Local certification of graph decompositions and applications to minor-free classes

Nicolas Bousquet, Laurent Feuilloley, Théo Pierron

Local certification of graph decompositions and applications to minor-free classes

Nicolas Bousquet
Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, F-69622 Villeurbanne, France
nicolas.bousquet@univ-lyon1.fr

Laurent Feuilloley
Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, F-69622 Villeurbanne, France
laurent.feuilloley@univ-lyon1.fr

Théo Pierron
Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, F-69622 Villeurbanne, France
theo.pierron@univ-lyon1.fr

Abstract

Local certification consists in assigning labels to the vertices of a network to certify that some given property is satisfied, in such a way that the labels can be checked locally. In the last few years, certification of graph classes received considerable attention. The goal is to certify that a graph G belongs to a given graph class $\mathcal{G}$. Such certifications with labels of size $O(\log n)$ (where n is the size of the network) exist for trees, planar graphs and graphs embedded on surfaces. Feuilloley et al. ask if this can be extended to any class of graphs defined by a finite set of forbidden minors.

In this work, we develop new decomposition tools for graph certification, and apply them to show that for every small enough minor H, H-minor-free graphs can indeed be certified with labels of size $O(\log n)$. We also show matching lower bounds using a new proof technique.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms → Distributed algorithms

Keywords and phrases Local certification, proof-labeling schemes, locally checkable proofs, graph decompositions, minor-free graphs

Funding This work was supported by ANR project GrR (ANR-18-CE40-0032).

1 Introduction

Local certification is an active field of research in the theory of distributed computing. On a high level it consists in certifying global properties in such a way that the verification can be done locally. More precisely, for a given property, a local certification consists of a labeling (called a *certificate assignment*), and of a local verification algorithm. If the configuration of the network is correct, then there should exist a labeling of the vertices that is accepted by the verification algorithm, whereas if the configuration is incorrect no labeling should make the verification algorithm accept.

Local certification originates from self-stabilization, and was first concerned with certifying that a solution to an algorithmic problem is correct. However, it is also important to understand how to certify properties of the network itself, that is, to find locally checkable proofs that the network belongs to some graph class. There are several reasons for that. First, because certifying some solutions can be hard in general graphs, while they become simpler on more restricted classes. To make use of this fact, it is important to be able to certify that the network does belong to the restricted class. Second, because some distributed algorithms work only on some specific graph classes, and we need a way to ensure that the network does belong to the class, before running the algorithm. Third, the distinction between certifying solutions and network properties is rather weak, in the sense that the techniques are basically

the same. So we should take advantage of the fact that a lot is known about graph classes to learn more about certification.

In the domain of graph classes certification, there have been several results on various classes such as trees [42], bipartite graphs [38] or graphs of bounded diameter [9], but until two years ago little was known about essential classes, such as planar graphs. Recently, it has been shown that planar graphs and graphs of bounded genus can be certified with $O(\log n)$-bit labels [21, 27, 28]. This size, $O(\log n)$, is the gold standard of certification, in the sense that little can be achieved with $o(\log n)$ bits, thus $O(\log n)$ is often the best we can hope for.

Planar and bounded-genus graphs are classic examples of graphs classes defined by forbidden minors, a type of characterization that has become essential in graph theory since the Graph minor series of Robertson and Seymour [49]. Remember that a graph H is a minor of a graph G, if it is possible to obtain H from G by deleting vertices, deleting edges, contracting edges. At this point, the natural research direction is to try to get the big picture of graph classes certification, by understanding all classes defined by forbidden minors. In particular, we want to answer the following concrete question. By the graph minor theorem, every minor-closed class of graphs can be defined by a finite list of forbidden minors, which underlines how a positive answer for this question would apply to wide range of applications.

▶ **Question 1** ([24, 27]). *Can any graph class defined by a finite set of forbidden minors be certified with $O(\log n)$-bit certificates?*

This open question is quite challenging: there are as many good reasons to believe that the answer is positive as negative.

First, the literature provides some reasons to believe that the conjecture is true. Properties that are known to be hard to certify, that is, that are known to require large certificates, are very different from minor-freeness. Specifically, all these properties (*e.g.* small diameter [9], non-3-colorability [38], having a non-trivial automorphism [38]) are non-hereditary. That is, removing a vertex or an edge may yield a graph that is not in the class. Intuitively, hereditary properties might be easier to certify in the sense that one does not need to encode information about every single edge or vertex, as the class is stable by removal of edges and vertices. Minor-freeness is a typical example of hereditary property. Moreover, this property, that has been intensively studied in the last decades, is known to carry a lot of structure, which is an argument in favor of the existence of a compact certification (that is a certification with $O(\log n)$-bit labels).

On the other hand, from a graph theory perspective, it might be surprising that a general compact certification existed for minor-free graphs. Indeed, for the known results, obtaining a compact certification is tightly linked to the existence of a precise constructive characterization of the class (*e.g.* a planar embedding for planar graphs [21, 28], or a canonical path to the root for trees [42]). Intuitively, this is because forbidden minor characterizations are about structures that are absent from the graphs, and local certification is often about certifying the existence of some structures. While such a characterization is known for some restricted minor-closed classes, we are far from having such a characterization for every minor-closed class. Note that there are a lot of combinatorial and algorithmic results on H-minor-free graphs, but they actually follow from properties satisfied by H-minor-free graphs, not from exact characterizations of such graphs. For certification, we need to rule out the graphs that do not belong to the class, hence a characterization is somehow necessary.

1.1 Our results

Answering Question 1 seems unfortunately out of reach, at the current state of our knowledge. We have explained above about why designing compact certification is hard for classes that do not have a constructive characterization. We will later give some intuition about why lower bounds seem equally difficult to get. In this paper, we intend to build the foundations needed to tackle Question 1. More precisely, we have four types of contributions.

First, we show how to certify some graph decompositions. Such decompositions state how to build a class based on a few elementary graphs and a few simple operations. They are essential in structural graph theory, and more specifically in the study of minor-closed classes. Amongst the most famous examples of these theorems is the proof of the 4-Color Theorem [3] or the Strong Perfect Graph Theorem [14].

Second, we show that by directly applying these tools, we can design compact certification for several H-minor-free classes, for which a precise characterization is known. See Fig. 1 and 2. That is, we answer positively Question 1, for several small minors, and show that our decomposition tools can easily be used.

Class	Optimal size	Result
K_3-minor-free	$\Theta(\log n)$	Equivalent to acyclicity [38, 42].
Diamond-minor-free	$\Theta(\log n)$	Corollary 29
K_4-minor-free	$\Theta(\log n)$	Corollary 29
$K_{2,3}$-minor-free	$\Theta(\log n)$	Corollary 29
$(K_{2,3}, K_4)$-minor-free (*i.e.* outerplanar)	$\Theta(\log n)$	Corollary 29
$K_{2,4}$-minor-free	$\Theta(\log n)$	Lemma 36

Figure 1 Our main results for the certification of minor-closed classes.

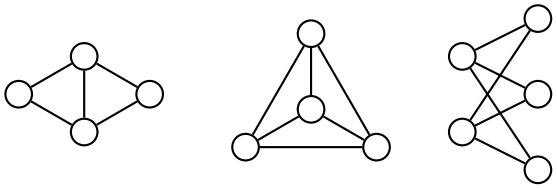

Figure 2 From left to right: the diamond, the clique on 4 vertices K_4, and the complete bipartite graph $K_{2,3}$.

Third, we do a systematic study of small minors to identify which is the first one that we cannot tackle. We first prove the following theorem.

▶ **Theorem 2.** *H-minor-free classes can be certified in $O(\log n)$ bits when H has at most 4 vertices.*

Then, we extend this theorem to minors on five vertices with a specific shape, proving along the way new purely graph-theoretic characterizations for the associated classes. After this study, we can conclude that the next challenge is to understand K_5-minor-free graphs.

Finally, we prove a general $\Omega(\log n)$ lower bounds for H-minor-freeness for all 2-connected graphs H of size at least 3. This generalizes and simplifies the lower bounds of [28] which apply only to K_k and $K_{p,q}$-minor-free graphs, and use ad-hoc and more complicated techniques.

At the end of the paper, we discuss why the current tools we have, both in terms of upper and lower bounds, do not allow settling Question 1. We list a few key questions that we need to answer before we can fully understand the certification of minor-closed classes, from the certification of classes with no tree minors to the certification of k-connectivity, for arbitrary k.

1.2 Our techniques

General approach and challenges

To give some intuition about our techniques, let us focus on a concrete example: K_4-minor-free graphs. Remember that a graph has K_4-minor if we can get a K_4 by deleting vertices and edges, and contracting edges. An alternative definition is that a graph has a K_4-minor, if it is possible to find four disjoint sets of vertices, called *bags*, such that: each bag is connected, there is a path between each pair of bags, these paths and bags are all vertex-disjoint (except for the endpoints of the paths that coincide with vertices of the bags). See Figure 3.

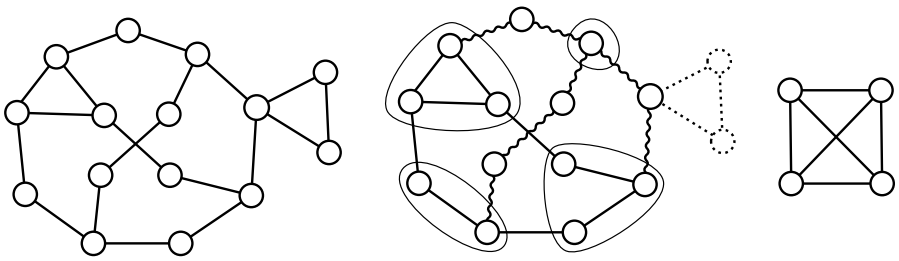

Figure 3 The graph on the left has a K_4 minor. Indeed, the bags of the second definition are depicted in the picture in the middle, and it is easy to find the six disjoint paths that link them. Alternatively, one can get a K_4 like the one of the right-most picture by contracting all the edges inside the bags, contracting the wavy paths between bags into edges, and deleting the dotted vertices and edges.

An important observation is that, if we take a collection $F_1, \ldots, F_k$ of K_4-minor-free graphs, and organize them into a tree, by identifying pairs of vertices like in Figure 4, we get a K_4-minor-free graph.

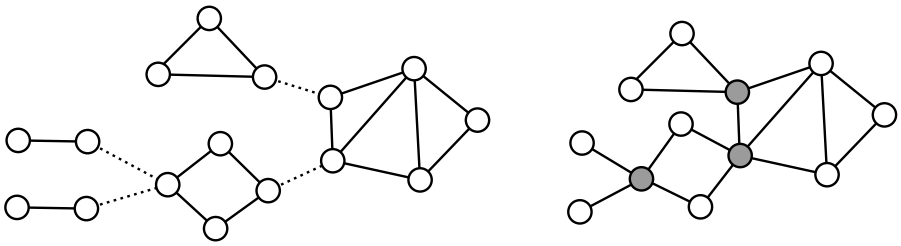

Figure 4 The five graphs with plain edges on the left picture are K_4-minor-free. Organizing them into a tree by identifying the vertices linked by dotted edges makes a larger K_4-minor-free graph.

To see that, suppose that the graph we created has a K_4-minor. Then there exist bags and paths as described above. If the bags and paths are all contained in the same former F_i, then this F_i would not be K_4-minor-free, which is a contradiction. If it is not the case, then the bags and paths use vertices that belong to different subgraphs F_i and F_j. And because

of connectivity, they should use a vertex v that connects two such subgraphs (grey vertices in Figure 4). Then the bags and paths cannot be vertex-disjoint as required, because v would them lie into at least two bags or interior of paths.

As a consequence of the observation above, a classic way to study K_4-minor-free graphs (as well as other classes) is to decompose the graph into maximal 2-connected components organized into a tree. This is called the *block-cut tree* of the graph, where every maximal 2-connected component is called a *block*. (Figure 4 actually shows the block-cut structure of the right-most graph.) This is relevant here because 2-connected K_4-minor-free graphs have a specific structure; we will come back to this later.

Now, from the certification point of view, there is a natural strategy: first certify the structure of the block-cut tree, and then certify the special structure of each block. There are several challenges to face with this approach. First, to certify the block-cut tree, it is essential to be able to certify the connectivity of the blocks. Second, we need to avoid what we call certificate congestion, which is the issue of having too large certificates because we use too many layers of certification on some vertices. We now detail these two aspects, starting with the latter.

Avoiding certificate congestion

In the block-cut tee of a graph, the blocks are attached to each other by shared vertices, the *cut vertices*. There is no bound on the number of blocks that are attached to a given cut vertex, and this is problematic for certification. Indeed, we cannot give to every vertex the list of the blocks it belongs to, as we aim for $O(\log n)$ certificates, and such a list could contain $\Omega(n)$ blocks. And even if we could fix the certification of the block-cut tree, the same problem would appear with the certification of the specific structure of each block: the cut vertices would have to hold a piece of certification for each block.

We basically have two tools to deal with this problem. The first one is not new, it is a degeneracy argument that already appeared in [27, 28]. A graph is k-degenerate if in every subgraph there exists a vertex that has degree at most k. Intuitively (and a bit incorrectly), this means that when we need to put a large certificate on a vertex, we can spread it on its some of its neighbors that have lower degree. A more precise statement is that, for k-degenerate graphs, we can transform a certification with $O(f(n))$ labels *on the edges of the graphs*, into a classic certification with $O(k \cdot f(n))$ labels on the vertices. This is relevant for our problem, as a priori there is less congestion on the edges, and minor-free classes have bounded degeneracy. Unfortunately, this is not enough for our purpose. We then build a second, more versatile tool. It consists in proving that it is possible to transform in mechanical way any certification of a graph or subgraph, into a certification that would put an empty certificate on some given vertex. Once we have this tool, we can adapt the certification of the blocks to work well in the block-cut tree: build the block-cut tree by adding blocks iteratively, making sure that the connecting vertex has an empty label in the certification of the newly added block.

