Local Certification of Graph Decompositions and Applications to Minor-Free Classes

Nicolas Bousquet
Univ. Lyon, Université Lyon 1, LIRIS UMR CNRS 5205, F-69621, Lyon, France

Laurent Feuilloley
Univ. Lyon, Université Lyon 1, LIRIS UMR CNRS 5205, F-69621, Lyon, France

Théo Pierron
Univ. Lyon, Université Lyon 1, LIRIS UMR CNRS 5205, F-69621, Lyon, France

Abstract

Local certification consists in assigning labels to the nodes 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 a 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 → Distributed algorithms

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

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2021.22

Related Version *Full Version*: `https://arxiv.org/abs/2108.00059` [4]

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

Acknowledgements The authors thank the reviewers for the their comments, and Jens M. Schmidt for pointing out a mistake a previous version.

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 nodes 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 [26], bipartite graphs [23] or graphs of bounded diameter [8], 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 [17, 18, 13]. 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 [31]. Remember that a graph H is a minor of a graph G, is it 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.

▶ **Question 1** ([18, 14]). *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 [8], non-3-colorability [23], having a non-trivial automorphism [23]) are non-hereditary. That is, removing a node 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 node, as the class is stable by removal of edges and nodes. 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 [17, 13], or a canonical path to the root for trees [26]). 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 [2] or the Strong Perfect Graph Theorem [10].

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 [26, 23].
Diamond-minor-free	$\Theta(\log n)$	Corollary 20
K_4-minor-free	$\Theta(\log n)$	Corollary 20
$K_{2,3}$-minor-free	$\Theta(\log n)$	Corollary 20
$(K_{2,3}, K_4)$-minor-free (*i.e.* outerplanar)	$\Theta(\log n)$	Corollary 20
$K_{2,4}$-minor-free	$\Theta(\log n)$	See full version.

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

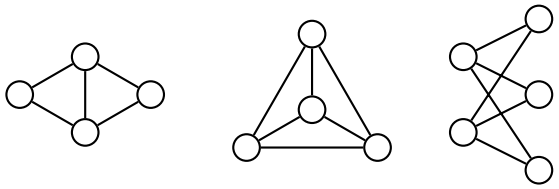

Figure 2 From left to right: the diamond, the clique on 4 vertices K_4, and the complete bipartite graph $K_{2,3}$.

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. This generalizes and simplifies the lower bounds of [17] 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 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.

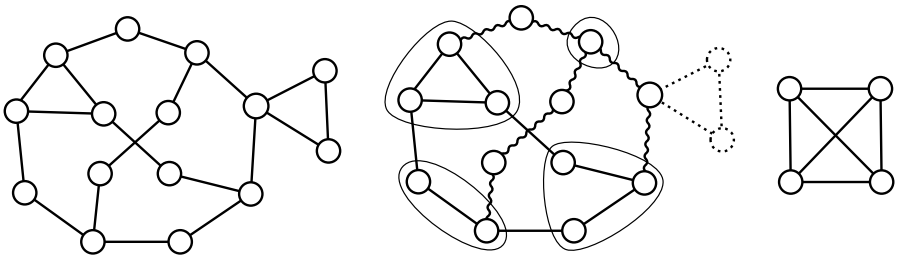

Figure 3 The graph on the left has a K_4 minor. Indeed, the bags of the second definition are depicted in the picture in the middle, and it is easy to find the six disjoint paths that link them. Alternatively, one can get a K_4 like the one of the right-most picture by contracting all the edges inside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

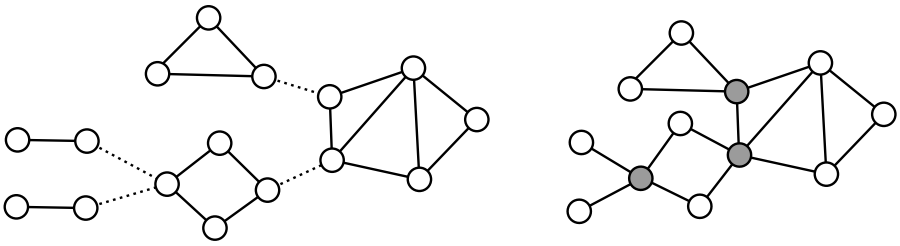

Figure 4 The five graphs with plain edges on the left picture are K_4-minor free. Organizing them into a tree by identifying the nodes linked by dotted edges makes a larger K_4-minor-free graph.

