1} according to w_i from message **(9)**. We know that these value are correct. Each w_i can count them (it knows that $w_{i+1} = \mathsf{co\text{-}twin}(v)$). Thus, $\mathsf{ant}(u)$ computes the size of the parts of $\mathsf{Pending}(u)$ according to the collection $\{w_i\}_i$. If these values do not match the size of $\mathsf{Pending}(u)$, it simply rejects.

Finally, as $\mathsf{ant}(u)$ knows that the partition is correct, it simply considers the values for $P(i,t)$ provided by w_i and computes the values for (a_v^i, b_v^i) at each step of the computation: At step i, it obtains a pair $(\bar{a}_v^i, \bar{b}_v^i)$, it compares its sum with that of $(a_{w_i}^\pi b_{w_i}^\pi)$ and rejects in case these values are not equal. Then, in order to obtain (a_v^{i+1}, b_v^{i+1}) it follows the rule from Lemma 3.9, deletes $P_i(t)$ from b_v^{i+1} and goes to the next step. At the end of this computation, it remains to check whether the obtained pair equals (a_v^π, b_v^π), and accept or reject accordingly.

- **Completeness.** An honest prover will provide the nodes with the correct ordering. Then, as we have described above, the nodes compute the correct values for each of their coordinates in a valid vector.

- **Soundness.** If G is not distance hereditary, because of the PERMUTATION protocol, we have that Merlin should provide a correct ordering with high probability. Therefore, it remains to check that π satisfies the above properties. Indeed, each v which is assigned as a leaf, trivially computes it has a unique neighbor with a higher value for π that matches $\mathsf{pending}(v)$ and therefore it accepts. For any node v we have that $\mathsf{ant}(v)$ should compute that the pair sent to v is correct at each

step of the computation and that all nodes u such that $\mathsf{twin}(u) = v$ are correct. We have that the values for the set of pending nodes in-between nodes assigned as twins is correctly computed as described above. Moreover, for a valid pair which has not been correctly computed at some step of the computation, it should occur that the seed t is bad for some canonical polynomial. A this family of functions is polinomially bounded beforehand, we have that at least one node should reject with high probability.

□

This improves on the result obtained by direct use of the NPY compiler (Distance-hereditary graphs can be recognized in linear time [DHP01]) which implies a three round protocol with cost $\mathcal{O}(n \log n)$. Observe that cographs are a subclass of the class of Distance-hereditary graphs). On the other hand, we obtained a protocol with cost $\mathcal{O}(\log n)$.

3.5 Detection of Triangle-free Graphs

The recognition of triangle-free graphs plays a central role in graph algorithms in the distributed setting as, for several graph problems, efficient algorithms are known for triangle-free graphs [HRSS14, PS15]. Its verification has been largely studied, and recent sub-linear protocols in the Congest model have been achieved in [ILG17], as well as proofs of the hardness of obtaining lower bounds when nodes are allowed to send different messages trough each channel [EFF+19]. Other results include protocols which use matrix multiplication techniques in the Congested Clique model [CHKK+19] and a $\mathsf{dMA}^{\mathrm{pub}}$ protocol based on a 2-party protocol for the problem DISJOINTNESS [FMO+19].

Despite its highly local nature, obtaining concise protocols in the dM model has proven to be rather difficult. In this section, by assuming the nodes know the identifiers of its neighbors, we show a distributed interactive proof with sub linear cost for this problem by extending the protocol of Crescenzi et al. [CFP19], which is in itself an adaption of a protocol for the DISJ problem in the 2-party MA model, obtained by Aaronson and Wigderson [AW09]. Assuming that the nodes know their neighbors beforehand is not a big obstacle to our model, as we show how to adapt it to our model by simply adding an extra round of interaction.

In order to prove our result, we begin by describing a simpler version of the protocol using four rounds of interaction and $\mathcal{O}(n^{\frac{1}{3}} \log n)$ bits of bandwidth. Then, we proceed to describe the general case.

Four rounds of interaction

We start describing a simplified version that uses four rounds of interaction, where the last round corresponds to a randomized verification process. Every node v creates a polynomial on three variables $\psi_v(x, y, z)$ that encodes its neighborhood, from which it can obtain a function $\Psi_v(x)$ which encodes its participation in a triangle. As this function depends

also on the vicinity of its neighbors, it can not obtain it right away, as they would have to send very large messages. Yet, the nodes can have Merlin provide such functions and then convince them of their correctness with high probability. For this, in the next round the nodes will (collectively) choose some random value r for the first variable of ψ_v, after which Merlin is required to send a partial construction of Ψ_v, setting $x = r$. Finally, in the verification round the nodes choose a second value $\bar{r}$ at random for ψ_v's second variable, by which they can now send the function $\psi_v(r, \bar{r}, z)$. With these values in mind, the nodes can now construct the value for Ψ_v by comparing its value at a random point to the function τ_v that Merlin previously committed to. We show that, in case that Merlin would have lied, then the nodes can detect an error with high probability and therefore can correctly compute Ψ_v.

If there is a node that participates in a triangle, with high probability it would have a point in its polynomial whose value is non-zero and one of the nodes would therefore reject.

Proposition 3.15 *If the nodes know the identifiers of their neighbors, then* Δ-FREE $\in$ $\mathsf{dMA}^{\mathrm{pub}}[4,\ n^{\frac{1}{3}} \log n]$.

Proof. Without loss of generality, we may assume that the nodes have identifiers in $[n]$, as in three rounds we can run a PERMUTATION protocol in parallel where nodes are assigned such identifiers and may verify that they are all distinct. Also, we may assume that $n(G)^{\frac{1}{3}}$ is an integer, as otherwise we take $\bar{n} = (\mathsf{ceil}(n^{\frac{1}{3}}))^3$, where $\mathsf{ceil}(\cdot)$ is the ceiling function. With this, the nodes can identify each identifier in $[n]$ with elements in $[n^{\frac{1}{3}}] \times [n^{\frac{1}{3}}] \times [n^{\frac{1}{3}}]$.

Before the protocol starts, each node represents its neighborhood $N(u)$ as a function $\psi_v : [n^{\frac{1}{3}}]^3 \to \{0, 1\}$ where $\psi_v(i, j, k) = 1$ if and only if there exists a u such that $\mathsf{id}(u)$ is identified with (i, j, k) and $u \in N(v)$. These functions can be extended to a polynomial $\psi_v : \mathbb{F}_q^3 \to \mathbb{F}_q$ with $q = \Theta(\mathrm{poly}(n))$ with degree $n^{\frac{1}{3}} - 1$ [CLF12]. Using this, we can define a polynomial $\Psi_{uv} = \psi_u \cdot \psi_v$ that represents the intersection between $N(u)$ and $N(v)$. In fact, $N(u) \cap N(v) = \emptyset$ if and only if $\Psi_{uv}(x, y, z) = 0$ for any $(x, y, z) \in [n^{\frac{1}{3}}] \times [n^{\frac{1}{3}}] \times [n^{\frac{1}{3}}]$. Notice that these polynomials have degree bounded by $(2n^{\frac{1}{3}} - 2)$.

We also have that u does not participate of any triangle if and only if $\Psi_{uv}(x, y, z) = 0$ for all $v \in N(u)$ and all $(x, y, z) \in [n^{\frac{1}{3}}]^3$.

Finally, we define the functions

$$\Psi_u(x) = \sum_{v \in N(u)} \sum_{y,z} \Psi_{uv}(x, y, z)$$

$$\tau_u(x, y) = \sum_{v \in N(u)} \sum_{z} \Psi_{uv}(x, y, z)$$

Both polynomials have degree at most $2(n^{\frac{1}{3}} - 1)$ and have the following property: a node u does not participate in any triangle if and only if $\Psi_u(x) = 0$ for any $x \in [n^{\frac{1}{3}}]$. This

occurs because, by extending the original function ϕ_u each term in $[n^{\frac{1}{3}}]$ for the sum in Ψ_u can be at most $n \cdot n^{\frac{2}{3}}$. Therefore, by considering $q > n^2$, we have that q is strictly larger than all values for Ψ_u.

Now we proceed to describe the protocol:

In the first round, Merlin sends to each node u a function $\Phi_u : \mathbb{F}_q \to \mathbb{F}_q$ of degree $O(n^{\frac{1}{3}})$ which is a candidate for Ψ_u. Then, the nodes collectively send (that is, by use of shared randomness) a seed $r \in \mathbb{F}_q$. In the first round Merlin sends $\mathcal{O}(n^{\frac{1}{3}} \log n)$ bits while in the second one the nodes send $\mathcal{O}(\log n)$ bits.

In the next round, Merlin sends to each node the function $\tau_u^r(y)$ which is a candidate for $\tau_u(r, y)$. This once again requires Merlin to send $\mathcal{O}(n^{\frac{1}{3}} \log n)$ bits.

Finally, during the verification round, the nodes collectively generate a second seed $\bar{r}$ of $\mathcal{O}(\log n)$ bits, locally construct a function $\psi_u(r, \bar{r}, z)$ and send it to their neighbors. With this, each node u reconstructs the value of $\tau_u(r, \bar{r})$ and compares it to $\tau^r(\bar{r})$, rejecting in case that these values differ.

Then they proceed to construct $\sum_y \tau_u^r(y)$ and compare it to the value of $\Phi_u(r)$. Once again they reject in case that these values are different. If u has not rejected so far, they continue to evaluate $\Phi_u(x)$ for $x \in [n^{\frac{1}{3}}]$ and accept if and only Φ_u vanishes at each of these values.

- **Completeness.** It follows that if G has no triangles then an honest prover will send the functions Ψ_u and $\tau(r, u)$ accordingly and therefore all nodes will always accept.

- **Soundness.** Suppose that G contains a triangle. We have that each node will construct $\tau_u(r, \bar{r})$ and, if Merlin sent a polynomial $\tau_u^r(y)$ that does not match $\tau_u(r, y)$ then, as he already committed to a value for r, with probability $\frac{1}{n^c}$ (where can choose $c > 3$ to be arbitrarily large) its valuations in $\bar{r}$ must differ and u rejects. Then, with high probability we can assume that each node u is able to construct the right value of $\Psi_u(r)$ and compare it to $\Phi_u(r)$. Once again, if Merlin was not honest then the probability that $\Psi_u(x)$ matches $\Psi_u(x)$ in r for some node is at most $1/n^{c-7/3}$.

It follows that Δ-FREE $\in \mathsf{dMA}^{\mathrm{pub}}[4,\ n^{\frac{1}{3}} \log n]$.

□

Extension to multiple rounds

Now we go over the description of the full protocol. In a similar fashion as previously described, a node v will encode its neighborhood (and the intersection between those of its neighbors) by a $k + 1$variable polynomial Ψ_v which he doesn't know right away, as obtaining its full description would require him to send large messages, but tries to reconstruct it by a series of partial evaluations τ_v^i provided by Merlin, where at each

round Merlin commits to a certain value for a variable in this function. After all $2k$ rounds occur, the only thing that v is sure of is that he computed the right value for Ψ_v when he evaluates it at all random values already sent, if all these evaluations are consistent to the previous information he is assured that the function provided by Merlin is the right one, therefore he will decide depending on the values provided by it.

Theorem 3.16 *Let $k \geq 2$ be an integer. If the nodes know their neighbors' identifiers, then Δ-FREE $\in \mathsf{dMA}^{\mathrm{pub}}[2k, n^{\frac{1}{k+1}} \log n]$.*

Proof. Once again, we may assume that the nodes have identifiers in $[n]$ which can be seen as elements from $[n^{\frac{1}{k+1}}]^{k+1}$.

Then, the nodes define the functions $\psi_u, \Psi_{uv} : \mathbb{F}_q^{k+1} \to \mathbb{F}_q$ analogously to Proposition 3.15, where $n^c \leq q \leq 2n^c$, with c to be chosen arbitrarily large. Finally, for $i \in [k]$ they define the functions τ_u^i as:

$$\Psi_u(x_1) = \sum_{x_j : j>1} \sum_{v \in N(u)} \Psi_{uv}(x_1, \dots x_{k+1})$$
$$\tau_u^i(x_1, \dots x_i) = \sum_{v \in N(u)} \sum_{x_j : j>i} \Psi_{uv}(x_1, \dots x_{k+1})$$

where all these polynomials have degree at most $2(n^{\frac{1}{k+1}} - 1)$ and we notice that $\tau_u^1 = \Psi_u$. It also holds that u does not participate of any triangle if and only if $\Psi_u(x_1) = 0$ for all $x_1 \in [n^{\frac{1}{k+1}}]$.

With this we can describe the protocol:

In the first round, Merlin sends to each node u the function Φ_u which is a candidate for Ψ_u. Then, consider round 2ℓ, for $\ell \in [k-1]$. Arthur sends a seed r_ℓ of $\mathcal{O}(\log n)$ bits. From this message, Merlin proceeds to answer to each node with the functions $\hat{\tau}_{\vec{r}}^{\ell+1}(u, x_{\ell+1})$, with $\vec{r} = (r_1, \dots r_\ell)$, corresponding to a candidate for the polynomial $\tau_u^{\ell+1}(r_1, \dots, r_\ell, x_{\ell+1})$.

At last, in round $2k$, Arthur generates a seed r_k and, during the verification round, each node sends $\psi_u(r_1, \dots, r_k, x_{k+1})$, which has degree at most $n^{\frac{1}{k+1}} - 1$.

Now, locally, each node u constructs $\Psi_{uv}(r_1, \dots, r_k, x_{k+1})$ and, summing over $v \in N(u)$ and over x_{k+1}, it constructs $\tau_u^k(r_1, \dots r_k)$ comparing it to $\hat{\tau}_{\vec{r}}^k(u, r_k)$. and accepting if these values match and rejecting otherwise. Afterwards, for each $\ell = 1, \dots k-1$, each u verifies that

$$\hat{\tau}_{\vec{r}}^{\ell}(u, r_\ell) = \sum_{x_{\ell+1}} \hat{\tau}_{\vec{r}}^{\ell+1}(u, x_{\ell+1})$$

and rejects otherwise, where we denote $\hat{\tau}^1(u, r_1) = \Phi_u(r_1)$.

If all these equalities hold, then node u accepts if and only if $\Phi_u(x_1) = 0$ for each $x_1 \in [n^{\frac{1}{k+1}}]$.

- **Completeness.** It follows that if G has no triangles, an honest Merlin will send for each ℓ the correct τ^ℓ, the same as Ψ_u, for which all nodes accept.

- **Soundness.** During the local verification round, each node u will reconstruct the right value of $\tau_u^k(r_1, \dots r_k)$. Therefore, given that Merlin has already committed to values for $r_1, \dots r_{k-1}$, if Merlin was not honest at the last round, the probability that his function evaluated at r_k matches $\tau_u^k(r_1, \dots r_k)$ for some node u is bounded by $1/n^{c-\frac{k+2}{k+1}}$. Therefore, we may assume that Merlin provided the functions the nodes requested during the last round.

  Now, set any $\ell \in [k-1]$, and suppose that $\hat{\tau}_{\vec{r}}^{\ell+1}(u, x_{\ell+1})$ equals $\tau_u^{\ell+1}(r_1, \dots, r_\ell, x_{\ell+1})$ in $\mathbb{F}_q$. Then, it follows that if u sums up over the values of $x_{\ell+1}$ then $\sum_{x_{\ell+1}} \hat{\tau}_{\vec{r}}^{\ell+1}(u, x_{\ell+1})$ matches the real value of $\tau_u^\ell(r_1, \dots, r_\ell)$. Now, given that Merlin has already committed to the values for $r_1, \dots r_{\ell-1}$, if Merlin were to send $\hat{\tau}_{\vec{r}}^{\ell}(u, x_\ell) \not\equiv \tau_u(r_1, \dots r_{\ell-1}, x_\ell)$, then the probability that their values match is bounded by $1/n^{c-\frac{2k+3}{k+1}}$. In the event that Merlin was honest at each phase of the protocol, then, if G has a triangle there will exist a node u for which $\Phi_u(x^*) \neq 0$ at some point x^* and therefore rejects the protocol.

□

We can see that this improves the result by [CFP19] which is a $\mathsf{dMA}^{\text{pub}}$ with cost $\mathcal{O}(\sqrt{n}\log n)$ and required all nodes to have identifiers in $[n]$, something that we can bypass by the use of more rounds. It is also interesting to compare to the protocol for the Congest model for triangle detection, which requires $\mathcal{O}((n\log n)^{2/3})$ rounds of communication. However, we are unable to improve the cost obtained by the compiler from [NPY20] in combination with Theorem 1.16 as it obtained a (non-constructive) protocol for triangle detection using $\mathcal{O}(1)$ rounds and bandwidth $\mathcal{O}(\log n)$ as it is a class which can be easily detected using logarithmic space. Regardless, the fact that our protocol uses shared randomness (while the one by the NPY compiler uses private randomness), indicates that this is still an improvement over what is currently known, as $\mathsf{dAM}^{\text{priv}}$ protocols can be stronger than $\mathsf{dAM}^{\text{pub}}$ protocols (we refer to Section 5.2 for more on this discussion).

The fact that, for the protocol to be correct, we need to assume that the nodes know the identifiers of their neighbors, relies on the fact that the last polynomial sent by the nodes depends on these values. This restriction can be avoided by adding an additional round of interaction at which Merlin sends to all nodes the function they would have sent had they known the identifiers, and then verify these value when the nodes exchange id's. In such case, we would have that $\Delta\text{-FREE} \in \mathsf{dAM}^{\text{pub}}[2k+1, n^{\frac{1}{k+1}}]$.

