A Meta-Theorem for Distributed Certification*

Pierre Fraigniaud[†1], Pedro Montealegre[‡2], Ivan Rapaport[3], and Ioan Todinca[4]

[1]IRIF, Université de Paris and CNRS, France. pierre.fraigniaud@irif.fr
[2]Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibañez, Santiago, Chile. p.montealegre@uai.cl
[3]DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Chile. rapaport@dim.uchile.cl
[4]LIFO, Université d'Orléans and INSA Centre-Val de Loire, France. ioan.todinca@univ-orleans.fr

Abstract

Distributed certification, whether it be *proof-labeling schemes*, *locally checkable proofs*, etc., deals with the issue of certifying the legality of a distributed system with respect to a given boolean predicate. A certificate is assigned to each process in the system by a non-trustable oracle, and the processes are in charge of verifying these certificates, so that two properties are satisfied: *completeness*, i.e., for every legal instance, there is a certificate assignment leading all processes to accept, and *soundness*, i.e., for every illegal instance, and for every certificate assignment, at least one process rejects. The verification of the certificates must be fast, and the certificates themselves must be small. A large quantity of results have been produced in this framework, each aiming at designing a distributed certification mechanism for specific boolean predicates. This paper presents a "meta-theorem", applying to many boolean predicates at once. Specifically, we prove that, for every boolean predicate on graphs definable in the monadic second-order (MSO) logic of graphs, there exists a distributed certification mechanism using certificates on $O(\log^2 n)$ bits in n-node graphs of bounded treewidth, with a verification protocol involving a single round of communication between neighbors.

Keywords: Proof-labeling scheme; Locally checkable proof; Fault-tolerance; Distributed decision.

1 Introduction

1.1 Context

Distributed certification is a concept that serves many purposes in distributed computing. One is fault tolerance. Indeed, the ability to certify the legality of a system-state with respect to some boolean predicate in a distributed manner guarantees that at least one process can launch a recovery procedure in case the system enters into an illegal state. Another application of distributed certification is safety. Indeed, distributed certification is a mechanism that guarantees that distributed algorithms dedicated to systems satisfying some specific property (e.g., algorithms dedicated to planar networks) can safely be used because, in case the system does not satisfy this property, at least one process can raise an alarm, and stop the computation.

Different certification mechanisms have been studied (cf. the related work section), all sharing the same principle. Distributed certification protocols involve a centralized *prover*, and a distributed *verifier*. The prover has complete knowledge of the system. It is computationally unbounded but not trustable. Given a boolean predicate $\mathcal{P}$ on system states, the prover assigns

*This work was partially done during the visit of the second and third authors to IRIF at Université de Paris, and LIFO at Université d'Orléans, partially supported by ANR project DUCAT and Fondecyt 1170021.

†Additional support for ANR projects QuData and DUCAT.

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

certificate to the processes, whose aim is to convince the processes that the system satisfies $\mathcal{P}$. The verifier is a distributed algorithm that runs at every process in the system, and is bounded to return a verdict (*accept* or *reject*) at each process after a limited communication among the processes. For instance, in a network, every processing node is bounded to communicate only once with its neighbors in the network before emitting its verdict.

To be correct, a distributed certification protocol for a boolean predicate $\mathcal{P}$ on system states must satisfy two properties. (1) Completeness: If the system satisfies $\mathcal{P}$, then there must exist a certificate assignment by the prover to the processes such that the verifier accepts at all processes. (2) Soundness: If the system does not satisfy $\mathcal{P}$, then, for every certificate assignment by the prover to the processes, it must be the case that the verifier rejects in at least one process. Network bipartiteness yields a simple example of distributed certification, using 1-bit certificates. For every bipartite network, every processing node in the network can be given a certificate 0 or 1, so that every processing node has a certificate different from the certificates assigned to its neighbors. The processing nodes can check these certificates in a single round of communication, where every processing node merely checks that the certificate of each of its neighbors is different from its own certificate. Completeness is satisfied by construction. Soundness is also satisfied. Indeed, if the network is not bipartite, then it is not 2-colorable. As a consequence, for every certificate assignment with certificates in $\{0, 1\}$, there are at least two neighboring processing nodes that receive the same certificate. These two processes will reject.

The main criterion measuring the quality of distributed certification is the *size* of the certificates. Indeed, the verification of $\mathcal{P}$ is typically performed frequently, for regularly checking that the system does satisfy $\mathcal{P}$, with the aim of reacting quickly if the system stops satisfying $\mathcal{P}$. As a consequence, there are frequent exchanges of certificates between the processes. Using small certificates limits the communication overhead caused by these exchanges.

1.2 Objective

A large collection of results related to distributed certification have been derived over the last twenty years (see Related Work), each result concerning a specific predicate. This paper is inspired by what has been achieved in the context of sequential computing where, instead of focusing on the design of an efficient algorithm for one specific problem, and then for another one, and so on and so forth, efforts have been made for deriving "meta-theorems", that is, results applying directly to large classes of problems. One prominent example is Courcelle's theorem [12] stating that every graph property definable in the monadic second-order (MSO) logic of graphs can be decided in linear time on graphs of bounded treewidth[1]. That is, even NP-hard problems such as vertex-coloring, minimum dominating set, minimum vertex cover, etc., have linear-time algorithms in the vast class of graphs with bounded treewidth. Each algorithm depends on the problem, but Courcelle's theorem essentially says that *every* problem expressible in the MSO logic has a linear-time algorithm in the class of graphs with bounded treewidth.

The objective of this paper is to address the existence of similar meta-theorems in the context of distributed certification applied to distributed computing in networks. Concretely, the question we address here is the following: is there a (large) class of boolean predicates on graphs for which one can guarantee the existence of a distributed certification mechanism with small certificates, say poly-logarithmic in the number of vertices of the graphs, for graphs taken from a (large) class of graphs?

1.3 Our Results

We present an analog of the aforementioned Courcelle's theorem in the context of distributed certification. Specifically, for every integer $k \geq 1$ and every MSO property φ on graphs, we

[1]Treewidth can be viewed as a measure capturing "how close" a graph is from a tree; roughly, a graph of treewidth k can be decomposed by a sequence of cuts, each involving a separator of size $O(k)$.

consider the following set:

$$\mathcal{P}_{k,\varphi} = \{\text{graph } G : (\mathsf{tw}(G) \leq k) \wedge (G \models \varphi)\},$$

where $\mathsf{tw}(G)$ is the treewidth of G. We provide a distributed certification mechanism for $\mathcal{P}_{k,\varphi}$ using certificates of poly-logarithmic size, as a function of the number n of vertices in the graphs. Specifically, given any network modeled as a connected simple graph $G = (V, E)$, with a process running at each vertex $v \in V$, our certification mechanism satisfies that $G \in \mathcal{P}_{k,\varphi}$ if and only if there is a certificate assignment to the vertices such that all vertices accept. The main result of the paper is the following.

Theorem 1 (Informal)**.** *For every $k \geq 1$ and every MSO property φ on graphs, there exists a distributed certification protocol for $\mathcal{P}_{k,\varphi}$ using certificates on $O(\log^2 n)$ bits.*

In fact, our theorem can be extended to properties including certifying solutions to maximization or minimization problems whose admissible solutions are defined by MSO properties. In the statement of Theorem 1, the big-O notation hides constants that depend only on k and φ. The theorem has many corollaries, as the universe of MSO properties is large. This includes predicates such as non 3-colorability, which is known to require certificates of quadratic size in arbitrary graphs [19], and diameter at most D, for a fixed constant D, which is known to require certificates of linear size in arbitrary graphs [11].

Corollary 1. *For every $c \geq 1$, there exists a distributed certification protocol for certifying* non c-colorability *in the family of graphs with bounded treewidth, using certificates on $O(\log^2 n)$ bits.*

For every $D \geq 1$, there exists a distributed certification protocol for certifying diameter at most D *in the family of graphs with bounded treewidth, using certificates on $O(\log^2 n)$ bits.*

Also, many natural graph families have bounded treewidth, as illustrated by the family of graphs excluding a planar graph as a minor, and thus we get the following corollary of Theorem 1.

Corollary 2. *For every planar graph H, and every MSO property φ on graphs, there exists a distributed certification protocol certifying φ in the family of H-minor-free graphs, using certificates on $O(\log^2 n)$ bits.*

Again, the big-O notation in the above statement hides constants that depend only on H and φ. Note that, as every 4-node graph is planar, Corollary 2 extends the recent results in [9], which applies to the families of graphs excluding a given 4-node graph H as a minor.

Interestingly, $\mathsf{tw}(G) \leq k$, and H-minor-freeness are themselves MSO properties for fixed k and H. It follows that Theorem 1 provides us with a distributed certification mechanism for treewidth and fixed-minor-freeness.

Corollary 3. *Let $k \geq 0$, and let H be a planar graph. There exist distributed certification protocols for certifying the class of graphs with treewidth at most k, and certifying the class of H-minor-free graphs, both using certificates on $O(\log^2 n)$ bits.*

Our Techniques. For establishing Theorem 1 we proceed in two steps. First, we provide a protocol for certifying 3-approximation of treewidth. Such a protocol satisfies the following: for any given $k \geq 1$, the protocol for k is such that, for every graph G,

$$\begin{cases} \mathsf{tw}(G) \leq k & \Rightarrow \text{ there exists a certificate assignment s.t. all vertices accept;} \\ \mathsf{tw}(G) > 3k+2 & \Rightarrow \text{ for every certificate assignment, at least one vertex rejects.} \end{cases}$$

Lemma 1 (Informal)**.** *For every $k \geq 1$ there exists a distributed protocol certifying a 3-approximation of the treewidth using certificates on $O(k^2 \log^2 n)$ bits.*

The proof of this lemma relies on a particular choice of a tree-decomposition, that we prove locally certifiable by "transferring" certificates between nodes that are far away from each other, which is typically the case of vertices in a same bag of the decomposition, without creating congestion.

Next, for any MSO property φ and integer k, we design a protocol which certifies $\mathcal{P}_{k,\varphi}$ on input graph G. The protocol exploits the tree decomposition in the proof of Lemma 1, for certifying a correct execution of a sequential dynamic programming algorithm for φ over this decomposition. Concretely, we design a distributed certification for a correct execution of a sequential dynamic programming algorithm a la Courcelle, using in fact the sequential MSO certification due to Borie, Parker and Tovey [8].

