Distributed Certification for Classes of Dense Graphs

Pierre Fraigniaud
IRIF, Université Paris Cité, CNRS, France

Frédéric Mazoit
LaBRI, Université de Bordeaux, France

Pedro Montealegre
Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago, Chile

Ivan Rapaport
DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Santiago, Chile

Ioan Todinca
LIFO, Université d'Orléans and INSA Centre-Val de Loire, France

Abstract

A *proof-labeling scheme* (PLS) for a boolean predicate Π on labeled graphs is a mechanism used for certifying the legality with respect to Π of global network states in a distributed manner. In a PLS, a *certificate* is assigned to each processing node of the network, and the nodes are in charge of checking that the collection of certificates forms a global proof that the system is in a correct state, by exchanging the certificates once, between neighbors only. The main measure of complexity is the *size* of the certificates. Many PLSs have been designed for certifying specific predicates, including cycle-freeness, minimum-weight spanning tree, planarity, etc.

In 2021, a breakthrough has been obtained, as a "meta-theorem" stating that a large set of properties have compact PLSs in a large class of networks. Namely, for every MSO_2 property Π on labeled graphs, there exists a PLS for Π with $O(\log n)$-bit certificates for all graphs of bounded *tree-depth*. This result has been extended to the larger class of graphs with bounded *tree-width*, using certificates on $O(\log^2 n)$ bits.

We extend this result even further, to the larger class of graphs with bounded *clique-width*, which, as opposed to the other two aforementioned classes, includes dense graphs. We show that, for every MSO_1 property Π on labeled graphs, there exists a PLS for Π with $O(\log^2 n)$-bit certificates for all graphs of bounded clique-width. As a consequence, certifying families of graphs such as distance-hereditary graphs and (induced) P_4-free graphs (a.k.a., cographs) can be done using a PLS with $O(\log^2 n)$-bit certificates, merely because each of these two classes can be specified in MSO_1. In fact, we show that certifying P_4-free graphs can be done with certificates on $O(\log n)$ bits only. This is in contrast to the class of C_4-free graphs (which does not have bounded clique-width) which requires $\tilde{\Omega}(\sqrt{n})$-bit certificates.

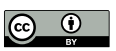
2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases CONGEST, Proof Labelling Schemes, clique-width, MSO

Digital Object Identifier 10.4230/LIPIcs.DISC.2023.20

Related Version *Full Version*: `https://arxiv.org/abs/2307.14292` [35]

Funding *Pierre Fraigniaud*: Additional support for ANR projects QuData and DUCAT.
Pedro Montealegre: This work was supported by Centro de Modelamiento Matemático (CMM), FB210005, BASAL funds for centers of excellence from ANID-Chile, and ANID-FONDECYT 1230599
Ivan Rapaport: This work was supported by Centro de Modelamiento Matemático (CMM), FB210005, BASAL funds for centers of excellence from ANID-Chile, and ANID-FONDECYT 1220142.



1 Introduction

Checking whether a distributed system is in a legal global state with respect to some boolean predicate occurs in several domains of distributed computing, including the following.

- Fault-tolerance: the occurrence of faults may turn the system into an illegal state that needs to be detected for allowing the system to return to a legal state.
- The use of subroutines as black boxes: some of these subroutines may contain bugs, and produce incorrect outputs that need to be checked before use in the protocol calling the subroutines.
- Algorithm design for specific classes of systems: an algorithm dedicated to some specific class of networks (e.g., algorithms for trees, or for planar networks) may cause deadlocks or live-locks whenever running on a network outside the class. The membership to the class needs to be checked before running the algorithm.

In all three cases above, the checking procedure may be impossible to implement without significant communication overhead. A typical example is bipartiteness, whether it be applied to the network itself, or to an overlay network produced by some subroutine.

1.1 Proof-Labeling Schemes

Proof-labeling scheme (PLS) [46] is a popular mechanisms enabling to certify correctness w.r.t. predicates involving some global property, like bipartiteness. A PLS involves a *prover* and a *verifier*. The prover has access to the global state of the network (including its structure), and has unlimited computational power. It assigns *certificates* to the nodes. The verifier is a distributed algorithm running at each node, performing in a single round, which consists for each node to send its certificate to its neighbors. Upon reception of the certificates of its neighbors, every node performs some local computation and outputs *accept* or *reject*. To be correct, a PLS for a predicate Π must satisfy:

$$\begin{array}{c}\text{the global state of the network satisfies } \Pi\\ \Updownarrow\\ \text{the prover can assign certificates such that the verifier accepts at all nodes.}\end{array}$$

For instance, for bipartiteness, the prover assigns a color 0 or 1 to the nodes, and each node verifies that its color is 0 or 1, and is different from the color of each of its neighbors. If the network is bipartite then the prover can properly 2-color the nodes such that they all accept, and if the network is not bipartite then, for every 2-coloring of the nodes, some of them reject as this coloring cannot be proper.

The PLS certification mechanism has several desirable features. First, if the certificates are small then the verification is performed efficiently, in a single round consisting merely of an exchange of a small message between every pair of adjacent nodes. As a consequence, verification can be performed regularly and frequently without causing significant communication overhead. Second, if the network state does not satisfy the predicate, then at least one node rejects. Such a node can raise an alarm or launch a recovery procedure for allowing the system to return to a correct state, or can stop a program running in an environment for which it was not designed. Third, the prover is an abstraction, for the certificates can be computed offline, either by the nodes themselves in a distributed manner, or by the system provider in a centralized manner. For instance, a protocol constructing an overlay network that is supposed to be bipartite, may properly 2-color the overlay for certifying its bipartiteness. It follows from their features that PLSs are versatile certification mechanisms that are also quite efficient whenever the certificates for legal instances are small.

Many PLSs have been designed for certifying specific predicates on labeled graphs, including cycle-freeness [46], minimum-weight spanning tree (MST) [45], planarity [31], bounded genus [25], H-minor-freeness for small H [6], etc. In 2021, a breakthrough has been obtained, as a "meta-theorem" stating that a large set of properties have compact PLSs in a large class of networks (see [27]). Namely, for every MSO_2 property[1] Π, there exists a PLS for Π with $O(\log n)$-bit certificates for all graphs of bounded *tree-depth*, where the tree-depth of a graph intuitively measures how far it is from being a star. This result has been extended to the larger class of graphs with bounded *tree-width* (see [36]), using certificates on $O(\log^2 n)$ bits, where the tree-width of a graph intuitively measures how far it is from being a tree. Although the class of all graphs with bounded tree-width includes many common graph families such as trees, series-parallel graphs, outerplanar graphs, etc., it does not contain families of *dense* graphs. In this paper, we focus on the families of graphs with bounded *clique-width*, which include families of dense graphs.

