New Distributed Interactive Proofs for Planarity: A Matter of Left and Right

Yuval Gil[1]
Reykjavík University, Iceland

Merav Parter
Weizmann Institute of Science, Rehovot, Israel

Abstract

We provide new distributed interactive proofs (DIP) for planarity and related graph families. The notion of a *distributed interactive proof* (DIP) was introduced by Kol, Oshman, and Saxena (PODC 2018). In this setting, the verifier consists of n nodes connected by a communication graph G. The prover is a single entity that communicates with all nodes by short messages. The goal is to verify that the graph G satisfies a certain property (e.g., planarity) in a small number of rounds, and with a small communication bound, denoted as the *proof size*.

Prior work by Naor, Parter and Yogev (SODA 2020) presented a DIP for planarity that uses three interaction rounds and a proof size of $O(\log n)$. Feuilloley et al. (PODC 2020) showed that the same can be achieved with a single interaction round and without randomization, by providing a proof labeling scheme with a proof size of $O(\log n)$. In a subsequent work, Bousquet, Feuilloley, and Pierron (OPODIS 2021) achieved the same bound for related graph families such as outerplanarity, series-parallel graphs, and graphs of treewidth at most 2. In this work, we design new DIPs that use exponentially shorter proofs compared to the state-of-the-art bounds. Our main results are:

- There is a 5-round protocol with $O(\log \log n)$ proof size for outerplanarity.
- There is a 5-round protocol with $O(\log \log n)$ proof size for verifying embedded planarity and $O(\log \log n + \log \Delta)$ proof size for general planar graphs, where Δ is the maximum degree in the graph. In the former setting, it is assumed that an embedding of the graph is given (e.g., each node holds a clockwise orientation of its neighbors) and the goal is to verify that it is a valid planar embedding. The latter result should be compared with the non-interactive setting for which there is lower bound of $\Omega(\log n)$ bits for graphs with $\Delta = O(1)$ by Feuilloley et al. (PODC 2020).
- The non-interactive deterministic lower bound of $\Omega(\log n)$ bits by Feuilloley et al. (PODC 2020) can be extended to hold even if the verifier is randomized. Moreover, the lower bound holds even with the assumption that the verifier's randomness comes in the form of an unbounded random string *shared* among the nodes.

We also show that our DIPs can be extended to protocols with similar bounds for verifying series-parallel graphs and graphs with tree-width at most 2. Perhaps surprisingly, our results demonstrate that the key technical barrier for obtaining $o(\log \log n)$ labels for all our problems is a basic sorting verification task in which all nodes are embedded on an oriented path $P \subseteq G$ and it is desired for each node to distinguish between its left and right G-neighbors.

2012 ACM Subject Classification Theory of computation → Interactive proof systems; Theory of computation → Distributed algorithms

Keywords and phrases Distributed interactive proofs, Planar graphs

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.34

Related Version *Full Version*: https://arxiv.org/pdf/2505.00338 [14]

Funding This project is partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 949083.

[1] This work was done while the author was a postdoc at Weizmann.

1 Introduction

Planarity is a fundamental graph property that has been widely studied due to its rich combinatorial structure and numerous algorithmic applications. While in the centralized setting, the task of verifying if a given graph is planar can be done in linear time [18], in the distributed setting the running time depends linearly on the diameter of the graph [13]. The non-local nature of planarity motivates the use of a powerful, but potentially untrusted, *prover* that can aid the distributed verification by providing each node with a short auxiliary string, a.k.a. a *proof label*. The nodes then engage in brief communication to collectively determine whether to accept or reject the provided proof. This framework has been formalized into *proof labeling schemes* by Korman, Kutten and Peleg [20]. In this work we focus on the interactive extension of this model to *distributed interactive proofs* (DIP) as proposed by Kol, Oshman, and Saxena [19]. In this setting, the nodes are allowed to interact with the prover through multiple rounds of communication. The key complexity measures are the number of interaction rounds and the proof size.