Lemma 2 (Informal)**.** *For every $k \geq 1$ and every MSO property φ on graphs, assuming given the certification protocol for 3-approximation k of treewidth from Lemma 1, there exists a distributed certification protocol for $\mathcal{P}_{k,\varphi}$ using additional certificates on $O(\log^2 n)$ bits.*

1.4 Related Work

The ability to detect illegal configurations of a distributed system was originally motivated by the design of fault-tolerant algorithms, especially self-stabilizing algorithms [1, 2, 21]. The notion of distributed certification as used in this paper originated from the seminal paper [23] defining *proof-labeling schemes* (PLS). We actually use a slight variant of PLS called *locally checkable proofs* (LCP) [19], which enables exchanging not only the certificates between the processing nodes, but also local states, including their IDs. Another related notion is *non-deterministic local decision* (NLD) [17] in which the certificates must not depend on the IDs given to the processing nodes. Distributed certification has been extended to various directions, including randomized PLS [18], approximate PLS [11, 14], local hierarchies [3, 15], interactive proofs [22, 25], and even, recently, zero-knowledge distributed certification [4]. All the aforementioned papers contain a vast collection of certification results for various graph problems. In these papers, each certification protocol is specific of the problem at hand. To our knowledge, the only "meta-theorem" in the context of distributed certification is the recent paper [10], which shows that every MSO formula can be locally certified on graphs with bounded *treedepth* using certificates on $O(\log n)$ bits. We show that the same result holds for the larger class of graphs with bounded *treewidth*, to the cost of slightly larger certificates, on $O(\log^2 n)$ bits. We are therefore partially answering the questions raised in [10], asking whether it is "possible to certify any MSO formula on bounded treewidth graphs", and "to certify that the graph itself has treewidth at most k", using small certificates.

In framework of sequential algorithms, there is a large literature on "meta-theorems" proving that large families of combinatorial properties (typically expressed using some form of logic formulae) can be efficiently decided on particular graph classes. In addition to Courcelle's (meta) theorem [12] on MSO properties on graphs with bounded treewidth, it is worth mentioning the recent results establishing that properties expressible in *first-order logic* can be verified in polynomial time on graphs of bounded *twinwidth* [7], as well as on *nowhere-dense* graphs [20]. Both graph classes include planar graphs, and thus include graphs with arbitrarily large treewidth. Our work is participating to the general objective of extending these results to the framework of distributed computing.

2 Preliminaries

2.1 Distributed Certification

We consider networks modeled as connected simple graphs. Every vertex is a processing element, and the vertices exchange messages along the edges of the graph. We systematically denote by

n the number of vertices in the considered graph. The vertices of a network/graph $G = (V, E)$ are given distinct identifiers (IDs), and we denote by $\mathsf{ID}(v)$ the identifier of vertex $v \in V$. These identifiers are not necessarily between 1 and n, but we adopt the standard assumption stating that IDs can be stored on $O(\log n)$ bits.

We consider boolean predicates on labeled graphs, i.e., graphs for which every vertex v is given a label $\ell(v) \in \{0,1\}^*$. These labels may represent a way to mark vertices (e.g., those in a dominating set), a color (e.g., in graph coloring), or any value depending on the graph property at hand. Given a boolean predicate $\mathcal{P}$ on labeled graphs, a *locally checkable proof* [19] for $\mathcal{P}$ is a prover-verifier pair. The prover is a non-trustable oracle with unbounded computing power. Given any labeled graph (G, ℓ), the prover assigns a certificate $c(v) \in \{0,1\}^*$ to every vertex $v \in V$. The verifier is a 1-round distributed algorithm running at all vertices of the graph. Given a labeled graph (G, ℓ) with a certificate assigned at every vertex, the vertices exchange their identifiers, labels, and certificates, between neighbors, and compute an output, accept or reject. To be correct, the pair prover-verifier must satisfy two conditions:

Completeness: If $(G, \ell) \models \mathcal{P}$, then, for every ID-assignment to the vertices, there must exist a certificate assignment by the prover to the vertices such that the verifier accepts at all vertices.

Soundness: If $(G, \ell) \not\models \mathcal{P}$, then, for every ID-assignment to the vertices, and for every certificate assignment by the prover to the vertices, it must be the case that the verifier rejects in at least one vertex.

2.2 Tree Decompositions and Terminal Recursive Graphs

Let us recall the classical definition of treewidth and tree decompositions, due to Robertson and Seymour [26].

Definition 1. A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, B) where $T = (I, F)$ is a tree, and $B = \{B_i, i \in I\}$ is a collection of subsets of vertices of G, called *bags*, such that the following conditions hold:

- For every $v \in V$, there exists $i \in I$ such that $v \in B_i$;
- For every $e = \{u, v\} \in E$ there is $i \in I$ such that $\{u, v\} \subseteq B_i$;
- For every $v \in V$, the set $\{i \in I : v \in B_i\}$ forms a connected subgraph of T.

The *width* of a tree decomposition is the maximum size of a bag, minus one. The *tree-width* of a graph G, denoted by $\mathsf{tw}(G)$, is the smallest width of a tree decomposition of G.

To facilitate the distinction between the original graph $G = (V, E)$ and the decomposition tree $T = (I, F)$, we will speak of the *nodes* $i \in I$ of T and of the *vertices* $v \in V$ of G.

We consider tree decompositions as rooted, i.e., we fix some node $r \in I$ as the root of $T = (I, F)$. For a node $i \in I \setminus \{r\}$, we denote by $p(i)$ its parent in T, and set $p(r) = \perp$. For $i \in I$, we denote by T_i the subtree of T rooted in i, and by V_i the subset of vertices of G in the bags of T_i, i.e., $V_i = \cup_{j \in V(T_i)} B_j$. Also, for $i \in I \setminus \{r\}$, we define $F_i = B_i \setminus B_{p(i)}$. For the root r, we set $B_{p(r)} = \varnothing$ and $F_r = B_r$. Given a rooted tree $T = (I, F)$, and two nodes of $i, j \in I$, we denote by $j \preceq i$ the property that j is a descendant of i in T.

Graphs of bounded treewidth can also be defined recursively, based on a graph grammar. Let w be a positive integer. A *w-terminal graph* is a graph (V, E) together with a *totally ordered* set $W \subseteq V$ of at most w distinguished vertices. Vertices of W are called the *terminals* of the graph, and we denote by $\tau(G)$ the number of its terminals. Since W is totally ordered, we can speak of the rth terminal, for $1 \leq r \leq w$. Since in our case vertices are given distinct identifiers, one can view W as ordered w.r.t. these identifiers.

The class of *w-terminal recursive graphs* is defined starting from *w-terminal base graphs* through a sequence of *composition operations*. A *w-terminal base graph* is a w-terminal graph of the form (V, W, E) with $W = V$. A *composition operation* f acts on one or two w-terminal graphs producing a new w-terminal graph as follows.

When f is of arity 2, graph $G = f(G_1, G_2)$ is obtained by firstly making disjoint copies of the two graphs G_1 and G_2, then "glueing" together some terminals of G_1 and G_2. The glueing performed by f is represented by a matrix $m(f)$ having $\tau(G) \leq w$ rows and two columns, with integer values between 0 and $\tau(G)$. At row r of the matrix, $m_{rc}(f)$ indicates which terminal of each $G_c, c \in \{1, 2\}$ is identified to terminal number r of graph G. If $m_{rc}(f) = 0$, then no terminal of G_c is identified to terminal r of G (in particular, if $m_{r1}(f) = m_{r2}(f) = 0$ it means that terminal r of G is a new vertex, but this situation will not occur in our constructions). Moreover, a terminal of G_c is identified to at most one terminal of G, i.e., each non-zero value in $1, \ldots, \tau(G_c)$ appears at most once in column c of $m(f)$. For an illustration, see, e.g., Figure 2.

When f is of arity 1, the corresponding matrix $m(f)$ has a unique column. Graph $G = f(G_1)$ is obtained as before, by identifying terminal m_{i1} of G_1 to terminal r of G. Note that in this case G and G_1 have exactly the same vertex and edge sets, and the terminals of G form a subset of the terminals of G_1.

We point out that the number of possible different matrices and hence of different operations is bounded by a function on w.

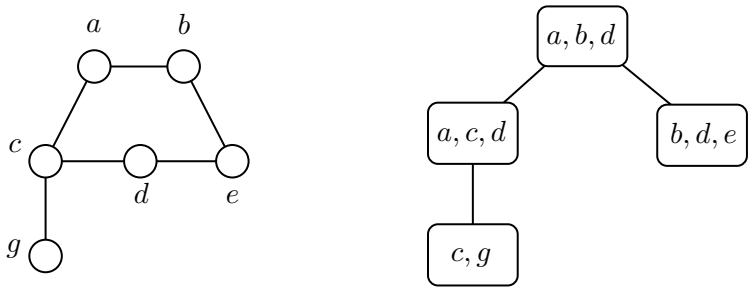

Figure 1: Graph G and a tree decomposition.

Proposition 1 (Theorem 40 in [6]). *Graph $H = (V, W, E)$ is $(w+1)$-terminal recursive if and only if there exists a tree decomposition of $G = (V, E)$, of width at most w, having W as root bag. Hence the grammar of $(w+1)$-terminal recursive graphs constructs exactly the graphs of treewidth at most w.*

Let us sketch briefly here how a tree decomposition of $G = (V, E)$ of width w can be transformed into a $(w+1)$-expression of the same graph. To each node i of the tree decompositions, we associate three $(w+1)$-terminal graphs:

- $G_i^b = (B_i, B_i, E(G[B_i]))$, the $(w+1)$-terminal base graph corresponding to graph $G[B_i]$ induced by bag B_i;
- $G_i = (V_i, B_i, E(G[V_i]))$, corresponding to $G[V_i]$, with bag B_i as set of terminals;
- If i differs from the root, $G_i^+ = (V_i \cup B_{p(i)}, B_{p(i)}, E(G[V_i \cup B_{p(i)}]))$ corresponding to the graph induced by $V_i \cup B_{p(i)}$, with $B_{p(i)}$ as set of terminals.

Let us describe how to compute the $(w+1)$-expression of these graphs, by parsing bottom-up the tree decomposition (see also Figure 2 applied to the tree decomposition of Figure 1).