Chapter 4

Lower Bounds

This chapter is devoted mainly to the study of lower bounds for interactive proofs for the problems studied in Chapters 2 and 3. We show, by adapting different techniques present in the literature, how to obtain lower bounds for the problems previously described in both the dM and dAM models. In Section 4.1, we show how to obtain a $\Omega(\log n)$ lower bound for the problem d-DEGENERATE as well as an extension to a $\Omega(\log \log n)$ lower bound in $\mathsf{dAM}^{\text{pub}}$, for any fixed k. In Section 4.2, we show a $\Omega(\log n)$ lower bound on any dM protocol for the problem COGRAPH. In Section 4.3, by adapting a technique by Göös and Suomela [GS16] for locally checkable proofs, we show a general result which provides lower bounds on both the dM and $\mathsf{dAM}^{\text{pub}}$ models, obtaining lower bounds on many problems described in the previous chapters. Specifically, we show that any protocol for the problems TWINS, DIST-HEREDITARY, INTERVAL, PROPER INTERVAL, CIRCULAR-ARC, PROPER CIRC-ARC and CHORDAL has cost $\Omega(\log n)$ and $\Omega(\log \log n)$ on the models dM and $\mathsf{dAM}^{\text{pub}}[k]$ (for any fixed k) respectively. Also, we include a way to extend these lower bounds to the dMA model, obtaining a bound of $\Omega(\log n)$ for the problems mentioned before. Finally, in Section 4.4, we extend a lower bound for the problem SYMMETRY obtaining a $\Omega(\log \log n)$ lower bound for $\mathsf{dMAM}^{\text{priv}}[k]$ protocols on any fixed k.

4.1 A dAM lower bound for the degeneracy

In Section 2.1, on Proposition 2.1 we showed a dM protocol for the problem d-DEGENERATE with cost $\mathcal{O}(\log n)$. It follows to ask whether this is tight. In this section we show a lower bound using a technique described in [FH18]. We construct a family of path-like Yes-instances that share the same proof, by which we can fool the network into accepting a No-instance that locally behaves like these graph. With this we show that there is no better protocol for computing the degeneracy, and that no more than an exponential improvement can be obtained if we add interaction in the shared randomness setting. We start by describing the construction.

Let $p \geq 1$ be an integer and, without loss of generality, suppose that $n = (d+1)(p+2)$. For each $r \in \{0, .., p+1\}$, we define a *node block* B_r consisting of d nodes with identifiers

in $r(d+1), r(d+1)+1, .., (d+1)(r+1)$ forming a clique.

We consider two special blocks: an *initial block* B_0 and a *final block* B_{p+1}. Any other block will be called *ordinary*.

Given a labeled clique C as defined above, we write C^r and C^ℓ to denote the sets of $\lceil |C|/2 \rceil$ and $\lfloor |C|/2 \rfloor$ nodes of C with largest and smallest identifiers, respectively.

From here we define a connection from a block B_i to a block B_j as a *join* between the sets B_i^r and B_j^ℓ.

Consider now π to be a permutation of p elements. To construct a π-path of block nodes, it suffices to define a series of connections from B_0 to $B_{\pi^{-1}(1)}$, from $B_{\pi^{-1}(1)}$ to $B_{\pi^{-1}(2)}$ and so on until we connect $B_{\pi^{-1}(p)}$ to B_{p+1}.

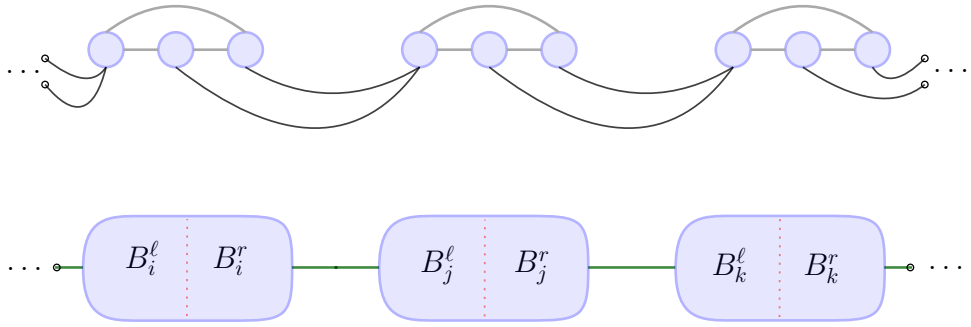

Figure 4.1: Case $d = 3$: Each tree node block is partitioned into left and right sub-blocks which are connected with all nodes in the previous right sub-block or next left sub-block respectively.

In a similar manner, we may define a π-*block cycle* as follows: given two values $c < c'$ lying between 1 and p, we make a connection from $B_{\pi^{-1}(\ell)}$ to $B_{\pi^{-1}(\ell+1)}$ for $\ell \in \{c, \dots c'-1\}$ and a connection from $B_{\pi^{-1}(c')}$ to $B_{\pi^{-1}(c)}$. It is important to mention that a block cycle only consider a portion of all blocks. Also, as the identifiers for each block are different, both block paths and block cycles are well defined as configurations.

Claim 4.1 *Let d be an integer. For any permutation π, P_π has degeneracy at most d*

Proof. It suffices to *trim* the blocks one by one, from B_0 to B_{p+1}. If the first block in a path is B followed by $\bar{B}$, by deleting all nodes in B^ℓ, the nodes in B^r are only adjacent to each other and to the nodes in $\bar{B}^\ell$. Therefore, these have degree $\lceil d/2 \rceil + \lfloor d/2 \rfloor = d$. □

Claim 4.2 *Let d be an integer bigger than 2. For any permutation π, any cycle C_π induced by it has degeneracy strictly bigger than d*

Proof. Given an arbitrary block B in the cycle, such that there is a connection from $\bar{B}$ to B and from B to $\hat{B}$, any node in B^ℓ has degree $d + \lfloor d/2 \rfloor > d$ whenever $d \geq 2$. For the case of nodes in B^r, the proof is analogous. □

To finally obtain our result, it is sufficient to combine both claims and show that it is impossible to distinguish between block paths and block cycles when Merlin's messages are small.

Proposition 4.3 *Let d be an integer. If* d-DEGENERATE $\in \mathsf{dM}[f(n)]$, *then* $f(n) = \Omega(\log n)$.

Proof. The case $d = 1$ corresponds to testing if a graph is acyclic, which is known [GS16] to require $\Omega(\log n)$ bits in a single round of interaction, therefore we ignore this case.

Consider then $d \geq 2$ and, assuming that there exists a dM protocol $\mathcal{P}$ with cost $g(n) = o(\log n)$, we will prove that there exists a pair of paths for which Merlin provides the same set of proofs.

Indeed, if we consider the set of possible proofs sent by Merlin that an arbitrary block B may receive, we have that there are at most $2^{(d-1)g(n)}$ such possible proofs.

Notice that there are at least $2^{(d-1)g(n)p}$ possible labeled blocks. Finally, given a path forming p different blocks with a valid label configuration according to some ordering π, there are $p!$ possible paths to be constructed, which will receive proofs from Merlin.

Considering that $p = \Theta(n)$, and comparing the logarithm of the previous values we notice that they have the following asymptotic behavior:

$$\log\left(2^{(d-1)g(n)p}\right) = o(n \log n) \quad \text{and} \quad \log(p!) = \Omega(n \log n)$$

By the pigeonhole principle, it follows that there exist two paths P and $\bar{P}$ which receive the same set of proofs, which respect the order by which these paths are conformed, such that the nodes in them accept the protocol for d-DEGENERATE.

Without loss of generality, we have that P is constructed according to the identity ordering. Then, for the permutation π that defines $\bar{P}$ we have that there exists a pair of positions $i < j$ such that $\pi_i > \pi_j$. From here we may construct a block cycle C as follows: the blocks $B_i, B_{i+1}, \ldots B_j$ are connected consecutively and we add a last connection from B_i to B_j as in Figure 4.2.

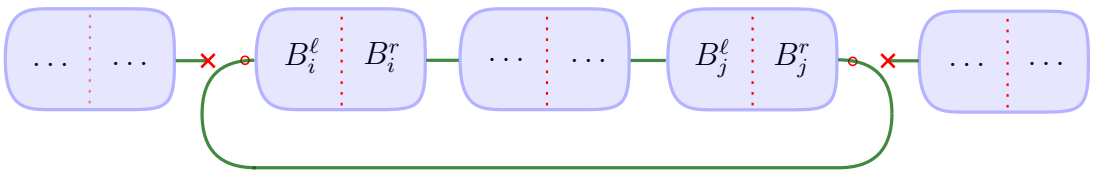

Figure 4.2: For small sized proofs, it is possible to find an ordering π where the block B_j mistakes the next block with B_i, as they both locally behave like an accepting instance.

With this, any node u in $\bigcup_{k=i+1}^{j-1} B_k \cup B_i^r \cup B_j^\ell$ believes that it belongs to the path P (as

locally they behave the same and have the same proof) and accepts. Finally, any node $w \in B_j^r \cup B_i^\ell$ believes itself to be in $\bar{P}$ and accept as well.

We conclude that there exists a graph with large degeneracy which receives proofs of size $o(\log n)$ such that its nodes accept, contradicting the correctness of $\mathcal{P}$. □

To study lower bounds for more interactions, it is possible to extend the previous result to obtain lower bounds in the $\mathsf{dAM}^{\text{pub}}$ model, where we have that any protocol using a constant number of rounds can have at most an exponential improvement.

Corollary 4.4 *Let $k \geq 2$ be an integer. If* d-DEGENERATE $\in \mathsf{dAM}^{\text{pub}}[k, f(n)]$ *, then* $f(n) = \Omega(\log \log n)$.

Proof. If we use the same block construction as before, it follows that, if $f(n) = o(\log \log n)$ and we consider a function $g(\cdot) : \{0,1\}^{\ell\, f(n)} \to \{0,1\}^{d \cdot p \cdot \ell \cdot g(n)}$ with $\ell = \lfloor k/2 \rfloor$ that assigns the random coins used in the k rounds of interaction to the corresponding proofs sent by Merlin at each round, we have that there are at most

$$2^{\ell f(n) k p \cdot 2^{\ell f(n)}}$$

such possible assignments, while there are at most $p!$ permutations, with $p = \Theta(n)$. Therefore, once again comparing the logarithms of both numbers and comparing their asymptotic behavior we have that, if $f(n) = o(\log \log n)$, then it is possible to replicate the previous fooling configuration such that all nodes locally believe they belong to some yes instances for all k rounds of interaction and will accept the protocol with constant probability. □

4.2 A dM lower bound for COGRAPH.

In this section, we construct a lower bound using the *edge crossing technique* defined in [FPSP19]. Given a proof m for a yes-instances with multiple copies of some structure, we can "mix" a set of edges between these structures and still use m as an accepting proof as long as we respect the port ordering of the edges involved.

Now, in order to prove the result, we first describe the edge crossing lemma (and its proof, for the sake of completeness) and for then to describe how to apply it to our case.

Given a graph $G = (V, E)$ and a pair of isomorphic subgraphs H_1 and H_2 such that they are node-disjoint and have no edges between them (we say that H_1 and H_2 are *independent*) with the isomorphism $\sigma : V(H_1) \to V(H_2)$, we define the *crossing* of G induced by σ, denoted by $\sigma_{\bowtie}(G)$, as the graph obtained by replacing all edges $\{u, v\} \in E(V_1)$ and $\{\sigma(u), \sigma(v)\} \in E(H_2)$ by the pair $\{u, \sigma(v)\}$ and $\{\sigma(u), v\}$.

Lemma 4.5 (by [FPSP19]) *Let $\mathcal{P}$ be a* dM *protocol for some language $\mathcal{L}$ with cost L. Suppose there exists a configuration (G, id, I) where G contains k pairwise independent iso-*

morphic subgraphs $\{H_j\}_{j=1}^k$ *with* s *edges each and let* $\sigma_j : H_1 \to H_j$ *be a Port-preserving isomorphism for each* $i \in [k]$. *If* $L < \frac{\log k}{2s}$, *then there exists a prover* $\mathcal{M}$ *and a pair* $i < j \in [k]$ *such that all nodes in the configuration* $\langle G, \mathsf{id}, I\rangle$ *accept given* $\mathcal{M}$ *if and only all nodes in the configuration* $(\hat{\sigma}_{\bowtie}(G), \mathsf{id}, I)$ *accept given* $\mathcal{M}$, *where* $\hat{\sigma} = \sigma_j \circ \sigma_i^{-1}$.

Proof. Let (G, id, I)be a configuration as previously described. Assume that $L < \frac{\log k}{s}$, and consider a collection $\{\sigma_j\}_{j=1}^k$ of port preserving isomorphisms. For each j, consider w_j to be the concatenation of proofs given by $\mathcal{M}$ to the nodes of H_j in the order induced by σ_j. As $n(H_j) \leq 2s$, then $|w_j| < \log k$ for each j . And so there are at most k different value for w_j in total. By the pigeonhole principle, there must exists a pair $i < j$ such that $w_i = w_j$. Set $\hat{\sigma} = \sigma_j \circ \sigma_i^{-1}$ and the graph $\hat{\sigma}_{\bowtie}$ with the same input and identifiers.

Now, suppose that all nodes in the configuration $\langle G, \mathsf{id}, I\rangle$ accept given a proof provided by $\mathcal{M}$. Then, all nodes in $V(G) \setminus V(H_i \cup H_j)$ accept. Now consider a node $v \in H_i$. Each neighbor $V(H_i)$ is replaced by its counterpart in H_j. Then, as w and $\hat{\sigma}(w)$ have the same label and v does not know its neighbors identifiers, then from v's perspective its vicinity behaves essentially the same as in G. From here it follows directly that, if the proof provided by $\mathcal{M}$ is accepted at every node of G, then it is accepted at every node at $\hat{\sigma}_{\bowtie}(G)$ and if there exists a node in G that rejects, its counterpart in $\hat{\sigma}_{\bowtie}(G)$ would also reject and the lemma follows.

□

Proposition 4.6 *If* COGRAPH $\in \mathsf{dM}[f(n)]$, *then* $f(n) = \Omega(\log n)$.

Proof. Without loss of generality set $n = 1 \mod 3$ and consider a graph G given by a collection of disjoint triangles $\{H_i\}_{i=1}^k$ with $k = \lfloor \frac{n}{3} \rfloor$ and a universal node $\bar{v}$. We have that G is a cograph as it is the join between $\bar{v}$ and a disjoint union of cliques.

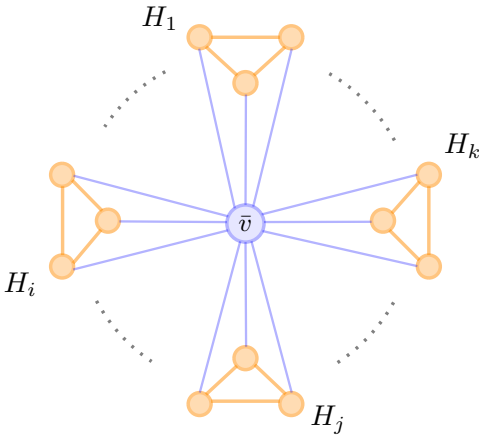

Figure 4.3: A Yes-instance for COGRAPH. There is a collection of disjoint small cliques (triangles H_j) which are joined to a single node $\bar{v}$.

Now suppose that there exists a dM protocol $\mathcal{P}$ with proofs of size $o(\log n)$. Then, by defining a collection $\{\sigma_j\}_{j=1}^{k}$ that preserves the order in each triangle we can use Lemma 4.5 to construct an instance $\hat{G}$ where the edges between a pair of triangles H_1 and H_2 are crossed according to $\sigma_2 \circ \sigma_1^{-1}$. Then, as G is a cograph and therefore there exists a prover $\mathcal{M}$ that makes G accept following $\mathcal{P}$, then $\hat{G}$ accepts for the same prover. But, by flipping a pair of triangles in $\hat{G}$, we created an induced 6-cycle, and therefore an induced P_4. This contradicts the correctness of $\mathcal{P}$.

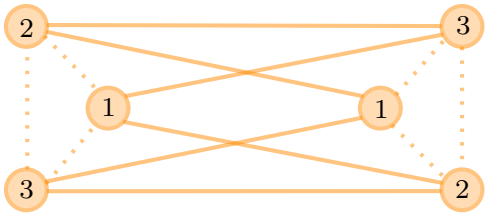

Figure 4.4: Fooling construction for the COGRAPH problem. As the graph contains several independent triangles, there exists two of them such that , if we connect them according to their port orderings, the corresponding nodes accept under the same proofs, as they can not see the identifiers of their neighbors.

□

Notice that this lower bound also holds when distance-hereditary graphs are considered: Our construction has an induces 8 cycle and it is known that this class of graphs can not have cycles of length at least 5 without any diagonals [BS+99].

Corollary 4.7 *If* DIST-HEREDITARY $\in$ dM$[f(n)]$ *then* $f(n) = \Omega(\log n)$.

We show this result again in Section 4.3 in the stronger setting where the nodes are aware of the identifiers of their neighbors.

4.3 A general lower bound for public dAM

In this section, we proceed to extend and apply some lower-bound techniques from other non deterministic models such as LCP with special interest in the "glueing" technique by Göös and Suomela [GS16]. This technique consists in combining several positive instances, which can be partitioned by a small cut, into a new, negative, instance. The fact that the cut is small allows to study the collection of accepting proofs for the whole graph solely through the proofs provided to this cut, which can then be extended to both parts. If the proofs are sufficiently small we can find a family of correct instances with a similar cut which can then be "glued" together, as the cut behaves essentially the same in both the original instances and in the glued instance.

We consider some modifications in order to be applicable in our model. We obtain different lower bounds for the problems studied in Chapters 2 and 3 problems in both the dMA and dAM$[k]$ (where k is fixed) interactive classes.

We start by describing a general version of the protocol that depends on two graphs G_A and G_B which we will define later. Depending on the properties of these two graphs we obtain lower bounds for different graph classes.

Theorem 4.8 *If any of the classes* TWINS, DIST-HEREDITARY, INTERVAL, PROPER INTERVAL, CIRCULAR-ARC, PROPER CIRC-ARC *and* CHORDAL *is in* $\mathsf{dM}[f(n)]$, *then* $f(n) = \Omega(\log n)$. *Moreover if, for any fixed* k, *any of them is in* $\mathsf{dAM}^{\text{pub}}[k, g(n)]$, *then* $g(n) = \Omega(\log \log n)$.

Proof. We first describe a construction for lower bounds in $\mathsf{dAM}^{\text{pub}}$, then we explain how to deduce stronger lower bound on dM protocols. We begin by obtaining lower bounds for the classes TWINS and DIST-HEREDITARY.