See Section 3 for the details on this topic.

Certifying connectivity properties

Connectivity properties have been studied before in distributed certification. Specifically, certifying that for two given vertices s and t, the st-connectivity is at least k has been studied in [42] and [38]. But here we are interested in the connectivity of the graph itself, or in other words, in the st-connectivity between any pair of vertices. Clearly, proving st-connectivity for

any pair using the schemes of the literature would lead to huge certificates. Instead, we use the characterizations of k-connected graphs that are known for small values of k. There are various such characterizations, but they are all based on the same idea of *ear decomposition*.

To explain ear decompositions, consider a graph that we can build the following way (see Figure 5). Start from an edge, and iteratively apply the following process: take two vertices u, v of the current graph and link them by a path whose internal vertices are new vertices of the graph. This is called an *ear decomposition* (where the ears are the paths added at each step). When each time u and v are distinct vertices, it is not hard to see that such a graph is always 2-connected. Remarkably, the converse is also true: any 2-connected graph can be built (or decomposed this way). This is called an *open* ear decomposition, and similar constructions characterize 2-edge connected graphs and 3–connected graphs.

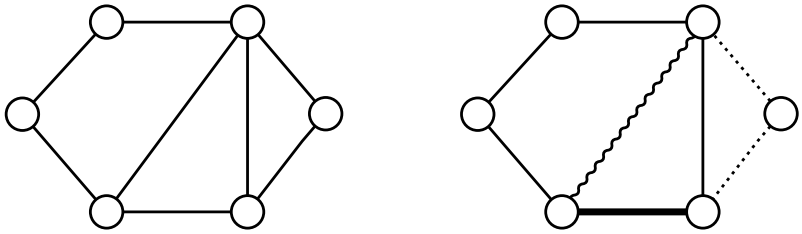

Figure 5 Illustration of an open ear decomposition. The graph on the left can be built with the ear decomposition described on the right. First, put the bold edge. Then add the path of plain edges. Finally, add the dotted path, and the wavy path, which is just one edge.

The good thing about these constructions is that we can certify them, by describing and certifying every step. This requires some care, as when certifying a new path, we could increase the size of the certificates of the endpoints, that are already in the graph. Fortunately, the tools developed to avoid certificate congestions allow us to control the certificate size.

The details about the connectivity certification can be found in Section 5.

Putting things together

Combining these techniques, we can prove the following theorem.

▶ **Theorem 3.** *For any 2-connected graph H, if the 2-connected H-minor-free graphs can be certified with $f(n)$ bits, then the H-minor-free graphs can be certified with $O(f(n) + \log n)$ bits.*

Going back to our example, K_4-minor-free graphs, given Theorem 3, we are left with certifying the 2-connected K_4-minor-free graphs. As said above, these have a specific shape. More precisely, 2-connected K_4-minor-free graphs have a nested ear decomposition, which is yet another type of ear decomposition, this time with additional constraints related to outerplanarity. We can certify this structure by adapting a construction from [28] for outerplanar graphs.

More generally the 2-connected graphs corresponding to most of the classes of Figure 1 have specific shapes that we can certify quite easily, which imply our compact certification schemes. We do this in Section 6. A special case is $K_{2,4}$, that has a more complicated structure, requiring to consider 3-connected components, and some more complicated substructures. We study this case in Section 7.

Finally, in Section 8, we study all the minors on at most 4 vertices, and in Section 9 all the minors on 5 vertices of some simple form. For these, we do not need new techniques

on the certification side, but we need to work on the graph theory side to establish new characterizations, as for these minors the literature does not help. The work we do in Section 9 might be of independent interest as we study the natural notion of H-minimal graph, which are the graph that have H as a minor, but for which any vertex deletion would remove this property.

Lower bounds

Towards the end of the paper, we show that $\Omega(\log n)$-bit labels are necessary to certify graph classes excluding a 2-connected minor. When it comes to $\Omega(\log n)$ lower bounds in our model, there are basically two complementary techniques (called *cut-and-plug techniques* in [24]). Both techniques basically show that paths cannot be differentiated from cycles, if the certificates use $o(\log n)$ bits. First, in [38], the idea is to use many correct path instances, and to prove that we can plug them into an incorrect cycle instance, thanks to a combinatorial result from extremal graph theory. Second, in [26], the idea is to consider a path, to cut it into small pieces, and to show via Stirling formula, that there exists a shuffle of these pieces that can be closed into a cycle.

Previous lower bounds for minor-free graphs in [28] followed the same kind of strategies as [38] and [26], with the same type of counting arguments, more complicated constructions, and tackled only minors that were cliques or bicliques.

In this paper, we are able to do a black-box reduction between the path/cycle problem and the H-minor-freeness for any 2-connected H. This way we avoid explicit counting arguments, and get a more general result with a simpler proof.

1.3 Related work

Local certification and graph classes

Local certification first appeared under the name of *proof-labeling schemes* in [42], inspired by works on self-stabilizing algorithms (see [16] for a book on self-stabilization). It has then been generalized under the name of *locally checkable proofs* in [38], and the field has been very active since these seminal papers. In the following, we will focus on the papers about local certification of graph classes, but we refer to [24] and [25] for an introduction and a survey of local certification in general.

As said earlier, certification was first mostly about checking that the solution to an algorithmic problem was correct, a typical example being the verification of a spanning tree [42]. Some graph properties have also been studied, for example symmetry in [38], or bounded diameter in [9]. Very recently, classes that are more central in graph theory have attracted attention. It was first proved in [47], as an application of a more general method, that planar graphs can be certified with $O(\log n)$ bits in the more general model of distributed interactive proofs. Then it was proved in [28] that these graphs can actually be certified with $O(\log n)$ bits in the classic model, that is, without interaction. This result was extended to bounded-genus graphs in [27]. Later, [21] provided a simpler proof of both results via different techniques. It was also proved in [41, 46] that cographs, distance-hereditary graphs, and some intersection graphs have compact distributed interactive proofs. On the negative side, it was established very recently that natural geometric classes, such as unit-disk graphs, require quasi-linear certificates [15].

Forbidden minors, meta-theorems and other labelings

After the publication of the first version of this paper, some progress has been done on Question 1. On the one hand, a positive answer has been given for an approximate version of the question. More precisely, by allowing mistakes for graphs that are close to being H-minor-free (in the spirit of property testing) one can define a compact certification [22] (follows from Theorem 6). (An approximate certification for bounded degree planar graphs with constant size labels had been established before, in [18].) On the other hand, two papers have established meta-theorems that answer the question for specific minor shapes. More precisely, by proving that monadic second order properties can be certified with $O(\log n)$ when the graph has bounded treedepth [29], and $O(\log^2 n)$ bits when the graph has bounded treewidth [32], these papers prove as corollaries that the same sizes suffice for path minors and planar minors, respectively. Indeed, graphs excluding a path (resp. planar) minor have bounded treedepth (resp. treewidth [50]). Let us also mention a generalization of the treewidth result to cliquewidth [33] (which has no further implications in terms of forbidden minors).

Graphs with forbidden minors have also attracted interest for other types of labelings. First, [34] established that one can design adjacency labelings of size $2\log n + O(\log\log n)$ for graphs excluding a specific minor. For approximate distance labelings, a key paper is [1], which introduced a specific decomposition of minor-free graphs.

Forbidden subgraphs, testing and detection

There are other structures that can be forbidden and give rise to interesting classes in graph theory, in particular subgraphs and induced subgraphs. These have been studied very recently for local certification, see [5].

Still in distributed computing, but outside local certification, a popular topic is the distributed detection of some subgraph H, which consists, in the CONGEST (or CONGEST-CLIQUE) model to decide whether the graph contains H as a subgraph or not (see [8] and the references therein). A related task is H-freeness testing, which is the similar but easier task consisting in deciding whether the graph is H-free or far from being H-free (in terms of the number of edges to modify to get a H-free graph). This line of work was formalized by [7] after the seminal work of [6] (see [31] and the references therein). To our knowledge, no detection or lower bounds have been designed for H-minor-freeness in general: examples like planarity testing [36] may however provide algorithms tailored to specific classes. Some results exists for H-minor-testing, see [10, 11], as well as for property testing on some minor-free classes [44].

Distributed algorithms for specific graph classes

Finally, we have mentioned in the introduction that certifying that the graph belongs to some given class is important because some algorithms are specially designed to work on some specific classes. For example, there is a large and growing literature on approximation algorithms for *e.g.* planar, bounded-genus, minor-free graphs. We refer to [23] for a bibliography of this area. There are also interesting works for exact problems in the CONGEST model, *e.g.* in planar graphs [35], graphs of bounded treewidth or genus [39] and minor-free graphs [40]. In particular the authors of [40] justify the focus on minor-free graphs by the fact that this class allows for significantly better results than general graphs, while being large enough to capture many interesting networks. Very recently, [37] proved

general tight results on low-congestion short-cuts (an essential tool for algorithms in the CONGEST model) for graphs excluding a dense minor.

2 Preliminaries

In this section, we define formally the notions we use and describe some useful known certification building blocks.

2.1 Graphs and minors

Let $G = (V, E)$ be a graph. Let $X \subseteq V$. The *subgraph of G induced by X* is the graph with vertex set X and edge set $E \cap X^2$. The graph $G \setminus X$ is the subgraph of G induced by $V \setminus X$. A graph G' is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. For every $v \in V$, $N(v)$ denotes the *neighborhood of v* that is the set of vertices adjacent to v. The graph G is *d-degenerate* if there exists an ordering $v_1, \dots, v_n$ of the vertices such that, for every i, $N(v_i) \cap \{v_{i+1}, \dots, v_n\}$ has size at most d. It refines the notion of maximum degree since any graph of maximum degree Δ are indeed Δ-degenerate (but the gap between Δ and the degeneracy can be arbitrarily large). Let $u, v \in V$, a *path* from u to v is a sequence of vertices $v_0 = u, v_1, \dots, v_\ell = v$ such that for every $i \leq \ell - 1$, $v_i v_{i+1}$ is an edge. It is a *cycle* if $v_\ell v_0$ also exists.

A graph G is *connected* if there exists a path from u to v for every pair $u, v \in V$. All along the paper, we only consider connected graphs. Indeed, in certification, the vertices can only communicate with their neighbors, so no vertex can communicate with vertices of another connected component.

A vertex v is a *cut-vertex* if $G \setminus \{v\}$ is not connected. If G does not contain any cut-vertex, G is *2-(vertex)-connected*. If the removal of any edge does not disconnect the graph, we say that G is *2-edge-connected*. A graph is *k-(vertex)-connected* if there does not exist any set X of size $k - 1$ such that $G \setminus X$ is not connected. To avoid cumbersome notations, we will simply write k-connected for k-vertex-connected.

A graph H is a *minor of G* if H can be obtained from G by deleting vertices, deleting edges and contracting edges. Equivalently, it means that, if G is connected, there exists a partition of V into connected sets $V_1, \dots, V_{|H|}$ such that there is (at least) an edge between V_i and V_j if $h_i h_j$ is an edge of H. We say that $V_1, \dots, V_{|H|}$ is a *model of H*. The graph G is H-minor-free if it does not contain H as a minor.

2.2 Local computation and certification

We assume that the graph is equipped with unique identifiers in polynomial range $[1, n^k]$, thus these identifiers can be encoded on $O(\log n)$ bits.

Local certification is a mechanism for verifying properties of labeled or unlabeled graphs. In this paper we will use a local certification at distance 1, which is basically the model called *proof-labeling schemes* [42]. A convenient way to describe a local certification is with a prover and a verifier. The *prover* is an external entity that assigns to every vertex v a certificate $c(v)$. The *verifier* is a distributed algorithm, in which every vertex v acts as follows: v collects the identifiers and the certificates of its neighbor and itself, and outputs a decision *accept* or *reject*. A local certification certifies a graph class $\mathcal{C}$ if the following two conditions are verified:

1. For every graph of $\mathcal{C}$, the prover can find a certificate assignment such that the verifier accepts, that is, all vertices output *accept*.

2. For every graph not in $\mathcal{C}$, there is no certificate assignment that makes the verifier accept, that is for every assignment, there is at least one vertex that rejects.

The size of the certificate of $\mathcal{C}$ is the largest size of a certificate assigned to a vertex of a graph of $\mathcal{C}$.

Note that to describe a local certification, the only essential part is the verifier algorithm, the prover is just a way to facilitate the description of a scheme.

In this paper, we are going to use a variant of the model above, called *edge certification*, where the certificates can be assigned on both the vertices and the edges. See Subsection 3.1.

2.3 Known building blocks for graph certification

There are few known certification schemes that we are going to use extensively as building blocks in the paper.

▶ **Lemma 4** ([2, 42]). *Acyclicity can be certified in $O(\log n)$ bits.*

The classic way to certify that the graph is acyclic, is for the prover to choose a root vertex, and then to give to every vertex as its certificate its distance to the root. The vertices can simply check that the distances are consistent.

The same idea can be used to certify a *spanning tree* of the graph, encoded locally at each vertex by the pointer to its parent, which is simply the ID of this parent. The scheme is the same, except that the prover, in addition to the distances, gives the ID of the root, and the verification algorithm checks that all vertices have been given the same root-ID, and only takes into account the edges that correspond to pointers (also the root checks that its ID is the root-ID). A spanning tree is a very useful tool to broadcast the *existence of a vertex satisfying a locally checkable property*: simply choose a spanning tree rooted at the special vertex, encode it locally with pointers and certify it. Then the root can check that indeed it has the right property, and all the other vertices know that such a vertex exists.

Finally, with the same ideas, one can easily deduce $O(\log n)$ certification for paths. We just add to the acyclicity scheme the verification that the degree of every vertex is at most 2. Note that cycles do not need certificates to be verified: every vertex just checks that it has degree exactly 2.

Let us now define a graph class that will appear in several decompositions.