When i is a leaf, $G_i = G_i^b$ is a $(w+1)$-terminal base graph. Assume now that i is not a leaf and let $Children(i)$ be the children of node i in the decomposition tree. For each $j \in Children(i)$, we already possess an expression of the $(w+1)$-terminal graph $G_j = (V_j, B_j, E(G[V_j]))$. Observe

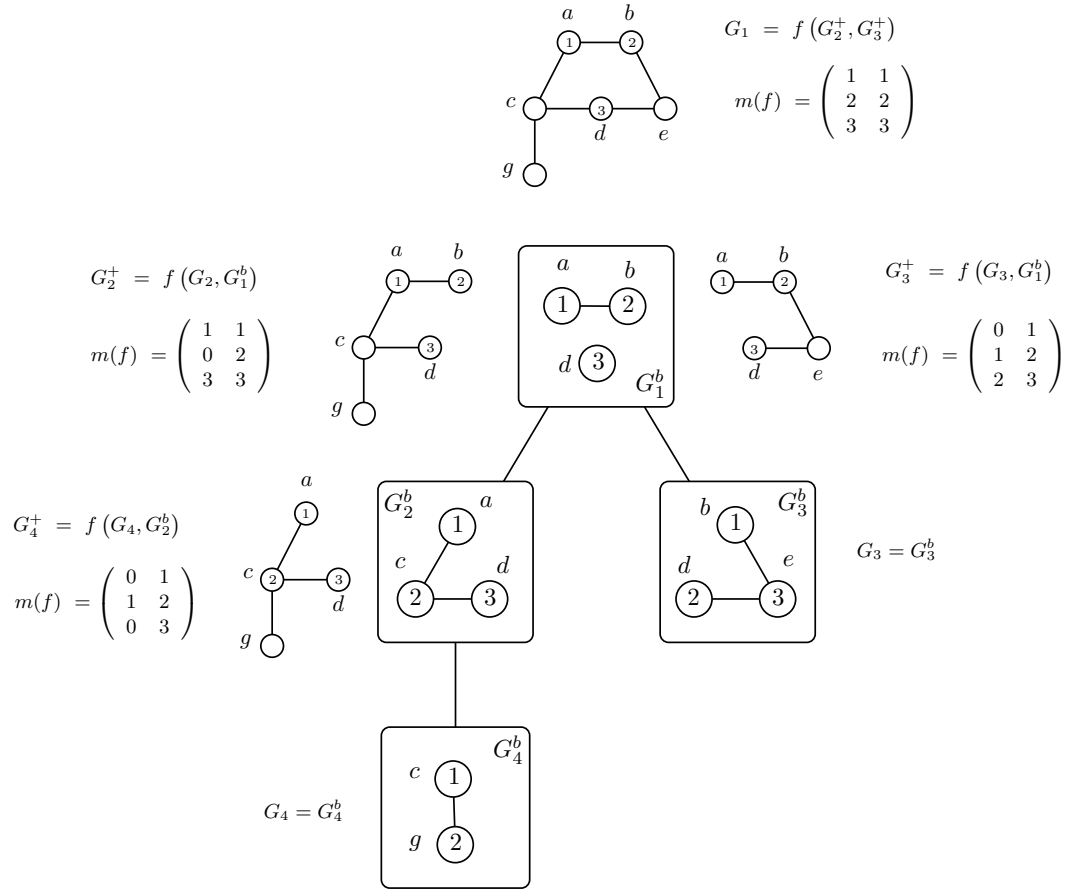

Figure 2: From a tree decomposition of width w to a $(w+1)$-expression.

that G_j^+ is obtained from a glueing of G_j and the base graph G_i^b, where the terminals of G_j contained in $B_j \cap B_i$ are glued on the corresponding terminals of G_j^+, and the others become non-terminals. Eventually, if i has more than one child, then G_i is obtained by the consecutive glueing of all G_j^+, $j \in Children(i)$, where the glueing is performed on B_i by the same matrix $m(f)$ having $m_{r1}(f) = m_{r2}(f) = r$, for $1 \leq r \leq |B_i|$.

2.3 Regular Properties and MSO

We consider graph properties $\mathcal{P}(G)$ assigning to each graph G a boolean value. We have in mind properties expressible in Monadic Second Order Logic, like "G is not 3-colourable", "G does not contain a given minor", etc. Nevertheless, technically, we do not need the definition of MSO formulae, and the interested reader may refer to [13]; we only need the fact that MSO properties are *regular*, in the sense defined below. By Courcelle's theorem, such properties can be decided in linear (sequential) time on graphs of bounded treewidth, if the tree decomposition (or the corresponding expression as a terminal recursive graph) is part of the input.

Definition 2 (regular property)**.** A graph property $\mathcal{P}$ is called *regular* if, for any value w, we can associate a finite set $\mathcal{C}$ of *homomorphism classes* and a *homomorphism function* h, assigning to each w-terminal recursive graph G a class $h(G) \in \mathcal{C}$ such that:

1. If $h(G_1) = h(G_2)$ then $\mathcal{P}(G_1) = \mathcal{P}(G_2)$.
2. For each composition operation f of arity 2 there exists a function $\odot_f : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ such that, for any two w-terminal recursive graphs G_1 and G_2,

$$h(f(G_1, G_2)) = \odot_f(h(G_1), h(G_2))$$

and for each composition operation f of arity 1 there is a function $\odot_f : \mathcal{C} \to \mathcal{C}$ such that, for any w-terminal recursive graph G,

$$h(f(G)) = \odot_f(h(G)).$$

We illustrate this definition on the property "G is not 3-colourable". We can choose, as homomorphism $h(G = (V, W, E))$, the set of all three-partitions (W_1, W_2, W_3) of the set W of terminals, such that graph G has, as three colouring, the one where each colour $i \in \{1, 2, 3\}$ intersects W exactly in the set W_i. Observe that graph G satisfies the property of not being 3-colourable if and only if its homomorphism class is the empty set. It is a matter of exercise to figure out how to compute the homomorphism class of a base w-terminal graph (by enumerating all its three-partitions into independent sets), and how to compute functions $\odot_f$ updating the class of the graph after a composition operation f.

The first condition of Definition 2 separates the classes into *accepting* ones (i.e., classes $c \in \mathcal{C}$ such that $h(G) = c$ implies that $\mathcal{P}(G)$ is true) and *rejecting* ones (i.e., classes $c \in \mathcal{C}$ such that $h(G) = c$ implies that $\mathcal{P}(G)$ is false). In full words, the second condition states that, if we perform a composition operation on two graphs (resp. one graph), the homomorphism class of the result can be obtained from the homomorphism classes of the graphs on which these operations are applied. Therefore, if a w-terminal recursive graph is given together with its expression in this grammar, and if moreover we know how to compute the homomorphism classes of the base graphs and the composition functions $\odot_f$ over all possible composition operations f, then the homomorphism class of the whole graph for a regular property $\mathcal{P}$ can be obtained by dynamic programming. We simply need to parse the expression from bottom to top and, at each node, we compute the class of the corresponding sub-expression thanks to the second condition of regularity. At the root, the property is true if and only if we are in an accepting class.

Proposition 2 ([8, 12])**.** *Any property $\mathcal{P}$ expressible by a MSO formula is regular. Moreover, given the MSO formula φ and parameter w, one can explicitely compute the set of classes, the homomorphism function for all w-terminal base graphs as well as the composition functions $\odot_f$ of all possible composition operations f.*

Altogether, this provides an effective algorithm for checking property $\mathcal{P}(G)$ in $O(n)$ time, by a sequential algorithm, given the w-expression (or, equivalently, the tree decomposition of width $w - 1$) of the input graph, by computing bottom-up the homomorphism classes.

The notions of MSO and regular properties extend to properties on graphs and vertex subsets, i.e., we can consider properties $\mathcal{P}(G, X)$ assigning to each graph G and vertex subset X of G a boolean value. This allows to capture properties as "X is an independent set of G", or "X is an dominating set of G". Moreover, the whole framework can capture the problem of computing a (or, in our case, certifying that) set X is of maximum weight among those satisfying $\mathcal{P}(G, X)$, for graphs with polynomial weights on their vertices. This issue is postponed to Appendix B.

2.4 Coherent Tree Decompositions

By a classic result of Bodlaender [5], an optimal tree decomposition of graph G can be transformed into a decomposition whose tree is of logarithmic depth, while the size of the bags is at most multiplied by 3. We strongly rely on such decomposition, plus a connectivity property that we call *coherence*. We say that a rooted tree decomposition of a graph $G = (V, E)$ is *coherent* if for every $i \in I$, the set F_i is non empty and the graph $G[V_i \setminus B_{p(i)}]$ is connected.

We show that such a decomposition exists and provide some of its properties used in our certification protocol. Due to space restrictions, the proofs of the results of this sub-section can be found in Appendix A.1.

Lemma 3. *Let $k \geq 1$, and let G be a connected n-vertex graph of treewidth at most k. Then, G admits a coherent tree decomposition of width at most $3k + 2$ and depth $\mathcal{O}(\log n)$.*

In our protocol we must be able to communicate, for any node i of the decomposition, some information about V_i to a vertex in the bag corresponding to the parent node $p(i)$, more precisely, to some vertex of $F_{p(i)}$. The following lemma shows the existence a vertex $\ell_i \in V_i \setminus B_{p(i)}$ adjacent in G to some vertex $w \in F_{p(i)}$.

Lemma 4. *Let $T = (I, F)$ be a coherent tree decomposition of $G = (V, E)$. Then, for every $i \in I$ different from the root there exists a pair of vertices $\ell_i \in V_i \setminus B_{p(i)}$ and $w \in F_{p(i)}$ such that $\{w, \ell_i\} \in E$.*

Vertex ℓ_i is called the exit vertex of i, *and w is called the* vertex of $F_{p(i)}$ in charge of node i.

In our certification protocols, for each node i of the decomposition tree, the vertices of F_i as well as the exit vertex ℓ_i will receive from the prover some information concerning graph $G_i = G[V_i]$. We will need to ensure that ℓ_i and all vertices of F_i received the same information. For this purpose we use trees contained in $G[V_i \setminus B_{p(i)}]$, spanning ℓ_i and F_i.

Lemma 5. *Consider a coherent tree decomposition $T = (I, F)$ of graph $G = (V, E)$, of depth $O(\log n)$. For each node i of the decomposition tree, there is a subtree $S(i)$ of $G[V_i \setminus B_{p(i)}]$ spanning F_i and the exit vertex ℓ_i.*

Moreover each vertex of G appears $O(\log n)$ times in the family of trees $\mathcal{T}(G) = \{S(i) \mid i \in I\}$.

3 A Protocol Certifying a 3-Approximation of the Treewidth

In this section we describe a protocol certifying a 3-approximation of treewidth. More precisely, we prove the following Lemma.

Lemma 6. *For each $k \geq 1$ there is a distributed certification protocol that uses messages of size $O(k^2 \log^2 n)$ and ensures, for any input graph G, that:*

$$\begin{cases} \mathsf{tw}(G) \leq k & \Rightarrow \text{ there exists a certificate assignment s.t. all nodes accept;} \\ \mathsf{tw}(G) > 3k+2 & \Rightarrow \text{ for every certificate assignment, at least one node rejects.} \end{cases}$$

Let us describe the messages that the prover sends to each vertex of G, if $\mathsf{tw}(G) \leq k$. These messages describe a coherent tree decomposition of width at mots $3k + 2$ and of logarithmic depth, which exists by Lemma 3.