1.2 Clique-Width

Intuitively, the definition of clique-width is based on a "programming language" for constructing graphs, using only the following four instructions (see [16] for more details):

- Creation of a new vertex v with some color i, denoted by $\mathsf{color}(v, i)$;
- Disjoint union of two colored graphs G and H, denoted by $G \parallel H$;
- Joining by an edge every vertex colored i to every vertex colored $j \neq i$, denoted by $i \bowtie j$;
- Recolor i into color j, denoted by $\mathsf{recolor}(i, j)$.

For instance, the n-node clique can be constructed by creating a first node with color blue, and then repeating $n-1$ times the following: (1) the creation of a new node, with color red, (2) joining red and blue, and (3) recoloring red into blue. Therefore, cliques can be constructed by using two colors only. Similarly, trees can be constructed with three colors only. This can be proved by induction. The induction statement is that, for every tree T, every vertex r of T, and every two colors $c_1, c_2 \in \{\text{blue, red, green}\}$, T can be constructed with colors blue, red, and green such that r is eventually colored c_1, and every other vertex is colored c_2. The statement is trivial for the single-node tree. Let T be a tree with at least two nodes, let r be one of its vertices, and let c_1, c_2 be two colors. Given an arbitrary neighbor s of r, removing the edge $\{r, s\}$ results in two trees T_r and T_s. By induction, construct T_r and T_s separately so that r (resp., s) is eventually colored c_1 (resp., c_2) and all the other nodes of T_r and T_s are colored $c_3 \notin \{c_1, c_2\}$. Then form the graph $T_r \parallel T_s$, and, in this graph, join colors c_1 and c_2, and recolor c_3 into c_2.

The clique-width of a graph G, denoted by $\mathsf{cw}(G)$, is the smallest $k \geq 0$ such that G can be constructed by using k colors. For instance, $\mathsf{cw}(K_n) \leq 2$ for every $n \geq 1$, and, for every tree T, $\mathsf{cw}(T) \leq 3$. A family of graphs has bounded clique-width if there exists $k \geq 0$ such that, for every graph G in the family, $\mathsf{cw}(G) \leq k$. Any graph family with bounded tree-depth or bounded tree-width has bounded clique-width [12, 48]. However, there are important graph families with unbounded tree-width (and therefore unbounded tree-depth) that have bounded clique-width. Typical examples (see [17]) are cliques (i.e., complete

[1] Monadic second-order logic (MSO) is the fragment of second-order logic where the second-order quantification is limited to quantification over sets. MSO_1 refers to MSO on graphs with quantification over sets of vertices, whereas MSO_2 refers to MSO on graphs with quantification over sets of vertices and sets of edges.

graphs), P_4-free graphs (i.e., graphs excluding a path on four vertices as an induced subgraph, a.k.a., cographs), and distance hereditary graphs (the distances in any connected induced subgraph are the same as they are in the original graph).

Many NP-hard optimization problems can be solved efficiently by dynamic programming in the family of graphs with bounded clique-width. In fact, every MSO_1 property on graphs has a linear-time algorithm for graphs of bounded clique-width [16]. In this paper we show a similar form of "meta-theorem", regarding the size of certificates of PLS for monadic second-order properties of graphs with bounded clique-width.

1.3 Our Results

Our main result is the following. Recall that a labeled graph is a pair (G, ℓ), where G is a graph, and $\ell : V(G) \to \{0,1\}^\star$ is a function assigning a label to every node in G.

▶ **Theorem 1.** *Let k be a non-negative integer, and let Π be an MSO_1 property on node-labeled graphs with constant-size labels. There exists a PLS certifying Π for labeled graphs with clique-width at most k, using $O(\log^2 n)$-bit certificates on n-node graphs.*

The same way several NP-hard problems become solvable in polynomial time in graphs of bounded clique-width, Theorem 1 implies that several predicates for which every PLS has certificates of polynomial size in arbitrary graphs have a PLS with certificates of polylogarithmic size on graphs with bounded clique-width. This is for instance the case of non-3-colorability (which is a MSO_1 predicate), for which every PLS has certificates of size $\tilde{\Omega}(n^2)$ bits in arbitrary graphs [42]. Theorem 1 implies that non-3-colorability has a PLS with certificates on $O(\log^2 n)$ bits in graphs with bounded clique-width, and therefore in graphs of bounded tree-width, cographs, distance-hereditary graphs, etc. This of course is extended to non-k-colorability, as well as other problems definable in MSO_1 such as detecting whether the input graph does not contain a fixed subgraph H as a subgraph, induced subgraph, minor, etc.

In fact, Theorem 1 can be extended to properties including certifying solutions to maximization or minimization problems whose admissible solutions are defined by MSO_1 properties. For instance maximum independent set, minimum vertex cover, minimum dominating set, etc.

In the proof of Theorem 1, we provide a PLS that constructs a particular decomposition using at most $k \cdot 2^{k-1}$ colors (the clique-width of the decomposition). It is through that decomposition that the PLS certifies that the input graph satisfies Π.

An application of Theorem 1 is the certification of certain families of graphs. That is, given a graph family $\mathcal{F}$, designing a PLS for certifying the membership to $\mathcal{F}$. Interestingly, there are some graph classes $\mathcal{F}$ that are expressible in MSO_1 and, at the same time, have clique-width at most k. Theorem 1 provides a PLS for certifying the membership to $\mathcal{F}$ in such cases. Indeed, the PLS first tries to build a decomposition of clique-width at most $k \cdot 2^{k-1}$. If there is no such decomposition, then the input graph does not belong to $\mathcal{F}$. Otherwise, the PLS uses the decomposition to check the MSO_1 property that defines $\mathcal{F}$.

▶ **Corollary 2.** *Let k be a non-negative integer, and let $\mathcal{F}$ be graph family expressible in MSO_1 such that all graphs of the family have clique-width at most k. Membership to $\mathcal{F}$ can be certified with a PLS using $O(\log^2 n)$-bit certificates in n-node graphs.*

For instance, for every $k \geq 0$, the class of graphs with tree-width at most k can be certified with a PLS using $O(\log^2 n)$-bit certificates. Indeed, "tree-width at most k" is expressible in MSO_1, and the class of graphs with tree-width at most k forms a family with clique-width at

most $3 \cdot 2^{k-1} + 1$ [12]. Another interesting application is the certification of P_4-free graphs. Indeed, "excluding P_4 as induced subgraph" is expressible in MSO_1, and P_4-free graphs form a family with clique-width at most 2 [17]. It follows that P_4-free graphs can be certified with a PLS using $O(\log^2 n)$-bit certificates. This is in contrast to the class of C_4-free graphs (i.e. graphs not containing a cycle on four vertices, whether it be as induced subgraph or merely subgraph), which requires certificates on $\tilde{\Omega}(\sqrt{n})$ bits [20]. In fact, in the case of cographs, the techniques in the proof of Theorem 1 can be adapted so that to save one log-factor, as stated below.

