Compact Distributed Certification of Planar Graphs

Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, Ioan Todinca

Compact Distributed Certification of Planar Graphs

Laurent Feuilloley*[1], Pierre Fraigniaud†[2], Ivan Rapaport‡[3], Éric Rémila§[4], Pedro Montealegre¶[5], and Ioan Todinca[6]

[1]Departamento de Ingeniería Industrial, Universidad de Chile, Chile
[2]IRIF, CNRS and Université de Paris, France
[3]DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Chile
[4]GATE Lyon St-Etienne (UMR 5824 CNRS), UJM St-Etienne, France
[5]Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibañez, Santiago, Chile.
[6]LIFO, Université d'Orléans and INSA Centre-Val de Loire, France

Abstract

Naor, Parter, and Yogev (SODA 2020) have recently demonstrated the existence of a *distributed interactive proof* for planarity (i.e., for certifying that a network is planar), using a sophisticated generic technique for constructing distributed IP protocols based on sequential IP protocols. The interactive proof for planarity is based on a distributed certification of the correct execution of any given sequential linear-time algorithm for planarity testing. It involves three interactions between the prover and the randomized distributed verifier (i.e., it is a dMAM protocol), and uses small certificates, on $O(\log n)$ bits in n-node networks. We show that a single interaction from the prover suffices, and randomization is unecessary, by providing an explicit description of a *proof-labeling scheme* for planarity, still using certificates on just $O(\log n)$ bits. We also show that there are no proof-labeling schemes — in fact, even no *locally checkable proofs* — for planarity using certificates on $o(\log n)$ bits.

*Additional support from MIPP and José Correa's Amazon Research Award

†Additional support from ANR project DESCARTES, and from INRIA project GANG

‡Additional support from CONICYT via PIA/Apoyo a Centros Científicos y Tecnológicos de Excelencia AFB 170001 and Fondecyt 1170021

§Additional support from IDEXLYON (project INDEPTH) within the Programme Investissements d'Avenir (ANR-16-IDEX-0005) and "Mathématiques de la Décision pour l'Ingénierie Physique et Sociale" (MODMAD)

¶Additional support from CONICYT via PAI + Convocatoria Nacional Subvención a la Incorporación en la Academia Año 2017 + PAI77170068 and FONDECYT 11190482

1 Introduction

Planar graphs are arguably among the most studied classes of graphs in the context of algorithm design and analysis. In particular, planar graphs enable the design of algorithms provably faster than for general graphs (see, e.g., [7] for a subquadratic algorithm for diameter in planar graphs, in contrast to the quadratic lower bound in [41]). In the context of distributed network computing too, planar graphs have been the source of many contributions. For instance, a distributed algorithm approximating a minimum dominating set (MDS) within a constant factor, in a constant number of rounds, has been designed for planar graphs [36]. In contrast, even a poly-logarithmic approximation of the MDS problem requires at least $\Omega(\sqrt{\log n/\log\log n})$ rounds in arbitrary n-node networks [35]. See Section 1.2 for further references to contributions on distributed algorithm design for planar graphs.

In this paper, we are concerned with *checking* whether a given network is planar. Indeed, we aim at avoiding the risk of performing distributed computations dedicated to planar graphs in networks that are not planar, which may lead to erroneous outputs, or even lack of termination. Note that it is sufficient that one node detects if the actual graph is not planar, as this node can then broadcast an alarm, or launch a recovery procedure (e.g., for overlay networks).

Efficient planarity tests have been designed in the *sequential* setting [31], but no such efficient algorithms are known in the *distributed* setting. In fact, it is known [22] that there are no *local* decision algorithms for planarity. That is, there are no algorithms in which every node of a network exchange information solely with nodes in its vicinity, and outputs accept or reject such that the network is planar if and only if all nodes accept.

Kuratowski's theorem states that a graph is planar if and only if it does not contain a subgraph that is a subdivision of the complete graph K_5, or the complete bipartite graph $K_{3,3}$. This characterization can easily be turned into a *proof-labeling scheme* [34] that a network is <u>not</u> planar, by certifying the presence of a subdivided K_5 or $K_{3,3}$ as a subgraph. However, it is not clear whether Kuratowski's theorem can be used to prove that a network <u>is</u> planar, in a distributed manner. Indeed, this would require to certify that no subdivided K_5 or $K_{3,3}$ are present in the network. Also, it is not clear whether using the coordinates of the nodes in a planar embedding would help, as checking whether two edges cross seems difficult if the extremities of these edges are embedded at far away positions in the plane. Similarly, a distributed proof based on checking the consistency of the faces resulting from a planar embedding of the graph appears uneasy, as one can construct embeddings of non planar graphs in which the faces look locally consistent.

Yet, Kuratowski's theorem can be used to show that planarity is at the second level of the local decision *hierarchy* with alternating quantifiers (see [19]). That is, for every assignment of $O(\log n)$-bit certificates to the nodes by a *disprover* aiming at convincing the nodes of a planar network that this network is not planar, a *prover* can assign other $O(\log n)$-bit certificates to the nodes for certifying that the distributed proof provided by the disprover is erroneous. The fact that planarity is at a low level of the local decision hierarchy is conceptually informative, but this fact does not provide a concrete distributed mechanism for certifying planarity.

A breakthrough has been recently achieved by Naor, Parter, and Yogev [38], who showed the existence of a *distributed interactive proof* for planarity. Such types of protocols are motivated by the possible access to services provided by a computationally powerful external entity, e.g., the cloud [33]. The interactive protocol for planarity is not explicit, but can be constructed automatically from any sequential linear-time algorithm for planarity testing. In fact, this protocol is just one illustration of a sophisticated generic *compiler* developed in [38] for constructing distributed IP protocols based on sequential IP protocols, which applies way beyond the case of planarity. Specifically, given an algorithm for planarity testing, the interactive protocol is roughly the following. The $O(n)$ steps corresponding to the execution of the algorithm on the actual network are distributed to the nodes, and the interactive protocol checks that these steps are consistent (e.g., the output of step i is the input of step $i+1$, etc.). The node handling the final step of the execution of the algorithm accepts or rejects according to the outcome of the algorithm. The steps of the execution (e.g., reading some memory location, or adding such and such registers) can be encoded on $O(\log n)$ bits, and it is shown that verifying the consistency of the $O(n)$ steps distributed over the n nodes can be achieved by a dMAM interactive protocol, using certificates on $O(\log n)$ bits. In such a protocol, the *non-trustable* prover (a.k.a. Merlin) provides each node with an $O(\log n)$-bit certificate. Then, at each node, the *honest* verifier (a.k.a. Arthur) running at that node challenges the prover with a random value. Finally the prover replies to each node by providing it with a second certificate, again on $O(\log n)$ bits. Once this is done, every node interacts with all its neighbors only once, and outputs accept or reject. The authors of [38] proved that, using their dMAM protocol, all nodes accept if and only if the network is planar.

The mechanism in [38] actually applies to all classes of graphs that can be recognized with a sequential polynomial-time algorithm. The resulting distributed interactive protocol is however efficient only for classes of sparse graphs, recognizable in quasi-linear time, among which the class of planar graphs plays a prominent role.

1.1 Our results

We show that several interactions between the prover and the verifier are not necessary for distributed certification of planarity, while the size of the certificates can be kept as small as in [38]. Indeed, we provide an explicit description of a *proof-labeling scheme* [34], still using certificates on just $O(\log n)$ bits. A proof-labeling scheme requires a single interaction between the prover and the distributed verifier, and no randomization is required. Specifically, the prover provides each node with an $O(\log n)$-bit certificate, and then the nodes can directly move on with the local verification stage, in which every node interacts with all its neighbors once, and outputs accept or reject such that all nodes accept if and only if the network is planar. Proof-labeling schemes can be implemented even in absence of services provided by external entities like the cloud. Actually, in many frameworks, including the one in this paper, the certificates can be computed in a distributed manner by the network itself during a pre-processing phase.

As mentioned before, it is not clear whether Kuratowski's theorem, or the use of coordinates can be applied to prove that a network is planar, in a distributed manner (Kuratowski's theorem can be used to certify that a network is not planar). Therefore, we adopt a different approach, by asking the prover to certify a specific form of planar embedding of the network, not relying on coordinates. Our approach is inspired by the work on planar graphs by Fraysseix and Rosentiehl [14], based itself on Tutte's crossing number theory [43]. Observe that any graph G can be embedded in the plane as a planar embedding of a spanning tree T of G, plus cotree edges (the *cotree* of G associated with T is the set of edges in G not in T), such that the crossings occur between cotree edges only. Such an embedding is called a T*-embedding* of G. A graph G is planar if and only if there exists a T-embedding of G in the plane with no crossing edges. Given a planar graph G, our prover provides the nodes with a distributed proof that there is T-embedding of G in which no cotree-edges cross.

We also show that our proof-labeling scheme has certificates of optimal size, in the sense that there are no *locally checkable proofs* [29] for planarity that use certificates of size $o(\log n)$ bits (locally checkable proofs are verification mechanisms stronger than proof-labeling schemes). In fact, we show a more general result regarding the class of graphs excluding a complete graph K_k as a minor, for any $k \geq 3$, and the class of graphs excluding a complete bipartite graph

$K_{p,q}$ as a minor, for any $p, q \geq 2$. The proof for graphs excluding K_k is an extension of the technique used in [21] for lower bounding the size of *global* certificates. The proof for graphs excluding $K_{p,q}$ is a non-standard adaptation of the original proof of a lower bound for checking spanning tree and leader election in [29]. Combining these results allows us to consider the class of graphs excluding both K_5 and $K_{3,3}$ as minors, from which the lower bound for planarity follows directly.

To sum up, the paper is dedicated to establishing the following results.

Theorem 1 *There is a 1-round proof-labeling scheme for planarity with certificates on $O(\log n)$ bits in n-node networks.*

For a finite family $\mathcal{H}$ of graphs, let $\mathrm{Forb}(\mathcal{H})$ be the class of graphs with all the graphs in $\mathcal{H}$ excluded as "forbidden minors".

Theorem 2 *Let $\mathcal{F} = \{K_k : k \geq 3\} \cup \{K_{p,q} : p, q \geq 2\}$. For any non-empty finite family $\mathcal{H}$ of graphs in $\mathcal{F}$, there are no locally checkable proofs for $\mathrm{Forb}(\mathcal{H})$ using certificates on $o(\log n)$ bits.*

The following result follows directly from Theorem 2 as the planar graphs form the class $\mathrm{Forb}(\{K_5, K_{3,3}\})$ by Wagner's theorem (see [15]).

Corollary 1 *There are no locally checkable proofs for planarity, using certificates on $o(\log n)$ bits.*

Theorem 2 can also be used for other graph classes. For example, outerplanar graphs, which are the planar graphs that can be drawn in the plane with all the vertices on the outerface, form the class $\mathrm{Forb}(\{K_4, K_{2,3}\})$, and thus have no locally checkable proof with certificates on $o(\log n)$ bits.