▶ **Definition 5.** *A* path-outerplanar *graph is a graph that admits a path P that can be drawn on a horizontal line, such that all the edges that do not belong to P can be drawn above that line without crossings. The edges outside of P are said to be* nested.

We are going to use the following result as a black box.

▶ **Lemma 6** ([28]). *Path-outerplanar graphs can be certified with $O(\log n)$-bit certificates.*

The following classic result will also be useful at some point of the paper.

▶ **Lemma 7** ([42]). *Every graph class can be certified with $O(n^2)$ bits.*

The idea of the scheme is that the prover gives to every vertex v the map of the graph, *e.g.* as an adjacency matrix, along with the position of v in this map. Then every vertex can check that it has been given the same map as its neighbors, and that the map is consistent with its neighborhood in the network.

3 Avoiding certificate congestion

One can obtain many structured graph classes like minor-free graphs with "gluing" operations, for instance, by identifying vertices of two graphs of the class. If we have a certification for both graphs, we would like to simply take both certificate assignments to certify the new graph. However, for the vertex on which the two graphs are glued, the size of the certificate might have doubled. While it is not a problem for bounded degree graphs, it can become problematic if many gluing operations occur around the same vertex, since this vertex would get an additional certificate from each operation. In this section, we present two ways to tackle these issues, that will be used in the forthcoming sections.

The first one consists in shifting the certification on edges instead of vertices, which helps in the sense that when gluing on vertices the edge certificate can remain unchanged. As we will see, the edge setting is equivalent to the usual vertex certification for nice enough classes. The second option uses that one can (almost) freely assume that a given vertex has an empty label in a correct certification.

3.1 Edge certification and degeneracy

Note that any vertex certification is a special case of an edge certification (taking empty edge certificates). We can even actually transfer the certificates from vertices to edges to get an equivalent edge certification with empty vertex certificates, without additional asymptotic costs: just copy on every edge the certificate of the two endpoints, and adapt the verification algorithm accordingly. Transforming an edge certification into a vertex certification is also always possible, by giving a copy of the edge label to each of its endpoints. But this transformation can drastically increase the certificate size: if an edge certification uses $\Omega(f(n))$-bit labels, the associated vertex certification might use $\Omega(n \cdot f(n))$-bit if the maximum degree of the graph is linear. The following theorem ensures that in degenerate graph classes there is a more efficient transformation that permits to drastically reduce the size of the certificate.

▶ **Theorem 8** ([27]). *Consider an edge certification of a graph class $\mathcal{C}$ where only the edges are labeled, and get $f(n)$-bit certificates. If $\mathcal{C}$ is d-degenerate, then there exists a (vertex) certification with $d \cdot f(n)$-bit certificates.*

Note that H-minor-free graphs have degeneracy $O(h\sqrt{\log h})$ where $h = |V(H)|$ [43, 52]. Therefore, we can freely put labels on edges when certifying classes defined by forbidden minors.

3.2 Certification with one empty label

In this part, our goal is to erase the certificate of a vertex. To this end, we first consider certification of spanning trees and strengthen both Lemma 4 and the discussion that followed in Subsection 2.3. We then extend this intermediate step to every graph class in Lemma 10.

▶ **Lemma 9.** *Let T be a spanning tree of G. There exists a certification of T that does not assign a label to the root, and uses the same certificate as the classic tree certification (cf. Subsection 2.3) on the other vertices.*

Proof. On *yes*-instances, the prover assigns the labels as in the classic scheme, and removes the label of the root. Then the verification proceeds like in the classic scheme except for a vertex that has no label or a vertex that has a neighbor with no label. If two adjacent

vertices have been given an empty label, then they reject. If a vertex with no label sees that two of its neighbors have been given different root-ID, then it rejects. Otherwise, every vertex simulates the computation where the vertex with empty label has been given distance 0, and the same root-ID as its neighbors. Because of the previous checks, the labels used in the simulation are consistent, and on correct instance are the same as the one used in the classic certification. Thus, the correctness follows from the correctness of the classic scheme. ◀

A *pointed graph* is a graph with one selected vertex. Given a class, one can build its pointed version by taking for each graph all the pointed versions of it.

▶ **Lemma 10.** *Consider a class $\mathcal{C}$ that can be certified with certificates of size $f(n)$. One can certify the pointed class of $\mathcal{C}$ with $O(f(n) + \log n)$ bits, without having to put certificates on the selected vertex.*

Proof. First, to certify that exactly one vertex is pointed, we can simply find a spanning tree rooted on the pointed vertex and assign to each vertex the spanning tree certification of Lemma 9 which uses $O(\log n)$ bits. For the rest of the certification, on a *yes*-instance, the prover first assigns the certificates following the original certification. Then it removes the certificate of the selected vertex and appends copies of it to the certificates of its neighbors.

Every vertex v runs the following verification. If v is not the selected vertex, nor one of its neighbors, then it does the same verification as before. If v is the selected vertex, it checks that its neighbors have been given the same label as "label of the selected vertex", and then takes this label as its own, and runs the previous verification algorithm. If v is a neighbor of the selected vertex, it runs the same verification algorithm as before, but simulating that the selected vertex has been given the certificate that was appended to its own certificate.

All vertices are simulating the computation in the graph where the selected vertex would have been given its certificates, thus the correctness of this new certification follows from the correctness of the original certification. ◀

Observe that the previous results can be easily iterated: one can always remove the labels of k vertices (as long as they are pairwise non-adjacent) to the cost of a factor k in the size of the certificates. Therefore, the result extends to the case of k-independent pointed classes (i.e. where an independent set of size at most k is selected instead of only one vertex).

▶ **Corollary 11.** *Consider a class that can be certified with certificates of size $f(n)$. One can certify the k-independent pointed class with $O(kf(n) + k \log n)$ bits, without having to put certificates on the selected vertices.*

Moreover, with more constraints on the structure of the set of pointed vertices (for instance if they are all at distance at least 3), one could even obtain certificate of size $O(f(n) + k \log n)$ (since every vertex receives the certificate of at most one selected vertex).

4 Compositions of certifications

In this section, we show how to combine certification algorithms for several classes to certify larger ones, and we illustrate this idea on two constructions. The first one considers classes defined by the existence of some subgraph: we settle the intuition stating that it is often easier to test the existence of a structure rather than its absence, since we can pinpoint which vertices/edges lie in the structure.

The second construction mimics a natural operation on graphs, consisting in replacing some vertex/edge by another graph. This operation occurs quite often in the literature:

many classes, especially the ones defined by forbidden minors, get a characterization using this operation.

Some results of this section will not be used to certify minor-free classes later in the paper. We nonetheless include them since they correspond to very natural operations which may be useful for future work.

4.1 Subgraphs

▶ **Proposition 12.** *Let $\mathcal{C}$ be a graph class that can be certified with $f(n)$-bit labels. Let $\mathcal{C}'$ be the class of the graphs that contain a graph of $\mathcal{C}$ as subgraph. Then $\mathcal{C}'$ can be certified with certificates of size $O(f(n) + \log n)$ on the vertices and $O(1)$ on the edges.*

Proof. On a *yes*-instance G, the prover assigns the certificates on vertices and edges in the following way. First, it chooses a subgraph H that belongs to $\mathcal{C}$ and assigns the certificates that certify that H is in $\mathcal{C}$, as if the rest of the graph did not exist. This takes at most $f(n)$ bits. Second to every vertex and edge that belongs to H, the prover assigns a special label. Third, the prover describes and certifies a spanning tree pointing to a vertex that has the special label.

The verification algorithm is the following. The vertices that have the special label, run the verification algorithm for $\mathcal{C}$, taking into account only the vertices and edges that have the special label. The vertices also check the spanning tree structure, and the root of the tree checks that it does have the special label.

Because of the spanning tree, there must exist a vertex with the special label, thus there are vertices that run the verification algorithm for $\mathcal{C}$, and if they succeed it means that a graph of $\mathcal{C}$ appears as a subgraph in the graph G. ◀

The edge certificates in Proposition 12 can be inconvenient if we want a classic certification (without edge certificates) and if the graph is not assumed to be degenerate, which prevents us from using Theorem 8. However, observe that we give non-empty certificates only to the edges of the subgraph, hence we can obtain a vertex-certification when the class $\mathcal{C}$ is degenerate.

▶ **Corollary 13.** *Let $\mathcal{C}$ be a d-degenerate graph class that can be certified with $f(n)$-bit labels. Let $\mathcal{C}'$ be the class of the graphs that contain a graph of $\mathcal{C}$ as a subgraph. Then $\mathcal{C}'$ can be certified with certificates of size $O(f(n) + d \log n)$ on the vertices.*

Observe also that when considering induced subgraphs, we only have to specify which vertices are special, hence we do not need edge certificates either. Note that, since we do not need to label edges, we do not need the class $\mathcal{C}$ to be degenerate.

▶ **Corollary 14.** *Let $\mathcal{C}$ be a graph class that can be certified with $f(n)$-bit labels. Let $\mathcal{C}'$ be the class of the graphs that contain a graph of $\mathcal{C}$ as an induced subgraph. Then $\mathcal{C}'$ can be certified with certificates of size $O(f(n) + \log n)$ on the vertices.*

4.2 Expansions

Two common operations in characterizations of graph classes are what we call vertex and edge expansions.

▶ **Definition 15.** *Consider two graph classes $\mathcal{C}_1$ and $\mathcal{C}_2$.*

- *The* vertex expansion of $\mathcal{C}_1$ by $\mathcal{C}_2$ *is the class of graphs obtained by the following operation. Take a graph G in $\mathcal{C}_1$ and replace every vertex v by a graph $H(v)$ in $\mathcal{C}_2$, in such a way that for every edge $uv \in E(G)$, there is (at least) one edge between $H(u)$ and $H(v)$ in G (and no such edge if $uv \notin E(G)$).*
- *The* edge expansion of $\mathcal{C}_1$ by $\mathcal{C}_2$ *is the class of graphs obtained by the following operation. Take a graph G in $\mathcal{C}_1$ and replace every edge uv by a graph $H(u,v)$ from $\mathcal{C}_2$, in such a way that the vertices of the original graph that are contained in $H(u,v)$ are exactly u and v.*

We would like to have results of the form: if $\mathcal{C}_1$ and $\mathcal{C}_2$ can be certified with $f(n)$ and $g(n)$-bit labels respectively, then the expansion can be certified with $O(f(n)+g(n))$-bit labels. While the natural approach (almost) works for edge-expansion, it does not give such a result for vertex-expansion. However, we can actually make it work with a bound that takes into account the maximum degree of the expanded graph.

▶ **Proposition 16.** *Consider two graph classes $\mathcal{C}_1$ and $\mathcal{C}_2$ that can be certified with $f(n)$-bit and $g(n)$-bit labels respectively, where all the graphs of $\mathcal{C}_1$ have maximum degree Δ. Then the vertex-expansion of $\mathcal{C}_1$ by $\mathcal{C}_2$ can be certified with $O(\Delta \cdot f(n) + g(n) + \Delta \log n)$-bit certificates.*

Proof. Consider a graph $G \in \mathcal{C}_1$ on ℓ vertices $v_1, \ldots, v_\ell$ of maximum degree Δ, and let $H_1, \ldots, H_\ell$ be graphs of $\mathcal{C}_2$. We consider the vertex expansion of G where every v_i is replaced by H_i.

On a *yes*-instance, the prover assigns the certificates the following way. First it assigns to every vertex the index i corresponding to the graph H_i it belongs to and the certification of the fact that H_i belongs to $\mathcal{C}_2$ (without taking into accounts the other vertices and edges). This takes at most $g(n) + \log n$ bits per vertex. Second, the prover gives to each vertex of H_i the original certificate of v_i that G belongs to $\mathcal{C}_1$ as well as the original certificate of all the vertices in $N(v_i)$ in G together with their names, which takes $O(\Delta f(n))$ bits. Finally, for every $v_j \in N(v_i)$, the prover chooses a vertex w_j in H_i adjacent to a vertex in H_j, and certifies a spanning tree of H_i rooted at w_j. This takes $O(\Delta \log n)$ bits.

The verification algorithm is the following. Every vertex (labeled as) in H_i checks that the number of trees corresponds to the degree of v_i in G. Every vertex checks the correctness of the different trees. Moreover, every root v of a spanning tree in H_i checks that it has a neighbor in the corresponding H_j. All the vertices of H_i check that their neighbors are in H_i or in some H_j with v_j incident to v_i in G. Every vertex of each H_i runs the verification algorithm to check that H_i does belong to $\mathcal{C}_2$. Finally, every vertex of H_i simulates the verification of the original vertex v_i, which is possible since every vertex of H_i receives the certificate of v_i and all its neighbors in G. And every vertex $w_i \in H_i$ incident to $w_j \in H_j$ checks that $v_j \in N_G(v_i)$ and that the certificate of w_j indeed contains the certificates of v_i and v_j given for G. ◀

▶ **Proposition 17.** *Consider two graph classes $\mathcal{C}_1$ and $\mathcal{C}_2$ that can be certified with $f(n)$-bit and $g(n)$-bit labels respectively. Then the edge-expansion of $\mathcal{C}_1$ by $\mathcal{C}_2$ can be certified with $O(f(n)+g(n)+\log n)$-bit certificates on the edges.*

Proof. We use a similar reasoning as for the proof of Proposition 16, except that we first transform the vertex certifications of $\mathcal{C}_1$ and $\mathcal{C}_2$ into edge certifications (by putting the label of a vertex on all the edges incident with it).

Consider a graph $G \in \mathcal{C}_1$. We consider the edge expansion of G where every uv is replaced by $H(u,v)$. Each edge e from $H(u,v)$ receives the labels of u and v, the certificate of uv

in G for $\mathcal{C}_1$, and the certificate of e in $H(u,v)$ for $\mathcal{C}_2$. Therefore, the certificates have size $O(f(n)+g(n)+\log n)$.