Without loss of generality, we may assume that n is even. Let A be a partition of $[1, n^2]$ into n sets of size n, and let B be a similar partition of $[n^2 + 1, 2n^2]$. Let $\mathcal{G}$ be a family of n-node graphs satisfying some property related to our problem: In the case of the problem TWINS, the graphs in $\mathcal{G}$ correspond to be the class of twin-free graphs. As for the problem DIST-HEREDITARY, we consider $\mathcal{G}$ to be the class of cographs.

Set now $\mathcal{G}_A$ to be the set of labeled graphs in $\mathcal{G}$, with label sets picked from A. Let F_a be a graph in $\mathcal{G}_A$, and let v^* be the node of F_a labeled with the smallest label. Similarly, we define $\mathcal{G}_B$ as the set of graphs in $\mathcal{G}$ labeled with labels in B.

For $(F_a, F_b) \in \mathcal{G}_A \times \mathcal{G}_B$ let $G(F_a, F_b)$ be the graph defined by the disjoint union of graphs F_a and F_b plus four additional nodes x_A, y_A, x_B, y_B. These nodes are labeled with different numbers in the set $C = [2n^2 + 1, 3n^2]$. Nodes x_A and x_B are both adjacent only to y_A and y_B. Node y_A is adjacent only to x_A, x_B and v_a, where v_a is some node of F_a. Node y_B is adjacent only to x_A, x_B and v_b, where v_b is some node of F_b. Observe that all nodes in F_a communicate with F_b only through the nodes x_A, y_A, x_B, y_B.

From the families $\mathcal{G}$ described above we have that the graph $G(F_a, F_b)$ is a Yes-instance for both problems: In TWINS, we have that there is a unique pair of twins (namely (x_A, x_B)), as both F_a and F_b are twin-free graphs. As for DIST-HEREDITARY, we have that, for any $(a, b) \in A \times B$, the graph $G(F_a, F_b)$ is distance hereditary: simply start by sequentially constructing a C_4 (namely $\{x_A, y_A, x_B, y_B\}$) , add two pending nodes, namely v_A and v_B and proceed to construct both graphs F_a and F_b in parallel by a sequence of (true or false) twins.

Let $\mathcal{P}$ be a k-round distributed interactive proof with shared randomness verifying a property P with bandwidth $K = \delta \log \log n$ and error probability ε. Let us call $\{x_A, y_A, x_B, y_B\}$ the *bridge* of $G(F_a, F_b)$. We can assume, without loss of generality, that $\mathcal{P}$ satisfies that for any (shared) random string generated by Arthur, the nodes in the bridge $\{x_A, y_A, x_B, y_B\}$ receive the same proof. Indeed, if we have a protocol that is not simple, with cost L, we can design a new protocol that is simple and whose proof has length $4L$ by making each node pick their portion of the proof and going by the original

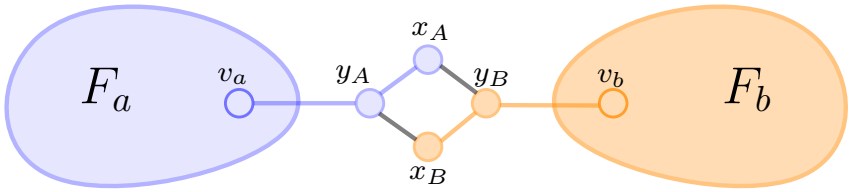

Figure 4.5: The auxiliary graph $G(F_a, F_b)$, with $a \in A$ and $b \in B$. This is a yes-instance for TWINS, as there is a unique pair of twins in the middle. It is also a yes-instance for DIST-HEREDITARY, as we can set both parts to hold a cograph, united by a 4-cycle.

protocol afterwards.

Given sequence of random strings $r = (r_1, r_2, \dots r_k)$, we call m^r the sequence indexed by nodes $v \in V(G[F_a, F_b])$, such that m^r_v is the set of certificates that Merlin sends to node v in protocol $\mathcal{P}$, when Arthur communicate string r_i on round i. Let $m_{ab} : \{0,1\}^{Kk} \to \{0,1\}^{Kk}$ be the function that associates to each sequence $r = (r_1, r_2, \dots r_k)$ the tuple $(m^r_{x_A}, m^r_{x_B}, m^r_{y_A}, m^r_{y_B})$ such that it extends to a proof assignment for the nodes in both F_a and F_b. that make them accept whenever the bridge accepts.

Now consider the complete bipartite graph $\hat{G} = A \cup B$. For each $a \in A$ and $b \in B$, color the edge $\{a, b\}$ with the function m_{ab}. There are at most $2^{Kk2^{Kk}}$ possible functions. Therefore, by the pigeonhole principle, there exists a monochromatic set of edges W of size at least $\frac{n^2}{2^{Kk2^{Kk}}}$.

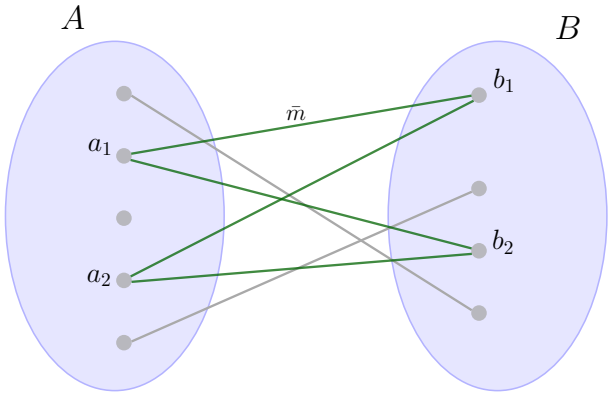

Figure 4.6: Auxiliary (complete bipartite) graph $\hat{G} = A \cup B$ with each node representing a set of identifiers $a \in A, b \in B$. The gray edges show a monochromatic set W colored by $\bar{m}$, the green edges show a 4-cycle present in W.

Observe that for sufficiently small δ and large n, $2^{Kk2^{Kk}} = (\log n)^{\delta k \log^{\delta k}(n)} = o(n^{1/2})$. Indeed, if $n > 2^{k\delta}$ and $\delta < 1/(4k)$ have that $\delta k \log^{\delta k}(n) \log\log(n) \leq \log^{2\delta k}(n) < \frac{1}{2}\log n$. Following a result of Bondy and Simunovitz given in Lemma 1.13, we have that there exists a 4-cycle a_1, b_1, a_2, b_2 in the subgraph $\hat{G}$ induced by W.

Consider now the graph $G(a_1, b_1, a_2, b_2)$ defined as follows: First. take a disjoint union of $F_{a_1}, F_{b_1}, F_{a_2}$ and F_{b_2}. Then, for each $i \in \{1, 2\}$ add nodes $x^i_A, x^i_B, y^i_A, y^i_B$, labeled with different labels in $[2n^2 + 1, 3n^2]$ corresponding to the yes instances formed by F_{a_i} and F_{b_j}.

For each $i \in \{1, 2\}$, the node y_A^i is adjacent to x_A^i, x_B^{i+1} and the node $v_{a_i} \in F_{a_i}$. Similarly, y_B^i is adjacent to the nodes x_A^i, x_B^{i+1} and the node $v_{b_i} \in F_{b_i}$, where the $i + 1$ is taken mod 2.

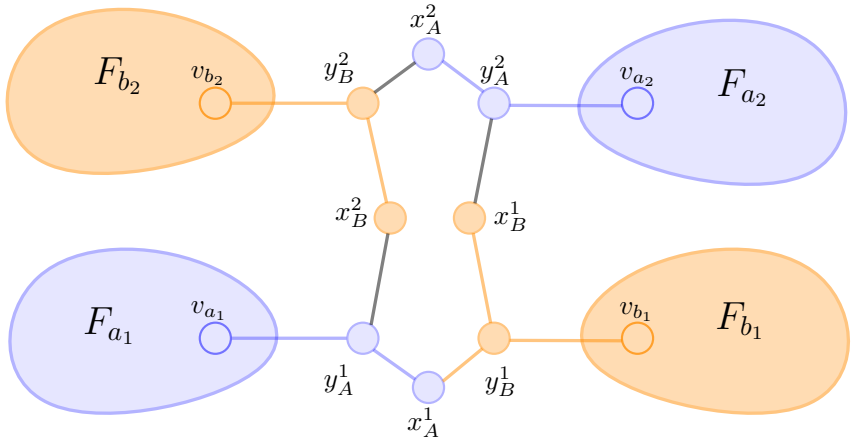

Figure 4.7: A No-instance for TWINS and DIST-HEREDITARY, with messages of size $o(\log n)$. From every node's perspective, the graph behaves like a Yes-instance, but there are no twins in the former, and a large cycle without diagonals for the latter.

It remains to show that the graph $G(a_1, b_1, a_2, b_2)$ is a No-instance for the properties P in {TWINS, DIST-HEREDITARY}.

- As we said above, for the class TWINS we have that both graphs F_a and F_b are twin-free graphs, for any pair (a, b). However, the graph $G(a_1, b_1, a_2, b_2)$ is a twin-free graph, which is easy to see as it is connected and does not have any 4-cycles.

- As for the problem DIST-HEREDITARY, it suffices to consider another characterization for the class DIST-HEREDITARY, namely these are the graphs such that any cycle of length at least 5 has a pair of crossing diagonals [BS+99] , we can easily show that the graph $G(a_1, b_1, a_2, b_2)$ can not be distance hereditary, as it contains an 8-cycle with no diagonals as an induced subgraph.

From here we show that this No-instance is capable of fooling the verifier. Indeed, all nodes of the set $x_A^1, x_B^1, y_A^1, y_B^1, x_A^2, x_B^2, y_A^2, y_B^2$ receive the same answers by Merlin which extends to assignments for the nodes in F_{a_i} and F_{b_i} that make them accept with the same probability as the nodes in the bridge, as they locally place themselves in a previously defined yes instance. Therefore all nodes accept two thirds of all possible random coins. This contradicts the fact that $\mathcal{P}$ was a correct distributed interactive proof for P.

Now, if we consider the classes PROPER INTERVAL, INTERVAL, CIRCULAR-ARC, PROPER CIRC-ARC and CHORDAL, we consider a slight modification in the graph construction $G(F_a, F_b)$ in order for the previous argument to work, with help from a result from [Erd64]. Indeed, consider again n to be even and A to be a partition of $[1, n^2]$ into n sets of size n, and B a partition of $[n^2 + 1, 2n^2]$ in a similar manner. Let $\mathcal{G}$ be a family of n-node graphs satisfying some property P among those mentioned.

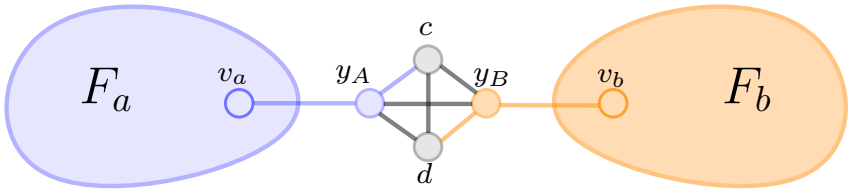

Figure 4.8: A yes-instance for INTERVAL and its super-classes, as F_a and F_b are two interval graphs which are connected through a 4-clique in the middle.

Set $\mathcal{G}_A$ and $\mathcal{G}_B$ to be the set of labeled graphs in $\mathcal{G}$, with label sets picked from A and B respectively. Let F_a be a graph in $\mathcal{G}_A$ and F_b a graph from $\mathcal{G}_B$. Finally consider two disjoint sets C and D in $[2n^2+1, 3n^2]$ of size n.

For $(F_a, F_b, c, d) \in \mathcal{G}_A \times \mathcal{G}_B \times C \times D$ let $G(F_a, F_b, c, d)$ be the graph defined by the disjoint union of graphs F_a and F_b plus four additional nodes y_A, y_B, c, d. The nodes y_A, y_B are labeled with different numbers in a set from $[2n^2+1, 3n^2]$ disjoint from C and D. The nodes y_A, y_B, c and d form a 4-clique, while the nodes y_A is connected to some node v_a in F_a. Node y_B is similarly adjacent some node v_b in F_b. Observe that all nodes in F_a communicate with F_b only through the nodes y_A, y_B, c and d.

Let $\mathcal{P}$ be a k-round distributed interactive proof with shared randomness verifying the property P with bandwidth $K = \delta \log\log n$ and error probability ε. Let us call $\{y_A, y_B, c, d\}$ the *bridge* of $G(F_a, F_b, c, d)$. We can assume, without loss of generality, that $\mathcal{P}$ satisfies that for any (shared) random string generated by Arthur, the nodes in the bridge $\{y_A, y_B, c, d\}$ receive the same proof. Indeed, if we have a protocol that is not simple, with cost L, we can design a new protocol that is simple and whose proof has length $4L$ by making each node pick their portion of the proof and going by the original protocol afterwards.

Given a sequence of random strings $r = (r_1, r_2, \dots r_k)$, we call m^r the sequence indexed by nodes $v \in V(G[F_a, F_b, c, d])$, such that m^r_v is the set of certificates that Merlin sends to node v in protocol $\mathcal{P}$, when Arthur communicate string r_i on round i. Let $m_{abcd} : \{0,1\}^{Kk} \to \{0,1\}^{Kk}$ be the function that associates to each sequence $r = (r_1, r_2, \dots r_k)$ the tuple $(m^r_{y_A}, m^r_{y_B}, m^r_c, m^r_d,)$ such that it extends to a proof assignment for the nodes in both F_a and F_b. that make them accept whenever the bridge accepts.

Now consider the complete 4-partite, 4-uniform hyper-graph graph $\tilde{G} = A \cup B \cup C \cup D$. For each $a \in A, b \in B, c \in C$ and $d \in D$, color the edge $\{a, b, c, d\}$ with function m_{abcd}. There are at most $2^{Kk2^{Kk}}$ possible functions. Therefore, by the pigeonhole principle, there exists a monochromatic set of hyper-edges H of size at least $\frac{n^4}{2^{Kk2^{Kk}}}$. Observe that for sufficiently small δ and large n, $2^{Kk2^{Kk}} = (\log n)^{\delta k \log^{\delta k}(n)} = o(n^{1/8})$. Indeed, if $n > 2^{k\delta}$ and $\delta < 1/(2^4 k)$ have that $\delta k \log^{\delta k}(n) \log\log(n) \leq \log^{2\delta k}(n) < \frac{1}{8}\log n$. Now, following a result from Ërdos, described in Lemma 1.14, by setting $\ell = 2, r = 4$ we have that there exists a $K^{(r)}(\ell)$ subgraph in $\tilde{G}$ induced by H. That is, the complete r-uniform, r-partite hyper-graph, where each part has size exactly ℓ. Let $\{a_i, b_i, c_i, d_i\}_{i=1}^2$ be the nodes involved in such a graph.

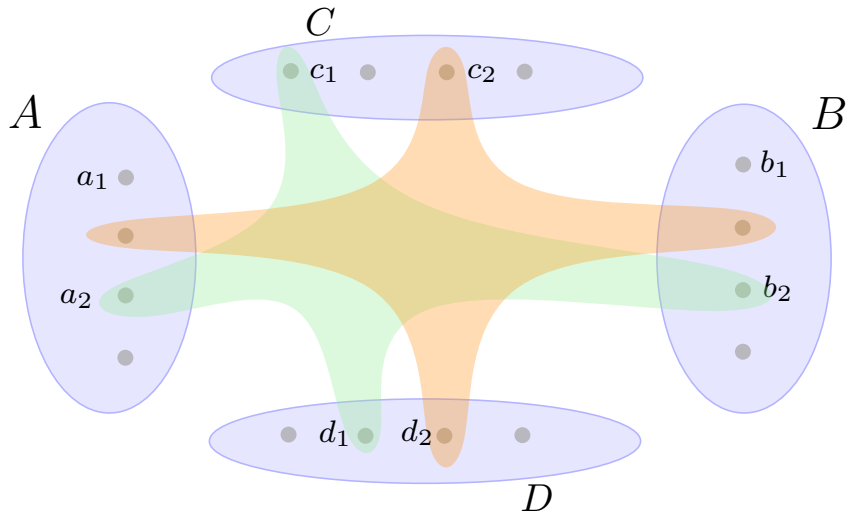

Figure 4.9: Auxiliary (complete) 4-uniform, 4 partite hyper-graph $\tilde{G} = A \cup B \cup C \cup D$ with each node in A and B representing a subgraph F_a, F_b and nodes in C and D representing single nodes in the original graph construction. The green and orange edge blocks are a pair of blocks from a monochromatic $K^(4)(2)$ structure present in the graph.

Consider now the graph $G(a_i, b_i, c_i, d_i)$ defined as follows: First. take a disjoint union of $F_{a_1}, F_{b_1}, F_{a_2}$ and F_{b_2}. Then, for each $i \in \{1, 2\}$ add nodes y^i_A, y^i_B, c^i, d^i, labeled with different labels in $[2n^2 + 1, 3n^2]$ correspondent to the yes instances formed by the graphs F_{a_i} F_{b_j}, and the nodes c_k and d_h . For each $i \in \{1, 2\}$, the node y^i_A is adjacent to y^i_B, c_i, d_{i+1} and the node v_{a_i} of F_{a_i}. Also, the node y^i_B is adjacent to c^i, d_i and the node $v_{b_i} \in F_{b_i}$ and c_i is adjacent to d_{i+1} Where the $i + 1$ is taken mod 2.

It remains to show that the graph $G(a_1, b_1, a_2, b_2)$ is a No-instance for the properties P in {PROPER INTERVAL, INTERVAL, PROPER CIRC-ARC , CIRCULAR-ARC, CHORDAL}.

- As for the classes INTERVAL and PROPER INTERVAL we simply define F_a and F_b to be a pair of proper interval graphs of size $\mathcal{O}(n)$, then $G(F_a, F_b, c, d)$ is also admits a representation through proper intervals as we simply connect both graphs through their extremes by a small clique. Finally, the newly constructed graph has an induced 6-cycle therefore can not have a representation by intervals.