Remark. All our lower bounds hold even if one allows verification algorithms performing an arbitrarily large constant number of rounds.

1.2 Related work

Distributed algorithms for planar graphs. As for the case of sequential computing, planar graphs have attracted significant interest in the context of distributed computing. Indeed, planar graphs do have structural properties that allow for fast distributed algorithms, in particular as far as approximation algorithms for classic problems such as minimum dominating set, or maximum matching are concerned. We refer to [9, 10, 11, 12, 13, 30, 36, 37, 44] for a non-exhaustive list of examples of such contributions.

Recently, it was proved that a combinatorial planar embedding that consists of each node knowing the clockwise order of its incident edges in a fixed planar drawing can be computed efficiently in the CONGEST model [26], and then used to derive $O(D)$-round distributed algorithms for MST and min-cut in planar networks of diameter D [27], hence bypassing the lower bounds $\tilde{\Omega}(D + \sqrt{n})$ for general networks [40, 42]. Even more recently, a randomized distributed algorithm for computing a DFS tree in planar networks has been designed [28], whose complexity beats the best known complexity bound $O(n)$ for general graphs [4].

More generally, it is worth to mention that there is also a large recent literature on distributed algorithms designed for other families of sparse graphs, beyond planar graphs. This is for instance the case of graphs of bounded genus, and graphs of bounded expansion (see, e.g., [2, 3]). We refer to [17] for a bibliography on distributed algorithms in classes of sparse graphs.

Distributed decision and verification. Locally checkable proofs (LCPs) were introduced in [29], as an extension of the seminal notion of *proof-labeling scheme* (PLS) introduced in [34]. There are only two differences between these two concepts, but they are subtle, and require some care. The first difference is that LCPs allow for verification procedures performing many rounds, while PLS are restricted to a single round of verification. Nevertheless, PLS can be easily extended to many rounds whenever the certificates are large enough to contain IDs, which enables to trade longer verification time for smaller certificates size (see, e.g., [20]). The second difference is more profound. While PLSs impose verification procedures exchanging the certificates only, LCPs allow the verification procedures to exchange additional information, e.g., the entire states of the nodes.

As a consequence, a PLS with certificates on $O(\log n)$ bits can be systematically implemented in one round in the $\mathsf{CONGEST}$ model, while this is not necessarily the case of LCPs, if the additional information is of size $\omega(\log n)$ bits. Nevertheless, this phenomenon has little impact on upper bounds in general, as the states of the nodes are often encoded on a logarithmic number of bits too (e.g., a constant number of pointers to neighbors, a constant number of bits marking the nodes, etc.). On the other hand, the difference between LCPs and PLSs has strong impact on the design of lower bounds, especially for demonstrating impossibility results when using sub-logarithmic certificates, as the IDs of the nodes (which are part of their states) are seen by the neighbors in LCPs, but not in PLSs with certificates on $o(\log n)$ bits. *Non-deterministic local decision* (NLD) [22] is yet another similar notion of distributed certification. It differs from the PLS and LCP in the fact that the certificates must be independent from the identity-assignment to the nodes. We refer to [16, 18] for recent surveys on distributed decision and verification.

Remark. Observe that the upper bound in Theorem 1 is obtained by designing a PLS for planarity, while the lower bounds in Theorem 2 hold even for LCPs.

All the aforementioned notions were extended by allowing the verifier to be randomized (see [24]). Such protocols were originally referred to as *randomized PLS* (RPLS), but are nowadays referred to as distributed Merlin-Arthur (dMA) protocols. The concept of distributed verification has also been extended [6, 19] in a way similar to the way NP was extended to the complexity classes forming the Polynomial Hierarchy, by alternating quantifiers.

Distributed interactive proofs. Recently, *distributed interactive proofs* were formalized [33], and the classes $\mathsf{dAM}[k]$ and $\mathsf{dMA}[k]$, $k \geq 1$, were defined, where k denotes the number of alternations between the centralized Merlin, and the decentralized Arthur. For instance, $\mathsf{dAM}[3] = \mathsf{dMAM}$ and $\mathsf{dMA}[2] = \mathsf{dMA}$, while LCP and PLS can be viewed as equal to $\mathsf{dAM}[1] = \mathsf{dM}$ (Merlin provides the nodes with their certificates, without challenges from Arthur). Distributed interactive protocols for problems like the existence of a non-trivial automorphism (AUT), and non-isomorphism ($\overline{\mathsf{ISO}}$) were designed and analyzed in [33]. The follow up paper [38] improved the complexity of some of the protocols in [33], either in terms of the number of interactions between the prover and the verifier, and/or in terms of the size of the certificates. A sophisticated generic way for constructing distributed IP protocols based on sequential IP protocols is presented in [38]. One of the main outcome of this latter construction is a dMAM protocol for planarity, using certificates on $O(\log n)$ bits. For other recent results on distributed interactive proof, see [8, 23].

2 Model and definitions

We consider the standard model for distributed network computing [39]. The network is modeled as a simple connected graph $G = (V, E)$ — note that self-loops and multiple edges can be eliminated without impacting planarity, and the planarity test could trivially be performed independently in each connected components if the graphs were not connected. The number of nodes is denoted by $n = |V|$. Each node v has an identifier $\mathrm{id}(v)$, which is unique in the network, and picked from a range of IDs polynomial in n. Therefore, every node identifier can be stored on $O(\log n)$ bits.

We recall the notion of *proof-labeling schemes*, and *locally checkable proofs* for certifying graph classes. (These mechanisms can also be used to certify classes of node- or edge-labeled graphs, but we are solely interested in unlabeled graph classes in this paper). Let $\mathcal{C}$ be a graph class, e.g., planar graphs. A locally checkable proof for $\mathcal{C}$ is a prover-verifier pair where the *prover* is a non-trustable oracle assigning *certificates* to the nodes, and the *verifier* is a distributed algorithm enabling the nodes to check the correctness of the certificates by performing a single round of communication with their neighbors. Note that the certificates may not depend on the instance G only, but also on the identifiers assigned to the nodes. In proof-labeling schemes, the information exchanged between the nodes during the verification phase is limited to the certificates. Instead, in locally checkable proofs, the nodes may exchange extra-information regarding their individual state (e.g., their IDs, if not included in the certificates, which might be the case for certificates of sub-logarithmic size). The prover-verifier pair must satisfy the following two properties.

Completeness: Given $G \in \mathcal{C}$, the non-trustable prover can assign certificates to the nodes such that the verifier *accepts* at all nodes;

Soundness: Given $G \notin \mathcal{C}$, for every certificate assignment to the nodes by the non-trustable prover, the verifier *rejects* in at least one node.

The main *complexity measure* for both locally checkable proofs, and proof-labeling schemes is the size of the certificates assigned to the nodes by the prover.

As an example, let us consider the class of paths. A possible proof-labeling scheme is as follows. Given a path $P = (v_1, \dots, v_n)$, the prover assigns the certificate $c(v_i)$ to node v_i, simply defined as the hop-distance between v_1 and v_i, for $i = 1, \dots, n$. The verifier executed at node u checks that u has degree 1 or 2, and, if u has degree 2, then it also checks that one of its two neighbors has certificate $c(u) - 1$ while the other has certificate $c(u) + 1$. If all tests are passed, then u accepts, else it rejects. Completeness holds by construction. For soundness, we simply observe that, if all nodes accept, then the nodes are necessarily forming a path as we assume the network to be connected.

Certifying the class of graphs *containing* a path of length at least k, for some fixed $k \geq 1$ can be done similarly, by also providing in the certificate of a node the IDs of its predecessor and successor in the path. The extremity of the path with positive hop-distance also checks that its hop-distance is at least k. In addition, the certificates must contain a distributed proof that the path exists somewhere in the network. This is achieved by asking the prover to provide the nodes with a distributed encoding of a tree spanning all nodes, rooted at one origin of the path, plus a distributed proof for this spanning tree. Such proof is known for long, as it is implicitly or explicitly present in the early work on self-stabilizing algorithm [1, 5, 32] — it simply consists of yet another hop-distance counter, plus the ID of the root.

The examples above are the main ingredients enabling the design of a proof-labeling scheme for certifying *non planarity*, whose existence is folklore in the context of distributed certification.

It is indeed sufficient to provide the nodes with a distributed proof that the graph contains a subdivided K_5 or a subdivided $K_{3,3}$. In both cases, this can be done by encoding the paths corresponding to these subdivided graphs in the certificates of the nodes in these paths. In addition, a spanning tree and its proof are given in the certificates of all nodes for establishing the existence of the subdivided graph (every node receives a pointer to a parent, its hop-distance to the root, and the ID of the root, which must be a node of the subdivided K_5, or subdivided $K_{3,3}$). We omit all the tedious details, and we expect that the reader has now understood the concept of proof-labeling schemes (and locally checkable proofs).

Observe that, in all the examples mentioned in this section, including certifying non-planarity, all certificates can be encoded on $O(\log n)$ bits. In the remaining part of the paper, we demonstrate that planarity can be distributedly certified, with $O(\log n)$-bit certificates too.

3 Upper bound

In this section, we establish our main result, namely, that there is a 1-round proof-labeling scheme for planarity with certificates on $O(\log n)$ bits in n-node networks. The design and analysis of our proof-labeling scheme for planarity is decomposed into three stages. First, in Section 3.1, we describe a proof-labeling scheme for a specific class of planar graphs, called *path-outerplanar* graphs. Roughly, these graphs are Hamiltonian graphs which can be drawn in the plane without crossing edges in such a way that the Hamiltonian path forms a line, and the edges not in the path, are all on the same side of the line. Note that a non-Hamiltonian tree is outerplanar, but is not path-outerplanar. Next, in Section 3.2, we show how to transform a planar graph into a path-outerplanar graph. More precisely, we show how to transform a T-embedding [14] of a graph into a drawing of a new graph such that, roughly, the new graph is path-outerplanar if and only if the original graph is planar. Finally, Section 3.3 shows how to use this transformation for extending the proof-labeling scheme for path-outerplanarity to a proof-labeling scheme for planarity. Throughout the scheme, we use the key property that every planar graph is 5-degenerate (i.e., every subgraph has a vertex of degree at most 5), in order to distribute the certificates evenly among the nodes.

3.1 Certifying path-outerplanar graphs

In this section, we present a proof-labeling scheme for a subclass of outerplanar graphs, that will be used as a building block for our scheme for planar graphs. Recall that a graph is outerplanar if it has a planar drawing with all vertices incident to the same face, called the outerface. We start with a combinatorial definition of path-outerplanarity, and then show the equivalence with a geometric definition.