Now each vertex can check that all the edges labeled in some $H(u,v)$ share the same certificate for uv. There are two kinds of vertices: some where all incident edges are labeled as in the same $H(u,v)$, and the others (the original vertices of G). All of them run the verification algorithm for $\mathcal{C}_2$ by considering each group of incident edges labeled as in the same $H(u,v)$. The latter also recover the certificates of their neighbors in G from the edge labeling, and run the verification algorithm for $\mathcal{C}_1$. ◀

Before giving deeper applications of these results in future sections, let us prove that the *existence* of a minor in the graph is easy to certify. This was already mentioned in previous papers without formal proofs [27, 28]. We prove it here to show a simple application of our techniques, and we think it is a meaningful illustration of the fact that certifying that a structure is present or absent are two very different tasks in our model.

▶ **Corollary 18.** *Given a graph H, one can certify that an n-vertex graph has H as a minor in $O(\log n)$ bits.*

Proof. As we already observed, a graph G has H as minor if and only if $V(G)$ can be partitioned into $|H|$ connected sets such that there is an edge between V_i and V_j when the corresponding vertices in H are connected. Free to delete edges, we can assume that each V_i is actually a spanning tree and there is a unique edge from V_i to V_j if and only if the corresponding vertices are connected in H. In other words, G has a subgraph that is a vertex expansion of H by trees. Moreover, we can choose such a subgraph with degree at most $|H|-1=O(1)$ since H is fixed.

Let us start from a certification of H and build a certification of G. The structure of H can be certified in a brute-force way, by providing to every vertex the complete map of the graph which takes constant space (since H is fixed). Then, since trees can be certified in $O(\log n)$ bits, thanks to Proposition 16, any vertex-expansion of H by trees can be certified with $O(\log n)$ certificates.

We finally get a vertex certification with certificates of size $O(\log n)$ using Corollary 13. ◀

5 Connectivity and connectivity decompositions

In this section, we explain how to certify connectivity properties and connectivity decompositions, in particular the block-cut tree mentioned in the introduction.

An *ear decomposition* is a way to build a graph by iteratively adding paths, the so-called *ears*. Ear decompositions are central tools for decades in structural graph theory and are used in many decomposition or algorithmic results. There exist various variants of this process, that characterize different classes and properties. For certification, these decompositions happen to be easier to manipulate than some other types of characterizations since they are based on iterative construction of the graph, and use paths, which are easy to certify. These paths are convenient since we can "propagate" some quantity of information on them as long as every vertex belongs to a bounded number of paths. In this section, we remind several such decompositions, and use them to certify various connectivity properties and decompositions.

5.1 Connectivity properties

Let us start with 2-connectivity. A graph G has an *open ear decomposition* if G can be built, by starting from a single edge, and iteratively applying the following process: take

two different vertices of the current graph and link them by a path whose internal vertices are new vertices of the graph (such a path is called *an ear*). Note that this path can be a single edge, and then there is no new vertex. Let an *inner vertex* of an ear be a vertex that is created with this ear, and let a *long ear* be an ear with at least one inner vertex.

▶ **Theorem 19** ([54] (reformulated)). *A graph is 2-connected if and only if it has an open ear decomposition.*

We use this characterization to certify 2-connectivity.

▶ **Lemma 20.** *2-connected graphs can be certified with $O(\log n)$ bits.*

Proof. First observe that one can obtain a long-ear decomposition from an ear decomposition (and vice versa) by removing/adding short ears, i.e. edges. Therefore, having an open ear decomposition is equivalent to having a subgraph with an open long-ear decomposition. Note that if a graph G has an open long-ear decomposition, then it is 2-degenerate. Indeed, the vertices of the last added long-ear have degree two and their removal is still a graph with an open long-ear decomposition. So in order to get the conclusion, Corollary 13 ensures that we only have to certify open long-ear decomposition with $O(\log n)$ bits per vertex.

The certification works as follows. First the prover gives to every vertex the identifiers of the very first edge, and describes and certifies a spanning tree pointing to one of the endpoints of this edge. The vertices of this edge are given an *index* 0. Second, the prover gives to every vertex the information related to the step when it has been added, and only about this step. That is, the prover gives the *index* of the addition (that is the number of the ear in which the vertex is created), along with two oriented paths spanning the path and pointing to the two extremities of the ear. By Corollary 11, these paths can be certified without certificates on the extremities of the paths.

Every vertex checks the correctness of the spanning tree pointing to the first edge, and the fact that only these vertices have index 0. Then, every vertex also checks that the spanning paths it has been given are correct, that is: (1) the distances and root-ID are consistent (2) all vertices have the same index, and that (3) the declared endpoints are different. Also, the two vertices that are adjacent to the endpoints of the paths check that the endpoints have a smaller index than their own.

Let us now prove the correctness of the certification. Because of the spanning tree, the original edge exists, is unique, and is the only set of vertices with index 0. Because of the certified paths spanning the ears, one can also recover the path structure and the fact that a path is added after its endpoints. Note that in an instance where all vertices accept, there might be two different paths with the same index i, but this is not a problem: the only important feature is the precedence order. ◀

With similar construction we can certify the edge connectivity instead of the vertex connectivity.

▶ **Corollary 21.** *2-edge-connected graphs can be certified with $O(\log n)$ bits.*

Proof. Robbins proved in [48] that a graph G is 2-edge connected if and only if G has an ear decomposition. An ear decomposition is the same as an open ear decomposition, except that it starts from a cycle and that the two endpoints of an ear do not need to be different. The proof above can thus be adapted to this class.

The only difference is that vertices with index 0 form a cycle (which can be certified). Then during the verification procedure we simply do not have to check that the extremities of the path of the ear decomposition are distinct, in other words we do not have to check (3). ◀

A more refined type of ear decomposition characterizes the 3–connected graphs.

▶ **Definition 22** ([12, 45, 51])**.** *Let ru and rt be two edges of a graph G. A* Mondshein sequence through rt, avoiding u *is an open ear decomposition of G such that:*

1. *rt is in the first ear.*
2. *the ear that creates vertex u is the last long ear, u is its only inner vertex, and it does not contain ru.*
3. *the ear decomposition is non-separating, that is, for every long ear except the last one, every inner vertex has a neighbor that is created in a later ear.*

▶ **Theorem 23** ([12, 51])**.** *Let ru and rt be two edges of a graph G. The graph G is 3–connected if and only if it has a Mondshein sequence through rt avoiding u, and there are three internally vertex-disjoint path between t and u.*

▶ **Corollary 24.** *3-connected graphs can be certified with $O(\log n)$ bits on vertices.*

Proof. On *yes*-instances the prover chooses an arbitrary edge ru and certifies the ear decomposition as in Lemma 20. The prover also adds a spanning tree pointing to the edges ru and rt, and gives to every vertex the index of the last long ear created. These new pieces of information allow the vertices to check that the ear decomposition is a Mondshein sequence. The prover also encode the three vertex disjoint paths, by pointer on the vertices of these paths, and number them 1, 2 and 3, to allow the vertices to check disjointness. ◀

5.2 Block-cut tree

Now that we can certify connectivity properties, we introduce a way to certify decomposition of graphs into parts of higher connectivity. Let us start with a few definitions.

A *2-connected component* of a graph G is a connected subgraph H maximal by inclusion such that the removal of one vertex does not disconnect H. Observe that a 2-connected component can consists of just one edge in the case of a bridge, i.e. an edge whose removal disconnects the graph.

The intersection of any pair C, C' of 2-connected components has size at most one. Indeed, if it had size at least two, then we could merge these into a larger 2-connected component, which would contradict the maximality. So we can define an auxiliary graph from G where every vertex corresponds to a 2-connected component and there is an edge between two components if and only if they intersect on exactly one vertex. This graph is a tree, because a cycle would again create a larger 2-connected component, contradicting maximality. This tree is called the *block-cut tree*.

Let T be a block-cut tree of G, and D a maximal 2-connected component chosen to be the root of this tree. (Note that if G is 2-connected then the graph is reduced to this component). Let C be a component that is not the root of the tree. The *connecting vertex* of a component C is the vertex lying both in C and in its parent component. The *interior* of C is the set of vertices of C minus the connecting vertex of C. Note that the interior of a component is always non-empty.

This section is devoted to proving the following result and apply it for certification:

▶ **Theorem 3.** *For any 2-connected graph H, if the 2-connected H-minor-free graphs can be certified with $f(n)$ bits, then the H-minor-free graphs can be certified with $O(f(n) + \log n)$ bits.*

Proof of Theorem 3. Since H is 2-connected, a graph G is H-minor-free if and only if each of its 2-connected components is. (This is basically the observation we made at the beginning

of Subesction 1.2.) This is the property we certify. On a *yes*-instance, the prover will assign the certificates the following way. It first computes the block-cut tree and root it on some vertex C. It then does the following:

1. For each 2-connected component, the prover chooses a vertex from the interior of the component to be the *leader* of this component. Every vertex of the interior of a component C is given the identifier of the leader of C as well as a spanning tree of C pointing towards it. Since the component is 2-connected, the component minus the leader of the component is connected and such a tree exists.
2. Every vertex is given a label stating whether it is a connecting vertex or not.
3. Every vertex is given the identifier of the connecting vertex of its component closest from the root in the block-cut tree (called the *component* of the vertex), as well as a spanning tree pointing to it, using the certification of Lemma 9 that uses an empty certificate on the root.
4. In order to check acyclicity of the block-cut tree, every vertex is given the distance of its component to the root-component (in terms of number of components).
5. The prover certifies the 2-connectivity of each component using the certification of Lemma 20 and the fact that it is H-minor-free using the certification with $f(n)$ bits of the theorem. By Lemma 10 this can be done by only assigning labels to the interior vertices of the component.

Before we move on to the verification and the correctness of the scheme, note that every vertex is given a certificate of size $O(f(n) + \log n)$. Indeed, each piece of information we have given to the vertex is of size $O(\log n)$ or $f(n)$, and we have given a constant number of those to every vertex. In particular, a connecting vertex in the interior of a component C, received only labels that are related C, and not labels related to other components it belongs to (since we consider pointed components).

Now, every vertex does the following verification. Every vertex checks that the spanning tree pointing to the leader is correct. If this step succeeds, we have a partition of the vertices in components. Every vertex also checks the correctness of the spanning tree pointing to the connecting vertex.

Every vertex v checks that, if it has an edge to a connecting vertex w with a different leader, then w is the connecting vertex of its own component. Every connecting vertex v checks that it is connected to a single vertex in its parent component and that it is the claimed neighbor in that component. If this step succeeds, we have a decomposition into components linked by connecting vertices. The consistency of the component distances are also checked by the vertices: this distance should be decremented at each connecting vertex, and only there. This ensures the acyclicity of the component structure. Finally, every vertex checks that the 2-connectivity and the H-minor-freeness of its component. Globally this verification ensures that the graph is H-minor-free. ◀

6 Application to C_4, C_5, Diamond, K_4 and $K_{2,3}$ minor-free graphs

This section is devoted to the certification of C_4-minor-free, diamond-minor-free graphs, K_4-minor-free graphs and $K_{2,3}$-minor-free graphs. All the proofs will follow the same structure: prove that the 2-connected components, which are more structured, can be certified with small labels, and then use Theorem 3 to conclude for the general case.

Before proving this, let us describe how to certify *series-parallel graphs*, which in addition to being interesting network topologies [30], are closely related to K_4-minor-free graphs [17].

▶ **Definition 25.** *A* (2-terminal) series-parallel graph *is a graph with two labeled vertices called the* source *and the* sink *that can be built recursively as follows. A single edge is a series-parallel graph where one endpoint is the source and the other is the sink. Let G_1, G_2 be two series-parallel graphs. The* series *of G_1 and G_2 which consists in merging the sink of G_1 and the source of G_2 is a series-parallel graph. The* parallel *of G_1 and G_2, which consists in merging the sources of G_1 and G_2 together and merging the sinks of G_1 and G_2 together, is a series-parallel graph.*[1]

A *nested ear decomposition* is an open ear decomposition that starts from a path, with two properties: (1) both ends of an ear e have to lie on a same ear added before e, and (2) for every ear e, the subpaths of e linking endpoints of ears plugged on e are pairwise either edge-disjoint or included one in the other. We say that the ears plugged on e are nested. Eppstein proved the following in [20] about series-parallel graphs.

▶ **Theorem 26** ([20])**.** *A 2-connected graph is series-parallel if and only if it has a* nested ear decomposition*.*

We will use this decomposition theorem for our certification.

▶ **Theorem 27.** *2-connected series-parallel graphs can be certified with $O(\log n)$-bit labels.*

Proof. The prover certifies the decomposition of Theorem 26. We have already described how to certify an open ear decomposition in the proof of Lemma 20. We can easily adapt it so that it starts from a path instead of an edge: there is a spanning tree pointing to one of the endpoints of the paths, and the path itself is certified with distances, the usual way.

It is also easy to certify that each ear e has both of its endpoints on the same older ear e': just give to each vertex of e the identifiers of the endpoints of e and e'. The endpoints of an ear can check the consistency of these announced identifiers with the identifiers of their paths. A more tricky part is to certify that the ears are nested. Remember that Lemma 6 states that a path with nested edges (a path-outerplanar graph) can be certified with $O(\log n)$-bit labels. This is exactly what we need except that we would like to have nested paths instead of nested edges. But then we can transfer the information from one endpoint of the paths to the other endpoint. ◀

▶ **Lemma 28.** *2-connected C_5-minor-free graphs are either graphs of size at most 4 or $K_{2,p}$ or $K'_{2,p}$ which is the complete bipartite graph $K_{2,p}$ plus an edge between the two vertices on the set of size 2.*

Proof. Since G is 2-connected, by Menger's theorem, for every pair x, y of non-adjacent vertices, there exist at least two vertex disjoint xy-paths. Since G is C_5-minor-free, these paths have size at most 2, in particular x, y are at distance at most 2.