- As for PROPER CIRC-ARC and CIRCULAR-ARC, by using the same construction (by assuring that each part has a large diameter), we also have a valid instance as (proper) interval graphs are in particular (proper) circular arc graphs. We simply consider their representation through intervals in the real line as a big arc in a portion of the circle. As for the newly obtained instance, we have a large, induced cycle which is consistent with a construction for a circular arc graph, while we also have large paths on each side which are in conflict with the cycle as a (proper) circular arc graph behaves locally like an interval graph and therefore can not have an asteroidal triple (three nodes at the extreme of each interval graph)

- Finally, for CHORDAL we can use the same graph described for the class INTERVAL

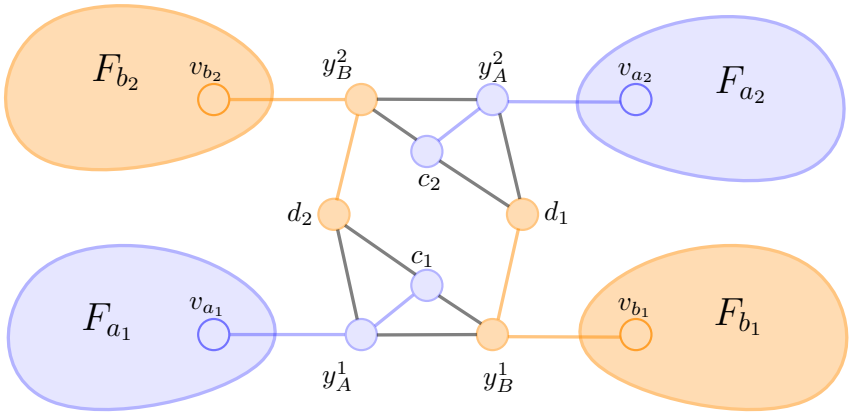

Figure 4.10: A No-instance for INTERVAL and its super-classes, as the graph admits a large cycle without any chords (and with large diameter subgraphs F_{a_i}, F_{b_j}), it can not admit a representation through intervals or circular arcs, nor can be a chordal graph.

as they are also chordal graphs. Finally, we have that the newly constructed graph has a 6-cycle without any chords. Therefore, it is not in CHORDAL.

□

This technique (in its two flavors) proves to be a very useful way to obtain simple lower bounds on the models dM and $\mathsf{dAM}^{\text{pub}}[k]$ for relatively simple graph classes, as this construction can be extended to other problems by choosing graph families $\mathcal{G}_A$ and $\mathcal{G}_B$ (and their inputs) appropriately as long as they admit a constant size cut. It is important to mention that this technique can not provide lower bounds stronger than $\Omega(\log n)$ for the dM class or $\Omega(\log \log n)$ for the $\mathsf{dAM}[k]$ class as a natural barrier in this construction is the fact that each configuration admits identifiers in a polinomially bounded set, and therefore the previously defined sets A and B can not be too large.

Now, by recycling the graph constructions from Theorem 4.8 and following a result by Fraigniaud et al. [FMO+19], we show that we can also obtain "strong" lower bounds for the models dMA in both its shared and private randomness versions if the acceptance probability is sufficiently good.

Corollary 4.9 *If any of the problems* TWINS *or* DIST-HEREDITARY *is in* $\mathsf{dMA}_{1/5}[f(n)]$, *then* $f(n) = \Omega(\log n)$. *If any of the problems* INTERVAL, PROPER INTERVAL, PROPER CIRC-ARC , CIRCULAR-ARC *or* CHORDAL *is in* $\mathsf{dMA}_{1/7}[g(n)]$, *then* $.g(n) = \Omega(\log n)$.

Proof. We use the constructions from Theorem 4.8 and change the ways each edge in the auxiliary (hyper) graph is colored. By bounding the acceptance probability according to what each node in the construction can see, we obtain the result.

We start with the classes TWINS and DIST-HEREDITARY. Indeed, suppose that we have a dMA protocol $\mathcal{P}$ with acceptance probability $\varepsilon \leq \frac{1}{5}$. We repeat the construction for the graph $G(F_a, F_b)$ given a set of identifiers $(a, b) \in A \times B$ as well as the bipartite

graph $\hat{G} = A \cup B$ and, using the result of Lemma 1.13, we obtain a monochromatic 4-cycle colored by the proof $\bar{m}$. Now we study the acceptance probability of the new instance $G(a_1, a_2, b_1, b_2)$ by going over the perspective of each node in the graph during the verification round. Let $V \coloneqq V(G[a_1, b_1, a_2, b_2])$ and let G_{ij} be the graph $G(F_{a_i}, F_{b_j})$. Then, the acceptance probability for our prover can be bounded as

$$\begin{aligned}\mathbf{Pr}[\text{Some } v \text{ in } V \text{ rejects }] \leq\ & \mathbf{Pr}[\text{Some } v \text{ in } V(F_{b_2}) \cup \{y_B^2, x_A^2\} \text{ rejects at } G_{22}] \\ & + \mathbf{Pr}[\text{Some } v \text{ in } V(F_{a_2}) \cup \{y_A^2, x_B^1\} \text{ rejects at } G_{21}] \\ & + \mathbf{Pr}[\text{Some } v \text{ in } V(F_{b_1}) \cup \{y_B^1, x_A^1\} \text{ rejects at } G_{11}] \\ & + \mathbf{Pr}[\text{Some } v \text{ in } V(F_{a_1}) \cup \{y_A^1, x_B^2\} \text{ rejects at } G_{12}] \\ & < \frac{4}{5}.\end{aligned}$$

As each term represents a portion of the graph that computes its decision according to the vicinity that it sees, each term is bounded by $1/5$. From here we have a contradiction as, by $G(a_1, b_1, a_2, b_2)$ being a bad instance, the acceptance probability should be smaller than $1/5$.

Now, for the classes PROPER INTERVAL, INTERVAL, PROPER CIRC-ARC , CIRCULAR-ARC and CHORDAL suppose we have a protocol $\mathcal{P}$ in dMA with acceptance probability $\varepsilon \leq \frac{1}{7}$. We repeat again the construction $G(F_a, F_b, c, d)$ with $(a, b, c, d) \in A \times B \times C \times D$, as well as the 4-partite hyper-graph $\tilde{G} = A \cup B \cup C \cup D$. Using the result of Lemma 1.14 we obtain a monochromatic $K^{(4)}(2)$ graph colored by the proof $\bar{m}$. Now in order to study the acceptance probability of $G(a_i, b_i, c_i, d_i)$, we define G_{ijkh} to be the graph $G(a_i, b_j, c_k, d_h)$, from here we bound the acceptance probability as

$$\begin{aligned}\mathbf{Pr}[\text{Some } v \text{ in } V \text{ rejects}] \leq\ & \mathbf{Pr}[\text{Some } v \text{ in } V(F_{b_2}) \cup \{y_B^2\} \text{ rejects at } G_{2222}] \\ & + \mathbf{Pr}[\text{Some } v \text{ in } V(F_{a_2}) \cup \{y_A^2, c_2\} \text{ rejects at } G_{2221}] \\ & + \mathbf{Pr}[\text{Some } v \text{ in } V(F_{b_1}) \cup \{y_B^1\} \text{ rejects at } G_{1111}] \\ & + \mathbf{Pr}[\text{Some } v \text{ in } V(F_{a_1}) \cup \{y_A^1, c_1\} \text{ rejects at } G_{1112}] \\ & + \mathbf{Pr}[\text{Some } v \text{ in } \{d_1\} \text{ rejects at } G_{2121}] \\ & + \mathbf{Pr}[\text{Some } v \text{ in } \{d_2\} \text{ rejects at } G_{1212}] \\ & < \frac{6}{7}\end{aligned}$$

Once again each term represents a portion of the graph that accepts with good probability (at least $6/7$) as any combination of the above vicinities is considered in the monochromatic $K^{(r)}(\ell)$. Finally, we have a contradiction on $\mathcal{P}$'s correctness as $G(a_i, b_i, c_i, d_i)$ is a bad instance that should accept with probability smaller than $1/7$.

□

We can extend this same technique for the lower-bound construction for the class

d-DEGENERATE described in subsection 2.1. By considering of the acceptance probabilities of each node block, these leads to a $\Omega(\log n)$ lower bound for dMA in both shared and private randomness.

Corollary 4.10 *If* d-DEGENERATE $\in \mathsf{dMA}^{\text{pub}}[f(n)]$, *then* $f(n) = \Omega(\log n)$.

4.4 A lower bound for Symmetry

In this section, we extend a lower bound previously constructed by [KOS18] for the problem SYMMETRY. That is, the class of graphs that have a non-trivial automorphism. This result shows the first $\mathsf{dAM}^{\text{priv}}$ lower bound for any fixed k which complements the original lower bound by Oshman et al. [KOS18], as well as complementing the $\mathsf{dAM}^{\text{pub}}$ lower bound to be described later in Section 5.1. As a particular result, we can study the problem DSYM corresponding to the problem SYMMETRY restricted to the family of *dumbbell graphs*, which are the graphs defined as in the previous construction. It is known that this class is in $\mathsf{dM}[\Omega(n^2)]$ [GS16](making it the hardest problem in this class) and $\mathsf{dAM}^{\text{priv}}[\Omega(\log\log n)]$ by the previous construction, as well as admitting a $\mathsf{dAM}^{\text{pub}}[\mathcal{O}(\log n)]$ protocol described in [KOS18]. It was shown in [NPY20] that by a clever use of their compiler such a protocol can be pushed to obtain a $\mathsf{dMAMAM}^{\text{priv}}[\mathcal{O}(\log\log n)]$, surpassing the $\log n$ barrier that appears natural in most problems. This shows that the cost on the result by Naor et al. is tight and can not be improved no matter the number of additional rounds considered.

We describe this construction with the proper adjustments.

Theorem 4.11 *For any integer* k, *if* SYMMETRY $\in \mathsf{dAM}^{\text{priv}}[k, f(n)]$, *then* $f(n) = \Omega\left(\frac{\log\log n}{k}\right)$.

Proof. Consider $\mathcal{G}$ to be a family of n-node asymmetric graphs of size $2^{\Omega(n^2)}$ which is known to exist [ER63]. Now, given $F_A, F_B \in \mathcal{G}$ we define $G(F_A, F_B)$ as follows.

We connect F_A from an arbitrary node, to F_B through a 4-path which we denote by $\{v_A, x_A, x_B, v_B\}$. Now, set $k \leq 2$ and consider $\mathcal{P}$ to be a $\mathsf{dAM}^{p}[k]$ protocol for SYMMETRY with bandwidth cost L. Without loss of generality we may assume that the protocol $\mathcal{P}$ is *simple*, in the sense that all nodes in the 4-path $\{v_A, x_A, x_B, v_B\}$ receive the same proof at any round of the protocol. Indeed, it suffices to send the concatenation of each node in the path in order, where each of them extracts its portion of the proof for themselves and proceed accordingly, thus increasing the proof size by a factor of 4.

Given $\ell = \lceil \frac{k}{2} \rceil$, set $F \in \mathcal{G}$, and consider $r \in \{0,1\}^{n \times L \cdot (\ell - 1)}$, $m \in \{0,1\}^{L \cdot \ell}$. We say that a proof m is (F, r) consistent for $\mathcal{P}$ if m can be seen as a sequence of ℓ proofs given by Merlin to the node x_A in the graph $G(F, F)$, where m_i is the answer to the messages $\{m_j\}_{j<i}$ sent by x_A with random coins $\{r_j\}_{j<i}$ sent by the whole graph, such that m

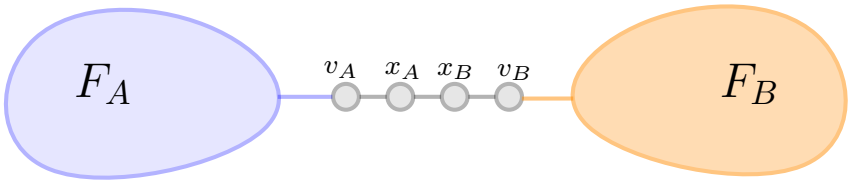

Figure 4.11: Lower-bound construction for SYMMETRY. Two asymetric graphs F_A and F_B are connected by a 4 node path. The graph obtained is in SYMMETRY if and only if $F_A = F_B$.

admits an extension to a set of proofs for F_A, where all nodes in V_A accept. With this, we can define the set

$$\mathcal{C}_A(F, r) = \{m \in \{0,1\}^{L \cdot \ell} \ : m \text{ is } (F, r) \text{ consistent for } \mathcal{P}\}$$

and define $\mathcal{C}_B(F, r)$ in a similar manner. Notice that $m \in \mathcal{C}_A(F_A, r) \cap \mathcal{C}_B(F_B, r)$ implies that all nodes accept in $G(F_A, F_B)$ when the random coin r is used.

Remark Given these sets we can characterize the acceptance probability of $\mathcal{P}$ over the graphs $G(F_A, F_B)$ as:

$$\max_{\mathcal{C}} \mathbf{Pr}(\mathcal{V} \text{ accepts } (G(F_A, F_B), \mathsf{id}) \text{ given } \mathcal{M}) = \mathbf{Pr}(\mathcal{C}_A(F_A, r) \cap \mathcal{C}_B(F_B, r) \neq \emptyset).$$

We may conclude then, by the correctness of $\mathcal{P}$ that, for all $F_A, F_B \in \mathcal{G}$:

- If F_A equals F_B, then $\mathbf{Pr}(\mathcal{C}_A(F_A, r) \cap \mathcal{C}_B(F_B, r) \neq \emptyset) \geq \frac{2}{3}$
- If F_A differs from F_B, then $\mathbf{Pr}(\mathcal{C}_A(F_A, r) \cap \mathcal{C}_B(F_B, r) \neq \emptyset) \leq \frac{1}{3}$

Now, notice that, as $\mathcal{C}(F, r)$ is in the power set of $\{0,1\}^{L \cdot \ell}$, there are at most $d = 2^{2^{L \cdot \ell}}$ possible values for $\mathcal{C}(F, r)$. Therefore, if we consider $\mu_A(F_A)$ to be the distribution of $\mathcal{C}(F, r)$ over r, we can picture $\mu_A(F_A)$ as a unit vector in $(\mathbb{R}^d, \|\cdot\|_1)$.

By the previous remark, we then have that any pair of vectors in $\{\mu_A(F)\}_{F \in \mathcal{G}}$ are at distance at least $2/3$ from each other, as there exists an event Q such that, for $F \neq F'$, $|\mu_A(F)[Q] - \mu_A(F')[Q]| \geq \frac{1}{3}$.

Finally, we consider the following lemma that we include without proof [KOS18].

Lemma 4.12 *Let $\mathcal{U}$ be a collection of vectors in $\mathbb{R}^d$ where for every pair $x \neq x' \in \mathcal{U}$ we have that $\|x - x'\| \geq \frac{1}{2}$, then $|\mathcal{U}| \leq 5^d$.*

By applying the previous lemma to the collection $\{\mu_A(F)\}_{F \in \mathcal{G}}$ we have that $|\mathcal{G}| \leq 5^d$, and considering that $|\mathcal{G}| \geq 2^{\Omega(n^2)}$, it follows that $L = \Omega\left(\frac{\log \log n}{k}\right)$.

□

Chapter 5

Shared versus Private Randomness

We take a step back from looking at protocols and focus on the intricacies of the model. In this chapter, we look at the two possible types of randomness used in the network to see how they relate to each other in terms of power and how the last interaction can affect this result. In Section 5.1 we show that any interactive protocol using shared randomness can be derandomized into a non-interactive proof, with an exponential-factor overhead in the bandwidth. Roughly, we prove that, if $\mathcal{L} \in \mathsf{dAM}^{\text{pub}}[k, f(n)]$, then $\mathcal{L} \in \mathsf{dM}(2^{O(k\, f(n))} + \log n)$. From this we conclude many lower bounds. For instance, we can conclude that SYMMETRY $\in \mathsf{dAM}^{\text{pub}}[k, \Omega(\log n)]$, for any fixed k. This result is tight, because it is already known that SYMMETRY $\in \mathsf{dMAM}^{\text{pub}}[\log n]$ (in fact, it is known that SYMMETRY $\in \mathsf{dMAM}^{\text{priv}}[\log n]$ [KOS18], but the private coin protocol can be easily adapted to work with shared randomness). Then, we show how to turn a k-round dAM protocol using shared randomness where the nodes accept with high probability into a protocol with perfect completeness, with an addition of an extra round of interaction in case that k is even.

Later, in Section 5.2, we separate the models with private and shared randomness through the language AMOS, which is the language of labeled graphs having at most one selected node. More precisely, AMOS is the language of n-node graphs with labels in $\{0, 1\}$, and where at most one node is labeled 1. In [FMO+19] it is shown AMOS is *easy* for private-coin Arthur-Merlin protocols, as AMOS $\in \mathsf{dAM}^{\text{priv}}[1]$. We prove that AMOS $\in \mathsf{dAM}^{\text{pub}}[k, \Theta(\log \log n)]$ and hence there exists an unbounded gap between the two models.

Interestingly, regarding private and shared randomness, roles are reversed when we address dMA protocols instead of dAM protocols. In fact, in Section 5.3, we get an analogous result to that in [CFP19] by proving that dMA protocols with shared randomness are more powerful than dMA protocols with private randomness. More precisely, if $\mathcal{L} \in \mathsf{dMA}^{\text{priv}}_{\varepsilon}[f(n)]$, then $\mathcal{L} \in \mathsf{dAM}^{\text{pub}}_{\varepsilon+\delta}[f(n) + \log n + \log(\delta^{-1})]$. We then separate the two classes. We introduce another language denoted 2-COL-EQ, which consists of graphs with n-bit labels corresponding to proper 2-colorings. In other words, the language consists of bipartite graphs where each part is colored with an n-bit label. We show that 2-COL-EQ

separates shared and private randomness on distributed Merlin-Arthur protocols. More precisely, we show first that 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{pub}}[\log n]$. Then, we show that, for $\varepsilon < 1/4$, 2-COL-EQ $\in \mathsf{dAM}^{\mathrm{priv}}_{\varepsilon}[\Theta(\sqrt{n})]$.

5.1 The Limits of Shared Randomness

In this section we show that the largest possible gap between non-interactive proofs and interactive proofs with shared randomness is exponential. More precisely, we show that any interactive protocol using shared randomness can be derandomized into a non-interactive proof, with an exponential-factor overhead in the bandwidth. From this result we can obtain lower bounds, some of them even tight, for the bandwidth of interactive-proofs with shared randomness.