We identify node i of the decomposition tree with the number corresponding to a binary representation of the set of vertices B_i contained in its bag, so $1 \leq i \leq n^{\mathcal{O}(k)}$. In full words, a node is simply identified by the content of its bag, which is possible since coherent tree decompositions have pairwise disjoint bags.

Our protocol distinguishes two types of certificates, namely *main messages* and *auxiliary messages*. Each vertex receives one main messages and $\mathcal{O}(\log n)$ auxiliary messages. Let us describe each one of them.

Main messages. These messages are used to encode a tree decomposition, following Definition 1. Each vertex v receives as a certificate the following messages, that we denote $m(v)$:

1. A number $d = d(v)$, representing the depth of the node i such that $v \in F_i$
2. A list of sets $\mathcal{B}(v) = B_d(v), B_{d-1}(v), \ldots, B_1(v)$, representing the path of bags from node $i = B_d(v)$ to the root node.
3. The list of sets $\mathcal{F}(v) = F_d(v), F_{d-1}(v), \ldots, F_1(v)$, representing the sets $F_j(v) = B_j(v) \setminus B_{j-1}(v)$, for each $j \in \{1, \ldots, d\}$.
4. A list of sets $\mathcal{E}(v) = E_d(v), \ldots, E_1(v)$, where, for each $j \in \{1, \ldots, d\}$, $E_j(v) \subseteq \binom{B_j(v)}{2}$ represents the edge set of $G[B_j(v)]$.

Observe that the size of a main message is $\mathcal{O}(k^2 \log^2 n)$.

Auxiliary messages. These messages allow to check the consistency of the main messages between vertices of a same set F_i, for each node i of the decomposition.

From Lemma 5, we have that for each node i there is a subtree $S(i)$ connecting all pair of vertices of F_i and the exit vertex ℓ_i. The vertices w of $S(i)$ are called *auxiliary vertices for* i. For a vertex w, let us call $Aux(w)$ the set of nodes i such that w is an auxiliary vertex for i. From Lemma 5, we know that for each $w \in V$, $|Aux(w)| = \mathcal{O}(\log n)$.

Each node w receives the set $Aux(w)$ and for each $i \in Aux(w)$ the message $m_{aux}(w, i)$ containing the following information where

- $d_{aux}(w, i)$ is the depth of node i.
- $\ell_i(w)$ is a vertex identifier of the exit vertex of F_i (cf. Lemma 4).
- $\alpha_i(w)$ is a vertex identifier of the vertex in $F_{p(i)}$ in charge of B_i (cf. Lemma 4).
- $F_i(w)$ is a set of vertices, representing F_i.
- $TreeCert(w)$ is the certificate that receives w in the protocol used to verify that $S(i)$ is a tree rooted at ℓ_i and spanning $F_i(w)$. More precisely $cert(F_i, w) = (parent(w), dist(w), sub(w))$, where:
  - $parent(w)$ represent the parent of w in $S(i)$ ($parent(w) = \perp$ if $w = \ell_i(w)$),
  - $dist(w)$ represents the distance from w to ℓ_i in $S(i)$, and
  - $sub(w)$ represents is the subset of $F_i(v)$ that are descendants of w in $S(i)$.

Observe that for any given vertex w and node i, the messages $m_{aux}(w, i)$ is of size $O(k \log n)$. Thanks to Lemma 5, a vertex w appears $\mathcal{O}(\log n)$ times as auxiliary vertex of some node i. Therefore, a vertex w receives in total $O(k \log^2 n)$ bits for auxiliary messages.

Verification round. Given two vertices u and v such that $d(u) \leq d(v)$, we say that the main message of u is a d-suffix of the main message of v if $B_j(u) = B_j(v)$ and $E_j(u) = E_j(v)$ for each $j \in \{1, \ldots, d\}$.

Let $d = d(v)$. In the verification round, vertex v verifies the following conditions.

Consistency of the tree decomposition.

1. The size of each $B \in \mathcal{B}(v)$ is at most $3k + 3$.
2. The set $F_d(v)$ contains v.
3. For each $j \in \{2, \ldots, d\}$, the set $F_j(v)$ equals $B_j(v) \setminus B_{j-1}(v)$.
4. For each $w \in V(G)$ and $j_1, j_2 \in \{1, \ldots, d\}$ with $j_1 < j_2$, if $w \in B_{j_1} \cap B_{j_2}$, then $w \in B_{i_j}$ for every $j \in \{j_1 + 1, \ldots, j_2 - 1\}$.
5. For each $j_1, j_2 \in \{1, \ldots, d\}$, each pair of vertices $u_1, u_2 \in B_{j_1}(v) \cap B_{j_2}(v)$ satisfies that $\{u_1, u_2\} \in E_{j_1}(v) \iff \{u_1, u_2\} \in E_{j_2}(v)$.
6. For each $u \in B_d(v)$, v checks that $\{u, v\} \in E \iff \{u, v\} \in E_d(v)$.
7. For each $u \in N(v)$ such that $d(u) \geq d(v)$, v checks that $m(v)$ is a $d(v)$-suffix of $m(u)$.
8. For each $u \in N(v)$ such that $d(u) \leq d(v)$, v checks that $u \in B_d(v)$.
9. v checks that it is an auxiliary vertex for $B_d(v)$ and that it has a neighbor that is also an auxiliary vertex for $B_d(v)$.
10. For each vertex $w \in N(v) \cup \{v\}$ such that w is an auxiliary tree vertex for $B_d(v)$, v checks that $d_{aux}(w, B_d(w)) = d$ and $F_i(w) = F_d(v)$.

Consistency of the auxiliary trees and the exit vertex. The following conditions are used to verify that the nodes marked as auxiliary vertices for node i form an auxiliary subtree $S(i)$ rooted at ℓ_i and spanning F_i. At the same time, we check that all de nodes in $S(i)$ have the same auxiliary information, corresponding to the depth d_i of bag i, the contents of F_i, the identity of exit vertex ℓ_i, and the identity of the node of $F_{p(i)}$ responsable of i, and the same d_i-suffix of the main messages.

For each $i \in Aux(v)$, vertex v checks te following conditions

11. For each vertex $w \in N(v)$ such that w is an auxiliary tree vertex for i, v checks that

$$(d_{aux}(w,i), \ell_i(w), \alpha_i(w), F_i(w)) = (d_{aux}(v,i), \ell_i(v), \alpha_i(v), F_i(v))$$

12. $d_{aux}(v,i) \leq d(v)$.

13. Uses $TreeCert(F_i(v), v)$ to verify that there is an auxiliary tree $S(i)$ rooted in $\ell_i(v)$ and spanning $F_i(v)$. More precisely, v checks the following conditions:

    (a) If $v \neq \ell_i(v)$ then v has a neighbor with the label $parent(w)$ which is also an auxiliary vertex for i;

    (b) If $v \neq \ell_i(v)$, then $dist(parent(v)) = dist(v) - 1$;

    (c) If $v = \ell_i$ then $dist(v) = 0$, $sub(v) = F_i(v)$, v is adjacent to $\alpha_i(v)$ and $d(\alpha_i(v)) = d_{aux}(v,i) - 1$.

    (d) Set $sub(v)$ is the union of all sets $sub(w)$ over the children w of v in $S(i)$ (i.e., for all w such that $parent(w) = v$), plus vertex v itself if $v \in F_i$.

Soundness and completeness. We now analyze the correctness of the protocol. The completeness follows directly by Lemmas 3, 4 and 5. In the following, we prove the soundness.

Soundness: Let us assume that all vertices accept a given certificate in the verification round. We now show that necessarily $\mathsf{tw}(G) \leq 3k+2$. For each node $v \in V$, let us call $B(v)$ and $F(v)$ the set $F_{d(v)}(v)$ and $B_{d(v)}(v)$, respectively. We say that a vertex v is in depth d if $d(v) = d$. The proof of the soundness is a consequence of the following claims.

Claim 1: For each $i \in Aux(v)$, there is a tree $S(i)$ rooted in $\ell_i(v)$ spanning $F_i(v)$. Moreover, all the vertices in $S(i)$ are in a depth greater or equal than $d_{aux}(v,i)$, and their main messages have the same $d_{aux}(v,i)$-suffix.

Proof of Claim 1. First, observe that by the verification of condition **13 (a)-(c)**, we have that $S(i)$ is defined by the set of all auxiliary vertices for i and the edges $\{w, parent(w)\}$. Since $S(i)$ is connected, by conditions **10** and **11**, all auxiliary vertices for node i agree in the same $F_i = F_i(v)$ and in the depth of i given by $d_{aux} = d_{aux}(v,i)$. By condition **13 (c)-(d)**, all vertices in F_i exist and are auxiliary vertices for node i. Finally, by condition **12** all nodes are in a depth greater or equal than d_{aux} and by condition **7**, the main messages of all vertices in $S(i)$ have the same d_{aux}-suffix. □

Claim 2: For every vertex v, all nodes in $F(v)$ receive the same main messages as v.

Proof of Claim 2. Let u be a vertex in $F(v)$. If u and v are adjacent the claim is true by condition **7**. Suppose then that $u \notin N(v)$. Since v verifies condition **9**, there is a set of auxiliary vertices for node $i = B(v)$. By **Claim 1**, $m(v)$ is a $d(v)$-suffix of $m(w)$, for every auxiliary vertex w for node i. Since all vertices in $F(v)$ are auxiliary vertices for i, we deduce that u has the same main messages than v. □

Claim 3: For every pair of vertices $u, v \in V$ either $F(v) = F(u)$ or $F(v) \cap F(u) = \emptyset$.

Proof of Claim 3. This is a direct corollary of **Claim 2**. Indeed, let us suppose that there exist a pair $u, v \in V$ such that $F(v) \neq F(u)$ but $F(v) \cap F(u) \neq \emptyset$. Then, without loss of generality, there is a node $w \in F(v) \cap F(u)$ such that $F(w) \neq F(v)$, which contradicts **Claim 2**. □

Claim 4: For every vertex v such that $d(v) > 1$, there exist a node u such that $m(u)$ is a $(d(v)-1)$-suffix of $m(v)$.

Proof of Claim 4. Let $d = d(v)$. **Claim 1** implies that the exit vertex ℓ_i for $i = B_d(v)$ exists and is the root of $S(i)$, which is in a depth greater or equal than $d_{aux} = d$. Condition **13 (c)** implies that ℓ_i is adjacent to a node α_i of depth $d-1$. Then, by condition **7**, $m(\alpha_i)$ is a $d-1$-suffix of $m(\ell_i)$. Since $m(v)$ is a d-suffix of $m(\ell_i)$, we deduce that $m(\alpha_i)$ is a $d-1$-suffix of $m(v)$. □

Claim 5: For every $u, v \in V$, the sets $F(u) \neq F(v)$ if and only if $B(u) \neq B(v)$.