▶ **Theorem 3.** *The class of (induced) P_4-free graphs can be certified with a PLS using $O(\log n)$-bit certificates in n-node graphs.*

Note that there is a good reason for the huge gap in terms of certificate-size between P_4-free graphs and C_4-free graphs. The point is that, for any graph pattern H, the class of H-free graphs has bounded clique-width if and only if H is an induced subgraph of P_4 [19]. Therefore, C_4-free graphs (as well as triangle-free graphs) do not have bounded clique-width, as opposed to P_4-free graphs (and P_3-free graphs, which are merely cliques).

1.4 Related Work

Proof-Labeling Schemes (PLSs) have been introduced and thoroughly studied in [46]. Variants have been been considered in [42] and [34], which slightly differ from PLSs: the former allows each node to transfer no only its certificates, but also its state, and the latter restricts the power of the oracle, which is bounded to produce certificates independent of the IDs assigned to the nodes. All these forms of distributed certifications have been extended in various directions, including tradeoffs between the size of the certificates and the number of rounds of the verification protocol [30], PLSs with computationally restricted provers [24], randomized PLSs [38], quantum PLSs [33], PLSs rejecting at more nodes whenever the global state is "far" from being correct [28], PLSs using global certificates in addition to the local ones [32], and several hierarchies of certification mechanisms, including games between a prover and a disprover [1, 29], interactive protocols [18, 44, 47], and even recently zero-knowledge distributed certification [3], and distributed quantum interactive protocols [41].

All the aforementioned distributed certification mechanisms have been used for certifying a wide variety of global system states, including MST [45], routing tables [2], and a plethora of (approximated) solutions to optimization problems [11, 23]. A vast literature has also been dedicated to certifying membership to graph classes, including cycle-freeness [46], planarity [31], bounded genus [25], absence of symmetry [42], H-minor-freeness for small H [6], etc. In 2021, a breakthrough has been obtained, as a "meta-theorem" stating that, for every MSO_2 property Π, there exists a PLS for Π with $O(\log n)$-bit certificates for all graphs of bounded *tree-depth* [27]. This result has been extended to the larger class of graphs with bounded *tree-width*, using certificates on $O(\log^2 n)$ bits [36]. To our knowledge, this is the largest class of graphs, and the largest class of boolean predicates on graphs for which it is known that PLSs with polylogarithmic certificates exist.

The class of H-free graphs (i.e., the absence of H as a subgraph), for a given fixed graph H, has attracted lot of attention in the distributed setting, mostly in the CONGEST model. Two main approaches have been considered. One, called distributed property testing, aims at deciding between the case where the input graph is H-free, and the case where the input graph is "far" from being H-free (see, e.g., [7, 9, 26, 39]). In this setting, the objective is to design (randomized) algorithms performing in a constant number of rounds. Such algorithms have been designed for small graphs H, but it is not known whether there is a

distributed algorithm for testing K_5-freeness in a constant number of rounds. The other approach aims at designing algorithms deciding H-freeness performing in a small number of rounds. For instance, it is known that deciding C_4-freeness can be done in $\tilde{O}(\sqrt{n})$ rounds, and this is optimal [20]. The $\tilde{\Omega}(\sqrt{n})$-round lower bounds for C_4-freeness also holds for deciding C_{2k}-freeness, for every $k \geq 4$. Nevertheless, the best known algorithm performs in essentially $\tilde{O}(n^{1-\Theta(1/k^2)})$ rounds [22], even if faster algorithms exists for $k = 2, 3, 4, 5$, running in $\tilde{O}(n^{1-\Theta(1/k)})$ rounds [10, 21]. Deciding P_k-freeness (as subgraph) can be done efficiently for all $k \geq 0$ [37]. However, this is not the case of deciding the absence of an *induced* P_k, and no efficient algorithms are known apart for the trivial cases $k = 1, 2, 3$. The first non-trivial case is deciding cographs, i.e., P_4-freeness (as induced subgraph).

The terminology *meta-theorem* is used in logic to refer to a statement about a formal system proven in a language used to describe another language. In the study of graph algorithms, Courcelle's theorem [13] is often referred to as a meta-theorem. It says that every graph property definable in the monadic second-order logic MSO_2 of graphs can be decided in linear time on graphs of bounded treewidth. This theorem was extended to clique-width, but for a smaller set of graph properties. Specifically, every graph property definable in the monadic second-order logic MSO_1 of graphs can be decided in linear-time on graphs of bounded clique-width [16]. Note that the classes of languages in MSO_1 and MSO_2 include languages that are NP-hard to decide (e.g., 3-colorability and Hamiltonicity, respectively). We remind that MSO_2 is as an extension of MSO_1 which also allows quantification on sets of edges – see Footnote 1 for a short description, or [14] for full details. Some graph properties, e.g., Hamiltonicity, are expressible in MSO_2 but not in MSO_1, nevertheless MSO_1 captures a large set of properties, including many classical NP-hard problems as explained above. Eventually, we emphasize again that, when comparing the two most famous meta-theorems, (1) MSO_2 *properties are decidable in linear time on bounded treewidth graphs* vs. (2) MSO_1 *properties are decidable in linear time on bounded clique-width graphs*, the former concerns a larger class of properties, but the latter concerns larger classes of graphs.

2 Models

In this section, we recall the main concepts used in this paper, including proof-labeling scheme, and cographs.

2.1 Proof-Labeling Schemes for MSO Properties

For a fixed integral parameter $\lambda \geq 0$, we consider vertex-labeled graphs (G, ℓ), where $G = (V, E)$ is a connected simple n-node graph, and $\ell : V \to \{0, \ldots, \lambda - 1\}$. The label may indicate a solution to an optimization problem, e.g., a minimum dominating set ($\ell(v) = 0$ or 1 depending on whether v is in the set or not), a λ-coloring, an independent set, etc. A labeling may also encode global overlay structures such as spanning trees or spanners, in bounded-degree graphs or in graphs provided by a distance-2 k-coloring, for $k = O(1)$. In the context of distributed computing in networks, nodes are assumed to be assigned distinct identifiers (ID) in $[1, n^c]$ for some $c \geq 1$, so that IDs can be stored on $O(\log n)$ bits. The identifier of a node v is denoted by $\mathsf{id}(v)$. We denote by $N_G(v)$ the set of neighbors of node v in a graph G, and we let $N_G[v] = N_G(v) \cup \{v\}$ be the closed neighborhood of v.

Given a boolean predicate Π on vertex-labeled graphs, a *proof-labeling scheme* (PLS) for Π is a prover-verifier pair. The *prover* is a non-trustable computationally unbounded oracle. Given a vertex-labeled graph (G, ℓ) with ID-assignment id, the prover assigns a *certificate* $c(v)$ to every node v of G. The *verifier* is a distributed algorithm running at every

node v of G. It performs a single round of communication consisting of sending $c(v)$ to all neighboring nodes $w \in N_G(v)$, and receiving the certificates of all neighbors. Given $\mathsf{id}(v)$, $\ell(v)$, and $\{c(w) : w \in N_G[v]\}$, every node v outputs *accept* or *reject*. A PLS is correct if the following two conditions are satisfied:

- Completeness: If (G, ℓ) satisfies Π then the prover can assign certificates to the nodes such that the verifier accepts at all nodes;
- Soundness: If (G, ℓ) does not satisfy Π then, for every certificate assignment to the nodes by the prover, the verifier rejects in at least one node.