The first evidence of the power of such proof systems for certifying planarity was provided by Naor, Parter, and Yogev [21]. Their result for planarity was in fact implied by a more general machinery that translates any (centralized) computation in $O(n)$ time – such as, the centralized planarity verification of [18] – into a three-round distributed interactive protocol with $O(\log n)$ proof size. Subsequent work by Feuilloley et al. [6, 7] demonstrated that the same proof size could be achieved with just a single interaction round, effectively reducing to the classical proof labeling scheme setting. Their work is accompanied by a matching lower bound of $\Omega(\log n)$ bits, that holds already for graphs with maximum degree of $O(1)$. These developments bring us back to the fundamental question of whether interaction truly provides an advantage in certifying planarity.

▶ **Question 1.1.** *What is the power of distributed interactive proofs for certifying planarity?*

We address this question by providing new DIPs for planarity and related graph families. Namely, we obtain constant-round protocols with a proof size of $O(\log \log n)$ for outerplanarity, embedded planarity, series-parallel graphs, and graphs of treewidth at most 2; and a proof size of $O(\log \log n + \log \Delta)$ for planarity in graphs of maximum degree Δ (we distinguish between embedded planarity in which we assume that a graph embedding is given in a distributed manner, and planarity in which no embedding is given; see full version [14] for formal definition). We also show that the $\Omega(\log n)$ lower bound of [6] can be extended to one-round DIPs. Therefore, our results give the first evidence to the advantage provided from interaction in planarity certification.

Model. In this paper, we consider *distributed interactive proofs (DIPs)* based on the model of [19].[2] In the DIP setting, instances are graphs $G = (V, E)$ taken from some universe $\mathcal{U}$ and the goal is to distinguish between *yes-instances* that come from a yes-family $\mathcal{F}_Y \subset \mathcal{U}$ and *no-instances* that come from a no-family $\mathcal{F}_N = \mathcal{U} - \mathcal{F}_Y$.[3] A DIP is an interactive protocol between a distributed *verifier* operating concurrently at all nodes of the graph and a centralized *prover* that can see the entire instance. The prover and verifier interact back and forth in *rounds*. Let $\mathcal{I}_{\texttt{vrf}}$ denote the rounds in which the verifier interacts with the prover and let $\mathcal{I}_{\texttt{prv}}$ denote the rounds in which the prover interacts with the verifier.

[2] We note that [19] uses the terms dAM and dMAM to denote the special cases of a DIP protocol with 2 and 3 rounds, respectively.

[3] One can easily adapt the setting so that instances also include some local information to the nodes (e.g., identifiers, weights, etc.). We chose to avoid this additional notation as the results of this paper apply to graphs without local node information.

Our protocols are *public-coin* which means that in each round $i \in \mathcal{I}_{\texttt{vrf}}$, the verifier at each node $v \in V$ interacts by drawing a random bitstring $\rho_v^i \in \{0,1\}^*$ and sending it to the prover (in particular, the verifier cannot hide any random bits from the prover). The prover interacts with the verifier in rounds $i \in \mathcal{I}_{\texttt{prv}}$ by sending a message $\mu_v^i \in \{0,1\}^*$ to each node $v \in V$. Keeping up with the terminology of [20], we sometimes refer to the messages sent by the prover as *labels*. The interaction ends with a round in which the prover interacts with the verifier, after which the verifier at each node $v \in V$ computes a local yes/no output based on: (1) the random bitstrings ρ_v^i drawn by v throughout the protocol; (2) the labels μ_v^i assigned to v by the prover throughout the protocol; and (3) the labels μ_u^i assigned to v's neighbors $u \in N(v)$ by the prover throughout the protocol. We say that the verifier *accepts* the instance if all nodes output "yes", and that the verifier *rejects* the instance if at least one node outputs "no".

As standard, the correctness of a proof system is defined by *completeness* and *soundness* requirements. The completeness requirements asks that if $G \in \mathcal{F}_Y$, then there exists an *honest* prover causing the verifier to accept the instance; whereas the soundness requirement asks that if $G \in \mathcal{F}_N$, then for any prover, the verifier rejects the instance. In the DIP setting, the correctness requirements are relaxed so that the completeness and soundness hold with probabilities $1-\epsilon_c$ and $1-\epsilon_s$, respectively, for some parameters $0 \leq \epsilon_c, \epsilon_s < 1/2$. In this case, we refer to ϵ_c as the *completeness error* and to ϵ_s as the *soundness error*. A protocol is said to have *perfect completeness* if $\epsilon_c = 0$. The performance of a DIP protocol is measured by the amount of prover-verifier communication it requires. Namely, the objective is to design protocols with a small number of interaction rounds and a small *proof size* which is defined as the size of the longest label assigned by the honest prover during the protocol.