Let u, v be two non-adjacent vertices. Then the removal of $N(u) \cap N(v)$ disconnects u from v since otherwise we can find two vertex disjoint uv-paths, one being of size at least 3, which provides a C_5. In particular, it implies that $|N(u) \cap N(v)| \geq 2$ since G is 2-connected.

Let $x \in N(u) \setminus (N(v) \cup \{v\})$. Since x, v are non-adjacent, they are at distance 2 hence there must be an edge between x and $N(v)$. But this creates a C_5 since $|N(u) \cap N(v)| \geqslant 2$. Therefore non-adjacent vertices have the same neighborhood.

Let I be a maximum independent set in G. Note that all the vertices of I have the same neighborhood. Therefore, by maximality, if $u \notin I$ then u is adjacent to every vertex of I.

[1] We only keep a single edge for each multi-edge created by this process.

Now either vertices of I have degree at least 3, and G contains $K_{3,3}$ hence a C_5-minor, or vertices of I have degree 2 and G is $K_{2,p}$ or $K'_{2,p}$. ◀

We can now easily prove the claimed certifications.

▶ **Corollary 29.** *The following classes of graphs can be certified with $O(\log n)$ bit certificates: C_4-minor-free graphs, C_5-minor-free graphs, diamond-minor-free graphs, house-minor-free graphs*[2]*, outerplanar graphs (that is $(K_{2,3}, K_4)$-minor-free graphs), $K_{2,3}$-minor-free and K_4-minor-free graphs.*

Proof. By Theorem 3, if we can certify the 2-connected graphs of these classes we obtain the conclusion. So we simply have to prove that for each class we can certify the 2-connected graph of the class.

- 2-connected C_4-minor-free graphs are K_2 and K_3 [13], which can be certified with $O(1)$ bits.
- 2-connected C_5-minor-free graphs are either graphs of size at most 4 or a complete bipartite graph $K_{2,p}$ (with a potential edge between the two vertices in the set of size 2 by Lemma 28. Since such graphs can be certified with $O(\log n)$ bits, the conclusion follows.
- 2-connected diamond-minor-free graphs are induced cycles. Cycles can be certified with $O(1)$ bits (see the discussion after Lemma 4).
- 2-connected house-minor-free graphs are either induced cycles or graphs of size at most four. Indeed, assume that there is a cycle of length at least 5. Then it should be induced, since otherwise it contains a house as a minor. Moreover, it should contain all the vertices of the graph otherwise there is an ear starting from this cycle and the cycle plus the ear provides a house. Since induced cycles can be easily certified with $O(\log n)$ bits, the conclusion follows.
- 2-connected outerplanar graphs are exactly path-outerplanar graphs with an edge between the first and the last vertex. Indeed, by 2-connectivity, the outer face must be a cycle, and removing any edge from it yields a path-outerplanar graph. One can then certify the existence and uniqueness of this edge using a spanning tree, and then certify that the rest of the graph is path-outerplanar. This yields a $O(\log n)$-bit certification by Theorem 6.
- Let G be a 2-connected $K_{2,3}$-minor-free graph. If G does not contain K_4 as a minor, then it is a 2-connected outerplanar graph and the result follows from the previous item. Otherwise, G contains K_4 as a minor. Observe that in particular G contains a subdivision of K_4 as a subgraph (by keeping only the edges of a spanning tree in each bag). But all such subdivisions contain $K_{2,3}$ as a minor except for K_4 itself. Therefore, G is K_4, which can be certified easily.
- The 2-connected K_4-minor-free graphs are exactly the 2-connected series-parallel graphs [17]. Then the results follow directly from Theorem 27. ◀

7 Application to $K_{2,4}$-minor-free graphs

When the size of the minors are increasing (and for most of the decomposition theorems known in structural graph theory), 2-connectivity is not enough. In this example we will illustrate how to use the certificate of 3-connectivity.

[2] The *house* being a C_4 plus a vertex connected to two consecutive vertices of the C_4.

Let us illustrate it for this section on the characterization of $K_{2,4}$-minor-free graphs from [19]. It is more involved than the other characterizations we have seen so far. We will follow the structure of [19], restricting first to 3-connected graphs, then to 2-connected graphs, and finally all $K_{2,4}$-minor-free graphs.

7.1 3-connected case

Let us start with the definition of a graph class. We use notations similar to [19] (Section 2.1). See Figure 6.

▶ **Definition 30.** *Let n, r, s be three integers, and p a Boolean. The graph $G_{n,r,s,p}$ consists of a path $v_1, ..., v_n$, the edges v_1v_{n-i} for $1 \leq i \leq r$, the edges v_nv_{1+j} for $1 \leq j \leq s$, and the edge v_1v_n if $p = 1$. For a function $f(n, r, s, p)$ that associates a boolean value with each combination of parameters, let $\mathcal{G}[f]$ be the set of graphs $G_{n,r,s,p}$ such that $f(n, r, s, p) = 1$.*

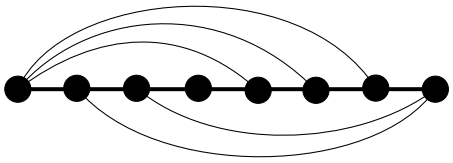

Figure 6 Example for Definition 30. This graph is $G_{8,3,2,0}$: it has 8 vertices, edges v_1v_{n-i} for $i \in \{1, 2, 3\}$, edges v_nv_{1+j} for $j \in \{1, 2\}$, and it does not have the edge v_1v_n.

▶ **Theorem 31** (Theorem 2.12 in [19] (adapted))**.** *There exists an f such that the set of 3-connected $K_{2,4}$-minor-free graphs is $\mathcal{G}[f]$, plus nine graphs on at most 8 vertices.*

In [19], the authors give an explicit description of f but we can avoid going into details here because of the following general lemma.

▶ **Lemma 32.** *For all f, $\mathcal{G}[f]$ can be certified with $O(\log n)$-bit labels.*

Proof. On a *yes*-instance, the prover certifies the spanning paths with root v_1, with last vertex v_n, and writes in each certificate the values n, r, s and p. The vertices check the structure of the path and the fact that n, r, s and p are the same on all vertices. Second v_1 checks the structure of its neighborhood, and in particular the values r and p. Similarly, v_n checks the structure of its neighborhood, and in particular the values s, and the fact that its distance to the root is indeed n. Finally, all vertices check that $f(n, r, s, p) = 1$. The correctness of the scheme is straightforward. ◀

This directly yields the following lemma.

▶ **Lemma 33.** *3-connected $K_{2,4}$-minor-free graphs can be certified with $O(\log n)$-bit labels.*

Proof. Consider a *yes*-instance. By Theorem 31, either it is one of the nine small graphs of Theorem 31, and then we can use a constant size certification, or it is a graph of $\mathcal{G}[f]$ for the specific f of Theorem 31, and then we can use Lemma 32. ◀

For the 2-connected case, one of the types of graphs that we want to certify is of the following form: a 3-connected graph, where a set of edges with a special property is expanded

with another graph class. To be able to certify this, we will need the vertices to check that the set of edges that has been expanded has the special property. To capture the notion of special property, without going into the intricate details of what this property is exactly, let us define an *edge-set decider*. A function h is an edge-set decider if it takes as input a 3-connected $K_{2,4}$-minor-free graph whose edges are either unlabeled, or labeled with a special label, and outputs a Boolean.

▶ **Lemma 34.** *Let h be an edge-set decider, such that for every graph G, there is at most $O(n)$ different sets of edges S such that $h(G,S)=1$. The set of 3-connected $K_{2,4}$-minor-free graphs G with labelled edges, where $h(G)$ is true, can be certified with $O(\log n)$ bits.*

Proof. First, for every graph G, we fix an indexing of edge sets S such that $h(G,S)=1$. The prover first uses the same certificates as in Lemma 33 for the certification of unlabeled 3-connected $K_{2,4}$-minor-free graphs. Then it gives to all vertices the index of the set of labeled edges. Following the certification of the proof of Theorem 31, every vertex knows in which graph it lives and what is its position in that graph. Then, every vertex just checks that the labeled edges in its neighborhood correspond to the index announced by the prover. The labels have size $O(\log n)$ because of Lemma 33 and because there are at most $O(n)$ different sets of edges S such that $h(G,S)=1$. ◀

7.2 2-connected case

We now state the characterization theorem of the 2-connected case.

▶ **Theorem 35** (Theorem 3.5 in [19])**.** *There exists a function h as in Lemma 34 such that the following holds. A graph G is 2-connected $K_{2,4}$-minor-free graph if and only if one of the following holds:*

1. *G is outerplanar.*
2. *G is the union of three path-outerplanar graphs H_1, H_2, H_3 with the same path endpoints x and y, and possibly the edge (x,y), where $|V(H_i)|\geq 3$, for each i and $V(H_i)\cap V(H_j)=x,y$ for $i\neq j$.*
3. *G is obtained from a 3-connected $K_{2,4}$-minor-free graph G_0 by choosing a subset S such that $h(G_0,S)=1$, and replacing each edges (x_i,y_i) of S by a path-outerplanar graphs H_i with endpoints (x_i,y_i) , where $V(H_i)\cap V(G_0)=\{x_i,y_i\}$ for each i, and $V(H_i)\cap V(H_j)\subset V(G_0)$ for $i\neq j$.*

In [19], h is called the set of subdividable edges, and is fully characterized. Our proof works for any h, as long as it satisfies the properties of Lemma 33, and it is the case for the h of [19].

▶ **Lemma 36.** *2-connected $K_{2,4}$-minor-free graphs can be certified with $O(\log n)$-bit labels.*

Proof. We show that each of the three cases can be certified with $O(\log n)$ bits.

1. Outerplanar graphs can be certified with $O(\log n)$ bits (Corollary 29).
2. This case basically consists in an edge expansion of a multigraph with three edges between two vertices by path outerplanar graphs. Note that the proof of Proposition 17 works here even if the original graph is a multigraph. Proposition 17 gives us an $O(\log n)$ edge certification because path-outerplanar graphs can be certified with $O(\log n)$ certificates, and the condition on the number of vertices can also be certified with $O(\log n)$ bits with a spanning tree counting the number of vertices (see *e.g.* in [24]). This edge certification can be transferred to a vertex certification with the same certificate size asymptotically because of Theorem 8, and because H-minor-free graphs have bounded degeneracy.

3. Again, this item basically corresponds to an edge-expansion: the edge expansion of a 3-connected $K_{2,4}$-minor-free graph by path-outerplanar graphs. We know by Lemma 33 and Lemma 6 that both these classes can be certified on $O(\log n)$ bits, so the vanilla edge-expansion can also be certified with $O(\log n)$ bits (using the degeneracy like in the previous item). The only issue left is the fact that the only edges of G_0 that are allowed to be expanded by something different from an edge need to belong to an S such that $h(G_0, S) = 1$. But this is easy with Lemma 34: the edges that have a path-outerplanar expansion are the one that are considered to have a special label. ◀

8 Certifying H-minor-free graphs with $|H| \leq 4$

In previous sections, we have proven that certifying H-minor-free graphs can be done with $O(\log n)$ bits for some graphs H. The graphs we have treated in previous sections are somehow amongst the hardest graphs of small size. When the connectivity of the graph H increases, the class of H-minor-free graph contains more and more graphs, and then is (morally speaking) harder to certify. Let us prove that the other graphs on 4 vertices (which have fewer edges, and then are less connected) can also be certified, with arguments either simpler than or similar to what has been done in previous sections, to establish the following theorem.

▶ **Theorem 2.** *H-minor-free classes can be certified in $O(\log n)$ bits when H has at most 4 vertices.*

We consider two cases depending on whether H contains a cycle.

▶ **Lemma 37.** *If $|H| \leq 4$, and H contains a cycle, then H-minor-free graphs can be certified with $O(\log n)$ bits.*

Proof. Since H contains a cycle, either it is C_4, and the result follows from Corollary 29, or it contains a triangle. Let us distinguish the cases depending on how the fourth vertex is connected to the triangle. If it is connected to two or three vertices, then H is either K_4, or a diamond, and then H-minor-free graphs can be certified with $O(\log n)$ bits by Corollary 29.

So we can assume that H is a triangle plus one vertex attached to at most one vertex of the triangle. If G contains a cycle, let C be a shortest cycle in G, that is a cycle that contains the minimum number of vertices. Then C must contain all the vertices of the graph. Indeed, otherwise, since G is connected, there exists a vertex v attached to C, and $v \cup C$ contains H as a minor. Therefore, G is either a cycle or a tree, which can be both certified in $O(\log n)$ bits, see Subsection 2.3. ◀

▶ **Lemma 38.** *If $|H| \leq 4$ and H is acyclic, then we can certify H-minor-free graphs with $O(\log n)$ bits.*

Proof. If H has an isolated vertex then any graph G contain H has a minor as long as G contains a (non necessarily induced) path on three vertices and an isolated vertex. Since this property holds for every connected graph on 4 vertices, the conclusion follows. If H consists of two independent edges, then each H-minor-free graph is either a triangle or a star, which can be certified with $O(1)$ bit certificates.

So we can assume that H is connected. There are only two acyclic connected graphs on 4 vertices: the star $S_{1,3}$ with 3 leaves, and the path P_4 on four vertices. If G does not contain a star with 3 leaves as a minor, it means that G is either a path or a cycle which can be easily certified. If G does not contain a path on four vertices as a minor, it means that G is a

star which, can be certified the following way. Give the identifier of the center to all vertices, and let the vertices check that they have been given the same ID, and that the non-center vertices have exactly one neighbor, and that this neighbor has this ID. ◀

This completes the picture for graphs H on at most 4 vertices.

9 Graphs on at most 5 vertices

Let us now focus on graphs H with at most 5 vertices. We were not able to deal with all of them, the most problematic one being K_5, as we will discuss later on. However, we proved that H-minor-freeness can be certified for some dense graphs like $K_{2,3}$. The goal of this section is to provide evidence that again, the hardest case will be the case where H is dense. Before entering into the details of the proof, let us study some necessary conditions on the graph to be minimally not H-minor-free.