Theorem 5.1 *Let $k \geq 1$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dAM}^{\mathrm{pub}}[k, f(n)]$. Then, $\mathcal{L} \in \mathsf{dM}[2^{O(k\, f(n))} + \log n]$.*

Proof. Let $\mathcal{P}$ be a protocol deciding $\mathcal{L}$ using shared randomness, k rounds of interaction, bandwidth $f(n)$, and with error probability $1/3$. We use $\mathcal{P}$ to define a protocol $\mathcal{P}'$ for $\mathcal{L}$ with only one round of interaction and bandwidth $2^{\mathcal{O}(k \cdot f(n))} + \log n$. Let us fix (G, id, I), an instance of $\mathcal{L}$.

For a prover $\mathcal{M}$ for protocol $\mathcal{P}$, we define a *transcript* of a node $v \in G$ as a k-tuple $\tau(\mathcal{M}, v) = (\tau_1, \tau_2, \ldots, \tau_k)$ such that $\tau_i \in \{0,1\}^{f(n)}$ is a sequence of bits communicated in the i-th round of interaction of $\mathcal{P}$, for each $i \in \{1, \ldots, k\}$. If both k and i are even, then τ_i is a message that $\mathcal{M}$ sends to node v in the i-th interaction. If k is even and i is odd, then τ_i is a random string drawn from the shared randomness. Finally, roles are reversed in the case where k is odd.

Let us fix $\ell = \lfloor \frac{k}{2} \rfloor$ and let R be the set of all ℓ-tuples $r = (r_1, \ldots, r_\ell)$ such that $r_i \in \{0,1\}^{f(n)}$, for each $i \in \{1, \ldots, \ell\}$. For $v \in G$ and $r \in R$ and a fixed prover $\mathcal{M}$, we call $\tau(\mathcal{M}, v, r)$ the transcript $\tau(\mathcal{M}, v)$ such that $\tau_{2i-1} = r_i$ when k is even and $\tau_{2i} = r_i$ otherwise, for each $i \in \{1, \ldots, \ell\}$. In words, $\tau(\mathcal{M}, v, r)$ is the transcript of the protocol, when the nodes draw the random strings from r.

We can construct a one-round protocol $\mathcal{P}'$, where the prover sends to each node v the following certificate:

1. A spanning tree T given by the id of a root ρ, the parent of v in the tree, denoted by t_v, and the distance in T from ρ to v, given by d_v.
2. The list $m_v = \{m_r^v\}_{r \in R}$, where $m_r^v \in \{0,1\}^{kf(n)}$ is interpreted as $\tau(\mathcal{M}, v, r)$.
3. A vector $\mathsf{acc}(v) \in \{0,1\}^{|R|}$ where $\mathsf{acc}(v)_r$ indicates that u accepts in the transcript given by m_r^u, for all u in the subtree T_v associated to v.

Given the messages received from the prover, the nodes first verify the consistency of the tree given by **(1)**, following the spanning tree protocol given in [KKP10]. Then, each node v checks that for each $r \in R$ the given transcript m_v^r is consistent with r. Then, for each $r \in R$, each node simulates the k rounds of protocol $\mathcal{P}$ using the certificates of its neighborhood, and decides whether to accept or reject. That information is stored in a vector $a^v \in \{0,1\}^{|R|}$. In order to check the consistency of the vector $\mathsf{acc}(v)$, for each $r \in R$ we say that $\mathsf{acc}(v)_r = 1$ if and only if $a_r^v = 1$ and $\mathsf{acc}(u)_r = 1$ for every children u in T_v. If all previous conditions are satisfied and v is not the root, then v accepts. Finally, the root ρ verifies previous conditions and counts the number of accepting entries in $\mathsf{acc}(\rho)$ and accepts if they are at least two-thirds of the total. In any other case, the nodes reject.

The number of bits sent by the prover is: $\mathcal{O}(\log n)$ in **(1)**, $(kf(n)) \cdot 2^{\mathcal{O}(k \cdot f(n))} = 2^{\mathcal{O}(k \cdot f(n))}$ in **(2)** and $2^{\mathcal{O}(k \cdot f(n))}$ in **(3)**. So, in total, the number of bits communicated in any round is $2^{\mathcal{O}(k \cdot f(n))} + \log n$. We now explain the completeness and soundness.

- **Completeness.** If an instance (G, id, I) is in $\mathcal{L}$, an honest prover will send the real answers that each node would have received in the k-round protocol. In such a protocol, all nodes accept for at least two-thirds of the coins, therefore the root accepts.

- **Soundness.** Suppose now that (G, id, I) is not in $\mathcal{L}$, and suppose by contradiction that there exist a prover $\tilde{\mathcal{M}}$ of protocol $\mathcal{P}'$ accepted by all nodes. Let m_v be the certificate that $\tilde{\mathcal{M}}$ gives to node v given by **(2)**. Now, let $\hat{\mathcal{M}}$ be a prover of $\mathcal{P}$ such that $\tau(\hat{\mathcal{M}}, v, r) = m_v^r$, for each $r \in R$. Since the root accepts, all nodes must accept two thirds of the transcripts, which contradicts the soundness of $\mathcal{P}$.

□

A direct consequence of previous result is the transfer of lower bounds from non-determinism to distributed interactive protocols with shared randomness.

Corollary 5.2 *Let $k \geq 1$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dM}[\Omega(f(n))]$. Then, $\mathcal{L} \in \mathsf{dAM}^{\text{pub}}[k, \Omega(\frac{\log f(n)}{k})]$.*

Corollary 5.3 *Let $k \geq 1$. Then, problems* SYMMETRY, DIAMETER, $\overline{\text{3-COL}}$ *and* Δ-FREE$\in \mathsf{dAM}^{\text{pub}}[k, \Omega(\log n)]$. *Also,* MST $\in \mathsf{dAM}^{\text{pub}}[k, \Omega(\log \log n)]$.

Proof. We just need to apply already known lower bounds: SYMMETRY $\in \mathsf{dM}[\Omega(n)]$ from [GS16], DIAMETER $\in \mathsf{dM}[\Omega(n)]$ from [CHPP20], $\overline{\text{3-COL}} \in \mathsf{dM}[\Omega(n)]$ from [GS16], $\triangle$-FREE $\in \mathsf{dM}[\Omega(n)]$ from [CFP19], MST $\in \mathsf{dM}[\Omega(\log^2 n)]$ from [KK07]. □

Remark The lower bound saying that SYMMETRY $\in \mathsf{dAM}^{\text{pub}}[k, \Omega(\log n)]$ is tight. More precisely, the $\mathsf{dMAM}^{\text{priv}}[\log n]$ protocol given by Kol, Oshman and Saxena [KOS18] for solving SYMMETRY can be easily adapted to work with shared randomness. In fact, the

protocol is somehow designed in that way, where one particular node generates the random string and shares it with the other nodes (through Merlin). Therefore, SYMMETRY $\in$ $\mathsf{dMAM}^{\text{pub}}[\Theta(\log n)]$. On the other hand, SYMMETRY $\in \mathsf{dM}[\Omega(n^2)]$ [GS16].

In the proof of Theorem 5.1, in order to design a dM protocol, we had to construct a spanning tree for verifying that two thirds of all coins are accepted by *all nodes*. In fact, it could be the case that, for negative instances, every node rejects a very small portion of the coins, getting the wrong idea that the instance is positive. For avoiding that, and coordinating the nodes, in the dM protocol we construct a spanning tree. This is where the additive $\log n$ term comes from. The next result states that the previous situation does not occur if, instead of two thirds, we ask the interactive protocol to accept *with high probability*.

Theorem 5.4 *Let $k \geq 1$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dAM}^{\text{pub}}_{\varepsilon}[k, f(n)]$, with $\varepsilon < \frac{1}{n+1}$. Then, $\mathcal{L} \in \mathsf{dM}[2^{\mathcal{O}(k\, f(n))}]$.*

Proof. Let $\mathcal{L}$ be a language over instance (G, id, I) with $n(G) = n$ and let $\mathcal{P}$ be a k round protocol for $\mathcal{L}$ with $f(n)$ bits such that its acceptance error is less than $\frac{1}{m}$, where $m > n + 1$.

In order to construct the protocol, we proceed in a similar way as in the proof of the previous theorem: Merlin sends to each node an enumeration m_r^v of the answers to each possible coin that the original prover sends to each node. Then, all nodes share their certificates to each coin sequence, simulate all of them and each node v accepts iff $(1 - \frac{1}{m})$ of the coin sequences are accepted by him.

We have that if G is a *yes instance*, an honest prover will return the answers to the original protocol and all nodes will accept $(1 - \frac{1}{m})$ of all possible coins, therefore all accept the protocol.

If G is a *no instance* and all nodes accept the protocol, we have that all nodes reject at most $\frac{2^{k\, f(n)}}{m}$ coin sequences, therefore the total amount of coins that are rejected by some node is at most $\frac{n 2^{k\, f(n)}}{m}$, which is strictly less than $(1 - \frac{1}{m})2^{k\, f(n)}$. □

Corollary 5.5 *Let $k \geq 1$ and let $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dM}[\Omega(f(n))]$. Then, $\mathcal{L} \in \mathsf{dAM}^{\text{pub}}_{\varepsilon}[k, \Omega(\frac{\log f(n)}{k})] = \mathsf{dAM}^{\text{pub}}_{\varepsilon}[k, \Omega(\log f(n))]$, with $\varepsilon < \frac{1}{n+1}$.*

Corollary 5.6 *Let $k \geq 1$. Then, problems* PLANAR, OUTERPLANAR, SPANNING-TREE *$\in \mathsf{dAM}^{\text{pub}}_{\varepsilon}[k, \Omega(\log \log n)]$, with $\varepsilon < \frac{1}{n+1}$.*

Proof. All these languages belong to $\mathsf{dM}[\Theta(\log n)]$ [GS16, FFR+20]. □

A know result in (centralized) AM protocols is that a public coin protocol can be assumed to have perfect completeness by an addition of an extra round of interaction, here we show how to extend this fact to the distributed setting whenever shared randomness is considered.

Theorem 5.7 *If $\mathcal{L} \in \mathsf{dAM}^{\text{pub}}_{\varepsilon}[k, f(n)]$ then $\mathcal{L} \in \mathsf{dAM}^{\text{pub}}[k', \frac{k \cdot f(n)^2}{\log(1/\varepsilon)}]$ with perfect completeness, where $k' = k + 1$ if k is even and $k' = k$ otherwise.*

Proof. We directly prove the case when k is odd, as the case when k is even is slightly simpler. Let $\mathcal{P}$ be a protocol deciding $\mathcal{L}$ using shared randomness, k rounds of interaction, bandwidth L and error probability ε. We use $\mathcal{P}$ to define a protocol $\mathcal{P}'$ for $\mathcal{L}$ using the same amount of interactions and perfect completeness, while increasing the bandwidth in a constant factor. Let us fix $\langle G, \mathsf{id}, I\rangle$, an instance of $\mathcal{L}$. Set $\ell = \lfloor \frac{k}{2} \rfloor$ and given a prover $\mathcal{M}$ for protocol $\mathcal{P}$ define $S_G(\mathcal{M})$ as the set of ℓ-tuples $(r_1, \dots r_\ell)$, where each r_j is in $\{0,1\}^{f(n)}$ such that for each $i \in [\ell]$ Merlin sends at round $2i-1$ a proof $m \in \{0,1\}^{f(n)}$ for each node that depends on all previous messages making all nodes accept.

We show by the probabilistic method that there exists a collection $\mathsf{Z} = \{z_i\}_{i=1}^{t}$ of coins (with $t = t(\varepsilon, n)$ to be set after) such that any element in $\{0,1\}^{\ell \cdot f(n)}$ can be obtained as a "translation" of $S_G(\mathcal{M})$ by some element in Z. That is, $\cup_i S_G(\mathcal{M}) \oplus z_i = \{0,1\}^{\ell \cdot f(n)}$ with $\oplus$ corresponds to adding two elements by bit-wise XOR.

Indeed, consider a collection Z of t elements chosen all independently and uniformly at random and study the event when this translation of $S_G(\mathcal{M})$ according to Z does not cover all coins possibly drawn during the protocol. If $\langle G, \mathsf{id}, I\rangle$ is a Yes-instance for $\mathcal{L}$, we have that $|S_G(\mathcal{M})| \geq (1-\varepsilon)2^L$. Then, if we define R to be the set of all possible random sequences drawn during a protocol

$$\begin{aligned}
\mathbf{Pr}_{\mathsf{Z}}[\exists r \notin \cup_{z \in \mathsf{Z}} S_G(\mathcal{M}) \oplus z] &= \mathbf{Pr}_{\mathsf{Z}}[\exists r \in \cap_i \overline{S_G \oplus z_i}] \\
&\leq \sum_{r \in R} \prod_{i=1}^{t} \mathbf{Pr}_{z_1}[r \in \overline{S_G \oplus z_1}] \\
&= \sum_{r \in R} \prod_{i=1}^{t} \mathbf{Pr}_{z_1}[r \oplus z_1 \notin S_G] \\
&\leq 2^{k \cdot f(n)} \varepsilon^t
\end{aligned}$$

Then, if we set $t > k \cdot f(n)/\log(1/\varepsilon)$ we have that $2^{k \cdot f(n)}\varepsilon^t$ is strictly smaller than 1. Therefore there exists a set Z satisfying our condition. Now, consider the case when $\langle G, \mathsf{id}, I\rangle \notin \mathcal{L}$, it follows then by $\mathcal{P}$'s correctness that $|S_G(\mathcal{M})| \leq 2^L \varepsilon$. Then, we have that:

$$\mathbf{Pr}[\cup_{z \in \mathsf{Z}} S_G \oplus z] \leq \sum_{z \in \mathsf{Z}} \mathbf{Pr}[S_G \oplus z] \leq t \cdot \varepsilon$$

Where the above probability is taken over all possible random sequences drawn. As for

any z a translation by it does not change the size of S_G, then we just need t to be smaller than $1/3\varepsilon$ for the above probability to be smaller than $\frac{1}{3}$.

And so, if we set $f(n)/\log(1/\varepsilon) \leq t \leq \mathcal{O}(1/\varepsilon)$. We can describe a k round protocol $\mathcal{P}'$ as follows:

Merlin first provides each node with the collection $\{z_i\}_{i=1}^{t}$ (with a cost of $t \cdot \ell \cdot f(n)$), then all nodes proceed through k rounds of interaction simulating protocol $\mathcal{P}$ with a slight modification: Suppose we are at round $2i$ and the verifier has drawn the coins $(r_1, \dots r_i)$, then Merlin at round $2i+1$ answers with t messages $(m_{i,1}, m_{i,2} \dots m_{i,t})$, where $m_{i,j}$ corresponds to the answer Merlin would have sent if the coin sequence $(r_1, \dots r_i) \oplus z_j^{(i)}$ was sent, with $z_j^{(i)}$ the first $i \cdot f(n)$ bits of z_j.

Finally, at the last round, consider $r = (r_1, \dots r_\ell)$ to be the coin sequence sent by the the nodes through all interactions, then Merlin sends in its last message an index $j \in [t]$ and the answer he would have sent if the challenge were $r \oplus z_j$. Then, at the verification round, all nodes check they received the same collection $\{z_i\}_i$ and the same index j and exchange messages running $\mathcal{P}$'s verification procedure on messages $(m_{1,j}, \dots m_{\ell+1,j})$ accepting or rejecting accordingly.

By the previous construction we have that, if $\langle G, \mathsf{id}, I\rangle$ is a Yes-instance, the nodes will always accept and, otherwise, then all nodes accept with probability at most $\frac{1}{3}$. As for the case when k is even, we simply need to define $S_G(\mathcal{M})$ ignoring the initial certificate, then we add an extra round in order for Merlin to provide the collection $\{z_i\}_i$. □

Notice that we require ε to be such that $f(n) \leq \mathcal{O}(1/\varepsilon)/\log(1/\varepsilon)$ which is an easy restriction to fulfill as we are mostly interested in the cases when $f(n)$ is sub-linear. Indeed, even if we had a protocol $\mathcal{P}$ for some class $\mathcal{L}$ with bandwidth $f(n)$ and fixed error probability ε it is known that we can amplify this error by parallel repetition and using standard techniques [CFP19] if we increase the bandwidth to $\mathcal{O}(K \cdot f(n)) + \log n$, as in such a case we can get an error δ with $1/\delta = 2^{\mathcal{O}(K)}$.

A direct application of this result is to consider $f(n) = \mathcal{O}(\log n)$, if we had a protocol that occurs with high probability (i.e. $\varepsilon(n) = 1/n$) then we can push our protocol to perfect completeness with an increase in bandwidth by at most a constant factor.

Corollary 5.8 *Let $\mathcal{L} \in \mathsf{dAM}^{\mathrm{pub}}_{1/n}[k, \mathcal{O}(\log n)]$ then $\mathcal{L} \in \mathsf{dAM}^{\mathrm{pub}}[k', \mathcal{O}(\log n)]$ with perfect completeness, where $k' = k+1$ if k is even and $k' = k$ otherwise.*

5.2 Shared dAM versus Private dAM

A recent result shows that dAM protocols with private randomness are more powerful than dAM protocols with shared randomness [CFP19]. The precise result corresponds to next proposition.

Proposition 5.9 ([CFP19]) *Let $k \geq 1$, $0 < \varepsilon < \frac{1}{2}$, and $\mathcal{L}$ be a language such that $\mathcal{L} \in \mathsf{dAM}^{\text{pub}}_{\varepsilon}[k, f(n)]$. Then, $\mathcal{L} \in \mathsf{dAM}^{\text{priv}}_{\varepsilon}[k, f(n) + \log n]$.*

A natural question is whether the two models are equivalent. In this section we give a negative answer. We separate them through the problem AMOS. Recall that AMOS is the language of labeled graphs where at most one node is selected. It is already known that AMOS $\in \mathsf{dM}[\Theta(\log n)]$ [GS16]. Moreover, in [FMO+19] the authors show that adding randomness *after* the nondeterministic round does not help. More precisely, AMOS $\in \mathsf{dMA}^{\text{priv}}_{\varepsilon}[\Omega(\log n)]$, for $0 < \varepsilon < \frac{1}{5}$.

The situation changes dramatically when randomness goes *before* nondeterminism, as explained in the following proposition.