Definition 1 *A graph $G = (V, E)$ is* path-outerplanar *if there is a total ordering $P = (V, <)$ of its vertices such that P forms a path (i.e., consecutive vertices in P are adjacent in G), and, for any pair of edges $\{a, b\}, \{c, d\} \in E$ with $a < b$ and $c < d$, one of the following inequalities holds: $a < b \leq c < d$, $c < d \leq a < b$, $a \leq c < d \leq b$, or $c \leq a < b \leq d$. The ordering P is called a* path-outerplanarity witness *of G.*

Lemma 1 *A graph is* path-outerplanar *if and only if it has a Hamiltonian path that can be drawn as a horizontal line such that all edges that do not belong to the path can be drawn above that line as semi-circles without crossings.*

Proof. Assume that G is path-outerplanar with witness $(1, \ldots, n)$. The vertices $\{1, \ldots, n\}$ can be drawn on a horizontal line, with vertex i at coordinate i. Each edge $\{a, b\}$ of G can be drawn as a semi-circle, with endpoints a and b, above the line. The drawing is planar since, by definition, for every two edges $\{a, b\}, \{c, d\}$ of G with $a < b$ and $c < d$, we have $|[a, b] \cap [c, d]| \leq 1$, or $[a, b] \subset [c, d]$, or $[c, d] \subset [a, b]$. Conversely, let us assume that there exists a planar drawing as described. Let us then consider the total order induced by the placement of the vertices along this horizontal path (say, from left to right). Since no two semi-circles cross, for every pair $\{a, b\}, \{c, d\}$ of edges, one of the four inequalities in Definition 1 must be satisfied, and thus the graph is path-outerplanar. □

Lemma 2 *There is a 1-round proof-labeling scheme for path-outerplanarity, with certificates on $O(\log n)$ bits in n-node networks.*

Proof. Given a path-outerplanar graph G, the prover computes a witness $P = (v_1, \dots, v_n)$ for G, and sends an $O(\log n)$-bit certificate to every vertex x, with the following information:

1. the number n of vertices of the graph;
2. the rank of x in P, i.e., the value i such that $x = v_i$;
3. the shortest interval $I(x) = [a, b]$ such that $\{v_a, v_b\}$ is an edge of G, and $a < i < b$ (if no such interval exists, then the prover sets $I(x)$ to $[0, n + 1]$).

In addition, the prover provides the nodes with information in their certificates enabling them to check that n is indeed the correct number of vertices, that the vertices are correctly ranked from 1 to n, and that this ranking induces a path. We do not detail these parts of the certificates, nor we describe the verification algorithm for these certificates, as this can be achieved using standard techniques [34] using $O(\log n)$-bit certificates.

For simplicity, in the algorithm as well as in the remaining part of the proof, we denote the vertices by their ranks, as numbers from 1 to n. We also add two virtual vertices 0 and $n + 1$, with virtual edges $\{0, 1\}, \{0, n + 1\}$, and $\{n, n + 1\}$. We set $I(0) = I(n + 1) = [-\infty, \infty]$. This ensures that any vertex x with $1 \leq x \leq n$ has at least one neighbor smaller than itself, and another larger than itself. Of course, the verification algorithm is only performed at the real vertices, i.e., those from 1 to n, with node 1 (resp., node n) simulating the behavior of its virtual neighbor 0 (resp., $n + 1$).

The verification algorithm performed at each node x is displayed in Algorithm 1.

We first show completeness.

Claim 1 *If G is path-outerplanar, and if the prover provides certificates corresponding to a witness P, then Algorithm 1 accepts at all nodes.*

Proof. For any node x, let $x_\ell^- < \cdots < x_0^- < x_0^+ < \cdots < x_k^+$, with $\ell \geq 0$ and $k \geq 0$, be the neighbors of x in the ordering P. In particular, $x_0^- < x < x_0^+$. Figure 1 illustrates the structure of the neighborhood of a node x, and the interval $I(x) = [a, b]$ (cf. Lemma 1). Note that, since G is planar, $a \leq x_\ell^-$, and $b \geq x_k^+$. The test of Line 3 succeeds since P is the witness for G. Observe that the vertices y with $I(y) = [a, b]$ are exactly the vertices incident to the unique face below the edge $\{a, b\}$, except a and b — this also holds for the nodes 0 and $n + 1$, as we could add a virtual edge between $-\infty$ and ∞. In particular, no edge incident to x can cross the edge $\{a, b\}$, which implies that the test of Line 5 is passed. The same arguments applied to x_i^+ explains why the test of Line 7 is passed (respectively, x_i^- and Line 9). For the rightmost neighbor x_k^+ of x, we distinguish between the cases $x_k^+ < b$, and $x_k^+ = b$. In the first case, x_k is also incident to the face below the edge $\{a, b\}$, and thus the test of Line 11 succeeds. The same arguments apply for the test at Line 13. The case $x_k = b$ is not explicitly addressed by

Algorithm 1: Verification procedure for path-outerplanarity at node x, $1 \le x \le n$.

```
let $x_\ell^- < \cdots < x_0^- < x_0^+ < \cdots < x_k^+$, with $\ell \ge 0$ and $k \ge 0$, be the neighbors of $x$;
collect the certificates of each neighbor: graph size, rank, and interval;
check that the ranks correspond to a spanning path of size $n$;
let $I(x) = [a,b]$;
check that $a < x < b$, and that all neighbors of $x$ are in the interval $[a,b]$;
for $i = 0$ to $k-1$ do
    check that $I(x_i^+) = [x, x_{i+1}^+]$;
for $i = 0$ to $\ell-1$ do
    check that $I(x_i^-) = [x_{i+1}^-, x]$;
if $x_k^+ < b$ then
    check that $I(x_k^+) = [a,b]$;
if $x_\ell^- > a$ then
    check that $I(x_\ell^-) = [a,b]$;
for every neighbor $y$ of $x$ do
    if one of the endpoints of $I(y)$ is $x$ then
        check that the other endpoint of $I(y)$ is adjacent to $x$;
        check that $I(y) \subsetneq I(x)$;
if all checks are passed then accept else reject.
```

Algorithm 1 applied on x. However, it is part of the tests performed at Lines 16 and 17, when applied to node x_k^+. These two tests succeed by definition of $I(x)$ and $I(y)$. It follows that node x accepts, as desired. $\diamond$

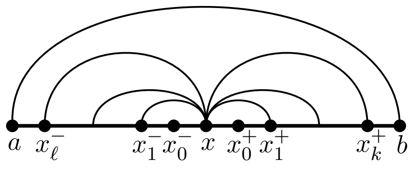

Figure 1: The neighborhood of a node x, and the edge $\{a,b\}$ covering x.

It remains to establish soundness. Namely, we prove that, if Algorithm 1 accepts at all nodes, then G is path-outerplanar. As mentioned before, we do not detail the test for the spanning path (Line 3), but this test ensures that the ranks are consistent, from 1 to n, and that consecutive vertices in this order are adjacent in G. We thus safely denote the vertices by their ranks. In fact, we show that if Algorithm 1 accepts at all nodes, then G is path-outerplanar with witness $P = (1, 2, \ldots, n)$. Let us assume, for the purpose of contradiction, that this is not the case. It follows that there are four nodes $x < y < z < t$ such that both edges $\{x,z\}$ and $\{y,t\}$ appear in G. Under these conditions, let us choose such four nodes with the additional conditions that (i) $z - y$ is minimum, and (ii) $t - x$ is minimum subject to (i). The following claim is a straightforward consequence of conditions (i) and (ii) — see Figure 2 for an illustration of the framework of the claim.

Claim 2 *For every edge $\{a,b\}$, if $y < a < z$, then $y \le b \le z$. Also, if $y = a < b < t$, then $b \le z$, and if $x < a < z = b$, then $a \ge y$. Finally, any two edges with both endpoints in the interval $[y,z]$ are non-crossing.* $\diamond$

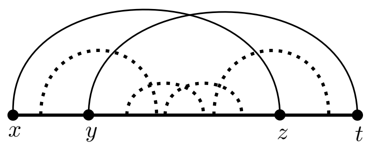

Figure 2: The crossings involving edges depicted by dotted lines cannot exist.

To understand the structure of the graph in between y and z, let us define:

- y_j^+, the rightmost neighbor of y strictly before t in the ordering P,
- z_i^-, the leftmost neighbor of z strictly after x in the ordering P.

Note that these two nodes exist, as the node immediately on the right of y in the path appears strictly before t, and the node immediately on the left of z in the path appears strictly after x.

Claim 3 *The following holds: $y < y_j^+ < z_i^- < z$, $I(y_j^+) = [y,t]$, and $I(z_i^-) = [x,z]$.*

Proof. By Claim 2, the inequality $y < y_j^+ \leq z$ holds. Also, as the test of Line 7 applied to vertex y succeeds, it follows that $I(y_j^+) = [y,t]$. Observe that $\{y,z\}$ cannot be an edge of the graph. Indeed, if $\{y,z\}$ is an edge of the graph, then it must be the case that $z = y_j^+$, and thus $I(z) = [y,t]$. Also, symmetrically, the equality $I(y) = [x,z]$ must hold too. As the test of Line 17 applied to vertex y succeeds, it follows that $I(z) \subsetneq I(y)$, which contradicts the fact that the two intervals overlap. Therefore $\{y,z\}$ is not an edge of the graph. It follows that $y_j^+ < z$. By the same arguments as for y_j^+, but applied to the left-hand side of z, it follows that $y < z_i^-$, and $I(z_i^-) = [x,z]$. Also observe that we have $y_j^+ \leq z_i^-$, by Claim 2 applied to edges $\{y, y_j^+\}$ and $\{z_i^-, z\}$. Finally, as $I(y_j^+) = [y,t]$ and $I(z_i^-) = [x,z]$, and as these two intervals are different, it must be that $y_j^+ < z_i^-$. $\diamond$

In the following, an edge $\{u,v\}$ is said to be *maximal* if v is the rightmost neighbor of u, and u is the leftmost neighbor of v. See Figure 3.

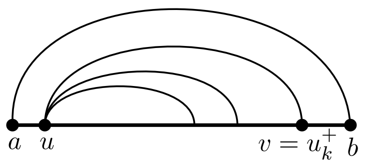

Figure 3: A maximal edge $\{u,v\}$.

The next claim describes the structure of the graph between y_j^+ and z_i^- (see Figure 4).

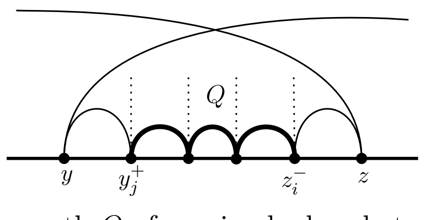

Figure 4: The path Q of maximal edges between y and z.

Claim 4 *There is a path $Q = (q_1, \dots, q_s)$ in G, with $s > 1$, $q_1 = y_j^+$, and $q_s = z_i^-$, such that, for every $i = 1, \dots, s-1$, $q_i < q_{i+1}$ and $\{q_i, q_{i+1}\}$ is a maximal edge.*

Proof. The path is constructed as follows: $q_1 = y_j^+$ and, as long as the rightmost vertex q_i of the partial path created so far satisfies $q_i < z_i^-$, we pick q_{i+1} as the rightmost neighbor of q_i. We claim that $q_{i+1} \leq z_i^-$. This is because:

1. by Claim 2, it cannot be the case that $q_{i+1} > z$;

2. $q_{i+1} = z$ cannot hold because q_i would then contradict the choice of z_i^- as leftmost neighbor of z larger than x;

3. $z_i^- < q_{i+1} < z$ is impossible as the crossing edges $\{q_i, q_{i+1}\}$ and $\{z_i^-, z\}$ would contradict Claim 2.