9.1 H-minimal graphs

A graph G is *H-minimal* if G admits a H-minor but, for any vertex v, $G \setminus v$ does not admit any H-minor. Note that if H is not connected, H-minimal graphs may not be connected. Our goal is to study the structure of H-minimal graphs when H has at most 4 vertices, and show that they fall into a few easily-certifiable types.

Consider a model $V_1, \dots, V_{|H|}$ of H in a H-minimal graph G. Intuitively, for all i, the important part of the subgraph induced by V_i is a spanning tree that makes it connected, and connected to the neighboring V_j's. For example, if a V_i contains a vertex that is only connected to other vertices of V_i, and whose removal does not disconnect the subgraph of V_i, then this vertex is unessential. In other words, such a vertex would not appear in a H-minimal graph, because we could remove it, and still have a model of H. Nevertheless, it is not true that the subgraph induced by every V_i is a tree (see Figure 7).

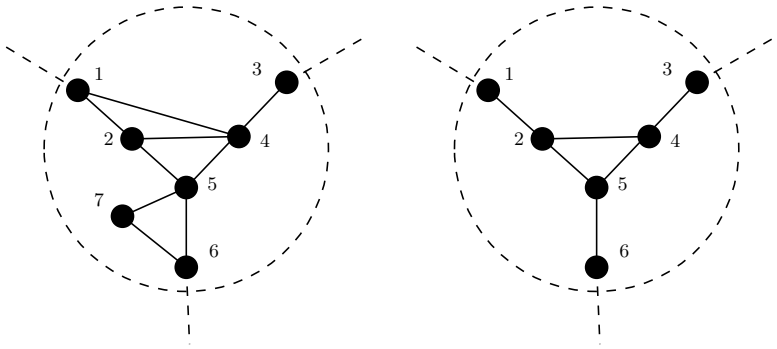

Figure 7 The two pictures represent some set V_i in a H-model. The dashed edges represent connections to other vertices of the model. In the first picture, the graph cannot be H-minimal, indeed we can remove the vertices 7 and 2, and still have a proper model. In the second picture, no vertex can be removed without disconnecting the subgraph induced by V_i.

We now describe what the V_i's subgraphs precisely look like in a H-minimal graph. Let T be a graph, and $S_1, \dots, S_r$ be some prescribed subsets of vertices of T. A *Steiner tree* of T with respect to the S_i's is a tree in T containing at least one element of each S_i. We say that T is an *almost tree* for the S_i's if any Steiner tree with respect to the S_i's contains all the vertices of T. Now, given a model $V_1, \dots, V_{|H|}$ of H, and $v_i \in H$, the prescribed sets we are

going to consider for V_i are the subsets $S_j \subseteq V_i$ containing all the vertices connected to V_j, for every j such that v_iv_j is an edge of H. A Steiner tree of V_i for the model $V_1, \dots, V_{|H|}$ of H is a Steiner tree containing at least one vertex of each prescribed set. When the model is clear from context, we simply say a Steiner tree of V_i.

With these notions, let us describe some properties of H-minimal graphs:

▶ **Lemma 39.** *Let H be a graph and let G a H-minimal graph. For every H-model of G, each V_i is an almost tree.*

Proof. The proof is straightforward. If some V_i is not an almost tree, then we can select a subset V_i' of V_i which is an almost tree. When we consider the subsets where all the V_j's are the same but V_i which is replaced V_i', we still have a model of H, and it does not contain all the vertices, a contradiction with the fact that G is H-minimal. ◀

It follows that we can characterize the form of the V_i's such that h_i has small degree in H.

▶ **Corollary 40.** *Let H be a graph, and G be a H-minimal graph. There exists a H-model of G such that:*

1. *If the degree of h_i in H is one, then V_i is reduced to a single vertex.*
2. *If the degree of h_i in H is two, then V_i is reduced to a single path P and, if h_j, h_k are the two neighbors of h_i then exactly one endpoint of P is connected to V_j, the other is connected to V_k, and all the other vertices of P are neither connected to V_j nor V_k.*
3. *If the degree of h_i in H is three, then the subgraph induced by V_i is of one of the three following types:*
   - *Type A: the subgraph is a triangle with a path attached to each of the three corners (which might be reduced to a single vertex) where the other endpoint of the path is attached to a V_j, and no other vertex is attached to V_j.*
   - *Type B: the subgraph is an induced subdivided star where only the last vertex of each branch is connected to a set V_j, and in that case it is connected to exactly one V_j.*
   - *Type C. the subgraph is a path, and there exists j, k such that the only connections with V_j and V_k are on the endpoints of the path. Any connection is possible for the vertices of the path with the last set V_ℓ.*

Proof. 1. If the degree of h_i is one, then a Steiner tree only needs the vertex that is connected to the rest of the model, so V_i has only one vertex.

2. If the degree is two, then any Steiner tree contains a path between a vertex in V_j and a vertex in V_k, and the shortest such paths is an induced path, thus the subgraph induced by V_i must be an induced path, with only the endpoints connected to the rest of the graph.

3. For degree 3, there are several cases.
   - If V_i contains a cycle, then by minimality the removal of any vertex on this cycle disconnects V_i from another branch. It follows that V_i has type A.
   - If V_i does not have a cycle, it has at most three leaves. If it has exactly three leaves then it has type B.
   - Otherwise, V_i is a path, and by the degree-2 case, only the endpoints can connect to some sets V_j and V_k, but the connections to the third set V_ℓ are not controlled. This is type C. ◀

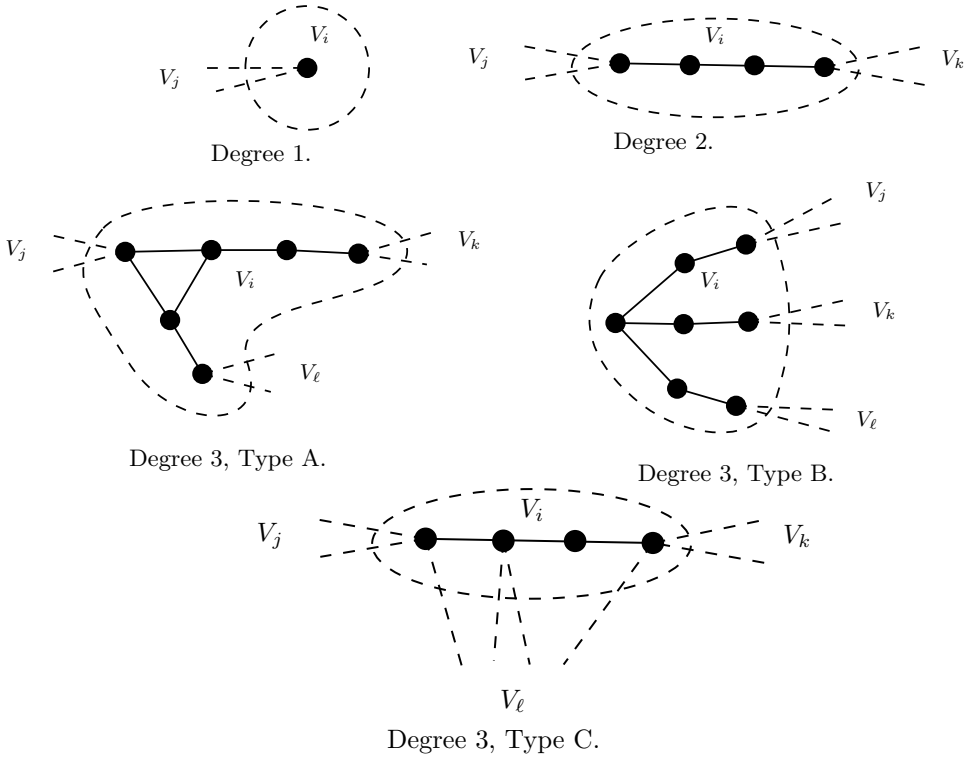

Figure 8 The types of the V_i's in Corollary 40

9.2 H with an isolated vertex and extension

▶ **Theorem 41.** *Let H be a graph on 5 vertices containing an isolated vertex. We can certify H-minor-free graphs with certificates of size $O(\log n)$.*

The rest of this section is devoted to the proof of Theorem 41.

Let H' be the graph H where an isolated vertex has been removed. A H-minor-free graph is either a H'-minor-free graph, or it is a graph G such that all the models of H' contain all the vertices of G. Since H'-minor-free graphs can be certified within $O(\log n)$ bits by Theorem 2, we can assume that G is a connected H'-minimal graph.

The core of the proof relies on the following lemma.

▶ **Lemma 42.** *For every graph H' on four vertices, any H'-minimal graph is in one of the following categories:*

1. *subdivided copies of H', that are graphs obtained from H' by replacing each edge by some path,*
2. *graphs of size 4,*
3. *induced cycles,*
4. *induced cycle plus a vertex,*
5. *the graphs of the type of Figure 9,*
6. *graphs with at most six vertices of degree larger than 2,*

Before proving the lemma, let us use it to conclude the proof of Theorem 41.

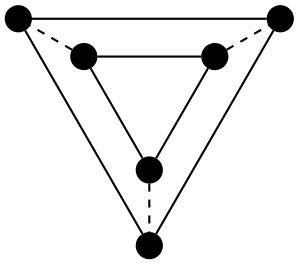

Figure 9 This drawing represents a class of graphs built by taking two vertex-disjoint triangles, and linking pairs of corners of the triangles by vertex-disjoint paths.

Proof of Theorem 41. Observe that all the classes of Lemma 42 can be certified with certificates of size $O(\log n)$, and in these classes it is easy to certify H'-minimality. This is because in all these classes there is a constant number of special vertices: the vertices before the subdivision for Item 1, the additional vertex for Item 4, the corners of the triangles for Item 5, the vertices of degree larger than 2 in Item 6, and none for Items 2 and 3. The vertices that are not special have degree 2. Certifying these classes boils down to having spanning trees pointing to the special vertices, and having a certification of every path of non-special vertices, to transfer the knowledge of the endpoints from one side of the paths to the other. Then basic consistency checks verify the certification. Because the structure is so constrained, it is easy also to check whether the graph is H'-minimal. ◀

Since the graph G is connected, the same proof holds for a graph H' plus a single vertex of degree one as long as all the vertices of H' are equivalent. A graph is *vertex transitive* if for every pair of vertices (u, v), there exists an automorphism of H mapping u to v. We thus get the following.

▶ **Lemma 43.** *Let H be a graph on 5 vertices obtained by adding a pending edge to a vertex transitive graph. Then H-minor-free graph can be certified with $O(\log n)$ bits.*

We end this section with the proof of Lemma 42. The proof is quite technical, so we split it into several claims. First let $v_1, \dots, v_4$ be the vertices of H', fix an H'-minimal graph G and denote by $V_1, \dots, V_4$ a model of H' in G.

▷ Claim 44. If H' is acyclic, then G has size at most 4 (Item 2).

Proof. If H' is acyclic, then it is either a star with 3 leaves or a disjoint union of paths. Using Corollary 40 in the latter case, observe that if G contains H' as a minor, it also contains H' as a subgraph. In particular, if G has more than 4 vertices, then G is not H'-minimal, since one can remove a vertex while keeping H' as a subgraph. ◀

▷ Claim 45. If H' has at most one degree-3 vertex, then G either has size 4, or is an induced cycle plus at most one vertex (Items 2, 3 and 4).

Proof. By the previous claim, we may assume that H' contains a cycle, which is C_3 or C_4 due to the size of H'.

If $H' = C_4$, it means that G contains a cycle of size at least 4 and that every such cycle contains all the vertices of G. In other words, either G has size exactly four or is an induced cycle.

If H' is a triangle plus a vertex, then we claim that G is an induced cycle plus a unique vertex. Indeed, since H' has at most one degree 3 vertex, the vertex not in the triangle has

degree at most one. Thus, for any cycle C of G, the cycle plus any vertex incident to C is a H'-minor. Since G is H'-minimal, the graph G is an induced cycle plus a vertex. ◀

Using the three claims above, it only remains to handle the cases where H is K_4 or K_4 minus an edge (namely, a diamond). In both cases, assume that V_3, V_4 (at least) are associated to degree 3 vertices of H'. If both V_3 and V_4 have type A or B, there is a unique vertex $x \in V_3$ incident to a vertex y in V_4. If we add y to V_3, and remove it from V_4, then the size of V_4 is decreasing, and the V_i's still form a model of H'. We can repeat this operation until V_4 does not have type A or B. Therefore, we can assume from now on that V_3 or V_4 has type C.

▷ Claim 46. If H' is a diamond, then $G = K_4$ or G is a subdivided diamond (Items 2 and 1).

Proof. In H', v_1 and v_2 have degree 2, and by Corollary 40, the subsets V_1 and V_2 are paths. Moreover, if there is an edge between them, and any of V_1, V_2, V_3 or V_4 has two vertices, we could remove one of these vertices, and still have a model of H'. Therefore, each V_i is reduced to a single vertex (otherwise G is not H'-minimal) hence G is K_4.

We may thus assume that there is no edge between V_1 and V_2. If V_3 or V_4 has type A, then $G \setminus V_1$ contains a diamond as a minor, a contradiction since G is H'-minimal. Moreover, as we already observed, at least one of V_3, V_4, say V_3, has type C. We now have two cases: V_4 has either type B or C.

Assume that V_4 has type B, and let $u \in V_4$ be the vertex of V_4 adjacent to V_3. If u sees two vertices of V_3, then replacing (V_3, V_4) by $(V_3 \cup \{u\}, V_4 \setminus u)$ gives a model where $V_3 \cup \{u\}$ has type A, a contradiction. Therefore, u sees a unique vertex of V_3 and G is a subdivided diamond.