The main parameter measuring the quality of a PLS is the *size* (i.e., number of bits) of the certificates assigned by the prover to each node of vertex-labeled graphs satisfying the predicate, and leading all nodes to accept.

MSO Predicates. We focus on predicates expressible in MSO_1. Recall that MSO_1 is the fragment of monadic second-order (MSO) logic on (vertex-labeled) graphs that allows quantification on vertices and on sets of (labeled) vertices, and uses the adjacency predicate (adj). For instance non 3-colorability is in MSO_1. Indeed, for every graph $G = (V, E)$, it can be expressed as: for all $A, B, C \subseteq V$, if $A \cup B \cup C = V$ and $A \cap B = A \cap C = B \cap C = \varnothing$ then

$$\exists (u, v) \in (A \times A) \cup (B \times B) \cup (C \times C) : (u \neq v) \wedge \mathsf{adj}(u, v).$$

We shall show that, although some MSO_1 predicates, like non-3-colorability, require certificates on $\tilde{\Omega}(n^2)$ bits in n-node graphs (see [42]), PLSs with certificates of polylogarithmic size can be designed for all MSO_1 predicates in a rich class of graphs, namely all graphs with bounded clique-width.

2.2 Cographs and Cotrees

We conclude this section by introducing a graph class that plays an important role in this paper. Recall that a graph is a cograph (see, e.g., [8]) if it can be constructed by a sequence of parallel operations (disjoint union of two vertex-disjoint graphs) and join operations (connecting two vertex-disjoint graphs G and H by a complete bipartite graphs between $V(G)$ and $V(H)$). Therefore, by definition, cographs have clique-width 2. In particular, cliques are cographs.

It is known [8] that cographs capture precisely the class of induced P_4-free graphs. We shall show that, as opposed to C_4-free graphs, which require $\tilde{\Omega}(\sqrt{n})$-bit certificates to be certified by a PLS [20][2], $O(\log n)$-bit certificates are sufficient for certifying P_4-free graphs. This result is of interest on its own, but proving this result will also play the role of a warmup before establishing our general result about graphs with bounded clique-width. Note that the class of P_4-free graphs (i.e., cographs) can be specified by an MSO_1 formula. Roughly, the formula states that if there exists four vertices v_1, v_2, v_3, v_4 such that $\mathsf{adj}(v_i, v_{i+1})$ for $i = 1, 2, 3$, then $\mathsf{adj}(v_1, v_3) \vee \mathsf{adj}(v_1, v_4) \vee \mathsf{adj}(v_2, v_4)$. C_4-freeness could be expressed in MSO_1 as well. However, P_4-free graphs have clique-width 2 whereas C_4-free graphs have unbounded clique-width – this is because there are $2^{\Omega(n\sqrt{n})}$ different C_4-free graphs of size n, but only $2^{\mathcal{O}(n \log n)}$ n-vertex graphs of bounded clique-width.

[2] The lower bound in [20] is expressed for the CONGEST and Broadcast Congested Clique models, but it extends directly to PLSs since Set-Disjointness has non-deterministic communication complexity $\Omega(N)$ on N-bit inputs.

Given a cograph G, there is actually a canonic way of constructing G by a sequence of parallel and join operations [8]. As explained before, this construction can be described as a tree T whose leaves are the vertices of G, and whose internal nodes are labeled $\parallel$ or $\bowtie$. This tree is called a *cotree*, and will be used for our PLS.

3 Overview of our Techniques

The objective of this section is to provide the reader with a general idea of our proof-labeling scheme. For a comprehensive description we refer to the full version of this article [35]. Our construction bears some similarities with the approach used in [36] for the certification of MSO_2 properties on graphs of bounded tree-width, with certificates of size $O(\log^2 n)$ bits. However, extending this approach to a proof-labeling scheme for graphs with bounded clique-width requires to overcome several significant obstacles. We therefore start by summarizing the main tools used for the certification of MSO_2 properties on graphs of bounded tree-width (see Section 3.1), and then proceed with the description of the new tools required for extending the result to graphs of bounded clique-width, to the cost of reducing the class of certified properties from MSO_2 to MSO_1 (see Sections 3.2-3.6).

3.1 Certifying MSO_2 Properties in Graphs of Bounded Tree-Width

Recall that a tree-decomposition of a graph G is a tree T where each node x of T, also called *bag*, is a subset of $V(G)$, satisfying the following three conditions: (1) for every vertex $v \in V(G)$ there is a bag $x \in V(T)$ that contains v, (2) for every edge $\{u, v\} \in E(G)$, there is a bag x containing both its endpoints, and (3) for each vertex $v \in V(G)$, the set of bags that contains v forms a (connected) subtree of T. Let Π be an MSO_2 property, and let T be a tree-decomposition of the graph G. The proof-labeling scheme aims at providing each vertex with sufficient information for certifying the correctness of T, as well as the fact that G satisfies Π. To do so, the certificate of each vertex is divided into two parts, one called *main messages*, and the other called *auxiliary messages*.

Main messages. The main message of a node v is a sequence seq_v representing a path of bags in T that connects a leaf with the root, such that v is contained in at least one bag of seq_v. For each bag $x \in \mathrm{seq}_v$, the main message includes, roughly: the set of vertices contained in x, the identifier of a vertex ℓ_x in x, called the *leader* of x, and a data structure c_x used to verify the MSO_2 property Π on G. The leader ℓ_x of x is chosen arbitrarily among the vertices of x that are adjacent to a vertex u belonging to the parent bag $p(x)$ of x in T. The vertex u is said to be *responsible* for x in $p(x)$. Let us assume the following consistency condition: for every bag x of T, every vertex in x received the same information about all the bags from x to the root of T. Under the promise that the consistency condition holds, it is possible to show that the vertices can collectively verify that T is indeed a tree-decomposition of G, and that G satisfies Π.

Auxiliary messages. The role of the auxiliary messages is precisely to check the above consistency condition. For each bag x, let τ_x be a Steiner tree (i.e. a minimal tree connecting a set of vertices denoted *terminals*) in G rooted at the leader ℓ_x, with all the nodes of x as terminals. Every vertex in τ_x receives an auxiliary message containing the certification of τ_x (each vertex of τ_x receives the identifier of a root, of its parent and the distance to the root), and a copy of the information about x given to the nodes in the bag x, through their main messages. By using the auxiliary messages, the leader ℓ_x can verify whether the

subgraph $G[x]$ of G induced by the union of the bags in $T[x]$ satisfies Π, where $T[x]$ the subtree of T containing x and all its descendants. Specifically, this verification is performed by simulating the dynamic programming algorithm in Courcelle's Theorem [13] as in the version of Boire, Parker and Tovey [5]. This uses a constant-size data structure c_x stored in the auxiliary messages that "encodes" the predicate $\Pi(G[x])$. Its correctness can be verified by a composition of the values c_y for each child y of x in T. The tree τ_x is actually used to transfer the information about c_y from the node ℓ_y in x responsible for y, to the leader ℓ_x.