The Challenge of Going Below the $\log n$ Barrier. As observed in [21], achieving sub-logarithmic proof lengths presents a significant challenge in the DIP setting and also serves as a lower bound for numerous problems in the non-interactive setting. This difficulty arises because many fundamental operations – such as identifying neighboring nodes, counting, or specifying node IDs – intrinsically require $\log n$ bits. While [21] made important initial progress in this area, their results apply to a more permissive variant of the DIP model, where nodes are allowed to send different messages to each of their neighbors. Indeed, their key technique is based on a rooted spanning tree provided by the prover such that every node identifies its tree-parent based on its internal port-numbering. Therefore, for each node to be able to learn its children in the tree (which is crucial to their protocols), every node has to send a distinct message to its parent.

In contrast, our work operates within the more restrictive DIP framework defined by Kol et al. [19], where nodes may only forward the proofs they receive to their neighbors. This constraint aligns with the non-interactive proof labeling model introduced in [20], where a node's decision is based solely on its own proof and those of its neighbors. This key difference in model assumptions becomes especially important in the sub-logarithmic setting, effectively preventing us from directly applying the techniques developed in [21].

Our Results. We present new distributed interactive proofs for various well-studied graph families. The first graph family considered is that of path-outerplanar graphs (see Section 2 for definition). Previously, [6] showed that path-outerplanarity admits a proof labeling scheme with a proof size of $O(\log n)$. We improve upon the communication complexity of that result by designing a protocol with exponentially shorter proof labels as specified in the following theorem.

▶ **Theorem 1.2.** *There exists a distributed interactive proof for path-outerplanarity running in* 5 *interaction rounds. The proof admits perfect completeness, a soundness error of* $1/\text{poly}\log n$*, and a proof size of* $O(\log\log n)$*.*

Building upon the path-outerplanarity protocol, we provide a protocol for (general) outerplanarity with the same asymptotic communication guarantees.

▶ **Theorem 1.3.** *There exists a distributed interactive proof for outerplanarity running in* 5 *interaction rounds. The proof admits perfect completeness, a soundness error of* $1/\text{poly}\log n$*, and a proof size of* $O(\log\log n)$*.*

We then move on to consider the case of planar graphs. In this context, we consider two verification tasks referred to as *planar embedding* and *planarity*. In the planar embedding task, an embedding of the graph is given in a distributed manner and the goal is to decide if it is a valid planar embedding (i.e., if no edges cross). In the planarity task, the goal is simply to decide if the given graph is planar. The details of our protocols for these tasks are given in the following two theorems.

▶ **Theorem 1.4.** *There exists a distributed interactive proof for planar embedding running in* 5 *interaction rounds. The proof admits perfect completeness, a soundness error of* $1/\text{poly}\log n$*, and a proof size of* $O(\log\log n)$*.*

▶ **Theorem 1.5.** *There exists a distributed interactive proof for planarity running in* 5 *interaction rounds. The proof admits perfect completeness, a soundness error of* $1/\text{poly}\log n$*, and a proof size of* $O(\log\log n + \log\Delta)$*.*

We also consider the two closely related graph families of series-parallel graphs and graphs of treewidth at most 2. We obtain the following two results.

▶ **Theorem 1.6.** *There exists a distributed interactive proof for series-parallel graphs running in* 5 *interaction rounds. The proof admits perfect completeness, a soundness error of* $1/\text{poly}\log n$*, and a proof size of* $O(\log\log n)$*.*

▶ **Theorem 1.7.** *There exists a distributed interactive proof for graphs of treewidth at most* 2 *running in* 5 *interaction rounds. The proof admits perfect completeness, a soundness error of* $1/\text{poly}\log n$*, and a proof size of* $O(\log\log n)$*.*

Finally, we provide the following lower bound.