Proposition 5.10 ([FMO+19]) *Let $0 < \varepsilon < \frac{1}{2}$. Then,* AMOS $\in \mathsf{dAM}^{\text{priv}}_{\varepsilon}[\log(\varepsilon^{-1})] = \mathsf{dAM}^{\text{priv}}_{\varepsilon}[1]$.

In the shared randomness framework, we can construct a protocol for AMOS that uses bandwidth $O(\log \log n)$. As we are going to see in Theorem 5.12, this upper bound is indeed tight.

Lemma 5.11 AMOS $\in \mathsf{dAM}^{\text{pub}}[\log \log n]$.

Proof. The protocol is the following. First, each node considers the smallest prime q such that $\log^{c+2} n \leq q \leq 2 \log^{c+2} n$ and constructs a polynomial over the field $\mathbb{F}_q$ associated to its id given by $p_v(x) = \sum_{i \leq \log(\mathsf{id}(v))} \mathsf{bin}_i(\mathsf{id}(v)) \cdot x^i$. Where $\mathsf{bin}_i(m)$ corresponds to i-th bit in the binary representation of m. All nodes generate a random string $s \in \mathbb{F}_q$ using the shared randomness. Then, the prover sends to each node the random evaluation of the selected node v_0. More precisely, $\bar{p} = p_{v_0}(s)$, which is of size $O(\log \log n)$. The nodes first check they received the same value of $\bar{p}$. If a node v is not selected, then it always accepts; otherwise, it accepts if and only if $p_v(s) = p_{v_0}(s)$. If an instance belongs to AMOS, then all nodes accept. Otherwise, there will exist at least two selected nodes that accept, such that their id's polynomial matches on s with probability at most $\frac{1}{\log^c n}$. □

From Corollary 5.5 we conclude that, for every $k \geq 1$, AMOS $\in \mathsf{dAM}^{\text{pub}}_{\varepsilon}[k, \Omega(\log \log n)]$ with $\varepsilon < \frac{1}{n+1}$. In other words, the protocol given in Lemma 5.11 matches the lower bound for all correct protocols that run with high probability. The next theorem shows that the upper bound is matched even when $\varepsilon = \frac{1}{3}$.

Theorem 5.12 *Let $k \geq 1$. Then,* AMOS $\in \mathsf{dAM}^{\text{pub}}[k, \Theta(\log \log n)]$.

Proof. We follow a technique by [GS16] for "glueing" solutions together to form a bad instance, with some modifications. Without loss of generality, we may assume that n is even. Let A be a partition of $[1, n^2]$ into n sets of size n, and let B be a partition of $[n^2 + 1, 2n^2]$ in a similar manner. Let $\mathcal{G}$ be a family of n-node graphs.

Set now $\mathcal{G}_A$ to be the set of labeled graphs in $\mathcal{G}$, with label sets picked from A. Let F_a be a graph in $\mathcal{G}_A$, and let v^* be the node of F_a labeled with the smallest label. Consider the input I_a such that $I_a(v) = 0$ for every node $v \in V(F_a)$ except for the node v^*, for which $I_a(v^*) = 1$. Similarly, we define $\mathcal{G}_B$ as the set of graphs in $\mathcal{G}$ labeled with labels in B. For each graph $F_b \in \mathcal{G}_B$, we define the input I_b such that $I_b(v) = 0$ for all $v \in V(F_b)$.

For $(F_a, F_b) \in \mathcal{G}_A \times \mathcal{G}_B$ let $G(F_a, F_b)$ be the graph defined by the disjoint union of graphs F_a and F_b plus four additional nodes x_A, y_A, x_B, y_B. These nodes are labeled with different numbers in the set $C = [2n^2 + 1, 3n^2]$. Nodes x_A and x_B are both adjacent only to y_A and y_B. Node y_A is adjacent only to x_A, x_B and v_a, where v_a is some node of F_a. Node y_B is adjacent only to x_A, x_B and v_b, where v_b is some node of F_b. Observe that $v^* \neq v_a$ and all nodes in F_a communicate with F_b only through the nodes x_A, y_A, x_B, y_B.

Consider now the input I of $G(F_a, F_b)$ such that $I(v) = I_a(v)$ if $v \in V(F_a)$, $I(v) = 0$ otherwise. Observe that $v^* \in V(F_a)$ is the only node in $V(G(F_a, F_b))$ satisfying $I(v^*) = 1$. Therefore, the graph $G = G(F_a, F_b)$ is a Yes-instance of AMOS.

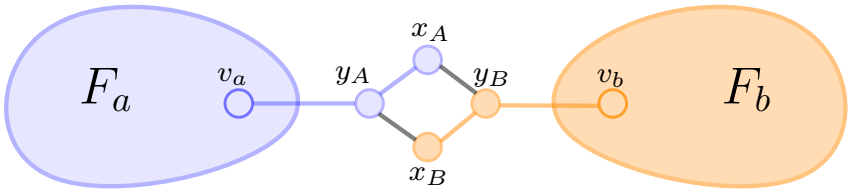

Figure 5.1: The auxiliary graph $G(F_a, F_b)$, with $a \in A$ and $b \in B$. This is a yes-instance for AMOS, as there is only one selected node (in F_a).

Let $\mathcal{P}$ be a k-round distributed interactive proof with shared randomness verifying AMOS with bandwidth $K = \delta \log \log n$ and error probability ε. Let us call $\{x_A, y_A, x_B, y_B\}$ the *bridge* of $G(F_a, F_b)$. We can assume, without loss of generality, that $\mathcal{P}$ is *simple*, that is, it satisfies that for any (shared) random string generated by Arthur, the nodes in the bridge $\{x_A, y_A, x_B, y_B\}$ receive the same proof. Indeed, if we have a protocol that is not simple, with cost L, we can design a new protocol that is simple and whose proof has length $4L$ by making each node pick their portion of the proof and going by the original protocol afterwards.

Given sequence of random strings $r = (r_1, r_2, \dots r_k)$, we call m^r the sequence indexed by nodes $v \in V(G[F_a, F_b])$, such that m^r_v is the set of certificates that Merlin sends to node v in protocol $\mathcal{P}$, when Arthur communicates string r_i on round i. Let $m_{ab} : \{0,1\}^{Kk} \to \{0,1\}^{Kk}$ be the function that associates to each sequence $r = (r_1, r_2, \dots r_k)$ the tuple $(m^r_{x_A}, m^r_{x_B}, m^r_{y_A}, m^r_{y_B})$ such that it extends to a proof assignment for the nodes in both F_a and F_b that make them accept whenever the bridge accepts.

Now consider the complete bipartite graph $\hat{G} = A \cup B$. For each $a \in A$ and $b \in B$,

color the edge $\{a, b\}$ with function m_{ab}. There are at most $2^{Kk2^{Kk}}$ possible functions. Therefore, by an averaging argument, there exists a monochromatic set of edges W of size at least $\frac{n^2}{2^{Kk2^{Kk}}}$.

Observe that for sufficiently small δ and large n, $2^{Kk2^{Kk}} = (\log n)^{\delta k \log^{\delta k}(n)} = o(n^{1/2})$. Indeed, if $n > 2^{k\delta}$ and $\delta < 1/(4k)$ have that $\delta k \log^{\delta k}(n) \log\log(n) \leq \log^{2\delta k}(n) < \frac{1}{2}\log n$. Following a result of Bondy and Simonovits [BS74], we have that there exists a 4-cycle a_1, b_1, a_2, b_2 in the subgraph $\hat{G}$ induced by W.

Consider now the graph $G(a_1, b_1, a_2, b_2)$ defined as follows: First. take a disjoint union of $F_{a_1}, F_{b_1}, F_{a_2}$ and F_{b_2}. Then, for each $i \in \{1, 2\}$ add nodes $x_A^i, x_B^i, y_A^i, y_B^i$, labelled with different labels in $[2n^2 + 1, 3n^2]$ correspondant to the yes instances formed by F_{a_i} and F_{b_j}. For each $i \in \{1, 2\}$, the node y_A^i is adjacent to x_A^i, x_B^{i+1} and the node v_{a_i} of F_{a_i}. Similarly y_B^i is adjacent to the x_A^i, x_B^{i+1} and the node v_{b_i} of F_{b_i}. Where the $i + 1$ is taken mod 2. Finally, define the input I as $I(v) = I_{a_i}(v)$ if v belongs to F_{a_i} and $I(v) = 0$ otherwise.

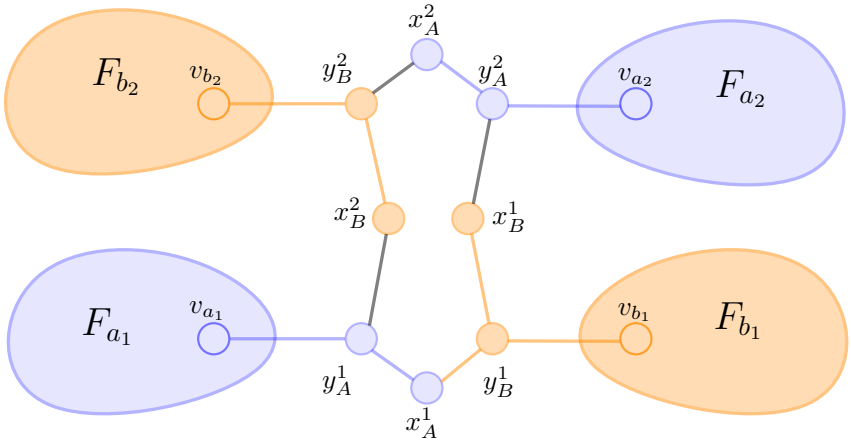

Figure 5.2: A No-instance for AMOS, with messages of size $o(\log n)$. From every node's perspective, the graph behaves like a Yes-instance, but there are two selected nodes.

Observe that $G(a_1, b_1, a_2, b_2)$ is a No-instance of AMOS, as there are two selected nodes: one in $V(F_{a_1})$ and another one in $V(F_{a_2})$. Moreover, all nodes of the set $x_A^1, x_B^1, y_A^1, y_B^1$, $x_A^2, x_B^2, y_A^2, y_B^2$ receive the same answers by Merlin which extends to assignments for the nodes in F_{a_i} and F_{b_i} that make them accept with the same probability as the nodes in the bridge, as they locally place themselves in a previously defined yes instance. Therefore all nodes accept two thirds of all possible random coins. This contradicts the fact that $\mathcal{P}$ was a correct distributed interactive proof for AMOS. □

5.3 Shared dMA versus Private dMA

In this section we first show that, continuing with the comparison between private and shared randomness, roles are reversed when we address dMA protocols instead of dAM protocols. In fact, we get a result analogous to that of Crescenzi, Fraigniaud, and Paz [CFP19] (Proposition 5.9) by proving that dMA protocols with shared randomness are more powerful than dMA protocols with private randomness.

Theorem 5.13 *Let $\varepsilon, \delta > 0$ with $\varepsilon + \delta < \frac{1}{2}$ and consider $\mathcal{L}$ to be a language with inputs of polynomial size over the graph's nodes such that $\mathcal{L} \in \mathsf{dMA}^{\mathrm{priv}}_{\varepsilon}[f(n)]$. Then, $\mathcal{L} \in \mathsf{dAM}^{\mathrm{pub}}_{\varepsilon+\delta}[f(n) + \log n + \log(\delta^{-1})]$.*

Proof. Let $\mathcal{L}$ be a distributed language in $\mathsf{dMA}^{\mathrm{priv}}_{\varepsilon}[f(n)]$ over a network configuration $\mathcal{I} = (G, \mathsf{id}, I)$, and let $\mathcal{P}$ be the protocol that witnesses that membership.

Let $Z(\mathcal{I}, m, r)$ be a random variable that equals 1 if and only if Arthur is wrong about the membership of $\mathcal{I}$ in $\mathcal{L}$ given the proof m and the coin sequence r in an occurrence of the protocol $\mathcal{P}$. Observe that both r and m are a sequence of $n \cdot f(n)$ bits, with the i-th portion of the sequence containing the message sent or received by the node identified as the i-th node of G.

We show by the probabilistic method that there exists a collection $\{r_i\}_{i=1}^{t}$ of random strings such that, for all network configurations $\mathcal{I}$, we can design a correct protocol that only relies on these coins, with a small increase in error.

Indeed, let $\{r_i\}_{i=1}^{t}$ be a collection of random strings of length $f(n)$ and consider a network configuration $\mathcal{I}$ such that $\mathcal{I} \in \mathcal{L}$. We define $Y_{\mathcal{I}}$ to be the following event over all random coin sequences $\{r_i\}_{i=1}^{t}$

$$Y_{\mathcal{I}} = \{\forall m \in \{0,1\}^{n \cdot f(n)} \quad \mathbf{E}_i(Z(\mathcal{I}, m, r_i)) > \varepsilon + \delta\}$$

where $\mathbf{E}_i(\cdot)$ is the expected value over a fixed sequence of random coins considered. We also define, for $\mathcal{I} \notin \mathcal{L}$, the event over a random coin sequences $\{r_i\}_{i=1}^{t}$

$$N_{\mathcal{I}} = \{\exists m \in \{0,1\}^{n \cdot f(n)} \text{ s.t. } \mathbf{E}_i(Z(\mathcal{I}, m, r_i)) > \varepsilon + \delta\}$$

Now, by the correctness of $\mathcal{P}$, we have that for $\mathcal{I} \in \mathcal{L}$ there exists a proof $m_{\mathcal{I}}$ such that Arthur errs with small probability, therefore we have that $\mathbf{E}_r(Z(\mathcal{I}, m_{\mathcal{I}}, r)) \leq \varepsilon$ and by a Chernoff bound:

$$\mathbf{Pr}(Y_{\mathcal{I}}) \leq \mathbf{Pr}\left[\left(\frac{1}{t}\sum_{i=1}^{t} Z(G, m_{\mathcal{I}}, r_i) - \varepsilon\right) > \delta\right] \leq 2\mathrm{e}^{-2\delta^2 t}$$

Where the above probability is taken over all possible random coin sequences $\{r_i\}_{i=1}^{t}$.

Now consider the case when $\mathcal{I} \notin \mathcal{L}$. It follows that, from the correctness of $\mathcal{P}$ we obtain that $\mathbf{E}_r(Z(\mathcal{I}, m_{\mathcal{I}}, r)) \leq \varepsilon$ for any $m \in \{0,1\}^{n \cdot f(n)}$. Thus, by a union bound and another Chernoff bound, we obtain:

$$\mathbf{Pr}(N_{\mathcal{I}}) = \mathbf{Pr}\left[\exists m \in \{0,1\}^{n \cdot f(n)} \text{ s.t. } \left(\frac{1}{t}\sum_{i=1}^{t} Z(\mathcal{I}, m, r_i) - \varepsilon\right) > \delta\right] \leq 2^{n \cdot f(n)} \cdot 2\mathrm{e}^{-2\delta^2 t}.$$

Where the above probability is taken over all possible random coin sequences $\{r_i\}_{i=1}^{t}$. Therefore, if we consider $B = \bigcup_{\mathcal{I} \in \mathcal{L}} Y_{\mathcal{I}} \cup \bigcup_{\mathcal{I} \notin \mathcal{L}} N_{\mathcal{I}}$ to be the union of $Y_{\mathcal{I}}$ over all $\mathcal{I} \in \mathcal{L}$ and of $N_{\mathcal{I}}$ over all $\mathcal{I} \notin \mathcal{L}$ and make a union bound, we get:

$$\mathbf{Pr}(B) \leq \sum_{G \in \mathcal{L}} 2\mathrm{e}^{-2\delta^2 t} + \sum_{G \notin \mathcal{L}} 2\mathrm{e}^{-2\delta^2 t} \cdot 2^{n \cdot f(n)}.$$

Now, remember from the definition of a distributed language that we assume that the set of all possible network configurations defined over graphs of size n is at most $2^{\mathrm{poly}(n)}$. Therefore, if we take $t = \Theta(\frac{\mathrm{poly}(n)+n\cdot g(n)}{\delta^2})$ the probability of B is strictly smaller than one.

And so, there exists a collection $\{r_i\}_{i=1}^t$ such that for any network configuration $\mathcal{I}$ defined over graphs of n nodes:

$$\begin{aligned}
\mathcal{I} \in \mathcal{L} &\implies \exists m\ \mathbf{E}_i\left(Z(\mathcal{I}, m, r_i)\right) \leq (\varepsilon + \delta) \Rightarrow\ \exists m\ \mathbf{Pr}(Z(\mathcal{I}, m, r_i) = 0) > 1 - (\varepsilon + \delta) \\
\mathcal{I} \notin \mathcal{L} &\implies \forall m\ \mathbf{E}_i\left(Z(\mathcal{I}, m, r_i)\right) \leq (\varepsilon + \delta) \Rightarrow\ \forall m\ \mathbf{Pr}(Z(\mathcal{I}, m, r_i) = 0) > 1 - (\varepsilon + \delta)
\end{aligned}$$

Now we can describe a $\mathsf{dMA}^{\mathrm{pub}}$ protocol for $\mathcal{L}$: Merlin sends Arthur the proof m that he would send in protocol $\mathcal{P}$. Then, Arthur proceeds to draw a random integer $i \in [t]$. Then, all nodes take their portion of r_i and proceed with the protocol $\mathcal{P}$ using proof m and r_i. The completeness and soundness of the protocol is guaranteed by the choice of the set $\{r_i\}_{i=1}^t$. The total bandwidth of the protocol is $f(n) + \log t = f(n) + \mathcal{O}(\log(n) + \log(\delta^{-1}) + \log(f(n)))$ bits. We deduce that $\mathcal{L}$ belongs to $\mathsf{dMA}^{\mathrm{pub}}[f(n) + \log n + \log(\delta^{-1})]$.

□

As we did in the previous section for dAM protocols, we are going to give here a negative answer to the question whether $\mathsf{dMA}^{\mathrm{pub}}$ and $\mathsf{dMA}^{\mathrm{priv}}$ are equivalent models. For obtaining such separation, we use the problem 2-COL-EQ. Recall that this language is the set of network configurations (G, id, I), where I is a function $I : V(G) \to \{0,1\}^n$, such that I is a proper two-coloring of G. In other words, (G, id, I) belongs to 2-COL-EQ if and only if there is a partition $\{V_0, V_1\}$ of $V(G)$, such that both V_0 and V_1 are independent sets and, for all $v, w \in V_i$, we have that $I(v) = I(w)$, for $i \in \{0,1\}$.