Certificate size. If T is of depth d, then the main messages are of size $O(d \log n)$ bits. Crucially, for every graph G, there is a tree-decomposition T satisfying that, for every bag x, there is a Steiner tree τ_x completely contained in $G[x]$. Such a decomposition is called *coherent* in [36] (Lemma 3). It follows that every node participates in a Steiner tree with at most d bags, which implies that the auxiliary messages can be encoded in $O(d \log n)$ bits. Thanks to a construction by Bodlaender [4], it is possible to choose a coherent tree-decomposition with depth $d = O(\log n)$, up to increasing the sizes of the bags by a constant factor only. It follows that the certificates are of size $O(\log^2 n)$ bits.

Our construction also follows the general structure described above. However, each element of this construction has to be adapted in a highly non-trivial way. Indeed, the grammar of clique-with, and the related structure of NLC decomposition, differ in several significant ways from the grammar of tree-width. The rest of the section is dedicated to providing the reader with a rough idea of how this can be done.

3.2 Clique-Width and NLC-Width

First, instead of working with clique-width, it is actually more convenient to work with the NLC-width, where NLC stands for *node-label controlled*. Every graph of clique-width at most k has NLC-width at most k, and every graph of NLC-width at most k has clique-width at most $2k$ [43]. As clique-width, NLC-width can be viewed as the following grammar for constructing graphs, bearing similarities with the grammar for clique-width:

- Creation of a new vertex v with color $i \in \mathbb{N}$, denoted by $\mathsf{newVertex}_i$;
- Given a set S of ordered pairs of colors, and an ordered pair (G, H) of vertex-disjoint colored graphs, create a new graph as the union of G and H, then join by an edge every vertex colored i of G to every vertex colored j of H, for all $(i, j) \in S$; this operation is denoted by $G \bowtie_S H$;
- Recolor the graph, denoted by $\mathsf{recolor}_R$ where $R : \mathbb{N} \to \mathbb{N}$ is any function.

If $k \geq 1$ colors are used, a recoloring function R is a function $R : [k] \to [k]$. When R is used, for every $i \in [k]$, vertices with color i are recolored $R(i) \in [k]$ (all colors are treated simultaneously, in parallel). Note that the recoloring operation in the definition of clique-width is limited to functions R that preserve all colors but one. Note also that, for $S = \varnothing$, the operation $G \bowtie_S H$ is merely the same as $G \parallel H$ for clique-width. We therefore use $G \bowtie_\varnothing H$ or $G \parallel H$ indistinctly. The NLC-width of a graph G is the smallest number of colors such that G can be constructed using the operations above. It is denoted by $\mathsf{nlcw}(G)$. For instance, the n-node clique can be constructed by creating a first node v_1 with color 1, and then repeating, for all $i = 1, \ldots, n-1$, (1) the creation of a new node v_{i+1}, with color 1 as well, and (2) applying $v_{i+1} \bowtie_{\{(1,1)\}} K_i$ to get the clique K_{i+1} on $i+1$ vertices. Therefore, cliques can be constructed by using one color only, i.e., $\mathsf{nlcw}(K_n) = 1$ for every $n \geq 1$.

NLC-decomposition. For every $k \geq 1$, the construction of a graph G with $\mathsf{nlcw}(G) \leq k$ can be described by a binary tree T, whose leaves are the (colored) vertices of G. In T, every internal node x has an identified left child x' and an identified right child x'', and is labeled by $\|$ or $\bowtie_S$ for some non-empty set $S \subseteq [k] \times [k]$. This label indicates the operation performed on the (left) graph G' with vertex-set equal to the leaves of the subtree $T_{x'}$ of T rooted at x', and the (right) graph G'' with vertex-set equal to the leaves of the subtree $T_{x''}$ of T rooted at x''. That is, node x corresponds to the operation $G_{x'} \| G_{x''}$ or $G_{x'} \bowtie_S G_{x''}$, depending on the label of x. In addition to its label ($\|$ or $\bowtie_S$ for some $S \neq \varnothing$), a node may possibly also include a recoloring function $R : [k] \to [k]$, which indicates a recoloring to be performed *after* the join operation, see Figure 1 for an example.

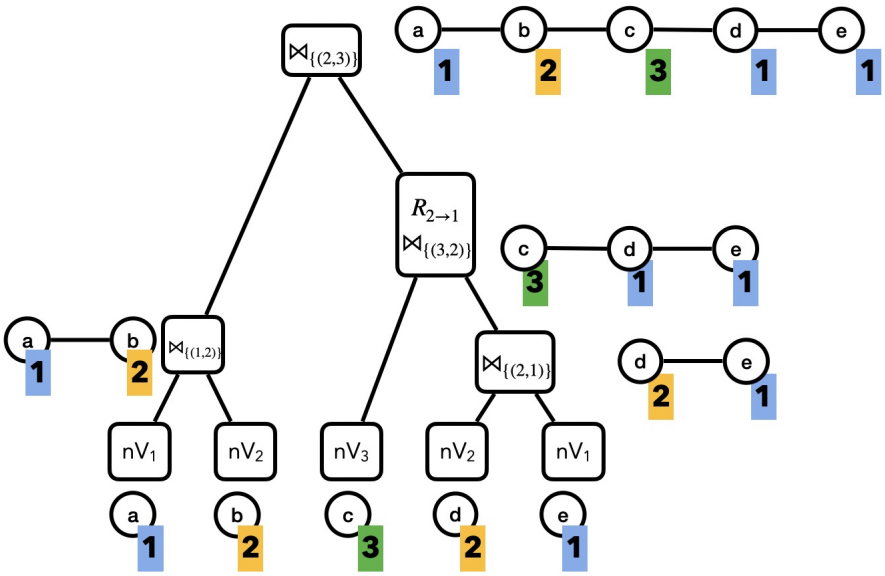


Figure 1 An NLC decomposition tree T. Next to each node x of the tree is displayed the colored graph $G[x]$ corresponding to subtree $T[x]$ of T rooted at x.

3.3 From Tree-Width to NLC-Width: The Main Messages

Let Π be an MSO_1 property, and let T be an NLC-decomposition tree of a graph G with $\mathsf{cw}(G) \leq k$. That is, we can choose the tree T as one using at most k colors. In the following, to avoid confusion, we call *vertices* the elements of the vertex set of G, and *nodes* the elements of the vertex-set of the decomposition tree T. The structure of our certificates differ from the one in [36], and now we decompose the certificate assigned to each node v into three parts: *main messages*, *auxiliary messages*, and *service messages*. This subsection focuses on the main messages.