Therefore the construction of Q correctly terminates at vertex z_i^-.

It remains to show that edges $\{q_i, q_{i+1}\}$ are maximal for all $i = 1, \dots, s-1$. By construction, q_{i+1} is the rightmost neighbor of q_i, and thus it is sufficient to prove that q_i is the leftmost neighbor of q_{i+1}. Assume that q_{i+1} has a neighbor $q' < q_i$. As for the construction of Q, observe that $q' \geq y$ by Claim 2 applied to the edge $\{q', q_{i+1}\}$. Also, q' cannot be one of the vertices $y \in \{q_1, \dots, q_i\}$ because, for every $j = 1, \dots, i$, q_j is the rightmost neighbor of q_{j-1} smaller than z. Eventually, q' cannot be strictly between q_{j-1} and q_j for some j with $1 \leq j \leq i$, by Claim 2 applied to edges $\{q', q_i\}$ and $\{q_{j-1}, q_j\}$. Therefore, q_i is the leftmost neighbor of q_{i+1}, and all edges $\{q_i, q_{i+1}\}$ are maximal. $\diamond$

We were interested in maximal edges because of the following claim.

Claim 5 *If the edge $\{u, v\}$ is maximal, and Algorithm 1 accepts at all nodes, then $I(u) = I(v)$.*

Proof. We refer to Figure 3 for an illustration of the arguments developed in the proof. Let $I(u) = [a, b]$ and $I(v) = [c, d]$. Let us assume, w.l.o.g., that $u < v$. Since $\{u, v\}$ is maximal, $v = u_k^+$ where u_k^+ is the largest neighbor of u. Since Algorithm 1 accepts at all nodes, all tests performed by the algorithm at all nodes have positive outcome. Therefore, by the test of Line 5 applied to $x = u$, we derive that $u_k^+ \leq b$, and thus $v \leq b$. Now let us consider three cases separately.

1. If $v < b$, then the test of Line 11 applied to u guarantees that $I(u_k^+) = I(u)$, and the claim follows.

2. If $v = b$ but $u = v_\ell^- > c$, where v_ℓ^- is the leftmost neighbor of v, then, by symmetry, the test of Line 9 applied to $x = v$ implies that $I(u) = I(v)$, as claimed.

3. If $v = b$ and $u = c$, then the test of Line 17 applied at $x = u$, and at $x' = b = v$ implies that $I(v) \subsetneq I(u)$. Symmetrically, by the same test at $x = v$ and $x' = u = c$, we also get that $I(u) \subsetneq I(v)$. This is a contradiction, and thus this case cannot appear.

This completes the proof of Claim 5. $\diamond$

We can now complete the soundness proof for our proof-labeling scheme. By Claims 4 and 5, all the vertices of the path Q (formed of maximal edges) were assigned the same interval I. This contradicts the fact that, according to Claim 3, $I(q_1) = [y, t]$ and $I(q_s) = [x, z]$. This contradiction completes the proof of Lemma 2. $\square$

3.2 From path-outerplanar graphs to planar graphs

The previous section has demonstrated how to construct a compact proof-labeling scheme for the class of path-outerplanar graphs. For extending the scheme to planar graphs, this section presents a transformation that maps any planar graph to a path-outerplanar graph. To get an intuition of this transformation, let us first explain how a tree T can be transformed into a path. The transformation is inspired by the classical approximation algorithm by Christofides for the traveling salesman problem. A depth-first search (DFS) traversal of the tree returns an ordering of the nodes by indices from 1 to $2n-1$, where a same node can be assigned several indices. This ordering can be transformed into a path on $2n-1$ nodes, by creating a copy of each node for each time it is visited by the DFS traversal, and linking such nodes according to the ordering of the traversal.

More formally, let T be a tree spanning a planar graph G. We fix a drawing of G in the plane without crossing edges (see Figure 5(a), ignoring node r' at this stage of the proof). We can assume, w.l.o.g., that all edges are drawn as straight lines (it is not crucial in the construction, but it helps for understanding it). The tree T is rooted at an arbitrary vertex r. For any vertex $v \neq r$, its neighbor on the path from v to r is called the *parent* of v, and the other neighbors of v are called its *children*. For any node v, $\deg_T(v)$ is the degree of v in T. A DFS traversal of T starting from r provides a *DFS-mapping* $f : \{1,\dots,2n-1\} \to V(T)$ satisfying the following: $f(1) = r$, and, for $1 \leq i < 2n-1$,

$$f(i+1) = \begin{cases} \text{a child of } f(i) \text{ that is not in } f(\{1,\dots,i-1\}) & \text{if there is such a node;} \\ \text{the parent of } f(i) & \text{otherwise.} \end{cases}$$

Note that f is onto but not one-to-one. In particular, each vertex v of T, different from the root is mapped by f^{-1} to $\deg_T(v)$ different vertices of the path $P = (1,2,\dots,2n-1)$ (see Figure 5(b)). The root r is mapped to $\deg_T(r)+1$ vertices of P. Intuitively, each edge of T is mapped by f^{-1} on exactly two edges of P. Note that if $1 \leq i,j \leq 2n-1$ are two integers such that i is the least integer satisfying $f(i) = v$, and j is the largest integer satisfying $f(j) = v$, then, for every $i \leq k \leq j$, $f(k)$ is either a descendent of v, or v itself.

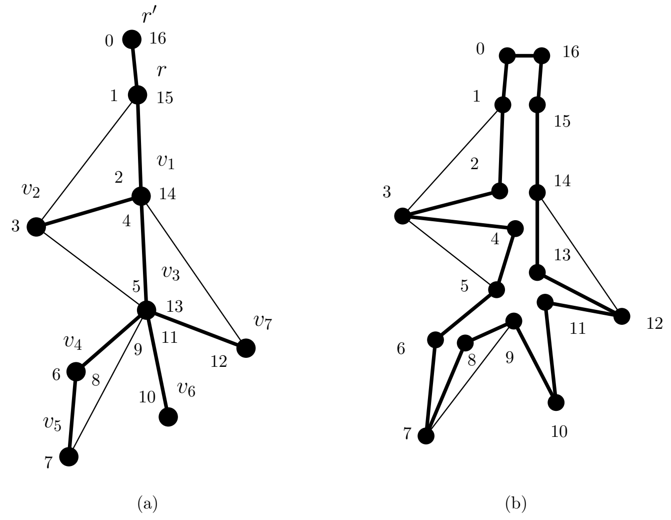

Figure 5: Transformation of a planar graph to a path planar graphs.

Given (G,T,f), we define a class of graphs induced by (G,T,f) in such a way that: (1) if G is planar, then there exists a choice of T and f for which a graph induced by (G,T,f) is path-outerplanar, and (2) if G is not planar, then for every T and f, no graph induced by (G, T, f) is path-outerplanar. Figure 5 provides an illustration of the definition below (again, ignore node r' at this stage of the proof).

Definition 2 *A graph H is said to be* induced by (G, T, f) *if $V(H) = \{1, \ldots, 2n-1\}$, and (1) $\{i, i+1\} \in E(H)$ for every $1 \leq i < 2n-1$, and (2) for every cotree edge $\{u, v\} \in E(G) \setminus E(T)$, there exists a unique edge $\{i, j\} \in E(H)$ such that $\{f(i), f(j)\} = \{u, v\}$.*

Note that the definition does not imply the uniqueness of H as a graph induced by (G, T, f). For example, in Figure 5(b), if we replace the edge between nodes 7 and 9 by an edge between 7 and 5, we get another graph induced by the graph on Figure 5(a) that satisfies Definition 2.

Lemma 3 *For every planar graph G, and every spanning tree T of G, there exists a DFS-mapping f of T, and a graph $G_{T,f}$ induced by (G, T, f), such that $G_{T,f}$ is path-outerplanar.*

Proof. Let $G = (V, E)$ be a planar graph, and consider a drawing of it in the plane. Let T be a spanning tree of G, rooted at an arbitrary node r. For every vertex $v \in V$, we set a numbering

$$\nu_v : \{0, \ldots, \deg_T(v) - 1\} \to N(v),$$

where $N(v)$ denotes the set of neighbors of v in T. In this numbering, $\nu_v(0)$ is the parent of v in T if $v \neq r$. These numberings satisfy that, for $i = 0, \ldots, \deg_T(v) - 1$, the edge $\{v, \nu_v(i)\}$ is immediately followed by the edge $\{v, \nu_v(i + 1 \bmod \deg_T(v))\}$ when one follows the counterclockwise ordering of the edges of T incident to v in the planar drawing of G. For every two distinct children u, u' of v, the fact that $u = \nu_v(k)$ and $u' = \nu_v(k')$ with $k < k'$ is denoted by $u \prec u'$.

We now define the DFS-mapping f of T as follows. The DFS traversal used for constructing f starts from r, and explores the children of each node v in counterclockwise order, i.e., in the order provided by the numbering ν_v. It follows that, for every $v \in V$, there exists a sequence $1 \leq i_1 < \cdots < i_d \leq 2n-1$ of integers, with $d = \deg_T(v)$ if $v \neq r$, and $d = \deg_T(r) + 1$ otherwise, such that $f^{-1}(v) = \{i_1, \ldots, i_d\}$, and

$$f(i_1 + 1) = f(i_2 - 1) \prec f(i_2 + 1) = f(i_3 - 1) \prec \cdots \prec f(i_{d-1} + 1) = f(i_d - 1).$$

To construct the desired path-outerplanar graph $G_{T,f}$, it is convenient to add an extra node r' to G, of degree 1, connected to the root r of T. In the drawing, r' is placed so that $\{r, r'\}$ appears between the edges $\{r, \nu_r(0)\}$ and $\{r, \nu_r(\deg_T(r) - 1)\}$ when one follows the clockwise ordering of the edges of T incident to r in the planar drawing of G. Moreover, r' is placed close enough to r in the plane so that the edge $\{r, r'\}$ drawn as a straight line does not cross any edge in the drawing of G (see Figure 5(a)). We extend f to $\{0, 2n\}$ by setting $f(0) = f(2n) = r'$. In this way, r becomes an internal node of the tree $T' = T + r'$, avoiding special considerations in the construction of $G_{T,f}$ regarding whether a node v is the root r or not. In particular, d systematically denotes the degree of the considered node $v \in V$, including r, and thus $f(i_1 - 1) = f(i_d + 1)$ is the parent of v in T', for every $v \in V$.