Otherwise, V_4 has type C, hence V_3 and V_4 induce two paths (with maybe edges between them). There cannot be two edges between V_3 and V_4, otherwise, $G \setminus V_1$ contains a diamond-minor, a contradiction. Therefore, there is only one edge between V_3 and V_4 and G is again a subdivision of a diamond. ◀

We now assume that $H' = K_4$. Let us first prove the following claim:

▷ Claim 47. If G contains two vertex-disjoint cycles C_1, C_2 such that $G \setminus C_1$ is connected and with three pairwise non-incident edges between C_1 and $G \setminus C_1$, then G is the graph depicted on Figure 9.

Proof. Let a_1b_1, a_2b_2, a_3b_3 be the edges from the statement, with $a_i \in C_1$.

Assume first that C_2 is not a triangle. Then we can remove a vertex of $G \setminus C_1$ in such a way it remains connected and still contains the b_i's. This gives a K_4-model, a contradiction. Hence, we assume that C_2 is a triangle.

We say that $u \in C_2$ has a private path to one of the a_i's if it has such a path that avoids $C_2 \setminus \{u\}$. If some vertex $u \in C_2$ has no such path, then $(G \setminus C_1) \setminus \{u\}$ is connected, hence $G \setminus u$ contains a K_4 minor, a contradiction. Moreover, if two vertices $u, v \in C_2$ have a private path to the same a_i, then we get a K_4 minor avoiding some other a_j. Therefore, each vertex of C_2 is associated with a unique a_i by considering private paths. Observe that there is exactly one path for each of the three choice of endpoints (since if there were two paths, one could remove a vertex which lies in one path but not in the other and get a K_4 minor).

It remains to show that C_1 is a triangle. To this end, observe that the structure we found on $G \setminus C_2$ ensures that the hypotheses of the statement are still met when exchanging C_1 with C_2, and the first argument of the proof shows that C_1 is a triangle. ◀

Recall that we can assume that V_3 or V_4 has type C. Applying the same argument to each pair V_i, V_j ensures that all but at most one set V_i (say V_4) are of type C. We now do a case analysis on the type of V_4.

▷ Claim 48. If V_4 has type A, then G is a graph from Figure 9 (Item 5).

Proof. Since V_4 has type A, it contains a triangle C. Moreover, $G \setminus C$ contains a cycle since it contains V_1, V_2, V_3 which are pairwise connected. We then conclude using Claim 47. ◀

▷ Claim 49. If V_4 has type B, then G is either a subdivided K_4 or a graph from Figure 9 (Items 1 and 5).

Proof. If V_4 has type B, then it is a subdivided star with three branches. Let x be the vertex of V_4 of degree three and a_1, a_2, a_3 be the endpoints of the subdivided star rooted in x (note that the a_i's are indeed distinct from x). Without loss of generality, each a_i is connected to V_i (and not to some other V_j since otherwise $G \setminus a_j$ contains a K_4-minor). If some a_i is connected to at least two vertices of V_i, then $G \setminus (\{a_i\} \cup V_i)$ contains a cycle as well as $V_i \cup \{a_i\}$ with the conditions of Claim 47. So the G is the graph of Figure 9. Therefore, each a_i is connected to exactly one vertex of V_i and G is a subdivided K_4. ◀

▷ Claim 50. If V_4 has type C, then G is a graph from Figure 9, (a subgraph of) a wheel, a subdivided K_4, or has at most five vertices of degree more than 2 (Items 5, 4, 1 and 6).

Proof. If V_4 has type C, then all four sets have type C. We may assume that V_1 is maximal, that is there is no model of K_4 where one of the sets strictly contains V_1. In particular, any addition of a neighbor of a vertex of V_j to V_1 does not keep a model.

Now we claim that $V_2 \cup V_3 \cup V_4$ is a cycle C. Indeed, it must contain a cycle C, since V_2, V_3, V_4 is a model of the triangle. If w is not in C, either it is adjacent to V_1 and then can be added to V_1 (a contradiction) or it is not, and then $G \setminus w$ still contains C and V_1, hence has a model of K_4 so G is not H'-minimal.

Let C be the cycle containing all the vertices of V_2, V_3, V_4 and X_1, X_2, X_3 the neighbors of V_1 in respectively V_2, V_3, V_4.

Case 1: C is not induced.

Any chord of the cycle separates the cycle into two sides. If there is a chord e of C that leaves one side of the cycle with at least one element of each of X_1, X_2, X_3, then we can remove a vertex on the opposite side of the cycle and still have a K_4-minor, a contradiction with the H-minimality of G. So, without loss of generality, e separates X_1 on one side and at least one element from X_2, X_3 on the other side. Let P, P' be the two parts of C separated by e where P' only contains X_1. In this case, we can apply Claim 47 with the cycle $e + P'$, and a cycle using V_1, a part of P and edges between V_1 and X_2, X_3. Hence, this case again boils down to the graphs of Figure 9.

Case 2: C is induced.

If V_1 is reduced to a single vertex, then the graph G is a wheel. So we can assume that V_1 has at least two vertices. And it is a path since it has type C and both endpoints of the path have neighbors in C (otherwise the model is not minimal). Since we have a K_4-model, we need the whole set V_1 to have at least three different neighbors on C.

First, note that every vertex of V_1 has at most 2 neighbors on C (otherwise, we have a $K_4 + K_1$ model since V_1 contains at least two vertices). More generally, if a subpath of V_1 has at least three neighbors on C, we have a contradiction. Consider an endpoint v of V_1. If all its neighbors in C have another neighbor in V_1, then the subpath $V_1 \setminus v$ has three

neighbors in C. Therefore, we can assume that both endpoints of V_1 have a private neighbor in C (that is a neighbor in C with no other neighbor in V_1). Similarly, V_1 is adjacent to at most 4 vertices in C (otherwise V_1 minus one endpoint has three neighbors in C).

Case 2a: V_1 has four neighbors in C.

We claim that only endpoints of V_1 have neighbors in C and each endpoint has exactly two neighbors. Let us denote by $v_1, \dots, v_\ell$ the vertices of V_1 with neighbors in C (in that order in the path V_1). Recall that v_1 and v_ℓ both have a private neighbor in C. For each $j \neq 1, \ell$, we have $N(v_j) \subset N(v_1)$ (otherwise $|N(V_1 \setminus v_\ell) \cap C| \geq 3$), and similarly $N(v_j) \subset N(v_\ell)$, so $N(V_1) \subset N(v_1) \cup N(v_\ell)$. If $\ell > 2$, then $N(v_1) \cap N(v_\ell) \neq \varnothing$ so $|N(C)| \leq |N(v_1) \cap N(v_\ell)| \leq 3$, which is impossible.

So $\ell = 2$, and v_1, v_2 and their (at most four) neighbors in C are the only vertices of degree more than 2 in G, which concludes.

Case 2b: V_1 has 3 neighbors in C.

Let $v_1, \dots, v_\ell$ be the vertices of V_1 with neighbors in C. Since v_1 and v_ℓ have private neighbors on C, all the other vertices are adjacent to the same vertex w of C. Let us denote by a and b respectively the private neighbors of v_1 and v_ℓ. Note that $v_\ell V_1 v_1 a P_{ab} b v_\ell$ is an induced cycle C' (where P_{ab} is the subpath of C from a to b avoiding w). If the only vertex outside of C' is w, then G is a subgraph of a wheel. If w sees only one v_i, then the graph is a subdivided K_4. Otherwise, there are at least 4 paths starting from w to C' and one of them is subdivided. The removal of a vertex in a subdivided path still leaves a graph with a K_4 minor, a contradiction. ◀

10 Lower bounds

In this section, we show logarithmic lower bounds for H-minor-freeness for every 2-connected graph H. These results generalize the lower bounds of [28] for K_k and $K_{p,q}$. Our technique is a simple reduction from the certification of paths, via a local simulation. In contrast, the proofs of [27] were ad-hoc adaptations of the constructions of [38] and [26], with explicit counting arguments. Moreover, our lower bounds apply in the stronger model of locally checkable proofs, where the verifier can look at a constant distance, see [38].

▶ **Theorem 51.** *For every 2-connected graph H with at least 3 vertices, certifying H-minor-freeness requires $\Omega(\log n)$ bits.*

Let us start by proving a couple of lemmas. Let H be a 2-connected graph, and let $e = uv$ be an arbitrary edge of H. Let H^- be the graph $H \setminus e$. Note that H^- is connected. We are going to consider copies of H^-, that we index as H_i^-'s, and where the copies of the vertices u and v will be called u_i and v_i. Let $\mathcal{P}$ be the class of all the graphs that can be made by taking some k copies of H^-, and by identifying for every $i \in [1, k-1]$, v_i with u_{i+1}. In other words, $\mathcal{P}$ is the set of paths, where every edge is a copy of H^-. The class $\mathcal{C}$ is the same as $\mathcal{P}$ except that we close the paths into cycles, that is, we identify v_k with u_1.

▶ **Lemma 52.** *The graphs of $\mathcal{P}$ are all H-minor-free, and the graphs of $\mathcal{C}$ all contain H as a minor.*

Proof. Let G be a graph of $\mathcal{P}$. Note that every vertex v_i (identified with u_{i+1}) for $i \in \{1, ..., k-1\}$, is a cut vertex of G. Therefore, since H is 2-connected, a model of H can only appear between two such vertices. By construction this cannot happen, as the graphs between the cut vertices are all H^-. Thus G is H-minor-free.

Now let G be a graph of $\mathcal{C}$. We claim that G contains H as a minor. Consider the following model of H. Any H_i^- is a model of H except for the edge uv. Since we have made a cycle of H_i^-'s, there is a path between v_i and u_i outside H_i^-, and this path finishes the model of H. ◀

▶ **Lemma 53.** *Let H be a 2-connected graph. If there is a certification with $O(f(n))$ bits for H-minor-free graphs, then there is a $O(f(n))$ certification for paths.*

Proof. Suppose there exists a certification with $O(f(n))$ bits for H-minor-free graphs. The certification of paths boils down to differentiate between paths and cycles, since the vertices can locally check that they have degree 2. Consider the following certification of paths. The idea is that the vertices of the path (or cycle) will simulate the computation they would do if instead of being linked by edges, they were linked by copies of H^-. The prover will give to every vertex the certificates of H-minor-freeness for these simulated graphs, that is, for every vertex the certificates of the two copies of H^- adjacent to it in the simulated graph. Every vertex will check with its neighbor in the real graph that they have been given the same certificates for these virtual H^-. Then every vertex will run the verification algorithm for H-minor-freeness in the simulated graph.

By construction, the simulated graph is either in $\mathcal{P}$ or in $\mathcal{C}$. Thus, if the verification algorithm accepts, that is, if the simulated graph is H-minor-free, then the graph is in $\mathcal{P}$, and then the real graph is a path. If the verification algorithm rejects, that is if the simulated graph is not H-minor-free then the graph is in $\mathcal{C}$, and then the real graph is a cycle. In other words we have designed a local certification for paths, with certificates of size $O(f(n))$. ◀

Proof of Theorem 51. Now Theorem 51 follows from the fact that paths cannot be certified with $o(\log n)$ bits [38, 42]. Note that the proof applies in the locally checkable proof setting, as soon as the number of copies of H^- is large enough, since the lower bound for paths also applies to locally checkable proofs. ◀

11 Discussion

Milestones to go further

In this paper, we develop several tools and use them to show that some minor closed graph classes can be certified with $O(\log n)$ bits. One can probably use the tools we developed to certify new classes, we simply wanted to illustrate the interest of these tools. Let us now discuss the tools that are missing in order to tackle the general question on H-minor-freeness and which steps can be interesting to tackle it.

First, as we explained in Section 9, certification of H-minor-free classes seems easier when H is sparse. One first question that might be interested to look at is the following:

▶ **Question 54.** *Let T be a tree. Can T-minor-free graphs be certified with $O(\log n)$ bits?*

The answer to this question for small graphs H (up to 5 vertices) is not very interesting, since the number of vertices of degree at least 3 is bounded (and then the whole structure of the graph is "simple"). Even if it remains simple for any H, there is no trivial argument allowing us to certify these vertices with $O(\log n)$ bits. In the light of the recent results that establish that $O(\log n)$ bits is doable for paths minors [29], and $O(\log^2 n)$ is doable for planar minors [32], Question 54 seems to be the simplest open question.

A natural approach to tackle Conjecture 1 would consist in an induction on the size of the excluded minor H. Indeed, knowing how to certify $(H \setminus x)$-minor-freeness for any

possible x may help to certify H-minor-freeness. The basic idea would consist in separating two cases. 1) When H is not heavily connected where we can heavily use the fact that $(H \setminus x)$-minor-freeness can be certified. And 2) when H is heavily connected, try to use a more general argument. A first step toward step 1) would consist in proving that if H-minor-freeness can be certified then so is $H + K_1$-minor-freeness[3]. We proved it for five vertices in Theorem 41, but the proof heavily uses the structure of the graphs on four vertices. One can then naturally ask the following general question:

▶ **Question 55.** *Let H be a graph. Can $(H + K_1)$-minor-free graphs be certified with $O(\log n)$ bits when H can be certified with $O(\log n)$ bits?*

As in the proof of Theorem 41, we know that we can assume that G is H-minimal. Even if most of the techniques for Lemma 41 are specific, Corollary 40 gives some (basics) general properties of H-minimal graphs which might be useful to tackle this question.

In structural graph theory, a particular class of H-minimal graphs received a considerable attention which are minimally non-planar graphs, in order words, graphs G that are minimal and that contains either a K_5 or a $K_{3,3}$ as a minor. It might be interesting to determine if minimally non-planar graphs can be certified with $O(\log n)$ bits.

Note that if we can answer positively Question 55 positively, the second step would consist in proving the conjecture when we add to H a vertex attached to a single vertex of H. Proving this case would, in particular, imply a positive answer to Question 54.