Our main messages have, to some extent, a structure similar to the main messages used in [36] for the tree-width. In particular, vertex v receives a sequence $\mathsf{path}(v)$, listing all the nodes, i.e., the whole set of operations, in the path from the root of T to the leaf of T where v was created. For each node x in $\mathsf{path}(v)$, the main message also includes the vertex identifier of a leader for x, called *exit vertex of* x, and denoted by $\mathsf{exit}(x)$. The main message also includes a data structure $h(x)$ that encodes the truth value of the MSO_1 property on $G[x]$.

However, unlike the case of tree-width, where the nodes of the tree-decomposition are sets of vertices (i.e., bags) of bounded size, the contents of a non-leaf node in an NLC-decomposition tree T does not necessarily include information about the vertices created in $T[x]$. For that reason, our proof-labeling scheme includes additional information in the main message of v in order to verify the correctness of the given decomposition. It may actually be worth providing a concrete example to explain the need for additional information.

For a node x different from the root, let us denote by $p(x)$ the parent of x in T. The main message of v includes a sequence $\mathsf{links}(v)$ that specifies, for each node x in $\mathsf{path}(v)$ different from the root, whether x is the left or right child of $p(x)$. For instance, in the example of Figure 1, we have $\mathsf{links}(c) = (1, 0)$, indicating that, to reach the leaf creating vertex c from the root, one must follow the right child (1), and then the left child (0). S imilarly, $\mathsf{links}(d) = (1, 1, 0)$. The sequences links are also used to determine the longest common prefixes of the main messages, when the same operations are repeated between two children of a same node (consider for instance the case where the same operation is performed at all the nodes of the decomposition tree). Back to our example above, let us suppose that the sequences $\mathsf{links}(u)$ and $\mathsf{links}(v)$ specify that $x_{t_3+1}(u)$ is the left neighbor of x_{t_3}, and $x_{t_3+1}(v)$ is the right neighbor of x_{t_3}. Using this information, u and v can infer that it is an operation $\bowtie_S$, with $(c(u), c(v)) \in S$ that is specified in the description of x_{t_3}. With the given information, each vertex can thus check that all its incident edges are indeed created at some node of the decomposition tree T.

It remains to check that the decomposition does not define non-existent edges. To do so, the main message of every vertex v also includes, for each node x in $\mathsf{path}(v)$, and for each $i \in [k]$, the integers $\mathsf{color}_i(x)$ representing the number of vertices of $G[x]$ that are colored i in the root of $T[x]$. (Recall that the subgraph $G[x]$ is the subgraph of G induced by the vertices created in the subtree $T[x]$ of T). Returning to our example, vertex v checks that it has exactly $\mathsf{color}_{c(u)}(x_{t_3+1}(u))$ neighbors with the same longest common prefix as u colored $c(u)$ in the left children of x_{t_3}. Also, vertex v checks, for each $i \in [k]$, that the number of vertices colored i in node x_{t_3} corresponds to the sum of the number of vertices colored j in $x_{t_3+1}(u)$ and $x_{t_3+1}(v)$, for each color j that is recolored i by the recoloring operation defined in x_{t_3}. So, let us assume that the following consistency condition (analogous to the one for the certification of tree-decompositions) holds:

C1: For every pair of vertices $u, v \in V(G)$, and for every node x in both $\mathsf{path}(u)$ and $\mathsf{path}(v)$, u and v receive the same information about all nodes in the path from x to the root of T in their main messages, and

C2: If x is the root of T, then the data structure $h(x)$ describes an accepting instance (i.e., G satisfies Π).

Assuming that the consistency condition is satisfied, it is not difficult to show that the vertices can collectively check that the given certificates indeed represent an NLC-decomposition tree, and that G satisfies Π. The difficulty is however in checking that the consistency condition holds. This is the role of the auxiliary and service messages, described next.

3.4 Checking Consistency: Auxiliary, and Service Messages

We use auxiliary and service messages for allowing our proof-labeling scheme to check the first condition **C1** of the consistency condition defined at the end of the previous subsection.

Auxiliary messages can easily be defined for every node x of T satisfying that $G[x]$ is connected. In that case, the auxiliary messages of all the vertices v in $T[x]$ contain the certificates for certifying a spanning tree τ_x of $G[x]$ rooted at the exit vertex of x. Each

vertex v can verify that the longest common prefix common to v and its parent in τ_x contains all the nodes from the root up to x, and that the information given in the main messages coincide for all such nodes. Observe that every vertex v may potentially contain one auxiliary message for each node in $\mathsf{path}(v)$.

The case where $G[x]$ is not connected is fairly more complicated, and we need to introduce another type of decomposition.

NLC+ decompositions trees. Observe that G itself is connected. Therefore, there must exist an ancestor z of x for which $G[z]$ is connected. We could provide the vertices in $G[z]$ with a spanning tree of $G[z]$ for checking the consistency in $T[x]$. However, the vertices in $G[z]$ do not necessarily contain x in the prefixes of their node sequences, so we would have to put a copy of the main message associated to x on every node participating in the spanning tree. Since an NLC-decomposition tree does not allow to provide a bound on the distance between z and x in the tree, we have no control on how many copies of main messages a vertex should handle.

Therefore, to cope with the case where $G[x]$ is disconnected, we define a specific type of NLC decompositions trees, called NLC+ decompositions trees. The NLC+ decomposition trees are similar to NLC-decomposition trees, up to two important differences.

- First, we allow the nodes corresponding to a $\|$ operation to have arbitrary large arity, and thus NLC+ decomposition trees are not binary trees, as opposed to NLC-decomposition trees.
- Second, if a node x induces a disconnected subgraph $G[x]$, then its parent node $p(x)$ must satisfy that $G[p(x)]$ is connected. Observe that $p(x)$ must then correspond to a $\bowtie$ operation, and thus $p(x)$ has only two children: x and another child, denoted by y.

Service trees. A *service tree* S_x for a node x such that $G[x]$ is disconnected is a Steiner tree in $G[p(x)]$ rooted at the exit vertex of x, and with all the vertices of $G[x]$ as terminals. Each vertex of S_x (i.e., all vertices in $G[x]$, plus some vertices in $G[y]$ is given a *service message*, which contains the certificate for the tree S_x, as well as a copy of the information about x given in the main messages of the vertices in $G[x]$. Each vertex in S_x can then check that it shares the same information about x than its parent. The properties of the NLC+ decomposition guarantee that a vertex v participates to at most two service trees, for each node x in the sequence $\mathsf{path}(v)$. Indeed, vertex v necessarily participates in S_x when x is of type $\|$, and may also participate in S_y whenever the sibling y of x is of type $\|$. There are significantly more subtle details concerning service trees, but they are described in the full version [35].

It remains to check the second condition **C2** of the consistency condition defined at the end of the previous subsection, which consists in verifying the correctness of $h(x)$, for every node x of T. This is explained next.