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 (gray vertices in Figure 4). Then the bags and paths cannot be vertex-disjoint as required, because at least two of them should use the vertex v.

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 show 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 nodes. 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 node 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 [17, 18]. 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 node 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 [26] and [23]. 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 different nodes of the current graph and link them by a path whose internal nodes are new nodes of the graph. 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-vertex-connected graphs.

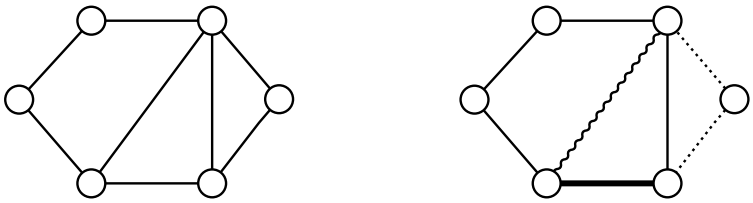

Figure 5 Illustration of an open ear decomposition. The graph on the left can be built with the ear decomposition described on the right. First, put the bold edge. Then add the path of plain edges. Finally, add the dotted path, and the wavy path, which is just one edge.

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 4.

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 [17] 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 5. For example, the 2-connected C_4-minor-free graphs are K_2 and K_3, and the 2-connected diamond-minor-free graphs are the induced cycles. A special case is $K_{2,4}$, that has a more complicated structure, requiring to consider 3-connected components, and some more complicated substructures. Due to space constraints, this section is deferred to the full version [4].

The full version also contains the study all the minors on at most 4 vertices, and 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. This 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 (2-connected) minor-free graph classes. When it comes to $\Omega(\log n)$ lower bounds in our model, there are basically two complementary techniques (called *cut-and-plug techniques* in [14]). Both techniques basically show that paths cannot be differentiated from cycles, if the certificates use $o(\log n)$ bits. First, in [23], 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 [19], the idea is to consider a path, to cut it into small pieces, and to show via Sterling formula, that there exists a shuffle of these pieces that can be closed into a cycle.

Previous lower bounds for minor-free graphs in [17] followed the same kind of strategies as [23] and [19], 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 first appeared under the name of *proof-labeling schemes* in [26], inspired by works on self-stabilizing algorithms (see [11] for a book on self-stabilization). It has then been generalized under the name of *locally checkable proofs* in [23], 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 [14] and [16] 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 [26]. Some graph properties have also been studied, for example symmetry in [23], or bounded diameter in [8]. Very recently, classes that are more central in graph theory have attracted attention. It was first proved in [30], 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 [17] 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 [18]. Later, [13] provided a simpler proof of both results via different techniques. It was also proved in [29] that cographs and distance-hereditary graphs have compact distributed interactive proofs.

Still in distributed computing, but outside local certification, the networks with some forbidden structures have attracted a lot of attention recently. 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 [7] 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 [6] after the seminal work of [5] (see [20] and the references therein). To our knowledge, no detection/testing algorithm or lower bounds have been designed for H-minor-freeness.

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 [15] for a bibliography of this area. There are also interesting works for exact problems in the CONGEST model, *e.g.* in planar graphs [21], graphs of bounded treewidth or genus [24] and minor-free graphs [25]. In particular the authors of [25] 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, [22] 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 $v_1, \ldots, v_n$ of the vertices such that, for every i, $N(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ 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 $v_0 = u, v_1, \ldots, v_\ell = v$ 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 nodes can only communicate with their neighbors, so no node can communicate with nodes 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 $V_1, \ldots, V_{|H|}$ 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 $V_1, \ldots, V_{|H|}$ 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* [26]. 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 node v a certificate $c(v)$. The *verifier* is a distributed algorithm, in which every node 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 nodes 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 node that rejects.

The size of the certificate of $\mathcal{C}$ is the largest size of a certificate assigned to a node 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 nodes 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 intensively as building blocks in the paper.