▶ **Theorem 1.8.** *For each of the following graph families, any one-round distributed interactive proof with completeness and soundness errors smaller than* $1/10$ *requires a proof size of* $\Omega(\log n)$*: (1) path-outerplanar graphs; (2) outerplanar graphs; (3) embedded planar graphs; (4) planar graphs; (5) series-parallel graphs; and (6) graphs of treewidth at most* 2*;*

We note that Theorem 1.8 strengthens the lower bound presented in [6] in the following ways. First, the lower bound of [6] only applies to one-round proofs with *deterministic* verifier. Theorem 1.8 states that the same bound holds even if the verifier is *randomized*. Combined with the upper bounds stated above, our results present a strong evidence of the power added from *interaction* in the context of distributed proofs for planarity and related tasks. We remark that our lower bound holds even if the nodes have access to (unbounded) *shared* randomness. We also note that the lower bound of [6] does not explicitly apply to some of the graph families that appear in Theorem 1.8 (namely, path-outerplanar graphs, embedded planar graphs, and series-parallel graphs).

Open problems. Our results leave some intriguing unresolved questions that can be explored in follow-up works. Here, we highlight three of them.

As the main open problem, we ask whether the additive $O(\log \Delta)$ term is necessary in the proof size for planarity. That is, we pose the following question.

▶ **Open Question 1.** *Is it possible to obtain a constant round protocol for planarity with a proof size of $O(\log \log n)$ even on graphs with maximum degree $\Delta = \omega(\text{poly} \log n)$?*

One may also ask whether 5 interaction rounds are necessary in order to obtain a proof size of $O(\log \log n)$ for the tasks discussed in this paper. Of course, we know by Theorem 1.8 that 1 round is insufficient. However, for any $1 < r < 5$, whether an r-round protocol exists remains open even if we simply look for a proof size of $o(\log n)$. This leads to the following open problem.

▶ **Open Question 2.** *Is it possible to obtain an r-round protocol for e.g., outerplanarity, with a proof size of $o(\log n)$ for some $1 < r < 5$?*

Finally, we ask whether it is possible to improve our protocol's communication bound.

▶ **Open Question 3.** *Is it possible to obtain a protocol for e.g., outerplanarity, where the prover communicates $o(\log \log n)$ bits with each node?*

2 Preliminaries and Definitions

Conventions. Throughout, if not specified otherwise, a graph $G = (V, E)$ is assumed to be undirected and connected. For each node $v \in V$, we stick to the convention that $N_G(v)$ denotes the set of v's *neighbors* in the graph, $E(v)$ denotes the set of edges incident on v, and $\deg_G(v) = |N_G(v)| = |E(v)|$ denotes v's degree in G. Whenever G is clear from the context, we may omit it from the notation and write $N(v)$ and $\deg(v)$ instead of $N_G(v)$ and $\deg_G(v)$. For a node-subset $V' \subseteq V$, we denote by $G(V')$ the subgraph induced on G by V'.

As part of our technical tools, we also consider *directed graphs*. If G is directed, we assume that the edge orientation is given to the nodes such that each node $v \in V$ can distinguish between its incoming and outgoing incident edges. For a directed edge e with endpoints u and v, we write $e = (u, v)$ to reflect that e is directed from u to v, and $e = (v, u)$ otherwise. In the context of a distributed interactive proof, we assume that the label assigned by the prover to node $v \in V$ can be viewed by both its incoming and outgoing neighbors.

Hamiltonian paths. Consider a graph $G = (V, E)$ with a Hamiltonian path P. For a pair $u, v \in V$ of nodes, define the relation $\prec_P$ so that $u \prec_P v$ if u precedes v in P. Naturally, this extends to $u \preceq_P v$ if $u \prec_P v$ or $u = v$. Going forward, when P is clear from context, we may omit it from our notation and write $u \prec v$ and $u \preceq v$ instead of $u \prec_P v$ and $u \preceq_P v$, respectively. Whenever we encounter a Hamiltonian path, it will be convenient to think of it drawn as a straight line from left to right. Keeping up with this convention, for each node $v \in V$, we can partition its non-path edges in G into v-*left* edges which are incident on neighbors $u \prec v$, and v-*right* edges which are incident on neighbors $v \prec u$. We say that a non-path edge (u, v) is the *longest* v-left (resp., v-right) edge if $u \prec v$ (resp., $v \prec u$) and $u \prec u'$ (resp., $u' \prec u$) for every neighbor $u' \in N(v)$.