For every $v \in V$, let us fix a circle C_v centered at the point representing v in the planar drawing of G. We choose C_v sufficiently small so that (i) v is the only node inside C_v, (ii) the only drawings of edges crossing C_v have v as endpoints, and (iii) $C_v \cap C_{v'} = \emptyset$ whenever $v \neq v'$. See Figure 6. For $1 \leq k \leq d$, let x_{i_k} be the point of the plane at the intersection of C_v with the drawing of the edge $\{f(i_k), f(i_k + 1))\}$. These points are met in the order $x_{i_1}, \ldots, x_{i_d}$ whenever one travels along C_v counterclockwise, starting from x_{i_1}.

We say that a point x of C_v is of type i_k if, traveling along C_v counterclockwise starting from x, the first point met among $\{x_{i_1}, \dots, x_{i_d}\}$ is x_{i_k}. The type of x is denoted by $\tau(x)$. For every cotree edge $\{u, v\} \in E \setminus E(T)$, the drawing of $\{u, v\}$ meets C_u in a point $x_{(u,v)}$, and meets C_v in a point $x_{(v,u)}$ (see Figure 6). We now consider the graph $G_{T,f}$ induced by (G, T, f), with the specific requirement that, for every cotree edge $\{u, v\}$ of $E(G) \setminus E(T)$, the unique edge $\{i, j\} \in E(G_{T,f})$ such that $\{f(i), f(j)\} = \{u, v\}$ is the edge $\{i, j\} = \{\tau(x_{(u,v)}), \tau(x_{(v,u)})\}$.

The proof of the lemma is completed by establishing that $G_{T,f}$ is outerplanar. To do so, it is convenient to state the following simple result. For a graph $G = (V, E)$ with vertices numbered from 1 to n, let G^+ be the graph obtained from G by adding two vertices 0 and $n + 1$, and $E(G^+) = E \cup \big\{\{0, 1\}, \{0, n + 1\}, \{n, n + 1\}\big\}$.

Claim 6 *If G is path-outerplanar with witness $(1, 2, \dots, n)$, then G^+ is outerplanar, and has a drawing in which the outerface corresponds to the cycle $(0, 1 \dots, n, n + 1)$. Conversely, if G^+ is outerplanar with a drawing in which the outerface forms a cycle $(0, 1, \dots, n, n + 1)$ in G^+, then G is path-outerplanar with witness $(1, 2, \dots, n)$.* $\diamond$

By Claim 6, it is sufficient to provide a planar embedding of $G^+_{T,f}$ in which the vertices $\{0, \dots, 2n\}$, which form a cycle, are on the same face, in order. For this purpose, at each node v of G, the circle C_v is split in d sections, such that section k, for $1 \leq k \leq d$, contains all the points of C_v with type i_k (see Figure 6). We draw node i_k on the middle of the section k. In this way, all the nodes of $G_{T,f}$ are placed in the plane. Each edge $\{i, j\}$ of $G_{T,f}$ is drawn as a straight line connecting the corresponding points in the plane. In addition, the extra node r' is split into two nodes 0 and $2n$, and three edges are added: $\{0, 1\}, \{0, 2n\}$, and $\{2n - 1, n\}$. The two nodes 0 and $2n$ are placed next to each other, close enough so that these three edges do not intersect.

The transformation simply corresponds to thickening the edges, and the internal nodes of the tree T', where the thickness of the tree is at most the maximum, taken over all the nodes v, of the radius of C_v, which can be chosen as small as desired. It follows that planarity is preserved.

By the setting of f, the nodes $(0, \dots, 2n)$ form a path. By construction of $G_{T,f}$, the cycle $(0, 1, \dots, 2n)$ in $G^+_{T,f}$ draws in a plane a Jordan curve, splitting the plane into two regions. The one containing the original drawing of T is also a face of $G^+_{T,f}$, all the cotree edges being drawn on the other region. It then follows from Claim 6 that $G_{T,f}$ is path-outerplanar. $\square$

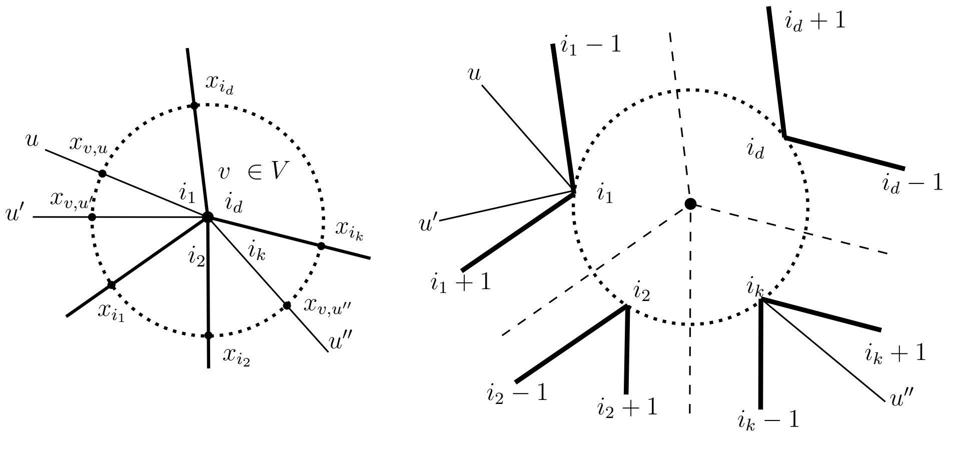

Figure 6: Planar drawing: from (G, T, f) to $G^+_{T,f}$.

The next result is a form of reciprocal of Lemma 3.

Lemma 4 *If G has a spanning tree T, and a DFS-mapping f of T, such that a graph H induced by (G, T, f) is path-outerplanar, then G is planar.*

Proof. Let us assume that G has a spanning tree T, and a DFS-mapping f of T such that some graph H induced by (G, T, f) is path-outerplanar. Let G_f be the graph with vertex set $\{1, 2, \ldots, 2n-1\}$ such that, for every $1 \leq i < j \leq 2n-1$, $\{i, j\}$ is an edge if and only if (1) $j = i+1$, or (2) $f(i) = f(j)$, and every $k \in \{i+1, \ldots, j-1\}$ satisfies $f(k) \neq f(i)$. The graph G_f is path-outerplanar by construction, due to the definition of the mapping f. Let G' be the graph defined as $V(G') = \{1, \ldots, 2n-1\}$, and $E(G') = E(H) \cup E(G_f)$. G' is planar, since both H and G_f are path-outerplanar graphs, and thus, for each of the two graphs, its edges can be drawn in one of the two sides of the path $(1, \ldots, 2n-1)$ — if these two graphs share edges, then they are considered as belonging to H. It follows that if all edges of G' belonging to $E(G_f) \setminus E(H)$ are contracted, then the resulting graph is planar. Now, observe that the resulting graph is precisely the graph G, because the contractions identify all pairs of nodes i, j for which $f(i) = f(j)$ holds. □

3.3 Certifying planar graphs

The previous sections provide us with the ingredients for the proof of Theorem 1. Lemma 3 provides a tool for transforming a planar graph into a path-outerplanar graph, by "cutting along" a spanning tree, and transforming the tree into a path through a specific DFS traversal. Importantly, this transformation is both ways, in the sense that, as stated in Lemma 4, if a graph G has been transformed into a path-outerplanar graph by cutting along a spanning tree, then G is necessarily planar. Our proof-labeling scheme for planar graphs aims at implementing this transformation, and then checking the path-outerplanarity of the resulting graph using the scheme provided in Lemma 2. The rest of the section contains the details of the implementation, including the delicate issue of keeping the certificate size small, which is not direct, as a same node of the graph may have to simulate the behavior of many vertices of the path-outerplanar graph after applying the transformation.

Proof of Theorem 1. Given a planar graph G, the prover draws it on the plane, constructs a spanning tree T, a DFS-mapping f on its vertices, and a path-outerplanar graph $G_{T,f}$ as in Lemma 3. The prover then provides the nodes with certificates allowing them to verify that:

1. T is a spanning tree of G, and f is a DFS-mapping of T;
2. $G_{T,f}$ is path-outerplanar, with witness f.

Certifying the conditions in the first item is quite simple and standard [34]. It is therefore omitted, apart from the issue of the certificates size, which we discuss later. We focus on the second item. The proof-labeling scheme of the path-outerplanarity of $G_{T,f}$ with witness f was presented in Lemma 2. Let us recall that a vertex x of G is transformed into several nodes in $G_{T,f}$, namely into the nodes in $f^{-1}(x) = \{i_1, \ldots, i_d\}$. The verification at node x involves the simulation of the verification protocol for path-outerplanarity for each of the nodes $i_1, \ldots, i_d$ of the graph $G_{T,f}$. The index d may however be large, even as large as $O(n)$. It follows that, for keeping certificates of logarithmic size, the prover cannot simply assign the d path-outerplanarity certificates of its d copies in $G_{T,f}$ to node x. We now explain how to cope with this issue for preserving $O(\log n)$-bit certificates.

Every edge $\{x, y\}$ of T is mapped on two edges $\{i, j\}$ and $\{i', j'\}$ of $G_{T,f}$, and every non-tree edge $\{x, y\}$ of G is mapped on one edge $\{i, j\}$ of $G_{T,f}$. The certificates corresponding to the

$2n-1$ vertices of $G_{T,f}$ are distributed to the n vertices of G in such a way that, after a single communication round in G, each vertex x collects the certificates of all its copies $i_1,\dots,i_d$ in the protocol for path-planarity, as well as the certificates of all the neighbors in $G_{T,f}$ of each copy i_j, $j=1,\dots,d$. To see how this is achieved, let us consider for now a virtual scenario in which the certificates can be distributed on the *edges* of G. Let $e=\{x,y\}$ be an edge of G, and let $\{i,j\}$ and $\{i',j'\}$ be its two associated edges in $G_{T,f}$ (if e is mapped to a unique edge of $G_{T,f}$, which is the case of cotree edges, we simply set $\{i',j'\}=\{i,j\}$). Then the certificate $c(e)$ of the edge e includes the following information:

- the identifiers of x and y in G;
- the values $i,j,i',j' \in \{1,\dots,2n-1\}$;
- the certificates of nodes i,j,i',j' in the proof-labeling scheme for path-outerplanarity in $G_{T,f}$ with witness f.

If we could assign $c(e)$ to every edge $e=\{x,y\}$ of G, then $c(e)$ could be sent to its endpoints x and y, and vertex x could simulate the verification protocol of Algorithm 1 for checking the path-outerplanar graph $G_{T,f}$ with witness f, for each copy $i_1\dots,i_d$ of x. Since assigning certificates to edges is not doable, the certificate $c(e)$ is actually assigned to one of its two endpoints. This assignment is performed so that each node receives at most five edge-certificates. For this purpose we use the fact that a planar graph $G=(V,E)$ has *degeneracy* at most 5. That is, there exists a total ordering σ of the vertices of G such that every node x has at most five neighbors y with $\sigma(y)>\sigma(x)$. Every edge e is associated to its smaller endpoint, according to σ. It follows that each node x is associated to at most five edges, all with x as endpoint. The prover assigns the certificate $c(x)$ to node x, defined as the concatenation of the at most five edge-certificates associated to x. Extra-informations are also added to every certificate, for allowing the nodes to check that T is a spanning tree of G. Overall, the certificates are of the due size $O(\log n)$ bits.