▶ **Lemma 4** ([26, 1]). *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 node, and then to give to every node as its certificate its distance to the root. The nodes 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 node 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 nodes 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 node is at most 2. Note that cycles do not need certificates to be verified: every node 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 are said to be* nested.

We are going to use the two following classic results as black boxes.

▶ **Lemma 6** ([17]). *Path-outerplanar graphs can be certified with $O(\log n)$-bit certificates.*

▶ **Lemma 7** ([26]). *Every graph class can be certified with $O(n^2)$ bits.*

The idea of the scheme is that the prover gives to every node v the map of the graph, *e.g.* as an adjacency matrix, along with the position of v in this map. Then every node 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.

Organization of the paper

The full version of the paper contains more material than this short version. First, we had to remove the majority of the proofs from the short version. As a consequence, some parts of the paper are mainly lists of lemmas, but the intuition provided in the introduction should be enough to follow the articulation of the reasoning. Also, as said before, several full sections of the paper appear only in the full version [4].

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

Transforming a node certification into an edge certification can always be done 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 node certification is also always possible, by giving a copy of the edge label to each of its endpoint. But this transformation can drastically increase the certificate size: if an edge certification uses $\Omega(f(n))$-bit labels, the associated node 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** ([18]). *Consider an edge certification of a graph class $\mathcal{C}$ where the edges are labeled with $f(n)$-bit certificates. If $\mathcal{C}$ is d-degenerate, then there exists a (node) certification with $d \cdot f(n)$-bit certificates.*

Note that H-minor free graphs have degeneracy $O(h\sqrt{\log h})$ where $h = |V(H)|$ [27, 33]. 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 node. 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 nodes.*

A *pointed graph* is a graph with one selected node. 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 node.*

The previous results can be easily iterated: one can always remove the labels of k nodes (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 nodes.*

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 node receives the certificate of at most one selected node).

4 Connectivity and connectivity decompositions

In this section, we study the certification of 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 exists 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.

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 nodes of the current graph and link them by a path whose internal nodes are new nodes 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 node. Let an *inner node* 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 node.

▶ **Theorem 12** ([35] reformulated). *A graph is 2-connected if and only if it has an open ear decomposition.*

As described in the introduction, we use this characterization to certify vertex and edge 2-connectivity.

▶ **Lemma 13.** *2-connected graphs can be certified with $O(\log n)$ bits.*

▶ **Corollary 14.** *2-edge-connected graphs can be certified with $O(\log n)$ bits.*

A more refined type of ear decomposition characterizes the 3-vertex-connected graphs, based on *Mondshein sequences*.

▶ **Definition 15** ([32, 28, 9])**.** *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 node 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 node has a neighbor that is created in a later ear.*

▶ **Theorem 16** ([9, 32])**.** *Let ru and rt be two edges of a graph G. The graph G is 3-vertex-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.*

We can translate this theorem into a compact certification.

▶ **Corollary 17.** *3-connectivity can be certified with $O(\log n)$ bits on vertices and $O(1)$ bits on edges.*

With the tools we previously introduced, we can now certify a well-known decomposition into parts of higher connectivity, called the *block-cut tree*. This allows to prove the following result. Due to space constraint, the needed definition and proof only appear in the full version [4].

▶ **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.*

5 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 can be found in the full version [4]. They will all 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.

A *nested ear decomposition* is an open ear decomposition that starts from a path, with two properties: (1) both ends of an ear have to be connected to the same ear, and (2) for every ear, the ears that are plugged onto it are nested. Eppstein proved in [12] that, for 2-connected graphs, being K_4-minor-free is equivalent to having a nested ear decomposition. Therefore, we get the following.

▶ **Theorem 18.** *2-connected K_4-minor-free graphs can be certified with $O(\log n)$-bit labels.*

We now extend the techniques to other small graphs, but before we prove a simple statement for the case of C_5.

▶ **Lemma 19.** *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.*

▶ **Corollary 20.** *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*[1]*, outerplanar graphs (that is $(K_{2,3}, K_4)$-minor-free graphs), $K_{2,3}$-minor-free and K_4-minor-free graphs.*

6 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 [17] 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 [17] were ad-hoc adaptations of the constructions of [23] and [19], 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.