Left-right sorting. We define a verification task called *left-right sorting (LR-sorting)* which is used as a sub-task in our protocols. In LR-sorting, a directed graph $G = (V, E)$ is given. The graph G admits a directed Hamiltonian path P which is given such that each node

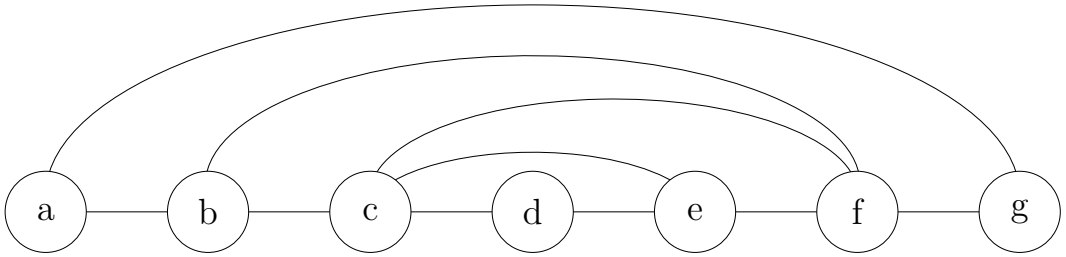

Figure 1 A path-outerplanar graph. The longest c-right edge is (c, f); the longest f-left edge is (b, f); the successor of (c, e) is (c, f).

$v \in V$ knows its incident edges in P. The path P is assumed to be directed from left to right. The goal of the task is to decide if $u \prec v$ for every directed edge $(u, v) \in E - P$. That is, a yes-instance is defined so that $u \prec v$ for every edge $(u, v) \in E$; whereas a no-instance admits at least one edge $(u, v) \in E$ such that $v \prec u$. Observe that equivalently, yes-instances are ones in which G is a DAG (in which case the P-ordering is the unique topological sort of G); and no-instances are ones in which G admits some cycle.

Path-outerplanar graphs. A graph $G = (V, E)$ is said to be *path-outerplanar* if it admits a Hamiltonian path P such that all non-path edges can be drawn above P without crossings. If the edges can be drawn in such a manner, we say that they are *properly nested* within P (or simply properly nested when P is clear from the context). Equivalently, a graph is path-outerplanar if no two edges $(u, v), (u', v') \in E$ satisfy $u \prec_P u' \prec_P v \prec_P v'$ with respect to some Hamiltonian path P (cf. [6]). Refer to Figure 1 for a pictorial example of a path-outerplanar graph and some of the related definitions.

The following simple observation will be useful in our protocol for path-outerplanar graphs in Section B.

▶ **Observation 2.1.** *Suppose that G is a path-outerplanar graph and let $(u, v) \in E$ be a non-path edge such that $u \prec v$. The edge (u, v) is either the longest u-right edge or the longest v-left edge.*

Proof. Assume that (u, v) is neither the longest u-right edge nor the longest v-left edge. Let (u, v') and (u', v) be the longest u-right and v-left edges, respectively. These edges satisfy $u' \prec u \prec v \prec v'$ which contradicts path-outerplanarity. ◀

Given a path-outerplanar graph G, we make the following definitions. For a non-path edge (u, v), $u \prec v$, define its *successor* as the edge (u', v') that satisfies: (1) $u' \preceq u \prec v \preceq v'$; and (2) $u'' \preceq u' \prec v' \preceq v''$ for every edge (u'', v'') that satisfies $u'' \preceq u \prec v \preceq v''$. Intuitively, the successor of an edge is the edge drawn directly above it. For cohesiveness, for any edge e that does not have a successor in the graph, define the successor to be a virtual edge $e^* = (u^*, v^*)$, $u^*, v^* \notin V$, defined so that $u^* \prec v \prec v^*$ for any $v \in V$. Notice that each edge has a unique successor. Naturally, we say that e is a *predecessor* of e' if e' is the successor of e. We say that two edges e and e' are *siblings* if they have a common successor.