The verification protocol executed at each node of G is displayed in Algorithm 2.

Algorithm 2: Verification procedure for planarity at node $x \in V(G)$

```
1  collect the certificate of each neighbor;
2  //Phase 1: recover local information regarding $T, f$, and $G_{T,f}$;//
3  compute ID of the parent of $x$ in $T$, and the IDs of all its children in $T$;
4  compute $f^{-1}(x) = \{i_1, i_2, \dots, i_d\}$, i.e., all nodes corresponding to $x$ in $G_{T,f}$;
5  for every $i \in f^{-1}(x)$ do
6      retrieve certificate of node $i$ in $G_{T,f}$ dedicated to the path-outerplanarity of $G_{T,f}$;
7      compute the neighbors of $i$ in $G_{T,f}$, and their certificates for path-outerplanarity;
8  //Phase 2: check that $T$ is a spanning tree of $G$, and $f$ is a DFS-mapping of $T$;//
9  check local consistency of certificates for spanning tree;
10 check local consistency of certificates for DFS-mapping;
11 // Phase 3: check the path-outerplanarity of $G_{T,f}$ with witness $f$;//
12 for every $i \in f^{-1}(x)$ do
13     simulate execution of Algorithm 1 at node $i$ of $G_{T,f}$;
14 if all checks are passed then return accept else return reject.
```

In Phase 1 of Algorithm 2, every node x retrieves information regarding x itself as a node of G, and all the "virtual" nodes $i \in f^{-1}(x)$ of $G_{T,f}$. Observe that, for every edge $e=$

$\{x, y\}$ incident to x in G, node x obtains all information contained in $c(e)$ after one single communication round with its neighbors. An edge $e = \{x, y\}$ is in T if and only if the pair (x, y) has been mapped on two disjoint pairs (i, j), (i', j') corresponding to distinct edges $G_{T,f}$. Node y is the parent of x if $\min\{i, i'\} < \min\{j, j'\}$, and is a child otherwise. By considering all incident edges in G, x retrieves all its corresponding nodes $i_1, \ldots, i_d$ in $G_{T,f}$, their neighbors in $G_{T,f}$, and all the corresponding certificates for the path-outerplanarity of $G_{T,f}$ with witness f.

Phase 2 checks that T is a spanning tree of G (see [34]), and checks that f is a DFS-mapping of T. This latter condition is rather simple to check, so we only provide some hints. Let $f^{-1}_{\min}(x)$ and $f^{-1}_{\max}(x)$ be the smallest and largest values in $f^{-1}(x)$. These values correspond to the first, and last time the DFS visited node x. The children $y_1, \ldots, y_k$ of x in T are then sorted by increasing value of $f^{-1}_{\min}(y_i)$. The main test consists to check that:

- if x has no children, then $f^{-1}_{\max}(x) = f^{-1}_{\min}(x)$;
- if x has children $y_1, \ldots, y_k$, then $f^{-1}_{\min}(x) = f^{-1}_{\min}(y_1) - 1$, $f^{-1}_{\max}(x) = f^{-1}_{\max}(y_k) + 1$, and, for every $j = 1, \ldots, k$, $f^{-1}_{\min}(y_{j+1}) = f^{-1}_{\max}(y_j) + 2$.

In addition, the root r of T checks that $\{1, 2n - 1\} \subseteq f^{-1}(r)$.

Finally, Phase 3 of Algorithm 2 at x applies the verification protocol for checking the consistency of the distributed proof of the path-outerplanarity of $G_{T,f}$, for every node $i \in f^{-1}(x)$ of $G_{T,f}$ corresponding to x.

Completeness holds as, if the input graph G is planar, and if the prover constructs T and f correctly, as well as the path-outerplanar graph $G_{T,f}$ whose existence is guaranteed by Lemma 3, then Phase 2 accepts by construction of T and f, and Phase 3 accepts since the verification algorithm for path-outerplanarity is correct by Lemma 2.

For the soundness, assume that Algorithm 2 returns accepts at all nodes of G. Then, thanks to Phase 2, T is necessarily a spanning tree of G, and f is a DFS-mapping of T. Also, thanks to Phase 3, $G_{T,f}$ is necessarily path-outerplanar with witness f. Therefore, by Lemma 4, G is planar. This completes the proof of Theorem 1. □

4 Lower bounds

This section is dedicated to the proof of Theorem 2. The proof is modular: Section 4.1 treats the case of $\text{Forb}(K_k)$, while Section 4.2 treats the case of $\text{Forb}(K_{p,q})$. Finally, Section 4.3 combines the two cases, and completes the proof. All proofs in this section can be generalized to the setting where the verification algorithm performs t rounds instead of just one round, for any constant $t \geq 1$, by replacing some edges of the graph instances constructed hereafter by a path of length $O(t)$.

4.1 Lower bound for $\text{Forb}(K_k)$

Our lower bound proof for $\text{Forb}(K_k)$ is based on extending the techniques developed in [21].

Lemma 5 *For every $k \geq 3$, there are no locally checkable proofs for* $\text{Forb}(K_k)$ *with $o(\log n)$-bit certificates.*

Proof. We first define two similar types of instances that we call *paths of blocks* and *cycles of blocks*. We prove that paths of blocks are legal instances, that is, they are K_k-minor-free, and that *cycles of blocks* are illegal instances, that is, they contain K_k as a minor. Then, we show by a counting argument that, if all the paths of blocks are accepted, and if the certificates used for these instances are too small, then there exists a cycle of blocks that is also accepted, which implies that the scheme is not correct.

Blocks and block connections. Let $p \geq 1$, and let $n = (k-1)(p+2)$. For every $r = 0, \ldots, p+1$, let us consider a path P_r of $k-1$ nodes with identifiers $r(k-1), r(k-1)+1, ..., (r+1)(k-1)-1$, in consecutive order. A *block* B_r is a graph formed of such a path P_r, plus edges between every pair of nodes in P_r. In other words, a block is a complete graph K_{k-1}, with an ordering of the nodes given by their identifiers. There are two special blocks: B_0, called the *starting block*, and B_{p+1}, called the *ending block*. The p other blocks are called *ordinary blocks*. Note that the identifiers of the blocks are all distinct.

Blocks are connected by *block connections*. A block connection from block B_i to block B_j is as follows. Place the block B_i on an horizontal line with the nodes in increasing order of their IDs, and then place the block B_j to the right of B_i, also with the nodes in increasing order of their IDs. Then, add all edges between the $\lceil (k-1)/2 \rceil$ rightmost nodes of B_i (which is called the *right part* of B_i) and the $\lfloor (k-1)/2 \rfloor$ leftmost nodes of B_j (which is called the *left part* of B_j). See Fig. 7 for an example with $k=4$, where the edges inside the blocks are grey, and the edges between the blocks are black (since $\lceil (k-1)/2 \rceil = 2$ and $\lfloor (k-1)/2 \rfloor = 1$, there are two edges between these blocks).

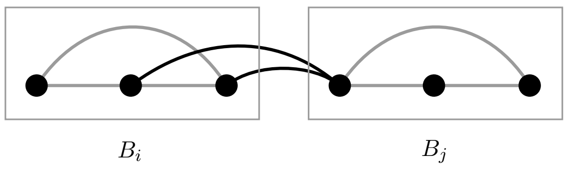

Figure 7: Block connexion for $k=4$, from a block B_i to a block B_j.

Note that two nodes that are linked by an edge have at most $k-3$ nodes between them in the horizontal ordering. In other words, no edge can "jump" over more than $k-3$ nodes. Also note that, if B_i has a block connection to B_j, and if B_j has a block connection to B_k, then, every node of B_j is connected either to a node of B_i, or to a node of B_k, but not both.

Path and cycles of blocks. Let π be a permutation on p elements. To create a path of blocks, B_0 has a block connection to $B_{\pi^{-1}(1)}$, $B_{\pi^{-1}(1)}$ has a block connection to $B_{\pi^{-1}(2)}$, etc., and $B_{\pi^{-1}(p)}$ has a block connection to B_{p+1}. A cycle of blocks is constructed in a similar way, as follows. Let k, k' with $1 \leq k < k' \leq p$. To create a cycle of blocks, $B_{\pi^{-1}(\ell)}$ has a connection to $B_{\pi^{-1}(\ell+1)}$ for every $\ell \in \{k, \ldots, k'-1\}$, and $B_{\pi^{-1}(k')}$ has a connection to $B_{\pi^{-1}(k)}$. Note that a cycle of blocks uses only a subset of blocks. Note also that, as the node identifiers given to different blocks are distinct, the paths and the cycles of blocks are well defined.

Claim 7 *Paths of blocks are K_k-minor-free.*

Proof. By symmetry, there is no loss of generality in proving the claim only for the identity permutation π. For simplicity of the notation, the nodes are referred to by their identifiers. Also, for the sake of clarity, we do not consider a path of blocks but an extension of it. That is, we add some edges, to make the path of blocks more symmetric, and hence avoid some case analysis. Note that, to prove the claim, it is sufficient to show that the larger graph G obtained by adding edges to the path of blocks, is K_k-minor-free.

For every $0 \leq r \leq p$, and every $0 \leq i \leq k-1$, the node $r(k-1)+i$ of B_r is connected in G to all nodes $(r+1)(k-1)+j$, $0 \leq j < i$, of B_{r+1}. See Figure 8 for an example. Note that, for every node ℓ, the closed neighborhood of ℓ (i.e., ℓ and its neighbors) is the interval $[\ell-k+2, \ell+k-2]$. It follows that, for every edge $\{\ell, \ell'\}$, $|\ell-\ell'| \leq k-2$.

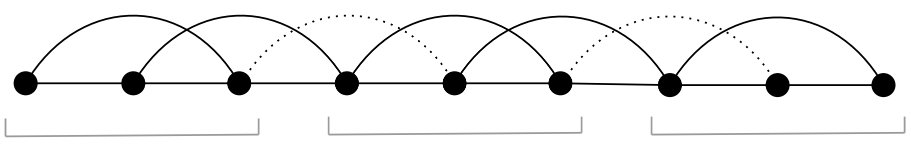

Figure 8: The graph G that extends the path of blocks with three blocks, for $k = 4$. The edges of the original path of blocks are plain edges and the new edges are dotted edges.