3.5 Dealing with MSO_1 Predicates

In their seminal work, Courcelle, Makowsky and Rotics [16] proved that every MSO_1 predicate Π on vertex-labeled graphs can be decided in linear time on graphs of bounded clique-width, and hence on graphs of bounded NLC-width, whenever a decomposition tree is part of the input. The running time of the algorithm is $O(n)$, i.e., linear in the number n of vertices of the input graph, with constants hidden in the big-O notation that depend on the clique-width bound, on the number of labels, and on the MSO_1 formula encoding the predicate Π. Note

that this result does not hold for MSO_2 predicates, which is why our proof-labeling scheme applies to MSO_1 predicates only. We discuss the possible extension to MSO_2 properties in the conclusion (see Section 4).

For our purpose it is convenient to see the linear-time decision algorithm as a dynamic programming algorithm over the NLC-decomposition tree of the input graph. We formalize this dynamic programming approach following the vocabulary and notations used by Borie, Parker and Tovey [5]. Note that the latter provided an alternative proof of Courcelle's theorem, but for graphs of bounded tree-width, i.e., specific to a graph grammar defining tree-width. To design our proof-labeling scheme, we adapt their approach to a graph grammar defining NLC-width.

Homomorphism Classes. For a fixed property Π and a fixed parameter k, there is a finite set $\mathcal{C}$ of *homomorphism classes* (whose size depends only on Π and k) such that we can associate to each graph G of clique-width at most k its class $h(G) \in \mathcal{C}$ (for more details see the full version [35]). Whenever G is obtained from two graphs G_1 and G_2 by a $\bowtie_S$ operation potentially followed by a recoloring operation R, the class $h(G)$ only depends on $h(G_1)$, $h(G_2)$, $S \subseteq [k] \times [k]$, and $R : [k] \to [k]$. This property also holds whenever $\bowtie_S$ is replaced by $\|$. Moreover, we also extend the notion to arbitrary arity so that it holds for the NLC+ decomposition trees. Importantly, Courcelle's theorem [16] provides a "compiler" allowing to compute $h(G)$ whenever G is formed by a single vertex of color $j \in [k]$, and to compute $h(G)$ from $h(G_1)$, $h(G_2)$, S and R whenever $G = R(G_1 \bowtie_S G_2)$.

Checking Condition C2. In our proof-labeling scheme, for each node x of the NLC+ decomposition tree, we specify $h(x)$ as the class $h(G[x])$. Following the same principles as before, the consistency of these classes can be checked by simulating a bottom-up parsing of the decomposition tree, in a way very similar to what we described before for checking the consistency of $\mathsf{color}(x)$, but replacing the mere additions by updates of the homomorphism classes as described above.

This completes the rough description of our proof-labeling scheme.

3.6 Certificate Size

For each vertex v, the main, auxiliary, and service messages of v can be encoded using $O(\log n)$ bits for each node x in $\mathsf{path}(v)$, for the following reasons.

- The main message associated to a node x contains the following information. First, the list of operations described in the node, which can be encoded in $O(k^2)$ bits. Second, the corresponding index of links, which is just one bit representing whether x is the left or right children of its parent. Third, the homomorphism class $h(x)$ that can be encoded in $f(k)$ bits for some function f depending on the MSO_1 property under consideration – see the remark further in the text for a discussion about f. Finally, it includes the node identifier of the exit vertex of x, and the integers color_i for each $i \in [k]$. All these latter items can be encoded on $O(\log n)$ bits.
- The auxiliary message associated to node x (whenever $G[x]$ is connected) corresponds to the certification of a spanning tree of $G[x]$, which can be encoded in $O(\log n)$ bits (see [46]).
- For the service messages, note that vertex v participates in at most two service trees associated to x: the one of x (whenever $G[x]$ is disconnected), plus the one of the sibling y of x (when $G[y]$ is disconnected). Again, each of these trees can be certified using $O(\log n)$ bits.

Therefore, the total size of the certificates is $O(d \cdot \log n)$ bits, where d is the depth of the NLC+ decomposition tree T. Our final certificate size depends then on how much we can bound the depth d of T. Courcelle and Kanté [15] show that there always exists an NLC decomposition tree of logarithmic depth, but it comes with a price: the width of the small depth decomposition can be exponentially larger than the width of the original decomposition. Specifically, Courcelle and Kanté have shown that every n-node graph of NLC-width k admits an NLC-decomposition of width $k \cdot 2^{k+1}$ such that the corresponding decomposition tree T has depth $\mathcal{O}(\log n)$. Fortunately, our construction of NLC+ decomposition trees does not increase the depth of a given NLC-decomposition tree. In other words, we can use the result of Courcelle and Kanté to also show that NLC+ decomposition trees have logarithmic depth. Overall, we conclude that the certificate size is $O(\log^2 n)$ bits.

Remark. Our asymptotic bound on the size of the certificates hides a large dependency on the clique-width k of the input graph. For certifying the NLC+ decomposition only, the constant hidden in the big-O notation is single-exponential in k, given that the width of the NLC+ decomposition tree with logarithmic depth grows to $k \cdot 2^{k+1}$. However, for certifying an MSO_1 property, the dependency on k can be much larger, as it depends on the number of homomorphism classes. It is known that, for MSO_1 properties, the number of homomorphism classes is at most a tower of exponentials in k, where the height of the tower depends on the number of quantifiers in the MSO_1 formula. Moreover, this non-elementary dependency on k can not be improved significantly [40]. This exponential or even super-exponential dependency on the clique-width k is however inherent to the theory of algorithms for graphs of bounded clique-width. The same type of phenomenon occurs when dealing with graphs of bounded tree-width (see [40]), and the proof-labeling scheme in [36] is actually subject to the same type of dependencies in the bound k. On the other hand, the certificate size of our proof-labeling scheme grows only polylogarithmically with the size of the graphs.

4 Conclusion

In this paper, we have shown that, for every MSO_1 property Π on labeled graphs, there exists a PLS for Π with $O(\log^2 n)$-bit certificates for all n-node graphs of bounded clique-width. This extends previous results for smaller classes of graphs, namely graphs of bounded tree-depth [27], and graphs of bounded tree-width [36]. Our result also enables to establish a separation, in term of certificate size, between certifying C_4-free graphs and certifying P_4-free graphs.

A natural question is whether the certificate size resulting from our generic PLS construction is optimal. Note that one log-factor is related to the storage of IDs, and of similar types of information related to other nodes in the graph. It seems hard to avoid such a log-factor. The other log-factor is however directly related to the depth of the NLC-decomposition, and our PLS actually uses certificates of size $O(d \cdot \log n)$ bits for graphs supporting an NLC-decomposition of depth d. Nevertheless, the best generic upper bound for the depth d of an NLC-decomposition preserving bounded width is $O(\log n)$. This log-factor seems therefore hard to avoid too. Establishing the existence of a PLS for MSO_1 properties in graphs of bounded clique-width using $o(\log^2 n)$-bit certificates, or proving an $\Omega(\log^2 n)$ lower bound on the certificate size for such PLSs appears to be challenging.