The following observation is now straightforward from the definitions.

▶ **Observation 2.2.** *Suppose that G is a path-outerplanar graph and let $e = (u, v)$ be a (possibly virtual) non-path edge such that $u \prec v$. There exists an ordering $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$ of e's predecessors such that $u \preceq u_1 \prec v_1 \preceq u_2 \prec v_2 \cdots \preceq u_k \prec v_k \preceq v$.*

Encoding a spanning forest in a planar graph. As a building block in our protocol, we would like for the prover to be able to communicate a spanning forest F of the graph G to the verifier. While it is trivial to achieve in general using $O(\log n)$-bit labels, in our case we would like much smaller labels. It turns out that this task can be achieved in planar graphs deterministically and with constant-sized labels. This is done by slightly extending a construction of [3] which is designed for the task of deciding whether a planar graph admits a perfect matching.[4] We state the construction's properties in the following lemma.

▶ **Lemma 2.3.** *Let G be a planar graph and let F be a rooted spanning forest of G (i.e., F is a collection of rooted trees). For some constant $c > 0$, there exists a label assignment $L : V \to \{0,1\}^c$ such that each node $v \in V$ can learn its parent and children in F only as a function of $L(v)$, and the labels $L(u)$ assigned to v's neighbors $u \in N(v)$.*

For completeness of presentation, we provide a proof for the lemma. We emphasize that this construction only allows the prover to communicate the forest F to the verifier and does not provide proof that F is indeed a spanning forest.[5]

Proof. For ease of presentation, let us assume that the prover tries to communicate a spanning tree T (i.e., a connected forest). The case of an unconnected forest admits a similar construction. Suppose that $G = (V, E)$ is a planar graph and T is a spanning tree rooted at some node $r \in V$. For each node $v \in V - \{r\}$, let $\mathsf{parent}(v)$ denote its parent and let $\mathsf{depth}(v)$ denote its depth. Define the graph G_{odd} (resp., G_{even}) to be the graph obtained by starting from G and contracting all edges $(v, \mathsf{parent}(v))$ that go from an odd (resp., even) depth node v to its parent in T. Observe that G_{odd} and G_{even} are both planar and thus, 4-colorable. Towards providing the label assignment, the prover computes 4-colorings of G_{odd} and G_{even}, respectively. For each node $v \in V$, let $c_1(v)$ the color of the node into which v contracted in G_{odd} and let $c_2(v)$ be the color of the node into which v contracted in G_{even}. The prover assigns each node $v \in V$ with the label $L(v) = (c_1(v), c_2(v), \mathtt{parity}(v))$ where $\mathtt{parity}(v) = \mathsf{depth}(v) \bmod 2$.

We argue that the label assignment allows each node $v \in V$ to deduce which of its neighbors are its parent and children in T. The idea is as follows. If a node $v \in V$ of odd depth receives a color $c_1(v)$, then due to the validity of the coloring on G_{odd}, it holds that $\mathsf{parent}(v)$ is the only neighbor of v with even depth for which $c_1(\mathsf{parent}(v)) = c_1(v)$. The case of even depth nodes is similar.

To make things more concrete, consider some node $v \in V$ with $\mathtt{parity}(v) = 1$ (resp., $\mathtt{parity}(v) = 0$). Node v identifies its parent as its only neighbor $u \in N(v)$ with $\mathtt{parity}(u) = 0$ and $c_1(v) = c_1(u)$ (resp., $\mathtt{parity}(v) = 1$ and $c_2(v) = c_2(u)$). Additionally, v identifies its children as the neighbors $u \in N(v)$ that satisfy $\mathtt{parity}(u) = 0$ and $c_2(v) = c_2(u)$ (resp., $\mathtt{parity}(v) = 1$ and $c_1(v) = c_1(u)$). ◀

Enabling edge-labels in planar graphs. In the technical sections, it will be convenient to describe protocols assuming that the prover can also assign edge-labels (such that both of the edge endpoints can see the label) rather than only node-labels. This assumption is facilitated by the following lemma.[6]

[4] The scheme extends to some classes of non-planar graphs; see [3] for full details.