Let us assume, for the purpose of contradiction, that G contains K_k as minor. Then, there are k disjoint sets of nodes $S_1, \ldots, S_k$ such that each set S_i is connected, and, for every pair (S_i, S_j) of sets, $S_i \cup S_j$ is a connected subgraph of G. Each node belongs to at most one set S_i. Let us visit the nodes of the path of blocks, by increasing order of their IDs, and let S_a be the last visited set. Let ℓ_a be the first node belonging to S_a, i.e., the node with minimum ID in S_a. Let us now consider a set S_b such that $S_b \cap [\ell_a - k + 2, \ell_a] = \emptyset$. Such a set exists because there are k disjoint sets, and the interval is of size $k - 1$. Let ℓ_b be a node of S_b such that $\ell_b < \ell_a$. Such a node exists because, by definition of S_a, S_b is visited before S_a when nodes are traversed according to the increasing order of their IDs. The interval $I = [\ell_a - k + 2, \ell_a - 1]$ contains $k - 2$ integers, and $(S_a \cup S_b) \cap I$ is empty. Now, on the one hand, there cannot exist an edge $\{\ell, \ell'\}$ with $\ell < \ell_a - k + 2$, and $\ell' > \ell_a - 1$, because $|\ell - \ell'| \geq k - 1$. However, on the other hand, the sets

$$S_a \cap [\ell_a, (r + 1)(p + 2)] \text{ and } S_b \cap [0, \ell_a - k + 1]$$

are both non-empty, as they respectively contain ℓ_a and ℓ_b. It follows that $S_a \cup S_b$ is not connected, and therefore K_k cannot be a minor of the path of blocks. $\diamond$

Claim 8 *Cycles of blocks are* not *K_k-minor-free.*

Proof. Let us consider a cycle of blocks, and let B be an arbitrary block of this cycle. By construction, the graph obtained by removing B from the cycle is connected. Let us contract this graph into one node v. The node v is connected to all nodes of B since, as pointed out before, every node of B is linked to at least one node outside B. Now, as B is in itself K_{k-1}, the resulting contracted graph is K_k. $\diamond$

To complete the proof of Lemma 5, it is sufficient to show that cycles and paths of blocks are indistinguishable whenever the certificates are too small.

Let us assume that there exists a locally checkable proof with certificates of size $g(n) = o(\log n)$ bits. We show that there exist two paths of blocks for which the prover gives to every block the exact same certificates. For this purpose, let us call *labeled block* a block in which every node is given a certificate of size $g(n)$. Every block can be labeled in $2^{(k-1)g(n)}$ manners. Let us consider a set of labeled blocks where one labeled version of each block appears as element of this set. There are $2^{(k-1)g(n)p}$ such sets. On the other hand, there are $p!$ different paths of blocks, as there are $p!$ permutations of the ordinary blocks. To compare these numbers, it is simpler to compare their logarithms. We get the following asymptotics (having in mind that k is a constant, and $p = \Theta(n)$):

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

It follows that, for large n, there are more paths of blocks than distinct sets of labeled blocks. Thus, by the pigeon-hole principle, for large enough n, there exist two paths of blocks, P and P', where all nodes accept with a same certificate assignment, i.e., certificates inducing identical labeled blocks. Let A be this certificate assignment.

Without loss of generality, let us assume that P corresponds to the identity permutation. In the permutation π corresponding to P', there must exist two indices i and j with $i < j$ such that $\pi(i) > \pi(j)$. We define a cycle of blocks C as follows. The blocks $B_i, B_{i+1}, ..., B_j$ are connected in this order, and the cycle is closed by connecting B_j to B_i. We claim that every node in C accepts whenever it is given certificates according to A. Indeed, let u be any node of C, except the ones incident to an edge between B_j and B_i. Node u has the same view in C as in P (i.e., it has the same neighbors, with the same identifiers, and the same certificates), thus it accepts. Now, let us consider a node that is incident to an edge between B_j and B_i. As pointed out before, such a node is not linked to any node from a third block. It follows that it has the same view in C as in P'. Therefore, all the nodes of C accept, in contradiction with the soundness condition. This completes the proof of Lemma 5. □

Note that the proof of Lemma 5 can be adapted to a larger verification radius t, by replacing each edge by a path of length t.

4.2 Lower bound for $\mathrm{Forb}(K_{p,q})$

Our lower bound proof for $\mathrm{Forb}(K_{p,q})$ is an adaptation of the lower bound proof for spanning tree and leader election, by Göös and Suomela [29]. Roughly speaking, we show that, if the certificates are of size $o(\log n)$, then it is possible to define a set of legal instances (i.e., $K_{p,q}$-free graphs) which can be "glued" together in order to obtain an illegal instance (i.e., a graph containing $K_{p,q}$ as a minor) in such a way that no vertices can locally distinguish which instance they belong to.

Lemma 6 *For every $p, q \geq 2$, there are no locally checkable proofs for* $\mathrm{Forb}(\{K_{p,q}\})$ *using certificates on $o(\log n)$ bits.*

Proof. If $p = q = 2$, then $K_{p,q}$ is a 4-cycle, and thus $\mathrm{Forb}(\{K_{p,q}\})$ is the family of graphs in which the only allowed cycles are of length 3. The fact that this class of graphs cannot be certified by a locally checkable proof using certificates on $o(\log n)$ bits follows directly for previously known constructions (see [29]).

Let $p \geq 2$ and $q \geq 3$, and let us assume, for the purpose of contradiction, that $\mathrm{Forb}(\{K_{p,q}\})$ admits a locally checkable proof with certificates of size $o(\log n)$. Without loss of generality, we assume that $q \geq p$, and we consider $n \geq 6q$. The range of identifiers is split into subsets, as in [29]. Let $n_A = \lfloor n/2 \rfloor$, and $n_B = \lceil n/2 \rceil$. Let us fix any partition of the identifiers $\{1, \ldots, n^2\}$ into $2n$ subsets $a_1, \ldots, a_n, b_1, \ldots, b_n$ so that each $a_i \in A = \{a_1, \ldots, a_n\}$ is of size n_A, and each $b_i \in B = \{b_1, \ldots, b_n\}$ is of size n_B. We denote by $a_i[j]$ (resp. $b_i[j]$) the j-th identifier of $a_i \in A$ (resp. $b_i \in B$) in increasing order of the IDs in a_i (resp. b_i).

For every pair $(a, b) \in A \times B$, let $I_{a,b}$ be a legal instance, i.e., a $K_{p,q}$-minor-free graph G with a specific identity-assignment to its nodes, defined as follows. Let $d = \lfloor \frac{n}{2q} \rfloor$. The graph G is defined as two disjoint paths, one of length n_A, and another of length n_B. In $I_{a,b}$, the nodes in the former path are given IDs in a, in increasing order, while the nodes in the latter path are given IDs in b, also in increasing order. Also, for every $j \in \{1, \ldots, q\}$, there is an edge in G between the node with ID $a[jd]$ and the node with ID $b[jd]$. An illustration of this construction is displayed on Figure 9.

Observe that G is outerplanar, and thus $G \in \mathrm{Forb}(\{K_{2,3}\})$, from which it follows that $G \in \mathrm{Forb}(\{K_{p,q}\})$ for every $p \geq 2$ and $q \geq 3$.

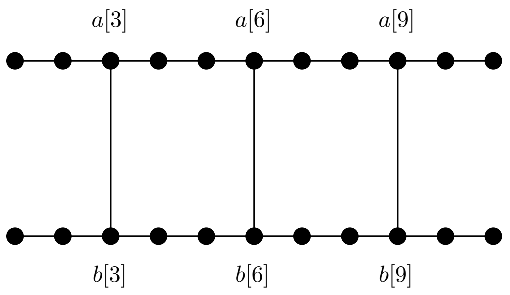

Figure 9: An instance $I_{a,b}$ for $n = 22$, $p = q = 3$, and $d = 3$.

For each node $v \in V(G)$, let $c_{a,b}(v)$ be the certificate provided to v in instance $I_{a,b}$ leading all nodes to accept, and define

$$c(a, b) = \big(c_{a,b}(a[d]), c_{a,b}(a[2d]), \dots, c_{a,b}(a[qd]), c_{a,b}(b[d]), c_{a,b}(b[2d]), \dots, c_{a,b}(a[qd])\big).$$

Since all certificates are of size $o(\log n)$, $c(a, b)$ forms a word on $o(\log n)$ bits. Now let $K_{n,n} = (A \cup B, E)$ be the complete bipartite graph with $E = \{\{a, b\} : a \in A, b \in B\}$, where every edge $\{a, b\} \in E$ is colored by $c(a, b)$. Pick n sufficiently large such that the number of bits of $c(a, b)$ is smaller than $\frac{\log(n)}{2q}$. Then, there exists a monochromatic subset F of edges of $K_{n,n}$ with

$$|F| > |E|/n^{\frac{1}{2q}} = n^{2-\frac{1}{2q}}.$$

It is known [25] that every graph with $n^{2-1/q} + o(n^{2-1/q})$ edges contains $K_{q,q}$ as subgraph. It follows that, for sufficiently large n, the graph $H = (A \cup B, F)$ contains $K_{q,q}$ as subgraph. Up to reindexing the elements in the sets A and B, the vertices of this subgraph are $a_1, \dots, a_q$ and $b_1, \dots, b_q$. Since all edges in H have the same color,

$$c(a_i, b_j) = c(a_{i'}, b_{j'}),$$

for every $(i, i', j, j') \in \{1, \dots, q\}^4$. We now create a new instance J, that roughly consists in gluing together the instances I_{a_i,b_j} with $1 \le i, j \le q$, for constructing an illegal instance. Let us consider q disjoint copies $P_1, \dots, P_q$ (resp., $Q_1, \dots, Q_q$) of a path P of length n_A (resp., of a path Q of length n_B), such that, in P_i (resp., Q_i), $i = 1, \dots, q$, the nodes are given the IDs in a_i (resp., b_i) in increasing order. For every $i, j \in \{1, \dots, q\}$, the nodes $a_i[jd]$ and $b_{i+j}[jd]$, where by $i + j$ is taken modulo q whenever greater than q, are connected by an edge. Figure 10 displays an example of the construction.

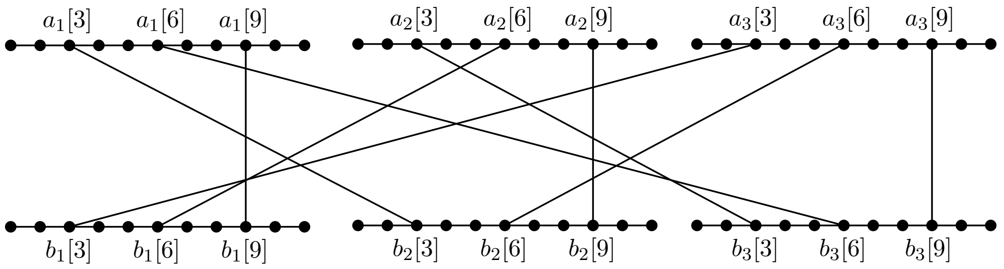

Figure 10: Example of the instance J for $p = q = 3$, and $d = 3$.

Observe that, by construction, if each of the q paths $P_1, \dots, P_q$, and each of the q paths $Q_1, \dots, Q_q$ is contracted into a single node, then the resulting graph is $K_{q,q}$. In other words, the underlying network in J contains $K_{q,q}$ as a minor, and therefore also $K_{p,q}$ as a minor. It follows that the instance J is illegal.