The next lemma shows that 2-COL-EQ is "easy" to solve using shared randomness.

Lemma 5.14 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{pub}}[\log n]$.

Proof. The prover sends each node a bit $c_v \in \{0,1\}$ to denote the coloring. Then each node considers a prime q such that $n^{c+2} \leq q \leq 2n^{c+2}$ and constructs a polynomial associated to $I(v)$ given by $p_v(z) = \sum_{i=1}^n I(v)_i\, z^i$, during the verification round, all nodes generate a random string $s \in \mathbb{F}_q$ using the shared randomness, and communicate $p_v(s)$. In the verification round, each node locally verifies the consistency of the 2-coloring and that $p_u(s)$ equals $p_w(s)$, for every pair of neighbors u, w. Accepting if both conditions are satisfied and rejecting otherwise. In total, the bandwidth of the protocol is $\mathcal{O}(\log n)$ bits.

- **Completeness.** If (G, id, I) is a yes-instance of 2-COL-EQ, then the input graph is bipartite, and $I(u) = I(v)$ for each u, v in the same partition. Obviously in this case $p_u(s) = p_v(s)$, and therefore the nodes always accept.

- **Soundness.** If G is not bipartite the nodes immediately reject because in this case the coloring c is not consistent (i.e. $c_v = c_u$ for two adjacent nodes u, v). Suppose now that G is bipartite with partitions V_0 and V_1, and suppose without loss of generality that there are two nodes u, v in V_0 such that $I(u) \neq I(v)$. Observe that, since G is connected, we can choose u, v with a common neighbor $w \in V_1$. Then w receives $p_u(s)$ and $p_v(s)$ in the verification round. Observe that the probability that $p_u(s) = p_v(s)$ is at most n/q. Indeed, $p(z) = p_u(z) - p_v(z)$ is a polynomial of degree at most n in $\mathbb{F}_q$, and then it has at most n roots in $\mathbb{F}_q$. We conclude that w fails to verify $I(u) \neq I(v)$ with probability at most $1/n^{c+1}$.

We deduce that 2-COL-EQ belongs to $\mathsf{dMA}^{\mathrm{pub}}[\log n]$. □

The goal now is to prove that 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{priv}}[\Theta(\sqrt{n})]$. We divide this proof in two subsections: the first for the upper bound and the second for the lower bound.

5.3.1 The upper bound

Babai and Kimmel devise a private coin, randomized protocol in the simultaneous messages model (SM) that solves EQUALITY communicating $\mathcal{O}(\sqrt{n})$ bits [BK97]. Problem EQUALITY consists in deciding whether two n-bit Boolean vectors, the inputs of Alice and Bob, are equal.

By using the protocol of Babai and Kimmel, we can directly construct a $\mathsf{dMA}^{\mathrm{priv}}$ protocol for 2-COL-EQ.

Lemma 5.15 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{priv}}[\sqrt{n}]$.

Proof. The prover sends each node v the bit c_v that defines the 2-coloring and then the nodes proceed to broadcast a message according to the protocol in Proposition 1.6. Then, they locally verify the consistency of the 2-coloring and each node w in V_i proceeds to act as referee for each pair of nodes u, v in its vicinity, accepting if for each pair of nodes the referee would accept.

- **Completeness.** If the input corresponds to a Yes-instance, then the nodes always accept: they receive and verify the 2-coloring and for all pairs of neighbors check that the equality protocol holds, as the protocol from Proposition 1.6 has one sided error, the graph accepts with perfect probability.

- **Soundness.** Suppose that the input corresponds to a No-instance. If G is not bipartite the nodes immediately reject because in this case the coloring c is not consistent (i.e. $c_v = c_u$ for two adjacent nodes u, v). Suppose now that G is bipartite with partitions V_0 and V_1, and suppose without loss of generality that there are two nodes u, v in V_0 such that $I(u) \neq I(v)$. Observe that, since G is connected, we can choose u, v with a common neighbor $w \in V_1$. Then, the probability that w accepts is at most ε, where ε is the acceptance error of the protocol described in Proposition 1.6.

□

5.3.2 The lower bound

In order to give a lower-bound on the bandwidth of any $\mathsf{dMA}^{\text{priv}}$ protocol solving 2-COL-EQ, we show that the result of Babai and Kimmel given by Proposition 1.5 can be extended to the scenario where Alice and Bob have access to random bits.

Theorem 5.16 *Let $f : X \times Y \to \{0,1\}$ be any Boolean function. Let $0 < \varepsilon < \frac{1}{2}$. Any ε-error MA^{sym} protocol for solving f using private coins needs the messages to be of size at least $\Omega\left(\sqrt{\mathsf{M}^{\text{sym}}(f)}\right)$.*

Proof. Let $f : X \times Y \to \{0,1\}$ be a boolean function and consider $\mathcal{P}$ to be a MA^{sym} protocol with two sided error ε, where the size of the messages sent by Alice and Bob and the size of the proof are bounded by K.

Let Γ be the set of all possible proofs sent by Merlin. Let Ω and Φ to be the set of all possible messages sent by Alice and Bob, with a and b bits, respectively. Now, given input x and proof m, we define $\mu_{x,m}$ to be distribution of messages sent by Alice given her input and the proof received. We define $\nu_{y,m}$ analogously for Bob.

Now, set $T = \{w_i\}_{i=1}^{t}$ to be a multiset of elements in Ω obtained uniformly at random, with $\mu(T) = \sum_{i=1}^{t} \mu(w_i)$ and define $\rho(\omega, \varphi)$ as the indicator function of whether the referee accepts given messages ω and φ.

We then have that $\mathcal{P}$'s correctness can be restated as follows:

$$f(x,y) = 1 \implies \exists m, \quad \sum_{\omega,\varphi} \mu_{x,m}(\omega) \cdot \nu_{y,m}(\varphi)\rho(\omega,\varphi) \geq 1 - \varepsilon$$

$$f(x,y) = 0 \implies \forall m, \quad \sum_{\omega,\varphi} \mu_{x,m}(\omega) \cdot \nu_{y,m}(\varphi)\rho(\omega,\varphi) \leq \varepsilon$$

Finally, consider the *strength* of φ over x, given the proof m to be defined as

$$F(x,\varphi,m) = \sum_{\omega \in \Omega} \mu_{x,m}(\omega) \cdot \rho(\omega,\varphi)$$

And, given an input $x \in X$, a proof m, φ and a multiset $T = \{w_i\}_{i=1}^{t}$ we define the variables

$$\xi_i(\varphi, m) = \begin{cases} 1 & \text{if the referee accepts } (\omega_i, \varphi) \text{ given } m \\ 0 & \text{if not} \end{cases}$$

Remark The variables $\xi_i(\varphi, m)$ are independent and their expected value is $F(x, \varphi, m)$

Claim 5.17 *For every input x and proof m there exists a multiset $T_{x,m} = \{w_1, \dots w_t\}$ with $t = O(\log(|\Phi|))$ such that for any $\varphi \in \Phi$.*

$$\left| \sum_{i=1}^{t} \xi_i(\varphi, m) - t \cdot F(x, \varphi, m) \right| \leq \delta \cdot t$$

Proof. Indeed, by choosing T uniformly at random, we define the event $\Lambda(\varphi, m, T)$ as $\left\{ |\sum_{i=1}^{t} \xi_i(\varphi, m) - t \cdot F(x, \varphi, m)| > \delta \cdot t \right\}$. Following the previous Remark, we obtain by a Chernoff bound that:

$$\mathbf{Pr}_T \left(\Lambda(\varphi, m, T_x) \right) < 2 \cdot \mathrm{e}^{-(\delta \cdot t)^2 / 2t} = 2\mathrm{e}^{-t\delta^2/2} < \frac{1}{2|\Phi|}$$

By taking a large enough constant for t. Then

$$\mathbf{Pr}_T(\exists \varphi \text{ s.t. } \Lambda(\varphi, m, T_x)) < 1/2$$

And so, by the probabilistic method, there exists a $T_{x,m}$ such that for any φ we have

$$\left| \sum_{i=1}^{t} \rho_m(\omega_i, \varphi) - t \cdot F(x, \varphi, m) \right| \leq \delta \cdot t$$

□

We construct a collection $\{T_{y,m}\}_{(y,m) \in Y \times \Gamma}$ for each input y for Bob and each proof m in a similar way .

Claim 5.18 *For any x, y and proof m the pair $(T_{x,m}, T_{y,m})$ induces a non deterministic protocol for f.*

Proof. We may first assume, without loss of generality, that the referee's decision is deterministic: for any tuple (ω, φ, m) the referee outputs the most probable answer over his random bits, duplicating the error [NS96]. And so we may consider $\rho_m(\omega, \varphi)$ to be the indicator function over the referee's decision given messages (ω, φ) and the proof m.

For $x \in X$, consider $T_{m,x} = (\omega_1, \dots, \omega_t)$ and for $y \in Y$, $T_{m,y} = (\varphi_1, \dots, \varphi_t)$ the collection of messages obtained by Claim 5.17.

Also, we consider the acceptance probability of the pair x, y given a proof m as

$$F(x, y, m) = \sum_{\omega, \varphi} \mu_{x,m}(\omega) \ \nu_{y,m}(\varphi) \cdot \rho_m(\omega, \varphi)$$

By the definition of $T_{x,m}$ and $T_{y,m}$ we have that:

$$\left| \sum_{i=1}^{t} \rho_m(\omega_i, \varphi) - t \cdot F(x, \varphi, m) \right| \leq \delta \cdot t \ \text{ and } \ \left| \sum_{j=1}^{t} \rho_m(\omega, \varphi_j) - t \cdot F(\omega, y, m) \right| \leq \delta \cdot t$$

with the strength of φ and ω with respect to x and y being defined in the same way as before. And so

$$\sum_{\omega} \mu_{x,m}(\omega)\rho_m(\omega,\varphi) \leq \frac{1}{t}\sum_{i=1}^{t} \rho_m(\omega_i,\varphi) + \delta$$
$$\sum_{\varphi} \nu_{y,m}(\varphi)\rho_m(\omega,\varphi) \leq \frac{1}{t}\sum_{j=1}^{t} \rho_m(\omega,\varphi_j) + \delta$$

this allows to bound the acceptance probability of x and y as:

$$\begin{aligned} F(x,y,m) &\leq \sum_{\omega} \mu_{x,m}(\omega)\left(\frac{1}{t}\sum_{j=1}^{t} \rho_m(\omega,\varphi_j) + \delta\right) \\ &\leq \delta + \left(\sum_{\omega} \mu_{x,m}(\omega)\frac{1}{t}\sum_{j=1}^{t} \rho_m(\omega,\varphi_j)\right) \\ &\leq \delta + \frac{1}{t}\sum_{j=1}^{t}\left(\sum_{\omega} \mu_{x,m}(\omega)\rho_m(\omega,\varphi_j)\right) \\ &\leq 2\delta + \frac{1}{t^2}\sum_{i,j=1}^{t} \rho_m(\omega_i,\varphi_j) \end{aligned}$$

by replicating the above procedure for the other direction we obtain:

$$\left|\sum_{i,j=1}^{t} \rho_m(\omega_i,\varphi_j) - t^2 \cdot F(x,y,m)\right| \leq 2\delta t^2$$

In other words, if the referee receives $T_{x,m}$ and $T_{y,m}$ he may estimate the value of $F(x,y,m)$ by a factor of 2δ and accept or reject accordingly.

From here we can define the protocol $\mathcal{P}^*$ simply as follows: Alice and Bob send $T_{x,m}$ and $T_{y,m}$ respectively. Then the referee takes de average answer for all pairs (ω_i,φ_j) given m and accepts if the majority of the cases accept.

- **Completeness.** If (x,y) is a yes-instance, then there exists a proof m such that $F(x,y,m)$ is large ($\geq 1-\varepsilon$) . As we know that $\frac{1}{t^2}|\sum_{i,j=1}^{t} \rho_m(\omega_i,\varphi_j) - t^2 \cdot F(x,y,m)| \leq 2\delta$ we have that $\frac{1}{t^2}\sum_{i,j=1}^{t} \rho_m(\omega_i,\varphi_j) \geq 1-\varepsilon-2\delta$. Therefore by choosing δ sufficiently small the referee accepts $T_{x,m}$ and $T_{y,m}$.

- **Soundness.** If (x,y) is a no-instance, then for any proof m if follows that $F(x,y,m)$ is small ($\leq \varepsilon$). And as we know that $\frac{1}{t^2}|\sum_{i,j=1}^{t} \rho_m(\omega_i,\varphi_j) - t^2 \cdot F(x,y,m)| \leq 2\delta$ then we have that $\frac{1}{t^2}\sum_{i,j=1}^{t} \rho_m(\omega_i,\varphi_j) \leq \varepsilon + 2\delta$. And so the referee rejects $T_{x,m}$ and $T_{y,m}$ for any m.

□

Thus given an $\mathsf{MA}^{\mathrm{sym}}$ protocol for f using $O(K)$ bits we obtained a $\mathsf{M}^{\mathrm{sym}}$ protocol that uses $O(K^2)$ bits. Therefore $K = \Omega\left(\sqrt{\mathsf{M}^{\mathrm{sym}}(f)}\right)$. □

Lemma 5.19 *If* 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{priv}}_{\varepsilon}[f(n)]$ *with* $\varepsilon < 1/4$, *then there exists a protocol* $\mathcal{P}$ *solving* EQUALITY *in the* MA^{sym} *model with bandwidth* $\mathcal{O}(f(n))$.

Proof. Indeed, let $\mathcal{P}$ be a protocol for 2-COL-EQ in the model dMA using random coins. We design a protocol $\mathcal{P}^*$ in the $\mathsf{MA}^{\mathrm{sym}}$ defined as follows. Let $x, y \in \{0,1\}^n$, and assume without loss of generality that n is even. Given $n \in \mathbb{N}$ Alice, Bob and the referee construct the following network configuration (G, id, I):

- G is a path of $2n+1$ nodes $v_1, \ldots, v_{2n+1}$.
- $\mathsf{id}(v_i) = i$ for each $i \in \{1, \ldots, 2n+1\}$.
- $I(v_i) = \begin{cases} 0^n & \text{if } i \text{ is odd} \\ x & \text{if } i \text{ is even and } i \leq n \\ y & \text{if } i \text{ is even and } i > n \end{cases}$

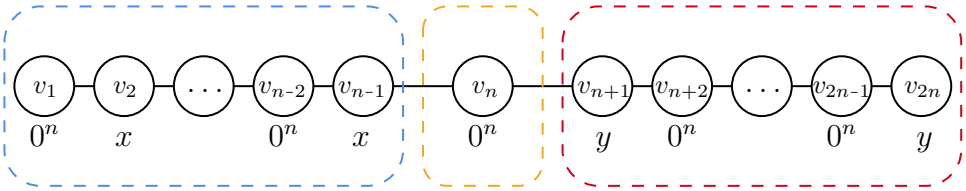

Figure 5.3: An instance (G, id, I) constructed by Alice and Bob: the blue box corresponds to the set of nodes assigned to Alice, along with input x, those in the red box are the ones assigned to Bob, along with the input y while the orange box containing a single node is assigned to the referee, whose input is fixed.

Given the input x for Alice and y for Bob, they proceed to construct the instance (G, id, I): Alice takes the first n nodes of G while Bob takes the last n. Finally, the central node is assigned to the referee.

For each $v \in G$, let $m(v)$ be the certificate that Merlin sends to node v according to protocol $\mathcal{P}$. In protocol $\mathcal{P}^*$, Alice receives from the prover the certificate $(m(v_n), m(v_{n+1}))$, and Bob receives the certificate $(m(v_{n+1}), m(v_{n+2}))$. Then, Alice proceeds to enumerate all possible certificates for the nodes $v_1, \ldots, v_{n+1}$, along with all possible random messages that these nodes may generate, depending on each certificate and input. Likewise, Bob enumerates all possible certificates and random messages for nodes $v_{n+1}, \ldots, v_{2n+1}$.

Having simulated all possible interactions between the nodes in their section and the referee, Alice draws a random string r_1 and communicate the following to the referee the message $(\mathsf{acc}_A, s_n, m(v_{n+1}))$, where:

- $\mathsf{acc}_A \in \{0,1\}$ and equals 1 if and only if all the nodes $v_1, \ldots, v_n$ accept in protocol $\mathcal{P}$ for more than $1-\varepsilon$ of all possible random bits;

- s_n is the message that node v_n sends to v_{n+1} in the verification round of protocol $\mathcal{P}$, given the string r_1, the input x and the certificate $m(v_n)$.

Analogously, Bob draws a random string r_2 and sends the ref the message $(\mathsf{acc}_B, s_{n+2}, m(v_{n+1}))$, such that $\mathsf{acc}_B \in \{0,1\}$ and equals 1 if and only if all the nodes $v_{n+2}, \ldots, v_{2n+1}$ accept in protocol $\mathcal{P}$ for more than $1-\varepsilon$ of all possible random bits; and s_{n+2} is the message that v_{n+2} sends to v_{n+1} in the verification round of $\mathcal{P}$, given r_2, the input y and the certificate $m(v_{n+2})$. We have that both Alice and Bob send $\mathcal{O}(K)$ bits each.

Having these messages, the referee verifies that $\mathsf{acc}_A = \mathsf{acc}_B = 1$ and that both send the same certificate for node v_{n+1}, rejecting if any of these fails. Then the referee draws a random string r_3 and simulates the verification round between the central node v_{n+1}, and nodes v_n and v_{n+2}. Given the messages received from Alice and Bob, the referee has the messages that v_{n+1} receives in the verification round of $\mathcal{P}$. Finally, the referee accepts if v_n accepts. We now analyze the soundness and completeness of $\mathcal{P}^*$.

- **Completeness.** If $x = y$ then (G, id, I) is a *yes*-instance of 2-COL-EQ. By the completeness of $\mathcal{P}$, all the nodes in G accept with probability greater than $1-\varepsilon$. This implies that Alice and Bob communicate $\mathsf{acc}_A = \mathsf{acc}_B = 1$ to the referee. It also implies that v_{n+1} accepts. Therefore, the referee accepts with probability greater than $1-\varepsilon$.