[5] Another way to formulate this construction is in terms of *advice*, based on the model of [9, 10]. Specifically, using the terminology of [10], the statement means that computing any spanning forest of a planar graph admits an $(O(1), 0)$-advising scheme.

[6] Transformations that enable edge-labels in planar graphs have been presented in previous papers (see, e.g., [6]). However, these constructions require the prover to assign an ordering to the nodes which

▶ **Lemma 2.4.** *Let Π be a class of planar graphs. Suppose that there exists a distributed interactive proof deciding whether $G \in \Pi$ in which the prover assigns labels of size ℓ to the nodes and edges. Then, there exists a distributed interactive proof in which the prover assigns labels of size $O(\ell)$ only to the nodes. Furthermore, the two proofs admit the same number of interaction rounds.*

Proof. It is well-known that planar graphs have arboricity at most 3. This means that the edge-set of any planar graph $G = (V, E)$ can be partitioned into three edge-disjoint forests F_1, F_2, F_3. By Lemma 2.3, the prover can inform each node $v \in V$ of its parent and children in each F_i using only constant-sized labels. Then, instead of assigning a label $L(u_i, v)$ to the edge (u_i, v) between v and its parent u_i in F_i, the prover simply writes $L(u_i, v)$ to a field in v's label which is designated for its parent in F_i. This allows both endpoints to learn the label $L(u_i, v)$, thus enabling the simulation of edge-labels. ◀

Spanning tree verification. Consider a graph $G = (V, E)$ and let T be a subgraph of G such that each node $v \in V$ knows its incident edges in T. We define *spanning tree verification* as the task of deciding whether T is a spanning tree of G. The following lemma is established in [21].

▶ **Lemma 2.5** ([21, Section 7.1])**.** *There is a distributed interactive proof for spanning tree verification with 3 interaction rounds and constant proof size. The proof admits perfect completeness and a constant soundness error.*

Observe that by standard parallel repetition, one can reduce the soundness error to $1/2^\ell$ at the expense of a $\Theta(\ell)$ proof size for any parameter $\ell > 0$. Throughout the paper, this fact will be used in a black-box manner.

Multiset equality. In the *multiset equality* problem, each node $v \in V$ receives as input two multisets $S_1(v), S_2(v)$ and the goal is to decide whether $S_1 = S_2$, where S_1 and S_2 are the multisets $S_1 = \bigcup_{v \in V} S_1(v)$ and $S_2 = \bigcup_{v \in V} S_2(v)$. Notice that in the definition of S_1 and S_2, the union is taken with respect to multisets, i.e., the multiplicity of element s in S_1 (resp., S_2) is the sum of its multiplicities over all multisets $S_1(v)$ (resp., $S_2(v)$). The multisets S_1 and S_2 are assumed to be of size at most k for some integer $k > 0$, and the elements are taken from a universe of size k^c for some constant $c \geq 1$. For our purposes, it would also be convenient to assume that the nodes are given a distributed encoding of a rooted spanning tree of the graph. The following lemma can be derived from the multiset equality protocol of [21].

▶ **Lemma 2.6** ([21])**.** *Given a multiset equality instance (G, S_1, S_2, k) such that $|S_1|, |S_2| \leq k$ and a rooted spanning tree T of G, there exists a 2-round distributed interactive proof for multiset equality. The proof admits perfect completeness, a soundness error of $1/k^c$, and a proof size of $O(\log k)$.*

Since the lemma above is not explicit in [21] and since we use the details of the multiset equality protocol in a white-box manner, we describe here the construction's details. The multiset equality protocol relies on the following idea. For a multiset S, define the polynomial $\varphi_S(x) = \prod_{s \in S}(s - x)$.[7] Now, observe that $S_1 = S_2$ if and only if $\varphi_{S_1} \equiv \varphi_{S_2}$. Moreover,

incurs an additive $\Theta(\log n)$ overhead to the label size. Thus, we cannot use these transformations for our purposes.