Another interesting research direction is whether our result can be extended to MSO_2 properties. It is known that the meta-theorem from [16] does not extend to MSO_2. Nevertheless, this does not necessarily prevent the existence of compact PLSs for MSO_2 properties

on graphs of bounded clique-width. For instance, Hamiltonicity is an MSO_2 property that can be easily certified in *all* graphs, using certificates on just $O(\log n)$ bits. Is there an MSO_2 property requiring certificates of $\Omega(n^\epsilon)$ bits, for some $\epsilon > 0$, on graphs of bounded clique-width? Finally, it might be interesting to study the existence of compact distributed interactive proofs [44] for certifying MSO_1 or even MSO_2 properties on graphs of bounded clique-width. Note that the generic compiler from [47] efficiently applies to sparse graphs only whereas the family of graphs with bounded clique-width includes dense graphs.

References

1 Alkida Balliu, Gianlorenzo D'Angelo, Pierre Fraigniaud, and Dennis Olivetti. What can be verified locally? *J. Comput. Syst. Sci.*, 97:106–120, 2018.

2 Alkida Balliu and Pierre Fraigniaud. Certification of compact low-stretch routing schemes. *Comput. J.*, 62(5):730–746, 2019.

3 Aviv Bick, Gillat Kol, and Rotem Oshman. Distributed zero-knowledge proofs over networks. In *33rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2426–2458, 2022.

4 Hans L Bodlaender. Nc-algorithms for graphs with small treewidth. In *Graph-Theoretic Concepts in Computer Science: International Workshop WG'88 Amsterdam, The Netherlands, June 15–17, 1988 Proceedings 14*, pages 1–10. Springer, 1989.

5 Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(5&6):555–581, 1992. doi:10.1007/BF01758777.

6 Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Local certification of graph decompositions and applications to minor-free classes. In *25th International Conference on Principles of Distributed Systems (OPODIS)*, volume 217 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

7 Zvika Brakerski and Boaz Patt-Shamir. Distributed discovery of large near-cliques. *Distributed Comput.*, 24(2):79–89, 2011.

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

9 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. *Distributed Comput.*, 32(1):41–57, 2019.

10 Keren Censor-Hillel, Orr Fischer, Tzlil Gonen, François Le Gall, Dean Leitersdorf, and Rotem Oshman. Fast Distributed Algorithms for Girth, Cycles and Small Subgraphs. In *34th International Symposium on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 33:1–33:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

11 Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. In *24th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, LNCS 10641, pages 71–89. Springer, 2017.

12 Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005.

13 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

14 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic. A language-theoretic approach.* Encyclopedia of Mathematics and its applications, Vol. 138. Cambridge University Press, June 2012. Collection Encyclopedia of Mathematics and Applications, Vol. 138. URL: https://hal.science/hal-00646514.

15 Bruno Courcelle and Mamadou Moustapha Kanté. Graph operations characterizing rank-width and balanced graph expressions. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 66–75. Springer, 2007.

16 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. doi:10.1007/s002249910009.

17 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discret. Appl. Math.*, 101(1-3):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.

18 Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In *33rd International Symposium on Distributed Computing (DISC)*, volume 146 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

19 Konrad K. Dabrowski and Daniël Paulusma. Clique-width of graph classes defined by two forbidden induced subgraphs. *The Computer Journal*, 59(5):650–666, 2016.

20 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *33rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 367–376, 2014.

21 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.

22 Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. Sublinear-time distributed algorithms for detecting small cliques and even cycles. *Distributed Computing*, 35(3):207–234, 2022.

23 Yuval Emek and Yuval Gil. Twenty-two new approximate proof labeling schemes. In *34th International Symposium on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

24 Yuval Emek, Yuval Gil, and Shay Kutten. Locally restricted proof labeling schemes. In *36th International Symposium on Distributed Computing (DISC)*, volume 246 of *LIPIcs*, pages 20:1–20:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

25 Louis Esperet and Benjamin Lévêque. Local certification of graphs on surfaces. *Theor. Comput. Sci.*, 909:68–75, 2022.

26 Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. Three notes on distributed property testing. In *31st International Symposium on Distributed Computing (DISC)*, volume 91 of *LIPIcs*, pages 15:1–15:30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

27 Laurent Feuilloley, Nicolas Bousquet, and Théo Pierron. What can be certified compactly? compact local certification of MSO properties in tree-like graphs. In *41st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 131–140, 2022.

28 Laurent Feuilloley and Pierre Fraigniaud. Error-sensitive proof-labeling schemes. *J. Parallel Distributed Comput.*, 166:149–165, 2022.

29 Laurent Feuilloley, Pierre Fraigniaud, and Juho Hirvonen. A hierarchy of local decision. *Theor. Comput. Sci.*, 856:51–67, 2021.

30 Laurent Feuilloley, Pierre Fraigniaud, Juho Hirvonen, Ami Paz, and Mor Perry. Redundancy in distributed proofs. *Distributed Comput.*, 34(2):113–132, 2021.

31 Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and Ioan Todinca. Compact distributed certification of planar graphs. *Algorithmica*, 83(7):2215–2244, 2021.

32 Laurent Feuilloley and Juho Hirvonen. Local verification of global proofs. In *32nd International Symposium on Distributed Computing*, volume 121 of *LIPIcs*, pages 25:1–25:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

33 Pierre Fraigniaud, François Le Gall, Harumichi Nishimura, and Ami Paz. Distributed quantum proofs for replicated data. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 185 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

34 Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60(5):35:1–35:26, 2013.

35 Pierre Fraigniaud, Frédéric Mazoit, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. Distributed certification for classes of dense graphs, 2023. arXiv:2307.14292.

36 Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. A meta-theorem for distributed certification. In *29th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 13298 of *LNCS*, pages 116–134. Springer, 2022.

37 Pierre Fraigniaud and Dennis Olivetti. Distributed detection of cycles. *ACM Trans. Parallel Comput.*, 6(3):12:1–12:20, 2019.

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

39 Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. Distributed testing of excluded subgraphs. In *30th International Symposium on Distributed Computing (DISC)*, volume 9888 of *LNCS*, pages 342–356. Springer, 2016.

40 Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Log.*, 130(1-3):3–31, 2004. doi:10.1016/j.apal.2004.01.007.

41 François Le Gall, Masayuki Miyamoto, and Harumichi Nishimura. Brief announcement: Distributed quantum interactive proofs. In *36th International Symposium on Distributed Computing (DISC)*, volume 246 of *LIPIcs*, pages 48:1–48:3. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

42 Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory Comput.*, 12(1):1–33, 2016.

43 Ö. Johansson. Clique-decomposition, nlc-decomposition, and modular decomposition - relationships and results for random graphs. *Congressus Numerantium*, 132:39–60, 1998.

44 Gillat Kol, Rotem Oshman, and Raghuvansh R. Saxena. Interactive distributed proofs. In *37th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 255–264. ACM, 2018.

45 Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Comput.*, 20(4):253–266, 2007.

46 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Comput.*, 22(4):215–233, 2010.

47 Moni Naor, Merav Parter, and Eylon Yogev. The power of distributed verifiers in interactive proofs. In *31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1096–115. SIAM, 2020.

48 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.