If we want to consider dense graphs, the questions seem to become even harder. In particular, one of the first main complicated H-minor class to deal with is probably the class of K_5-minor-free graphs. There are several reasons for that. First, it is the smallest 4-connected graph and the hardness to certify seem to be highly related to the connectivity of the graph that is forbidden as a minor. The second reason is that it is the smallest graph for which H-minor-free graphs is a super class of planar graphs. In other words, we cannot take advantage of the "planarity" of the graph (formally or informally) to certify the graph class. We then ask the following question:

▶ **Question 56.** *Can K_5-minor-free graphs be certified with $O(\log n)$ bits?*

Wagner proved in [53] that a graph is K_5-minor-free if and only if it can be built from planar graphs and from a special graph V_8 by repeated clique sums. A *clique sum* consists in taking two graphs of the class and gluing them on a clique and then (potentially) remove edges of that clique. While it should have been easy to certify this sum if we keep the edges of the clique, the fact that they might disappear makes the work much more complicated for certification.

More generally, many decompositions are using the fact that we replace a subgraph by a smaller structure (a single vertex or an edge for instance) only connected to the initial neighbors of that structure in the graph. Certifying such structures is a challenging question whose positive answer can probably permit to break several of the current hardest cases.

Obstacles towards lower bounds

There are also several obstacles preventing us to prove extra-logarithmic lower bounds for the certificate size of H-minor-free graphs. Basically, the only techniques we know consist in (explicit or implicit) reductions to communication complexity. In particular [38] and [9]

[3] $H + K_1$ is the graph H plus an isolated vertex.

designed lower bounds for respectively non-3-colorable graphs and bounded diameter graphs as reductions from the disjointness problem in non-deterministic communication complexity.

Let us remind what these reductions look like. In such a reduction, one considers a family of graphs with two vertex sets A and B, with few edges in between. These graphs are defined in such a way that the input of Alice for the disjointness problem can be encoded in the edges of A and the input of Bob in the edges of B. Then, given a certification scheme, Alice and Bob can basically simulate the verification algorithm, and deduce an answer for the disjointness problem. If a certification with small labels existed for the property at hand, then the communication protocol would contradict known lower bounds which proves a lower bound for certification.

The difficulty of using this proof for H-minor-free graphs comes from the fact that it is difficult to control where a minor can appear, that is, to control the models of H. For example, it is difficult to control that if H appears in the graph, then the vertices V_i associated with some vertex i of H are on Alice's side. As a comparison, for proving properties on the diameter, [9] used a construction where all the longest paths in the graph had to start from Alice side and finish in Bob side, but such a property seems difficult to obtain for minors.

Connectivity questions

A large part of the paper is devoted to certify connectivity and related notions that are of independent importance, for instance to certify the robustness of a network. For these, we do not have lower bounds, and leave the following question open.

▶ **Question 57.** *Does the certification of k-connectivity require $\Omega(\log n)$ bits?*

For this question it is tempting to try a construction close to the one we have used for H-minor-free graphs. For example, one could think that the vertices of the path/cycle could simulate the k-th power of the graph which is k-connected if and only if the graph is a cycle. But this does not work: we want the *yes*-instances for the property (*e.g.* the k-connected graphs) to be in mapped to *yes*-instances for acyclicity (*e.g.* paths), and not with the *no*-instances, which are the cycles.

An interesting open problem about k-connectivity also is on the positive side:

▶ **Question 58.** *Can k-connectivity be certified with $O(\log n)$ bits for any $k \geq 4$?*

Beyond the question of certifying the connectivity itself, we would like to be able to decompose graphs based on k-connected components, like what we did with the block-cut tree for 2-connectivity. Such decomposition are more complicated and less studied than block-cut trees, but for 3-connectivity such a tool is SPQR trees [4]. Unfortunately, similarly to the clique sum operation we mentioned earlier, some steps of the SPQR tree construction are based on edges that can be removed in later steps, making it hard to certify this structure.

Acknowledgments

We thank Jens M. Schmidt for pointing out a mistake in the characterization of 3-connexity we used in an earlier version of this paper. We thank Cyril Gavoille for discussions on adjacency and distance labelings. We also thank the reviewers of the conference and journal versions for their comments and improvement suggestions.

References

1 Ittai Abraham and Cyril Gavoille. Object location using path separators. In Eric Ruppert and Dahlia Malkhi, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006*, pages 188–197. ACM, 2006. doi:10.1145/1146381.1146411.

2 Yehuda Afek, Shay Kutten, and Moti Yung. Memory-efficient self stabilizing protocols for general networks. In *Distributed Algorithms, 4th International Workshop, WDAG '90*, volume 486, pages 15–28, 1990. doi:10.1007/3-540-54099-7_2.

3 Kenneth Appel, Wolfgang Haken, et al. Every planar map is four colorable. *Bulletin of the American mathematical Society*, 82(5):711–712, 1976.

4 Giuseppe Di Battista and Roberto Tamassia. Incremental planarity testing (extended abstract). In *30th Annual Symposium on Foundations of Computer Science,*, pages 436–441, 1989. doi:10.1109/SFCS.1989.63515.

5 Nicolas Bousquet, Linda Cook, Laurent Feuilloley, Théo Pierron, and Sébastien Zeitoun. Local certification of forbidden subgraphs. *CoRR*, abs/2402.12148, 2024. doi:10.48550/ARXIV.2402.12148.

6 Zvika Brakerski and Boaz Patt-Shamir. Distributed discovery of large near-cliques. *Distributed Comput.*, 24(2):79–89, 2011. doi:10.1007/s00446-011-0132-x.

7 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. *Distributed Comput.*, 32(1):41–57, 2019. doi:10.1007/s00446-018-0324-8.

8 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 2020*, volume 179 of *LIPIcs*, pages 33:1–33:17, 2020. doi:10.4230/LIPIcs.DISC.2020.33.

9 Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof-labeling schemes. *Theor. Comput. Sci.*, 811:112–124, 2020. doi:10.1016/j.tcs.2018.08.020.

10 Yi-Jun Chang. Efficient distributed decomposition and routing algorithms in minor-free networks and their applications. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 55–66, 2023.

11 Yi-Jun Chang and Hsin-Hao Su. Narrowing the local-congest gaps in sparse networks via expander decompositions. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, pages 301–312, 2022.

12 Joseph Cheriyan and S. N. Maheshwari. Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs. *J. Algorithms*, 9(4):507–537, 1988. doi:10.1016/0196-6774(88)90015-6.

13 Markus Chimani, Martina Juhnke-Kubitzke, Alexander Nover, and Tim Römer. Cut polytopes of minor-free graphs. *arXiv preprint arXiv:1903.01817*, 2019.

14 Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of mathematics*, pages 51–229, 2006.

15 Oscar Defrain, Louis Esperet, Aurélie Lagoutte, Pat Morin, and Jean-Florent Raymond. Local certification of geometric graph classes. *CoRR*, abs/2311.16953, 2023. doi:10.48550/ARXIV.2311.16953.

16 Shlomi Dolev. *Self-Stabilization*. MIT Press, 2000. ISBN 0-262-04178-2. URL `http://www.cs.bgu.ac.il/%7Edolev/book/book.html`.

17 R. J. Duffin. Topology of series-parallel networks. *J. Math. Anal. Appl.*, 10:303–318, 1965. ISSN 0022-247X. doi:10.1016/0022-247X(65)90125-3.

18 Gábor Elek. Planarity can be verified by an approximate proof labeling scheme in constant-time. *J. Comb. Theory, Ser. A*, 191:105643, 2022. doi:10.1016/j.jcta.2022.105643.

19 Mark N. Ellingham, Emily A. Marshall, Kenta Ozeki, and Shoichi Tsuchiya. A characterization of $K_{2,\,4}$-minor-free graphs. *SIAM J. Discret. Math.*, 30(2):955–975, 2016. doi:10.1137/140986517.

20 David Eppstein. Parallel recognition of series-parallel graphs. *Inf. Comput.*, 98(1):41–55, 1992. doi:10.1016/0890-5401(92)90041-D.

21 Louis Esperet and Benjamin Lévêque. Local certification of graphs on surfaces. *Theor. Comput. Sci.*, 909:68–75, 2022. doi:10.1016/j.tcs.2022.01.023.

22 Louis Esperet and Sergey Norin. Testability and local certification of monotone properties in minor-closed classes. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022*, volume 229 of *LIPIcs*, pages 58:1–58:15, 2022. doi:10.4230/LIPIcs.ICALP.2022.58.

23 Laurent Feuilloley. Bibliography of distributed approximation on structurally sparse graph classes. *CoRR*, abs/2001.08510, 2020.

24 Laurent Feuilloley. Introduction to local certification. *Discrete Mathematics & Theoretical Computer Science*, vol. 23, no. 3, 2021. doi:10.46298/dmtcs.6280.

25 Laurent Feuilloley and Pierre Fraigniaud. Survey of distributed decision. *Bulletin of the EATCS*, 119, 2016. url: bulletin.eatcs.org link, arXiv: 1606.04434.

26 Laurent Feuilloley and Juho Hirvonen. Local verification of global proofs. In *32nd International Symposium on Distributed Computing, DISC 2018*, volume 121 of *LIPIcs*, pages 25:1–25:17, 2018. doi:10.4230/LIPIcs.DISC.2018.25.

27 Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Eric Rémila, and Ioan Todinca. Local certification of graphs with bounded genus. *CoRR*, abs/2007.08084, 2020.

28 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. doi:10.1007/s00453-021-00823-w.

29 Laurent Feuilloley, Nicolas Bousquet, and Théo Pierron. What can be certified compactly? compact local certification of MSO properties in tree-like graphs. In *PODC '22: ACM Symposium on Principles of Distributed Computing*, pages 131–140. ACM, 2022. doi:10.1145/3519270.3538416.

30 Paola Flocchini and Flaminia L. Luccio. Routing in series parallel networks. *Theory Comput. Syst.*, 36(2):137–157, 2003. doi:10.1007/s00224-002-1033-y.

31 Pierre Fraigniaud and Dennis Olivetti. Distributed detection of cycles. *ACM Trans. Parallel Comput.*, 6(3):12:1–12:20, 2019. doi:10.1145/3322811.

32 Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. A meta-theorem for distributed certification. In *Structural Information and Communication Complexity - 29th International Colloquium, SIROCCO 2022*, page To appear., 2022.

33 Pierre Fraigniaud, Frédéric Mazoit, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. Distributed certification for classes of dense graphs. In *37th International Symposium on Distributed Computing, DISC 2023*, volume 281 of *LIPIcs*, pages 20:1–20:17, 2023. doi:10.4230/LIPICS.DISC.2023.20.

34 Cyril Gavoille and Arnaud Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In *Algorithms - ESA 2007, 15th Annual European Symposium*, volume 4698, pages 582–593, 2007. doi:10.1007/978-3-540-75520-3_52.

35 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, MST, and min-cut. In *Proceedings of the Twenty-Seventh*

Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, pages 202–219. SIAM, 2016. doi:10.1137/1.9781611974331.ch16.

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

37 Mohsen Ghaffari and Bernhard Haeupler. Low-congestion shortcuts for graphs excluding dense minors. In *PODC '21: ACM Symposium on Principles of Distributed Computing*, pages 213–221. ACM, 2021. doi:10.1145/3465084.3467935.

38 Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory of Computing*, 12(19):1–33, 2016. doi:10.4086/toc.2016.v012a019.

39 Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Near-optimal low-congestion shortcuts on bounded parameter graphs. In *Distributed Computing - 30th International Symposium, DISC 2016*, volume 9888, pages 158–172. Springer, 2016. doi:10.1007/978-3-662-53426-7_12.

40 Bernhard Haeupler, Jason Li, and Goran Zuzic. Minor excluded network families admit fast distributed algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018*, pages 465–474, 2018.

41 Benjamin Jauregui, Pedro Montealegre, and Ivan Rapaport. Distributed interactive proofs for the recognition of some geometric intersection graph classes. In *Structural Information and Communication Complexity - 29th International Colloquium, SIROCCO 2022*, page To appear., 2022.

42 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010. doi:10.1007/s00446-010-0095-3.

43 Alexandr V Kostochka. The minimum hadwiger number for graphs with a given mean degree of vertices. *Metody Diskret. Analiz.*, (38):37–58, 1982.

44 Reut Levi, Moti Medina, and Dana Ron. Property testing of planarity in the congest model. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2018.

45 Lee F. Mondshein. *Combinatorial Ordering and the Geometric Embedding of Graphs*. PhD thesis, M.I.T. Lincoln Laboratory / Harvard University, 1971.

46 Pedro Montealegre, Diego Ramírez-Romero, and Ivan Rapaport. Compact distributed interactive proofs for the recognition of cographs and distance-hereditary graphs. In *Stabilization, Safety, and Security of Distributed Systems - 23rd International Symposium, SSS 2021*, volume 13046, pages 395–409, 2021. doi:10.1007/978-3-030-91081-5_26.

47 Moni Naor, Merav Parter, and Eylon Yogev. The power of distributed verifiers in interactive proofs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 1096–115. SIAM, 2020. doi:10.1137/1.9781611975994.67.

48 H. E. Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939. ISSN 00029890, 19300972.

49 Neil Robertson and Paul D Seymour. Graph minors—a survey. *Surveys in combinatorics*, 103:153–171, 1985.

50 Neil Robertson and Paul D Seymour. Graph minors. v. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986.

51 Jens M. Schmidt. Mondshein sequences (a.k.a. (2, 1)-orders). *SIAM J. Comput.*, 45(6): 1985–2003, 2016. doi:10.1137/15M1030030.

52 Andrew Thomason. An extremal function for contractions of graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 95, pages 261–265. Cambridge University Press, 1984.

1453 **53** K. Wagner. Uber eine eigenschaft der ebenen komplex. In *Math. Ann.*, volume 114, pages
1454 570–590, 1937.
1455 **54** Hassler Whitney. Non-separable and planar graphs. *Transactions of the American*
1456 *Mathematical Society*, 34:339–362, 1932. doi:10.1090/S0002-9947-1932-1501641-2.