▶ **Theorem 21.** *For every 2-connected graph H, 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 nodes 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 22.** *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 nodes. 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 23.** *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.*

Now Theorem 21 follows from the fact that paths cannot be certified with $o(\log n)$ bits [26, 23]. Note that this also 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.

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

7 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. While these tools probably permit certifying new classes, we simply wanted to illustrate their interest. 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 the section of the full version concerned with 5 vertices, 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 24.** *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 nodes with $O(\log n)$ bits.

A natural approach to tackle Conjecture 1 would consist in an induction on the size of H. Indeed, knowing how to certify $H \setminus x$ for any possible x may help to certify H. The basic idea would consist in separating two cases. 1) When H is not heavily connected where we can heavily use the fact that we can $H \setminus x$ 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[2]. We proved it for five vertices in the full version, but the proof heavily uses the structure of the graphs on four vertices. One can then naturally ask the following general question:

▶ **Question 25.** *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 the analogue theorem for 5 vertices in the full version, we know that we can assume that G is H-minimal. Even if most of the techniques we use are specific, some (basic) 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 25 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 24.

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-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:

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

▶ **Question 26.** *Can K_5-minor free graphs be certified with $O(\log n)$ bits?*

Wagner proved in [34] 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 super-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, 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 nodes V_i associated with some node i of H are on Alice's side. As a comparison, for proving properties on the diameter, [8] 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 27.** *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 nodes 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 28.** *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 [3]. 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.

References

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

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

3 Giuseppe Di Battista and Roberto Tamassia. Incremental planarity testing (extended abstract). In *FOCS 89*, pages 436–441, 1989. doi:10.1109/SFCS.1989.63515.

4 Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Local certification of graph decompositions and applications to minor-free classes. *CoRR*, abs/2108.00059, 2021. arXiv:2108.00059.

5 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.

6 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.

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

8 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.

9 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.

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

11 Shlomi Dolev. *Self-Stabilization*. MIT Press, 2000. URL: http://www.cs.bgu.ac.il/%7Edolev/book/book.html.

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

13 Louis Esperet and Benjamin Lévêque. Local certification of graphs on surfaces. *CoRR*, abs/2102.04133, 2021.

14 Laurent Feuilloley. Introduction to local certification. *CoRR*, abs/1910.12747, 2019.

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

16 Laurent Feuilloley and Pierre Fraigniaud. Survey of distributed decision. *Bulletin of the EATCS*, 119, 2016. URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/411/391, arXiv:1606.04434.

17 Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and Ioan Todinca. Compact distributed certification of planar graphs. In *PODC '20*, pages 319–328. ACM, 2020. doi:10.1145/3382734.3404505.

18 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.

19 Laurent Feuilloley and Juho Hirvonen. Local verification of global proofs. In *DISC 2018*, volume 121 of *LIPIcs*, pages 25:1–25:17, 2018. doi:10.4230/LIPIcs.DISC.2018.25.

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

21 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut. In *SODA 2016*, pages 202–219. SIAM, 2016. doi:10.1137/1.9781611974331.ch16.

22 Mohsen Ghaffari and Bernhard Haeupler. Low-congestion shortcuts for graphs excluding dense minors. In *PODC 2021*, page To appear., 2021.

23 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.

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

25 Bernhard Haeupler, Jason Li, and Goran Zuzic. Minor excluded network families admit fast distributed algorithms. In *PODC 2018*, pages 465–474, 2018.

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

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

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

29 Pedro Montealegre, Diego Ramírez-Romero, and Iván Rapaport. Compact distributed interactive proofs for the recognition of cographs and distance-hereditary graphs. *CoRR*, abs/2012.03185, 2020.

30 Moni Naor, Merav Parter, and Eylon Yogev. The power of distributed verifiers in interactive proofs. In *SODA 2020*, pages 1096–115. SIAM, 2020. doi:10.1137/1.9781611975994.67.

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

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

33 Andrew Thomason. An extremal function for contractions of graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 95(2), pages 261–265. Cambridge University Press, 1984.

34 K. Wagner. Uber eine eigenschaft der ebenen komplex. In *Math. Ann.*, volume 114, pages 570–590, 1937.

35 Hassler Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34:339–362, 1932. doi:10.1090/S0002-9947-1932-1501641-2.