Let us consider the following certificate assignment c_J to the nodes in instance J. For every $i \in \{1, \ldots, q\}$, every node u of the path P_i (resp., Q_i) is given the certificate that it would receive in I_{a_i,b_i}. It follows from this assignment that, for every node u whose ID is different from $a_i[jd]$ and $b_i[jd]$), for all $i, j \in \{1, \ldots, q\}$, then the closed neighborhood of u in J with certificate assignment c_J is identical to the closed neighborhood of the same node u in I_{a_i,b_i} with certificates assignment c_{a_i,b_i}. It follows that such a node u accepts in J. Let u be a node of P_i with ID $a_i[jd]$) for some $i, j \in \{1, \ldots, q\}$. The closed neighborhood of u in J with certificate assignment c_J is identical to the view of the same node u in $I_{a_i,b_{i+j}}$ with certificates assignment $c_{a_i,b_{i+j}}$. It follows that such a node u accepts in J as well. The case of a node of Q_i with ID $b_i[jd]$) for some $i, j \in \{1, \ldots, q\}$ is identical, i.e., such a node accepts in J too. Therefore, with certificate assignment c_J, all nodes accept in J, and thus soundness is not satisfied. This contradicts the existence of a locally checkable proof for Forb($\{K_{p,q}\}$) with $o(\log n)$ bits, and completes the proof of Lemma 6. □

Again, the proof can be adapted to a larger verification radius t. The instances are formed by paths with identifiers only in a, or only in b, and edges having an endpoint in a, and an endpoint in b. We replace each edge of the former type (identifiers only in a, or only in b) by a path of length t. The same proof as for Lemma 6 applies after this change, in particular the legal instances are still outerplanar.

4.3 Proof of Theorem 2

Let $\mathcal{F} = \{K_k : k \geq 3\} \cup \{K_{p,q} : p, q \geq 2\}$, and $\mathcal{H}$ be a finite set of graphs in $\mathcal{F}$. If $\mathcal{H}$ contains a unique graph in $\mathcal{F}$, the fact that there are no locally checkable proofs for Forb($\mathcal{H}$) using certificates on $o(\log n)$ bits has been established in Lemma 5 and Lemma 6. We are left with the case of $|\mathcal{H}| \geq 2$.

Let us first make a few remarks about the family $\mathcal{H}$. If $\mathcal{H}$ contains two cliques K_k and $K_{k'}$ with $k' > k$, then we can remove $K_{k'}$, because K_k-minor-freeness implies $K_{k'}$-minor-freeness. Thus $\mathcal{H}$ contains at most one clique. Similarly, if $\mathcal{H}$ contains two complete bipartite graphs $K_{p,q}$ and $K_{p',q'}$, with $p' \geq p$ and $q' \geq q$, then we can safely remove $K_{p',q'}$ from the family. Also, if $\mathcal{H}$ contains K_3, then, Forb($\mathcal{H}$) is the class of trees, for which it is known that $o(\log n)$ bits are not sufficient for certification. Similarly, if $\mathcal{H}$ contains $K_{2,2}$ (and no K_3) then Forb($\mathcal{H}$) is the class of the graphs that do not contain cycles larger than 4, and, as noted in the proof of Lemma 6, this is also a case where $o(\log n)$ is not enough.

We are then left with the case of a family $\mathcal{H}$ included in $\{K_k : k \geq 4\} \cup \{K_{p,q} : p \geq 2, q \geq 3\}$, containing exactly one clique K_k, $k \geq 4$, and one or many complete bipartite graphs $K_{p,q}$, $p \geq 2, q \geq 3$. Consequently, the class of graphs Forb($\mathcal{H}$) contains the class Forb($\{K_4, K_{2,3}\}$). The latter is precisely the class of outerplanar graphs. Recall that in the construction of proof of Lemma 6 the legal instances are outerplanar, therefore these instances must be in Forb($\mathcal{H}$). The construction of the proof of Lemma 6 shows that if a scheme with $o(\log n)$ bits accepts all these instances, it also accepts an instance with $K_{p,q}$ as a minor, which is a contradiction. The result follows. □

5 Conclusion and further work

This paper provides a proof-labeling scheme for planarity, using certificates of optimal size $\Theta(\log n)$ bits, hence improving the previously known dMAM interactive protocol [38], by decreasing the number of interactions between the prover and the verifier, and avoiding the use of randomization.

This work could find extensions in many directions, fitting with the aforementioned interest of the distributed computing community for classes of sparse graphs beyond planar graphs (see [17]). For instance, the approach in this paper appears to be generalizable to the design of a proof-labeling scheme for the class of graphs with bounded genus $g > 1$, still using certificates on $O(\log n)$ bits, by cutting along so-called *nooses* until the resulting surface is planar. Certifying the correctness of such cuts, and the consistency of the embedding after these cuts requires to fix many technical details, but appear doable with certificates on $O(\log n)$ bits.

Perhaps more interesting is to extend our result to the certification of graph classes defined by a finite set of forbidden minors. The technique in this paper also enables to certify the class of outerplanar graphs with certificates on $O(\log n)$ bits, but the general case of classes defined by forbidden minors is open. A standard tool to consider in general for graphs excluding a fixed minor, that is, to certify $G \in \text{Forb}(H)$ for a fixed graph H, is the characterization of the topological embeddings for these graphs provided by Robertson and Seymour's Theorem. However, certifying the correctness of such embeddings seems challenging using certificates of small size. In fact, the minor-excluding graphs have a linear number of edges, and therefore they admit proof-labeling schemes using certificates on $O(n \log n)$ bits [29, 34]. Even the design of proof-labeling schemes for these graph classes with certificates on $O(n^\epsilon)$ bits, $\epsilon < 1$, appears challenging.

References

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

[2] Saeed Akhoondian Amiri, Patrice Ossona de Mendez, Roman Rabinovich, and Sebastian Siebertz. Distributed domination on graph classes of bounded expansion. In *30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 143–151, 2018.

[3] Saeed Akhoondian Amiri, Stefan Schmid, and Sebastian Siebertz. Distributed dominating set approximations beyond planar graphs. *ACM Trans. Algorithms*, 15(3):39:1–39:18, 2019.

[4] Baruch Awerbuch. A new distributed depth-first-search algorithm. *Inf. Process. Lett.*, 20(3):147–150, 1985.

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

[6] Alkida Balliu, Gianlorenzo D'Angelo, Pierre Fraigniaud, and Dennis Olivetti. What can be verified locally? In *34th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 8:1–8:13, 2017.

[7] Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Trans. Algorithms*, 15(2):21:1–21:38, 2019.

[8] Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In *33rd International Symposium on Distributed Computing (DISC)*, LIPIcs 146, pages 13:1–13:17. Dagstuhl, 2019.

[9] Andrzej Czygrinow, Michał Hańćkowiak, and Edyta Szymanska. Distributed approximation algorithms for planar graphs. In *6th Italian Conference on Algorithms and Complexity (CIAC)*, pages 296–307, 2006.

[10] Andrzej Czygrinow, Michał Hańćkowiak, Edyta Szymanska, Wojciech Wawrzyniak, and Marcin Witkowski. Distributed local approximation of the minimum k-tuple dominating set in planar graphs. In *18th Int. Conference on Principles of Distributed Systems (OPODIS)*, pages 49–59, 2014.

[11] Andrzej Czygrinow, Michał Hańćkowiak, Edyta Szymanska, Wojciech Wawrzyniak, and Marcin Witkowski. Improved distributed local approximation algorithm for minimum 2-dominating set in planar graphs. *Theor. Comput. Sci.*, 662:1–8, 2017.

[12] Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Distributed packing in planar graphs. In *20th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 55–61, 2008.

[13] Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *22nd Int. Symp. on Distributed Computing (DISC)*, pages 78–92, 2008.

[14] Hubert de Fraysseix and Pierre Rosenstiehl. A characterization of planar graphs by trémaux orders. *Combinatorica*, 5(2):127–135, 1985.

[15] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, 2000.

[16] Laurent Feuilloley. Introduction to local certification. *CoRR*, abs/1910.12747, 2019.

[17] Laurent Feuilloley. Bibliography of distributed approximation beyonf bounded degree. *CoRR*, abs/2001.08510, 2020.

[18] Laurent Feuilloley and Pierre Fraigniaud. Survey of distributed decision. *Bulletin of the EATCS*, 119, 2016.

[19] Laurent Feuilloley, Pierre Fraigniaud, and Juho Hirvonen. A hierarchy of local decision. In *43rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 118:1–118:15, 2016.

[20] Laurent Feuilloley, Pierre Fraigniaud, Juho Hirvonen, Ami Paz, and Mor Perry. Redundancy in distributed proofs. In *32nd International Symposium on Distributed Computing (DISC)*, LIPIcs, pages 24:1–24:18. Dagstuhl, 2018.

[21] Laurent Feuilloley and Juho Hirvonen. Local verification of global proofs. In *32nd Int. Symp. on Distributed Computing (DISC)*, LIPIcs 121, pages 25:1–25:17. Dagstuhl, 2018.

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

[23] Pierre Fraigniaud, Pedro Montealegre, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. On distributed Merlin-Arthur decision protocols. In *26th Int. Colloquium Structural Information and Communication Complexity (SIROCCO)*, LNCS 11639, pages 230–245. Springer, 2019.

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

[25] Zoltán Füredi. An upper bound on Zarankiewicz'problem. *Combinatorics, Probability and Computing*, 5(1):29–33, 1996.

[26] Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks I: planar embedding. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 29–38, 2016.

[27] Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, MST, and min-cut. In *27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 202–219, 2016.

[28] Mohsen Ghaffari and Merav Parter. Near-optimal distributed DFS in planar graphs. In *31st Int. Symp. on Distributed Computing (DISC)*, LIPIcs, pages 21:1–21:16. Dagstuhl, 2017.

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

[30] Miikka Hilke, Christoph Lenzen, and Jukka Suomela. Brief announcement: local approximability of minimum dominating set on planar graphs. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 344–346, 2014.

[31] John E. Hopcroft and Robert Endre Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

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

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

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

[35] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *23rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 300–309, 2004.

[36] Christoph Lenzen, Yvonne Anne Oswald, and Roger Wattenhofer. What can be approximated locally?: case study: dominating sets in planar graphs. In *20th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 46–54, 2008.

[37] Christoph Lenzen, Yvonne Anne Pignolet, and Roger Wattenhofer. Distributed minimum dominating set approximations in restricted families of graphs. *Distributed Computing*, 26(2):119–137, 2013.

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

[39] David Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

[40] David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM J. Comput.*, 30(5):1427–1442, 2000.

[41] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *45th ACM Symposium on Theory of Computing (STOC)*, pages 515–524, 2013.

[42] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

[43] William T. Tutte. Toward a theoryof crossing number. *J. Combinatorial Theory*, 8:45–53, 1970.

[44] Wojciech Wawrzyniak. A strengthened analysis of a local algorithm for the minimum dominating set problem in planar graphs. *Inf. Process. Lett.*, 114(3):94–98, 2014.