Proof of Claim 5. First, observe that if $F(u) = F(v)$, then by condition **2** and **Claim 2**, $B(v) = B(u)$. For the reciprocal, let us suppose by contradiction that there exist $u, v \in V$ such that $F(u) \neq F(v)$ and $B(u) = B(v)$. Let us call $d_1 = d(u)$ and $d_2 = d(v)$. Since $F(u) \neq F(v)$, necessarily $B_{d_1-1}(u) \neq B_{d_2-1}(v)$. Let us assume, without loss of generality, that there exists a vertex $w \in F(v) \setminus F(u)$. Since w belongs to $F(v)$, we have that $F(w) = F(v)$ by **Claim 2**, and w does not belong to $B_{d_1-1}(v)$. Since $w \notin F(u)$ we have that w belongs to $B_{d_2-1}(u)$. Let us call d_3 the maximum in $\{1, \ldots, d_1 - 1\}$ such that $B_{d_3}(u)$ belongs to $\mathcal{B}(v)$. Observe that d_3 exists, because applying condition **7** on all the vertices in G we deduce that $B_1(u) = B_1(v)$. If $B_{d_3}(u)$ contains w, then v fails to verify condition **4**. If $B_{d_3}(u)$ does not contain vertex w, there exists a $d_4 \in \{d_1, \ldots, d_3 - 1\}$ such that $w \in F_{d_4}(u) = B_{d_4}(u) \setminus B_{d_4-1}(u)$. Then, **Claim 4** applied to the vertices in the sequence $F_{d_1}(u), F_{d_1-1}(u), ..., F_{d_4}(u)$ implies that there is a node w' such that $F(w') = F_{d_4}(u)$. Then, by **Claim 2**, $F(w) = F_{d_4}(u)$. We deduce that $B(v) = B_{d_4}(u)$, which is a contradiction with the choice of d_3. □

Let us define I as the set of indexes $i \in [n^{\mathcal{O}(k)}]$ for which there is a $v \in V(G)$ such that i is the binary representation of $B(v)$. By **Claim 2**, **3** and **5**, we have a partition $\{F_i\}_{i \in I}$ of $V(G)$, such that, for each $i \in I$, all nodes in F_i receive the same main messages. In particular, for every vertex v in F_i, we have that i is the binary representation of $B(v)$. For each $v \in F_i$, we define $p(i)$ as the binary representation of $B_{d(v)-1}(v)$ ($p(i) = \perp$ if $v \in B_1(v)$). From **Claim 4** we know that the binary representation of $B_{d(v)-1}(v)$ is also in I. In other words, the nodes in $F_{d(v)-1}(v)$ have certificates that are consistent with the certificate of v. In particular, all vertices of G agree on the contents of the root node, that we call B_1. We then define the pair $(T, \{B_i\}_{i \in I})$, where T is defined by the tree with vertex set I and edge set $\{i, p(i)\}$, for each $i \in I$ different than the root.

Claim 6: The pair $(T, \{B_i\}_{i \in I})$ forms a tree decomposition of G of width $3k + 2$.

Proof of Claim 6. According to Definition 1 we have to check that the following three properties are satisfied:

- For every $v \in V$, there exists $i \in I$ such that $v \in B_i$;
- For every $e = \{u, v\} \in E$ there is $i \in I$ such that $\{u, v\} \subseteq B_i$;
- For every $v \in V$, the set $\{i \in I : v \in B_i\}$ forms a connected subgraph of T.

The first two properties are directly verified as every vertex is given one bag that contains it in the main message. The second property is verified by condition **8**. Finally, for the third condition, let us suppose that there exists a vertex $v \in V$ such that $I_v = \{i \in I : v \in B_i\}$ is not connected. Let C_1 and C_2 be two different components of I_v, and let i_1 and i_2 be, respectively, the nodes in C_1 and C_2 of minimum depth. Observe that $F_{i_1} \neq F_{i_2}$ and by condition **3**, v must

be contained in $F_{i_1} \cap F_{i_2}$, which contradicts **Claim 2**. We deduce that for every $v \in V$, the set $\{i \in I : v \in B_i\}$ forms a connected subgraph of T. We conclude that $(T, \{B_i\}_{i \in I})$ forms a tree decomposition of G. Finally, the width of the decomposition is verified by condition **1**. □

We finish this section showing one more property of our verification algorithm, that is not required for the certification of the 3-approximation of the treewidth, but will be useful in the next section.

Claim 7: For every $v \in V$ and every $j \in \{1, \ldots, d(v)\}$, the set $E_j(v)$ corresponds to the edges of graph induced by $B_j(v)$.

Proof of Claim 7. We prove this claim by induction on $d(v)$. Suppose first that $d(v) = 1$. Since $F_1 = B_1$, we have that $F(u) = F(v)$ for every other vertex u in B_1. By **Claim 2** we obtain that v and u agree on the same set E_1. Then, by condition **5** on all the vertices in B_1, we deduce that $E_1 = E[G_1]$. Now suppose that the claim is true for every vertex of depth smaller than $d > 1$ and suppose that $d(v) = d$. By the induction hypothesis, for every $j \in \{1, \ldots, d-1\}$ the set $E_j(v)$ corresponds to the set of edges of $G[B_j(v)]$. Then, it remains to prove that $E_d(v)$ corresponds to the set of edges of $G[B_d(v)]$. Let w_1, w_2 be an arbitrary pair of vertices in $B(v)$, and call d_1 and d_2 the depth of w_1 and w_2, respectively. Without loss of generality assume that $d_1 \leq d_2$. By **Claim 4** applied to all vertices in the path of nodes between $B_d(v)$ and $B_{d_2}(w_2)$, we have that $E_{d_2}(w_2) = E_{d_2}(v)$. By condition **6**, we have that w_1, w_2 are adjacent if and only if $\{w_1, w_2\}$ belongs to $E_{d_2}(w_2)$. Suppose that $d_2 = d$. By **Claim 2**, we know that all nodes in $F(v)$ have the same main messages, in particular, they agree in the set $E_d(v)$. Then $E_{d_2}(v) = E_d(v)$. If $d_2 < d$, we have by condition **5**, that $w_1, w_2 \in E_{d_2}(v)$ if and only if $\{w_1, w_2\}$ belongs to $E_d(v)$. In both cases we deduce that $\{w_1, w_2\} \in E$ if and only if $\{w_1, w_2\} \in E_d(v)$. □

4 Certifying regular properties

In this section, we prove our main result, Theorem 1.

Theorem 2. *For every $k \geq 1$ and any regular graph property $\mathcal{P}(G)$, there exists a distributed certification protocol certifying that $\mathsf{tw}(G) \leq k$ and $\mathcal{P}(G)$ is true, using certificates on $O(\log^2 n)$ bits in n-node networks.*

For simplicity, we integrate the condition $\mathsf{tw}(G) \leq k$ to property $\mathcal{P}$, by setting $\mathcal{P}(G) = (\mathsf{tw}(G) \leq k) \wedge \mathcal{P}(G)$. The new property is regular because property $\mathsf{tw}(G) \leq k$ is regular (see, e.g., [24] for a discussion), and a conjunction of regular properties is regular by [8]. Basically, we enrich the protocol of Section 3 as follows. Either the protocol rejects because $\mathsf{tw}(G) > k$, or it constructs and certifies a tree decomposition at most $3k + 2$. In the latter case, we also certify property $\mathcal{P}$ using the tree decomposition of width $3k+2$ and the homomorphism classes $\mathcal{C}$ of the property on $(3k + 3)$-terminal graphs.

Fix the tree decomposition of width $3k + 2$. As in the sketch of proof of Proposition 1, for each node i of the decomposition tree, G_i denotes the $(3k + 3)$-terminal graph corresponding to $G[V_i]$, with set of terminals B_i. Also, for each $w \in F_i$, let $Children(w)$ denote the set of children j of i such that w is in charge of node j (see Lemma 4 applied to j). In particular, the sets $Children(w)$ for $w \in F_i$ form a partition of the children nodes of i in the decomposition tree. Denote by $G_i[w]$ the $(3k + 3)$-terminal graph obtained from $G[B_i \cup \bigcup_{j \in Children(w)} V_j]$ by choosing B_i as set of terminals. Note that if $Children(w)$ is empty, then $G_i[w]$ is simply the $3k + 3$-terminal base graph G_i^b corresponding to $G[B_i]$, as illustrated in Figure 3.

The **prover** appends two new informations to the previous main messages of each vertex $v \in F_i$: the homomorphism class of G_i as well as the homomorphism class of $G_i[v]$. Moreover the homomorphism class of G_i is also added to the auxiliary message $m_{aux}(w, i)$ for every vertex w of the auxiliary tree $S(i)$. Note that this only ads a constant size to the previous main messages,

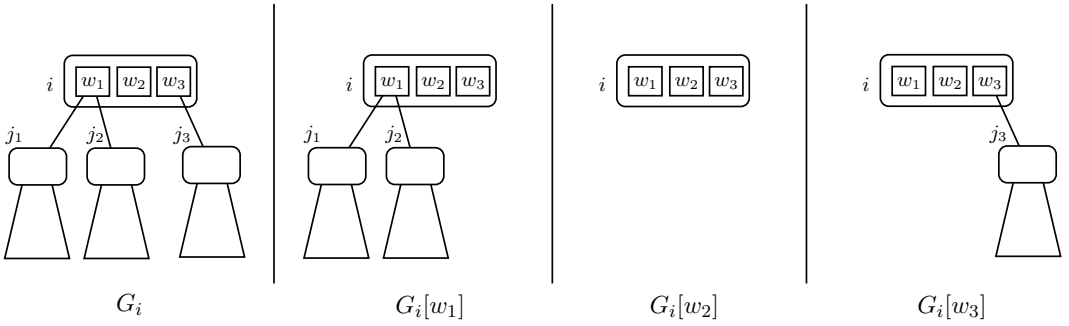

Figure 3: Graphs G_i and $G_i[w]$.

since property $\mathcal{P}$ has a constant number of homomorphism classes. Auxiliary messages are increased by $O(\log n)$ bits, since each vertex w is in $O(\log n)$ auxiliary trees $S(i)$ by Lemma 5. Nevertheless, the constants here depend on k and on property $\mathcal{P}$.