- **Soundness.** In the case that $x \neq y$, we have that, by the correctness of $\mathcal{P}$, the probability that all nodes accept is strictly less than ε.

  Now consider A to be the variable that equals 1 if Alice's portion of the graph accepts when v_n receives r_3 from v_{n+1} and B be the variable that equals 1 if Bob's portion accepts given r_3. As $\mathsf{acc}_A = \mathsf{acc}_B = 1$ we have that

$$\begin{aligned}\mathbf{Pr}(\text{The referee accepts}) = \ & \mathbf{Pr}(\mathsf{A}, \mathsf{B} \text{ and the referee accept}) \\ & + \ \mathbf{Pr}(\text{ The referee accepts and } \mathsf{A} = 0 \text{ or } \mathsf{B} = 0) \\ & < \varepsilon + 2\varepsilon\end{aligned}$$

  As Alice and Bob each reject the protocol with probability less than ε. Therefore, with probability $1 - 3\varepsilon > \frac{3}{4}$ we have that the referee rejects.

Finally, as there is a gap between both acceptance probabilities, the error can be reduced by standard amplification. We conclude that $\mathcal{P}^*$ is a protocol for EQUALITY in the MA^{sym} model with bandwidth $\mathcal{O}(f(n))$. □

Theorem 5.20 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{priv}}_{\varepsilon}[\Theta(\sqrt{n})]$ *for any* $\varepsilon < \frac{1}{4}$ *and* 2-COL-EQ $\in \mathsf{dMA}^{\mathrm{pub}}_{1/3}[\Theta(\log n)]$.

Proof. Indeed, it is known that in the classic 2-party communication model of Alice and Bob the problem EQUALITY has complexity $\Theta(n)$ even with the help of no-determinism [Kus97]. This bound translates naturally to the simultaneous messages model, and so $\mathsf{N}(\text{EQUALITY}) = \Theta(n)$. From Theorem 5.16 we deduce that any protocol in the model $\mathsf{MA}^{\mathsf{sym}}$ for EQUALITY using random bits requires $\Theta(\sqrt{n})$ bits. Now, set $\varepsilon < 1/4$. If there exists a protocol $\mathcal{P}$ for 2-COL-EQ using $o(\sqrt{n})$ bits and with and error smaller than ε, then by Lemma 5.19 there would exist a protocol $\mathcal{P}^*$ for EQUALITY in the model $\mathsf{MA}^{\mathsf{sym}}$ using $o(\sqrt{n})$ bits and error smaller than $1/3$, a contradiction.

Moreover, for every $\varepsilon \leq 1/3$, if 2-COL-EQ belongs to $\mathsf{dMA}^{\mathrm{pub}}_{\varepsilon}[f(n)]$ then $f(n) = \Omega(\log n)$ as we can derandomize the protocol and it would contradict the bound for EQUALITY. Thus by Lemma 5.14 we conclude that the protocol is tight. □

Conclusions and future work

Throughout this work, first we saw how the help of a prover can allow us to locally recognize properties which can be global in nature, or properties that, even while local in nature may require the network to communicate very large messages (such as triangle-free graphs). Moreover, by interacting for several rounds, the prover can provide even more help, allowing us to recognize more graph classes with smaller messages. As such, we obtained protocols for different types of graph classes that are hard to pin down because of their high connectivity such as cographs, distance hereditary graphs, or chordal graphs. As many of them are shown to be tight, we improve on the previous known results in this model, such as the compiler by Naor, Parter and Yogev. Also, we explored how the presence of different information bottlenecks allow us to use different lower bound techniques, combining tools from extremal graph theory (Theorem 4.8) and packing arguments in vector spaces (Theorem 4.11), to obtain $\Omega(\log n)$ or $\Omega(\log \log n)$ lower bounds for the classes dM and $\mathsf{dAM}^{\text{pub}}$ respectively.

Second, we saw how the type of randomness that is used by a protocol can impact greatly on the power of interactive proofs. We showed in Theorem 5.1 how the existence of large lower bounds ($\omega(\log n)$) in the dM model can force an, at most, exponential gap when a larger number of interactions using shared randomness occur. Then, in Sections 5.2 and 5.3, we showed how dAM and dMA protocols can simulate one another depending on the type of randomness that is chosen by paying a small additive cost. Also, we provided separation results between the two variants of distributed interactive proofs by studying two problems (namely AMOS and 2-COL-EQ) that admit a large separation between the costs of using shared and private randomness.

Now, as it is natural for this type of work, several questions arise that can lead to interesting lines of future work. Starting with the design of interactive proofs, a first natural question is whether we can find better protocols for the problems we considered here by either reducing their costs (as Δ-FREE) or reducing the number of rounds of interaction by exploiting some structure in the graph (e.g. COGRAPH DIST-HEREDITARY or CIRCULAR-ARC). On the contrary, it would be interesting to study new types of lower bound techniques for classes that do not admit a small cut (e.g. CLIQUE and COGRAPH) or simply finding a general method for obtaining bounds in the $\mathsf{dAM}^{\text{priv}}$ model, where only a lower bound by a packing argument has been found for the problem SYMMETRY.

Another interesting problem, which has not been considered in this work, is under which conditions we can simulate protocols in other models: We saw in Section 3.3 that

we can simulate a BCC protocol for the problem COGRAPH given its special structure which involves a join between two graphs. Given this highly arbitrary condition we would like to look for some other cases by which we could obtain such a result.

Another model that we are interested in, and which we would naturally compare to, is the Congest model. As described in Theorem 1.9 any protocol in the Congest model where messages are broadcast in nature can be easily simulated by dAM protocols and therefore distributed interactive proofs are simply stronger. As for its unicast variant, it is easy to show that if our graph has bounded degeneracy, we can simulate it by assigning each node the messages that go through each edge that comes with him in an elimination ordering, increasing the cost by a factor of $\mathsf{deg}(G)$. Yet this leaves open the question of whether this is the only case when it is possible, or there exists another condition by which we can do it with a small increase in cost. Finding such a result (or its negative) would be specially interesting as it opens a broader set of tools for us to use in the obtainment of efficient protocols or in the proof of hardness of lower bounds, if we were to follow the routing schemes of Ghaffari et al. [GKS17] or the recent results by Oshman et al. [EFF+19] which would relate the existence of lower bounds to the possibility of simulating some families of circuits in graphs with small mixing time.

Besides the main questions regarding the actual power of dAM and dMA, and a general method to obtain lower-bounds on these models, this work leaves open other interesting research lines regarding the study of the distributed interactive proofs model. First, Theorem 5.1 shows that, if $f = \Omega(\log n)$, then $\mathsf{dAM}^{\text{pub}}[f] \subseteq \mathsf{dM}[2^f]$. Is it possible to obtain such an inclusion even for $f = o(\log n)$? We were able to show that this is indeed the case when the interactive proof is restricted to randomized protocols with low error (see Theorem 5.4), but it is unclear whether this holds in general.

Another question along this line is about the maximum gap between dAM protocols with private and shared randomness. More precisely, is there a language contained in both $\mathsf{dAM}^{\text{pub}}[\Omega(n)]$ and $\mathsf{dAM}^{\text{priv}}[\mathcal{O}(1)]$? Such a result would further increase our separation gap between both variants and give some insight into the structure of problems for which such gap holds.

Finally, we can consider another model variant, which combines the power of shared and private randomness on any round. Namely, a model where the nodes use private randomness to interact with the prover and shared randomness in the verification round. How powerful is this model compared to the one with only private coins?

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABCP92] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Low-diameter graph decomposition is in nc. In *Proceedings of the Scandinavian Workshop on Algorithm Theory*, pages 83–93. Springer, 1992.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1):1–54, 2009.

[BE10] Leonid Barenboim and Michael Elkin. Sublogarithmic distributed mis algorithm for sparse graphs using nash-williams decomposition. *Distributed Computing*, 22(5-6):363–379, 2010.

[BK97] László Babai and Peter G Kimmel. Randomized simultaneous messages: Solution of a problem of Yao in communication complexity. In *Proceedings of Computational Complexity. Twelfth Annual IEEE Conference*, pages 239–246. IEEE, 1997.

[BMRT14] 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 *Proceedings of the International Colloquium on Structural Information and Communication Complexity*, pages 83–95. Springer, 2014.

[BMRT20] 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.

[BNBYF+01] 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*, 48(5):1069–1090, 2001.

[Bod98] Hans L Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[BS74] John A Bondy and Miklós Simonovits. Cycles of even length in graphs. *Journal of Combinatorial Theory, Series B*, 16(2):97–105, 1974.

[BS+99] Andreas Brandstadt, Jeremy P. Spinrad, et al. *Graph classes: a survey*, volume 3. SIAM, 1999.

[C+52] Herman Chernoff et al. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[CDE+13] 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: Google's globally distributed database. *ACM Transactions on Computer Systems*, 31(3):1–22, 2013.

[CDSF01] 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.

[CFP19] Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In *33rd International Symposium on Distributed Computing*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[CHKK+19] 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.

[CHPP20] Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theoretical Computer Science*, 811:112–124, 2020.

[CLF12] 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.

[DGS14] 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.

[DHP01] 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.

[Die06] R. Diestel. *Graph Theory*. Electronic library of mathematics. Springer, 2006.

[DKO14] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *Proceedings of the 2014 ACM symposium on*

Principles of distributed computing, pages 367–376, 2014.

[EFF+17] 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*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[EFF+19] 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*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[EJT10] Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of bodlaender and courcelle. In *IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 143–152. IEEE, 2010.

[ER63] Paul Erdős and Alfréd Rényi. Asymmetric graphs. *Acta Mathematica Academiae Scientiarum Hungarica*, 14(3-4):295–315, 1963.

[Erd64] P Erdös. On extremal problems of graphs and generalized graphs. *Israel Journal of Mathematics*, 2(3):183–190, 1964.

[ERSR12] 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.

[FFR+20] Laurent Feuilloley, Pierre Fraigniaud, Ivan Rapaport, Éric Rémila, Pedro Montealegre, and Ioan Todinca. Compact distributed certification of planar graphs. *arXiv preprint arXiv:2005.05863*, 2020.

[FGKO18] Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and impossibilities for distributed subgraph detection. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 153–162, 2018.

[FH18] Laurent Feuilloley and Juho Hirvonen. Local verification of global proofs. *arXiv preprint arXiv:1803.09553*, 2018.

[FKP13] Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *Journal of the ACM*, 60(5):1–26, 2013.

[FMO+19] 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.

[FOZ16] Orr Fischer, Rotem Oshman, and Uri Zwick. Public vs. private randomness in simultaneous multi-party communication complexity. In *Proceedings of the 2016 International Colloquium on Structural Information and Communication Complexity*, volume 9988 of *Lecture Notes in Computer Science*, pages 60–74, 2016.

[FPSP19] Pierre Fraigniaud, Boaz Patt-Shamir, and Mor Perry. Randomized proof-labeling schemes. *Distributed Computing*, 32(3):217–234, 2019.

[Gar07] Frédéric Gardi. The roberts characterization of proper and unit interval graphs. *Discrete Mathematics*, 307(22):2906–2908, 2007.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM*, 62(4):27, 2015.

[GKS17] Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed MST and routing in almost mixing time. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 131–140, 2017.

[GP12] 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.

[GS16] Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory of Computing*, 12(1):1–33, 2016.

[HMPV00] 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.

[Hoà94] Chính T Hoàng. Efficient algorithms for minimum weighted colouring of some classes of perfect graphs. *Discrete Applied Mathematics*, 55(2):133–143, 1994.

[Hoe48] W Hoeffding. Probability inequalities for sums of random variables. *Annals of Mathematical Statistics*, 10:293–325, 1948.

[HRSS14] Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Large cuts with local algorithms on triangle-free graphs. *arXiv preprint arXiv:1402.2543*, 2014.

[HT74] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[ILG17] Taisuke Izumi and François Le Gall. Triangle finding and listing in congest networks. In *Proceedings of the ACM Symposium on Principles of*

Distributed Computing, pages 381–389, 2017.

[KK07] Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20(4):253–266, 2007.

[KKP10] Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.

[KMRS15] 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.

[KOS18] Gillat Kol, Rotem Oshman, and Raghuvansh R. Saxena. Interactive distributed proofs. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 255–264. ACM, 2018.

[KT96] Lefteris M Kirousis and Dimitris M Thilikos. The linkage of a graph. *SIAM Journal on Computing*, 25(3):626–647, 1996.

[Kus97] Eyal Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44. Elsevier, 1997.

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

[McC03] Ross M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003.

[MP06] 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.

[MPSRT18] 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.

[NPY20] Moni Naor, Merav Parte, and Eylon Yogev. The power of distributed verifiers in interactive proofs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1096–115. SIAM, 2020.

[NS96] Ilan Newman and Mario Szegedy. Public vs. private coin flips in one round communication games. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, STOC 2009, pages 561–570, 1996.

[PS15] Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Information and Computation*, 243:263–280, 2015.

[Rob69] Fred S. Roberts. Indifference graphs. proof techniques in graph theory. In *Proceedings of the Second Ann Arbor Graph Conference, Academic Press, New York*, 1969.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th annual ACM symposium on Theory of Computing*, pages 49–62. ACM, 2016.

[RS04] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[RTL76] 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.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

[Tuc70] Alan Tucker. Characterizing circular-arc graphs. *Bulletin of the American Mathematical Society*, 76(6):1257–1260, 11 1970.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979.

Appendix A

Graph problems

Here we include the list of problems to be considered in this work, following the definitions used in [Die06] and [BS+99], as well as the graph-theoretic definitions described in Section 1.1. For the sake of notation, we define N_G to be the set of nodes in a graph G which are adjacent to v (where we ignore the sub-index when there is no confusion on the graph considered). We define $d_G(v)$ to be the size of $N_G(v)$ and $\delta_G = \delta(G)$ to be the minimum degree among all nodes in graph G. We define P_k to be the path-graph on k nodes, C_k to be the cycle-graph on k nodes, K_m to be the complete graph (i.e. where all pairs of nodes are adjacent) of m nodes, $K_{m,n}$ to be the complete bipartite graph with parts of size n and m respectively.

Consider a set $\mathcal{O}$ of polynomial size in n, where the inclusion and intersection between objects is well-defined, which we call *object space*. An intersection model for a graph G is defined as a set of objects $\{O_v\}_{v \in V(G)}$ from a object space $\mathcal{O}$ such that u and v are adjacent in G if and only if their respective objects O_u and O_v is non-empty and belongs to $\mathcal{O}$.

Given a configuration $\langle G, \mathsf{id} \rangle$ with G a graph of n nodes and a permutation $\sigma : [n] \to [n]$ we define $\sigma(G, \mathsf{id})$ as the resulting configuration from permuting the id's of the nodes of G according to σ when sorting the nodes of G increasingly according to their identifiers. Finally, we denote $\iota : [n] \to [n]$ to be the identity function.

- d-DEGENERATE$= \{\langle G, \mathsf{id}, d \rangle$ s.t. $\max_{H \subseteq G} \quad \delta_H\}$
- TWINS$= \{\langle G, \mathsf{id} \rangle$ s.t. there exists $u, v \in G$ with $N(u) = N(v)\}$
- Δ-FREE$= \{\langle G, \mathsf{id} \rangle$ s.t. K_3 is not a subgraph of $G\}$
- CHORDAL$= \{\langle G, \mathsf{id} \rangle$ s.t. for any $k \geq 4, C_k$ is not an induced subgraph of $G\}$
- INTERVAL$= \{\langle G, \mathsf{id} \rangle$ s.t. G admits an intersection model of intervals on the real line$\}$

  In which case, we say that G is an interval graph.

- PROPER INTERVAL= $\{\langle G, \mathsf{id}\rangle$ s.t. G is an interval graph with no inclusions between intervals$\}$
- CIRCULAR-ARC= $\{\langle G, \mathsf{id}\rangle$ s.t. G admits an intersection model of arcs in a circle$\}$

  In which case, we say that G is an circular-arc graph.
- PROPER CIRC-ARC $= \{\langle G, \mathsf{id}\rangle$ s.t. G is a circular arc graph with no inclusions between arcs$\}$
- COGRAPH $= \{\langle G, \mathsf{id}\rangle$ s.t. G has no P_4 as an induced subgraph$\}$
- DIST-HEREDITARY $= \{\langle G, \mathsf{id}\rangle$ s.t. for every $H \subseteq G,$ and $u, v \in V, d_H(u, v) = d_G(u, v)\}$
- CLIQUE $= \{\langle G, \mathsf{id}\rangle$ s.t. for any $u, v \in V(G), u$ and v are adjacent$\}$
- H-SUBGRAPH= $\{\langle G, \mathsf{id}\rangle$ s.t. G has isomorphic copy of H as a subgraph$\}$
- H-MINOR= $\{\langle G, \mathsf{id}\rangle$ s.t. G contains H as a minor$\}$
- $\overline{\text{3-COL}}$= $\{\langle G, \mathsf{id}\rangle$ s.t. G is not 3-coloreable$\}$
- PLANAR= $\{\langle G, \mathsf{id}\rangle$ s.t. $K_{3,3}$ is not a minor of $G\}$
- OUTERPLANAR $= \{\langle G, \mathsf{id}\rangle$ s.t. $K_{2,3}$ is not a minor of $G\}$
- DIAMETER $= \{\langle G, \mathsf{id}, k\rangle$ s.t. $\max_{u,v} \quad d_G(u, v) \leq k\}$
- SYMMETRY $= \{\langle G, \mathsf{id}\rangle$ s.t. exists $\sigma : [n] \to [n]$ with $\sigma(G, \mathsf{id}) \cong \langle G, \mathsf{id}\rangle$ and $\sigma \neq \iota\}$
- AMOS $= \{\langle G, \mathsf{id}, I\rangle$ s.t. $I : V(G) \to \{0, 1\}$ and $|\{v \in V(G) : I(v) = 1\}| \leq 1\}$,
- 2-COL-EQ $= \{\langle G, \mathsf{id}, I\rangle$ s.t. $I : V(G) \to \{0, 1\}^n$ is a proper two-coloring of $G\}$.