We now update the **verification round** to exploit these new messages and check the property. As before, we use the auxiliary tree $S(i)$ to ensure that ℓ_i, and all vertices $v \in F_i$, have received from the prover the same isomorphism class for G_i.

It remains to check the consistency of the homomorphism classes for property $\mathcal{P}$ in the respective subgraphs.

Consistency of the homomorphism class of $G_i[v]$. Firstly, each vertex $v \in F_i$ in charge of some nodes must certify the homomorphism class of $G_i[v]$, in the sense that it compares the message received from the prover with the homomorphism class that he constructs from the nodes $j \in Children(v)$. Vertex v receives, for each $j \in Children[v]$, a message from ℓ_j with the homomorphism class of $\mathcal{P}$ restricted to the $(3k+3)$-terminal graph G_j. Using Definition 2, it constructs the homomorphism class on G_j^+. Recall that $G_j^+ = f(G_j, G_i^b)$, i.e., G_j^+ is obtained by glueing G_j and the base graphs G_i^b induced by B_i, the glueing being performed by identifying the terminals of $B_j \setminus B_i$ in G_j to the corresponding vertices of B_i. Vertex v knows both sets B_i (which is in its initial message) and B_j (received from ℓ_j), so it has full knowledge of matrix $m(f)$ of the composition operation f. (There is a hidden technicality here. Node ℓ_j sends its main message to v in the unique communication round, and this message contains all bags $\mathcal{B}(\ell_j)$, in particular bag B_j. Node v can retrieve this bag, since its order in the list $\mathcal{B}(\ell_j)$, starting from the end of the list, is exactly the depth $i(v)$ of node i, plus one.) Then the homomorphism class of $h(G_j^+)$ is obtained as $\odot_f(h(G_j), h(G_i^b))$(see Figure 3, Proposition 1 and its sketch of proof). Again v knows graph $G[B_i]$ hence it can compute its homomorphism class $h(G_i^b)$. It also knows $h(G_j^+)$ from ℓ_j, altogether v is able to compute the homomorphism class $h(G_j^+)$. Eventually, since $G[v]$ is obtained by glueing on B_i all graphs $G_j^+, j \in Children(v)$, v computes the homomorphism class of $G_i[v]$. If this class is not the same as the one received from the prover, vertex v rejects.

Consistency of the homomorphism class of G_i. Every vertex $v \in F_i$ checks the consistency between the message received from the prover as class of $\mathcal{P}$ on G_i, and the one it constructs from the glueing of all classes of $G_i[w]$ (that vertex w has received from the prover), for all $w \in F_i$, on B_i. Indeed, G_i is equal to the glueing, on B_i, of all graphs $G_i[w]$ with $w \in F_i$. Again, in case of inconsistency, vertex v it rejects.

Yes-instance. Every vertex belonging to F_r (the root node of the decomposition tree) accepts if the class of property $\mathcal{P}$ on G_r is an accepting one, otherwise it rejects.

Due to space restrictions, the soundness and completeness of the protocol are detailed in Appendix A.2. In a nutshell, the completeness is quite straightforward by construction of the messages. For the soundness, assume that all vertices accept. We proceed by induction on nodes i on the decomposition tree, from the leaves to the root, and show that the messages received by each $v \in F_i$ from the prover as homomorphism classes for $G_i[v]$ and G_i are correct. Eventually,

since vertices of the root node accept, we conclude the homomorphism class of $\mathcal{P}$ on the whole graph is an accepting one, so $\mathcal{P}(G)$ is true.

5 Conclusion

To sum up, we proved that for every $k \geq 1$ and every MSO property on graphs, there exists a distributed protocol certifying that the input graph is of treewidth at most k and satisfies the required property, using certificates on $O(\log^2 n)$ bits. The result extends to optimisation problems, where we certify that a given vertex subset is of optimal weight (e.g., of maximum or of minimum size) for some MSO property, and the treewidth of the input graph is at most k.

The first natural question is whether we can reduce the size of certificate to $O(\log n)$ instead of $O(\log^2 n)$. We believe that such an improvement requires considerably different techniques, even for certifying that the treewidth of the input graph is at most k.

Another further research direction concerns certification versions for other algorithmic "meta-theorems". For example, given a graph property expressible by a first-order boolean formula, is there a distributed protocol certifying that the input graph is planar and satisfies the property, using certificates of logarithmic size?

Acknowledgment. The authors are thankful to Eric Remila for fruitful discussions on certification schemes related to the one considered in this paper.

References

[1] Yehuda Afek, Shay Kutten, and Moti Yung. The local detection paradigm and its application to self-stabilization. *Theor. Comput. Sci.*, 186(1-2):199–229, 1997.

[2] Baruch Awerbuch, Boaz Patt-Shamir, and George Varghese. Self-stabilization by local checking and correction. In *32nd Symposium on Foundations of Computer Science (FOCS)*, pages 268–277, 1991.

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

[4] Aviv Bick, Gillat Kol, and Rotem Oshman. Distributed zero-knowledge proofs over networks. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2022.

[5] Hans L. Bodlaender. NC-algorithms for graphs with small treewidth. In *14th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 344 of *LNCS*, pages 1–10. Springer, 1988.

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

[7] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 601–612. IEEE, 2020.

[8] 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.

[9] Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Brief announcement: Local certification of graph decompositions and applications to minor-free classes. In *35th International Symposium on Distributed Computing (DISC)*, volume 209 of *LIPIcs*, pages 49:1–49:4. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[10] Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Local certification of MSO properties for bounded treedepth graphs. arXiv 2110.01936, 2021.

[11] Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theor. Comput. Sci.*, 811:112–124, 2020.

[12] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[13] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic*. Cambridge University Press, 2012.

[14] 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.

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

[16] Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM J. Comput.*, 44(1):54–87, 2015.

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

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

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

[20] Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *J. ACM*, 64(3):17:1–17:32, 2017.

[21] Gene Itkis and Leonid A. Levin. Fast and lean self-stabilizing asynchronous protocols. In *35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 226–239, 1994.

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

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

[24] Mathieu Liedloff, Pedro Montealegre, and Ioan Todinca. Beyond classes of graphs with "few" minimal separators: FPT results through potential maximal cliques. *Algorithmica*, 81(3):986–1005, 2019.

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

[26] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1):49–64, 1984.

A Detailed proofs

This section contains detailed proofs moved to the Appendix due to space restrictions.

A.1 Coherent tree decompositions

Proof of Lemma 3

Proof. Firstly, choose a tree decomposition $(T = (I, F), \{B_i, i \in I\})$ of G where T is of logarithmic depth and each bag is of size at most $3k + 3$. Such a decomposition exists by [5].

Let us show how to transform this tree decomposition into a coherent one. Firstly we focus on the connectivity condition. Assume the decomposition is not coherent, and let i a node that violates the connectivity condition, closest to the root. Observe that i is different from the root: we can assume w.l.o.g that all bags are non empty, in particular F_r is non empty and $G[V_r] = G$ is connected. Then suppose that $G[V_i \setminus B_{p(i)}]$ is not connected and let $W^1, W^2, \ldots, W^p$ be the vertex sets of the connected components of $G[V_i \setminus B_{p(i)}]$. Denote by N^j the neighbourhood of $W^j \in G$, for $1 \leq j \leq p$, and observe that every N^j is a subset of $B_i \cap B_{p(i)}$. For each j, $1 \leq j \leq p$, construct a tree decomposition T_i^j of $G[W^j \cup N^j]$ by taking a copy of T_i and the corresponding bags, then restricting the bags to their intersection with $W^j \cup N^j$. Eventually replace, in the tree decomposition T of G, the subtree T_i by the p copies $T_i^1, \ldots, T_i^p$, by making their roots adjacent to $p(i)$. Observe that we obtain indeed a tree decomposition of G, and the connectivity condition on node i has been mended, in the sense that the p new nodes corresponding to copies of node i satisfy it: for each copy i^j of i, we have that $V_{i^j} \setminus B_{p(i^j)} = W^j$.

Now that the connectivity condition is satisfied for every node, we fix the condition stating that sets F_i must not be empty. If F_i is empty for some node i, then B_i is a subset of $B_{p(i)}$. Therefore we can remove node i from the decomposition tree and attach its children directly to the parent $p(i)$ of the deleted node, obtaining a new tree decomposition, without increasing the depth. This process can be iterated as long as necessary, hence we my assume that for any node i, F_i is non empty. Also observe that the removal of node i does not modify the set $V_j \setminus B_{p(j)}$ for any child j of i in the initial tree T, therefore the connectivity condition is preserved for all nodes. □

Proof of Lemma 4

Proof. Denote $W_i = V_i \setminus B_{p(i)}$. By definition of a tree decomposition the neighbourhood $N_G(W_i)$ of W_i in graph G is a subset of $B_{p(i)}$. We must show that $N_G(W_i)$ contains at least one vertex w in $F_{p(i)}$, which allows to take a $\ell_i \in W_i$ adjacent to w in G. Assume by contradiction that $N_G(W_i)$ does not intersect $F_{p(i)}$. In this case $p(i)$ is not the root vertex, and $N_G(W_i) \subseteq B_{p(i)} \setminus F_i$, which is also equal to $B_{p(i)} \cap B_{p(p(i))}$. Therefore $B_{p(i)} \cap B_{p(p(i))}$ separates W_i from the $F_{p(i)}$ in graph G. This contradicts the coherence of the tree decomposition, more precisely the connectivity property at node $p(i)$, since W_i and $F_{p(i)}$ are disconnected in $G[V_{p(i)} \setminus B_{p(p(i))}]$. □

Proof of Lemma 5

Proof. Since our tree decomposition is coherent, for each node i graph $G[V_i \setminus B_{p(i)}]$ is connected so it contains the required subtree $S(i)$.

Observe is that, if i and j are different nodes of the tree decomposition such that none is ancestor of the other, then sets $V_i \setminus B_{p(i)}$ and $V_j \setminus B_{p(j)}$ are disjoint, by definition of tree decompositions. Therefore, if we fix a vertex v of G, the nodes i such that v appears in $S(i)$ are pairwise comparable w.r.t the ancestor relation in the decomposition tree. The decomposition tree is of depth $O(\log n)$ and the conclusion follows. □

A.2 Theorem 2: soundness and completeness

We detail the soundness and completeness of the protocol of Theorem 2, certifying a regular property $\mathcal{P}$ on graph G of treewidth at most k.

For the completeness part, assume that our graph G has treewidth at most k and satisfies property $\mathcal{P}$. By Lemma 1, the prover can construct the messages for the 3-approximation of treewidth, such that the verifier passes all the tests certifying the tree decomposition. Moreover the tree decomposition is correct, and so are, for each node i of the decomposition, the exit vertex ℓ_i of i and the vertex of $F_{p(i)}$ in charge of node i. It remains to prove that vertices $v \in F_i$ accept. The proof is done bottom-up, by considering i from the leaves to the root. If i is a leaf of the decomposition tree, then v is not in charge of any other node (i.e., $Children(v)$ is empty). In this case $G_i = G_i[v]$, and the homomorphism classes are all equal and correspond to the $(3k+3)$-terminal base graph $G[B_i]$, and all vertices $v \in F_i$ accept. Now if i is not the root, every $v \in F_i$ is assigned a (possibly empty) set $Children(v)$ of children of i in the decomposition tree. For each $j \in Children(v)$, vertex v receives from ℓ_j the homomorphism class of G_j, so v computes the class of G_j^+. By Proposition 1 and Definition 2, the homomorphism class of $G_i[v]$ is consistent withe the one obtained with the glueing of all G_j^+ on the set B_i of terminals. Eventually, by glueing on B_i all graphs $G_i[v]$, for all $v \in F_i$, we obtain G_i, and the homomorphism classes of G_i and $G_i[v]$ are consistant, so v accepts. At the root node $i = r$, each $v \in F_r$ also checks that the homomorphism class of $\mathcal{P}$ is an accepting one (and it is), so v accepts.

For the soundness, assume that all nodes accept. We must show by induction, for each node i of the decomposition from leaves to the root, that the messages that each $v \in F_i$ received as homomorphism class of $\mathcal{P}$ on graphs $G_i[v]$ and G_i are correct. We rely again on the fact that the tree decomposition is correct, as well as the exit nodes and their neighbours in the parent node. When i is a leaf node, each v knows that its set $Children(v)$ is empty, since it has received no message from some exit node. Also v knows the graph $G[B_i]$ (recall that all edges of $G[B_i]$ have been sent in the main messages). Therefore it checks that the homomorphism classes received from the prover for $G_i[v]$ and G_i received are correct: they must be equal, and must correspond to the base graph $G[B_i]$. If i is not a leaf node, we rely on the fact that, collecting the messages from the exit nodes ℓ_j, vertex $v \in F_i$ correctly constructs $Children(v)$. For each $j \in Children(v)$, v has received from ℓ_j the homomorphism class c of G_j (which is correct by induction hypothesis). Therefore v correctly constructs the class of G_j^+ from c and the class of the $(3k+3)$-terminal base graph $G[B_i]$. Then, by glueing all G_j^+, $j \in Children(v)$, v it gets the class of $G_i[v]$. Since at this stage v has not rejected, the class of $G_i[v]$ received from the prover is correct. Eventually, v constructs the homomorphism class of G_i by glueing the classes of all $G_i[w], w \in F_i$. Since v knows F_i and B_i, it correctly performs the glueing. By the fact that v has not rejected up to now, we deduce that the homomorphism class of G_i obtained from the prover is correct.

Since vertices v of the root bag accept, il means that the homomorphism class of $\mathcal{P}$ on the whole graph is an accepting one, so the property holds, which completes the proof of Theorem 2.

B More preliminaries: MSO and regular properties for optimization

Let us enrich our framework to properties on graphs and vertex subsets, i.e., properties $\mathcal{P}(G, X)$ assigning to each graph G and each vertex subset X of G a boolean value. Properties like "X is an independent set of G" or "X is a dominating set of G" are expressible in (Counting) Monadic Second Order Logic, and they are still regular as we shall see below. More importantly, in the sequential setting this allows to solve efficiently optimisation problems on graphs of bounded treewidth, namely to compute a vertex subset X of maximum (or minimum) size such that $\mathcal{P}(G, X)$ holds.

Composition operations on w-terminal recursive graphs naturally extend to pairs (G, X), where G is a w-terminal recursive graph and X is a vertex subset. Let f be a composition operation of arity 1, and $G = f(G_1)$. Then for every vertex subset X_1 of G_1, we take $f(G_1, X_1) = (G, X_1)$. Consider composition operation of arity 2 such that $G = f(G_1, G_2)$. When we perform this composition on pairs (G_1, X_1), (G_2, X_2), the result is the pair (G, X), where X is obtained by the the glueing of X_1 and X_2. Therefore the intersections of sets X_1 and X_2 with the terminals of G_1 and respectively G_2 must be coherent with the gluing, in the sense that if two terminals x_1 of G_1 and x_2 of G_2 are identified in G, then we either have $x_1 \in X_1$ and $x_2 \in X_2$, or we have $x_1 \notin X_1$ and $x_2 \notin X_2$ (see [8, 16] for more details). To be complete, we restate the notion of regularity to properties $\mathcal{P}(G, X)$ – the only difference being that the property and the homomorphism classes now depend on both parameters, the graph and the vertex subset.

Definition 3 (regular property on graphs and vertex sets)**.** Graph property $\mathcal{P}(G, X)$ is called *regular* if, for any value w, we can associate a finite set $\mathcal{C}$ of *homomorphism classes* and a *homomorphism function* h, assigning to each w-terminal recursive graph G and to each vertex subset X a class $h(G, X) \in \mathcal{C}$ such that:

1. If $h(G_1, X_1) = h(G_2, X_2)$ then $\mathcal{P}(G_1, X_1) = \mathcal{P}(G_2, X_2)$.
2. For each composition operation f of arity 2 there exists a function $\odot_f : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ such that, for any two pairs (G_1, X_1) and (G_2, X_2),

$$h(f((G_1, X_1), (G_2, X_2)) = \odot_f(h(G_1, X_1), h(G_2, X_2))$$

and for each composition operation f of arity 1 there is a function $\odot_f : \mathcal{C} \to \mathcal{C}$ such that, for any pair (G, X),

$$h(f(G, X)) = \odot_f(h(G, X)).$$

The first condition separates the classes into *accepting* ones (i.e., classes $c \in \mathcal{C}$ such that $h(G, X) = c$ implies that $\mathcal{P}(G, X)$ is true) and *rejecting* ones (s.t. $h(G, X) = c$ implies that $\mathcal{P}(G, X)$ is false).

We also have:

Proposition 3 ([8, 12])**.** *Any property $\mathcal{P}(G, X)$ expressible by a MSO formula is regular.*

Moreover, given the MSO formula φ and parameter w, one can explicitely compute the set of classes, the homomorphism function for all w-terminal base graphs as well as the homomorphism functions $\odot_f$ over all possible composition operations f.

E.g., for the property "X is an independent set of G", we can choose as homomorphism class $h(G(V, W, E), X)$ formed by a boolean indicating whether $\mathcal{P}(G, X)$ is true, and the intersection of X with the set of terminals.

We may assume w.l.o.g. that the homomorphism class $c = h(G, X)$, for $G = (V, W, E)$ always encodes the intersection of X with the set of terminals. This is not explicitly required by the definition of regular properties, but it can be done since it only costs w bits to encode the subset of the terminals contained in X. Therefore we assume there is a function $term(c, W)$ that, given a homomorphism class c and an ordered set of terminals W returns the unique possible set $X \cap W$, over all pairs $(G = (V, W, E), X)$ mapped to c. Thanks to this function, when we glue two terminal recursive graphs with their corresponding vertex subsets, we will be able to check that the glueing is coherent. Moreover, we can perform optimisation tasks as follows.

Assume that we deal with weighted graphs, i.e., we have a function $weigth$ associating to every vertex an integer weight in the interval $[-MAXW, +MAXW]$. Given a w-terrminal recursive graph G and a regular property $\mathcal{P}(G, X)$, we aim to compute the maximum weight vertex subset X satisfying $\mathcal{P}(G, X)$. For this purpose, for any homomorphism class $c \in \mathcal{C}$ of property $\mathcal{P}$, let

$$MaxWeight(G, c) = \max\{weight(X) \mid X \subseteq V(G) \text{ s.t. } h(G, X) = c\}.$$

For convenience, we set $MaxWeight(G, c)$ to $-\infty$ if no such set exists, and more generally we consider that the maximum value of an empty set is $-\infty$. Then we have:

Lemma 7. *For any w-terminal recursive graph $G = (V, W, E)$ and any homomorphism class c of property $\mathcal{P}$,*

1. *If G is a w-terminal base graph,*

$$MaxWeight(G, c) = weight(term(c, W)).$$

2. *If $G = f(G_1)$ for some composition operation f of arity 1, then*

$$MaxWeight(G, c) = \max_{c_1 \ s.t. \ c=\odot_f(c_1)} MaxWeight(G_1, c_1).$$

3. *If $G = f(G_1, G_2)$ for some composition operation f of arity 2, then*

$$MaxWeight(G, c) = \max_{c_1,c_2 \ s.t. \ c=\odot_f(c_1,c_2)} MaxWeight(G_1, c_1) + MaxWeight(G_1, c_1) - weight(term(c_1, W_1) \cap (term(c_2, W_2)),$$

where W_j denotes the set of terminals of graph G_j, for $j \in \{1, 2\}$.

Proof. The first two items are simple consequences of the definitions, let us focus on the third item.

Firstly, let us prove that $MaxWeight(G, c)$ is at least equal to the right-hand side of the expression. Let c_1, c_2 be the homomorphism classes realising the maximum, and for each $j \in \{1, 2\}$ let X_j be the vertex subset of G_j such that $h(G_j, X_j) = c_j$ and $MaxWeight(G_j, c_j) = weight(X_j)$. Observe that, by taking the vertex subset X of G obtained from X_1 and X_2 by glueing the corresponding terminal vertices according to composition operation f, $weight(X) = weight(X_1)+weight(X_2)-weight(term(c_1, W_1)\cap(term(c_2, W_2))$, since the negative term avoids overcounting the vertices of X appearing as terminals in both X_1 and X_2. By construction $(G, X) = f((G_1, X_1), (G_2, X_2)$ so $MaxWeight(G, c)$ is at least $weight(X)$.

Conversely, let X be a maximum weight vertex subset of G such that $h(G, X) = c$. For $j \in \{1, 2\}$, let X_j be the intersection of X with the vertex set of G_j, and $c_j = h(G_j, X_j)$. By construction, $c = \odot_f(c_1, c_2)$ and $weight(X) = weight(X_1)+weight(X_2)-weight(term(c_1, W_1)\cap(term(c_2, W_2))$. We claim that $weight(X_j) = MaxWeight(G_j, c_j)$, for both values j. Assume by contradiction there is a set, say X_1', of larger weight than X_1 and such that $h(G_1, X_1) = c_1$. Note that X_1' and X_1 may only differ on non-terminal vertices of G_1, otherwise they would not correspond to the same homomorphism class. Then set X' obtained by glueing X_1' and X_2 is of larger weight than X, moreover $h(G, X') = c$, contradicting the maximality of X. We conclude that the right-hand side of the expression is at least equal to $MaxWeight(G, c)$ and the conclusion follows.

□

C Certifying optimal sets for regular properties

We can now extend out certification protocol to optimisation problems on weighted graphs, with polynomial weights.

Theorem 3. *For every $k \geq 1$ and any regular graph property $\mathcal{P}(G, X)$, there exists a distributed certification protocol certifying that $\mathsf{tw}(G) \leq k$ and X is the maximum weight vertex set such that $\mathcal{P}(G, X)$ is true, on graphs with polynomial weights, using certificates on $O(\log^2 n)$ bits in n-node networks.*

If instead of polynomial weights we use weights in the interval $[-MAXW, +MAXW]$, the protocol requires $O(\log n(\log n + \log MAXW))$ bits.

We only describe the differences of the new protocol with respect to the protocol of Section 4. As for the protocol of Theorem 2, we assume that the condition $\mathsf{tw}(G) \leq k$ is integrated to property $\mathcal{P}$. Here the input is also formed of vertex set X. On one hand we certify $\mathcal{P}(G, X)$ (this part of the protocol being almost identical to the one of Theorem 2), and in the meantime we certify, for each homomorphism class c and at each node i of the decomposition, the weight of an optimal partial solution (G_i, Y) for graph G_i, of homomorphism class c. Then we simply compare at the root node the weight of X with the weight of an optimal solution.

Let us detail how we deal with set X.

The first issue is that, for each node i of the decomposition tree and each vertex $v \in F_i$, vertex v must know the set $B_i \cap X$. For this purpose, The prover adds to the main messages of vertex v, a sequence of sets $\mathcal{X}(v) = (X_d(v), \dots X_1(v))$ where $X_j(v)$ represents the intersection of the solution X with bag $B_j(v)$.

The verification is very similar to the one of the edge sets of $G[B_i]$. In the verification round, v verifies for each $j_1, j_2 \in \{1, \dots, d\}$ and for each $u \in B_{j_1}(v) \cap B_{j_2}(v)$, that $u \in X_{j_1}(v) \iff u \in X_{j_2}(v)$. By **Claim 2**, all vertices in $F(v)$ receive the same main messages, then all nodes in $F(v)$ agree in the part of the solution X that intersect the bags in the nodes from $i = B_d(v)$ up to the root. By **Claim 4** the vertices globally agree on the set X. Also, each vertex $v \in X$ verifies that it belongs to $X_d(v)$, ensuring that the set X claimed by the prover is consistent with the input.

A second issue to deal with is the overall weight of set X. Here we use a completely different but standard technique to collect the weight of X in a vertex v_r belonging to the root bag, using $O(\log n)$ supplementary bits per vertex, see [23]. We encode a spanning tree of the whole graph rooted in v_r, by giving to each vertex its distance to the root and the identifier to the parent vertex. Moreover, each vertex v receives the total weight $weightX(v)$ of the vertices of X contained in the subtree rooted at v. The situation is very similar to the tree certificates $TreeCert$ that we have used in Section 3 for the auxiliary messages, where we encoded a subtree $S(i)$ rooted in a given vertex ℓ_i and spanning the vertex subset F_i. The verification follows exactly the same principles for certifying the tree structure, moreover each vertex v checks that $weightX(v)$ corresponds to the sum of weights $weightX(w)$ for its children w, plus the weight of v if the latter belongs to X.

A third issue is that, for each node i of the decomposition tree, the prover sends to each $v \in F_i$ and the exit vertex ℓ_i the homomorphism class $h(G_i, X \cap V_i)$ (instead of the class of G_i). Also, v receives the homomorphism class $h(G_i[v], X \cap V(G_i[v])$ and the weight of $X \cap V(G_i[v])$. This part is a simple update of the the protocol of Theorem 2, adapted to properties on graphs and edge subsets.

The main novelty is that each $v \in F_i$ and ℓ_i receive from the prover, for each homomorphism class of property $\mathcal{P}$, value $MaxWeight(i, c)$ corresponding to the maximum weight of a partial solution (G_i, Y) of homomorphism class c on graph G_i, and v also receives value $MaxWeight(i, c; v)$, the maximum weight of $Y \subseteq V(G_i[v])$ such that $h(G_i[v], Y) = c$.

Let describe the verification performed by each vertex. We already ensured that vertices of a same set F_i, for each node i of the decomposition tree, posses the correct set $X \cap B_i$. Checking property $\mathcal{P}(G, X)$ is has no significant difference compared to Theorem 2, we simply use the homomorphism functions of Definition 3 instead of Definition 2. The construction is similar, we simply use the fact that each vertex $v \in F_i$ knows $X \cap B_i$, allowing it to compute the homomorphism class for base graphs G_i^b.

A supplementary effort is required to compute the weight of an optimal solution, then to compare it to the weight of X. For this purpose, at each node i of the decomposition tree, the verifier performs the following operations on each $v \in F_i$.

- Firstly, vertex v checks that values $MaxWeight(i, c; v)$ received from the prover for each homomorphism class $c \in C$, claimed to be equal to $MaxWeight(G_i[v], c)$ are indeed con-

sistent with the information it receives from nodes $j \in Children[v]$, the graph $G[B_i]$ and $X \cap B_i$.

For this purpose, v computes $MaxWeight(G^b, c)$, for each homomorphism class c, using Lemma 7 applied to the $3k+3$-terminal base graph G^b and set $X \cap B_i$. Recall that v has $G^b = (B_i, B_i, E(G[B_i]))$ and $X \cap B_i$ in its own message. Then, for each $j \in Children(v)$, it retrieves all values $MaxWeight(G_j, c)$ from the exit vertex ℓ_i. Using again Lemma 7 for graph $G_j^+ = f(G_j, G_i^b)$, it computes all values $MaxWeight(G_j^+, c)$, from $MaxWeight(G_j, c_1)$, $MaxWeight(G_j, c_2)$ and $weight(term(c_1, B_j)) \cap term(c_2, B_i))$, over all classes c_1, c_2 with $c = \odot_f(c_1, c_2)$.

Then v must deduce $MaxWeight(G_i[v], c)$ based on the fact that $G_i[v]$ is obtained by consecutively glueing $G_{j_1}^+$, $G_{j_2}^+, \ldots,$ $G_{j_p}^+$, where $Children(v) = \{j_1, \ldots, j_p\}$ (e.g., we can order the nodes j of $Children(v)$ by increasing size of the identifier of ℓ_j). The glueing (composition operation) f is the same at each step, performed on the same set of terminals B_i. Let H^r denote the result of the glueing of $G_{j_1}^+ \ldots, G_{j_r}^+$, for each $r, 1 \leq r \leq p$. In particular $H^1 = G^+ j_1$, $H^r = G_i[v]$ and $H^r = f(H^{r-1}, G_{j_r})$ for each $r, 2 \leq r \leq p$. Therefore, for each r from 2 to p, vertex v computes all values $MaxWeight(H^r, c)$ from values $MaxWeight(H^{r-1}, c_1)$ and $MaxWeight(G_{j_r}, c_2)$ using the equation of Lemma 7 on operation f. Eventually v has all values $MaxWeight(G_i[v], c)$ for all homomorphism classes c. If one of these values differs from the message $MaxWeight(i, c; v)$ received from the prover, then v rejects.

- Secondly, vertex v checks that values $MaxWeight(i, c)$ correspond, for each homomorphism class c, to the value $MaxWeight(G_i, c)$ obtained by expressing G_i as the consecutive glueing of all $G_i[w]$, for all $w \in F_i$, on the set of terminals B_i. Value $MaxWeight(G_i, c)$ is obtained by iteratively performing the $|F_i| - 1$ glueings of $G_i[w]$ and using Lemma 7 and values $Max(i, c; w)$. As above, at iteration r, $2 \leq r \leq |F_i|$, we glue the first r graphs of the form $G[w]$, where vertices w are ordered by increasing identifiers. Again, in case of inconsistency between $MaxWeight(i, c)$ and $MaxWeight(G_i, c)$ for some homomorphism class c, vertex v rejects.

- At the root node r, recall that we must have a vertex $v_r \in F_r$ that knows the weight of X – it corresponds simply to $weightX(v_r)$. The node v_r firstly checks that it belongs indeed to the root of the decomposition tree by testing its depth, i.e., checking that $d(v_r) = 1$. Then v_r computes the maximum weight $MaxWeight(G_r, c)$ as the maximum of $MaxWeight(r, c)$ over all accepting classes c. If one of those is larger than the $weightX(v_r)$, vertex v_r rejects.

Soundness and completeness. We already know that the protocol correctly encodes the tree decomposition, the homomorphism classes of $\mathcal{P}(G, X)$ on partial graphs G_i and $G_i(v)$, and that the weight of set X is encoded in $weightX(v_r)$ for some vertex v_r belonging to the root bag. It remains to deal with quantities $MaxWeight(i, c)$ and $MaxWeight(i, c; v)$.

For the completeness part, the prover simply needs to correctly compute the intersection of X with the bags, and values $MaxWeight(i, c)$ and $MaxWeight(i, c; v)$ for each homomorphism class c, each node i of the decomposition tree and each vertex $v \in F_i$. The proof that vertex v accepts when certifying messages $MaxWeight(i, c)$ and $MaxWeight(i, c; v)$ assigned to it follows the same steps as the completeness part for the decision problem, certifying that homomorphism classes of G_i and $G_i[v]$ are correct. We need to use Lemma 7, allowing to obtain the optimal weight of a homomorphism class after glueing, instead of simply using Definition 2. Therefore, we prove by bottom-up induction on nodes i that all vertices $v \in F_i$ accept, if i is not the root. When i is the root r, vertex $v_r \in F_r$ also check that the homomorphism class of $\mathcal{P}(G_r, X)$ is an accepting one, and that the weight of X corresponds to the maximum weight of an accepting class, and both conditions hold for a yes-instance.

For the soundness condition, assume that all vertices accept. We prove as before, by bottom-up induction (from leaves to the root) on nodes i, that homomorphism classes as well as quantities $MaxWeight(i, c)$ and $MaxWeight(i, c; v)$ are correct, in the sense that they correspond to graphs G_i and $G_i[v]$. Again, for values $MaxWeight(i, c)$ and $MaxWeight(i, c; v)$, the glueing is performed using Lemma 7.

At the root, since vertex $v_r \in F_r$ accepts, it means that $\mathcal{P}(G_r, X)$ is true and moreover the weight of X (which is equal to $weightX(v_r)$) is the maximum possible weight over all vertex subsets Y such that $\mathcal{P}(G, Y)$ accepts (by the last item of the verification protocol). Therefore X is the optimal set for property $\mathcal{P}$.