[7] Notice that we assume here w.l.o.g. that multiset elements are integers.

notice that the degree of $\varphi_{S_1}(x)$ and $\varphi_{S_2}(x)$ is at most k. Define p as the smallest prime number that satisfies $p > k^{c+1}$ and let z be a variable drawn uniformly at random from $\{0, \ldots, p-1\}$. By polynomial identity testing properties, if $\varphi_{S_1} \not\equiv \varphi_{S_2}$ and $\varphi_{S_1}(z), \varphi_{S_2}(z)$ are computed over the field $\mathbb{F}_p$, then $\Pr[\varphi_{S_1}(z) = \varphi_{S_2}(z)] \leq k/p \leq 1/k^c$.[8]

Following this idea, the multiset equality problem essentially reduces to evaluating a polynomial at a random point $z \in \{0, \ldots, p-1\}$. Recalling that we assume that a distributed encoding of a rooted spanning tree is provided to the nodes, the polynomial evaluation is implemented as follows. First, the point $z \in \{0, \ldots, p-1\}$ is sampled by the root and sent to the prover. Then, the prover assigns each node $v \in V$ with the value z and the values $\varphi_{S_1^v}(z), \varphi_{S_2^v}(z)$ (computed over $\mathbb{F}_p$) where S_1^v (resp., S_2^v) is the multiset of elements from S_1 (resp., S_2) in v's subtree. Given the assigned values, it is well-known that their validity can be checked at each node $v \in V$ based on its input, its label, and its children's labels. This is because polynomial evaluation is an aggregation task that can be verified "up the tree" (see, e.g., [20, Lemma 4.4] for details). Following these checks, the root r can check that $\varphi_{S_1^r}(z) = \varphi_{S_2^r}(z)$ (which implies that $\varphi_{S_1}(z) = \varphi_{S_2}(z)$).

To summarize, given a rooted spanning tree of the graph, the multiset equality protocol runs for 2 interaction rounds, admits perfect completeness, a soundness error of $1/k^c$, and a proof size of $O(\log p) = O(\log k)$.[9]

3 Technical Overview

In this section, we provide an overview of the techniques used to obtain the protocols presented in the paper. To exemplify the challenge of planarity certification with labels of size $O(\log \log n)$, let us first sketch a seemingly natural (yet unsuccessful) approach to which we refer as the *clustering* approach. Suppose that the prover computes a partition of the graph into node-disjoint connected clusters of size $\operatorname{poly} \log n$. Then, the prover provides a proof that: (1) the subgraph induced by each cluster is planar; and (2) the graph obtained by contracting all clusters is planar. Notice that in terms of *proof size* this approach is promising (and indeed, a similar approach was used in [21] to achieve sub-logarithmic proofs for various other problems). This is because one can use , e.g., the logarithmic proof for planarity of [6] on each cluster to obtain a proof of size $O(\log \log n)$ for (1). As for (2), since each cluster has size $\operatorname{poly} \log n$ and acts as a single node in the contracted graph, one can hope that it is possible to distribute the proof of a single cluster among the nodes within that cluster using only $O(\log \log n)$-sized labels per node. For the sake of this example, let us assume that it is indeed possible to obtain a proof for (2) with a proof size of $O(\log \log n)$.

It is not hard to see that the proof provided from the clustering approach is complete – if G is planar, then so are the subgraphs induced by the clusters as well as the graph obtained from contracting every cluster. The fundamental problem however, is that a proof for planarity obtained from this approach *cannot* be sound. To see why this is the case, consider a non-planar graph that contains a single 5-clique H.[10] A cheating prover can then define the partition such that, e.g., two nodes of H are assigned to one cluster and the other three are assigned to a different cluster. In this case, H does not violate planarity within any cluster and translates to a single edge in the contracted graph. Therefore, in this instance the verifier is likely to accept which violates the soundness requirement.

[8] Recall that $\mathbb{F}_p$ is the field whose elements are $\{0, \ldots, p-1\}$ and operations are done modulo p.
[9] Recall that standard density of primes properties assure that p cannot be too large. In particular, $p < k^{c+2}$, and thus $\log p = O(\log k)$.
[10] Recall that a graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor.

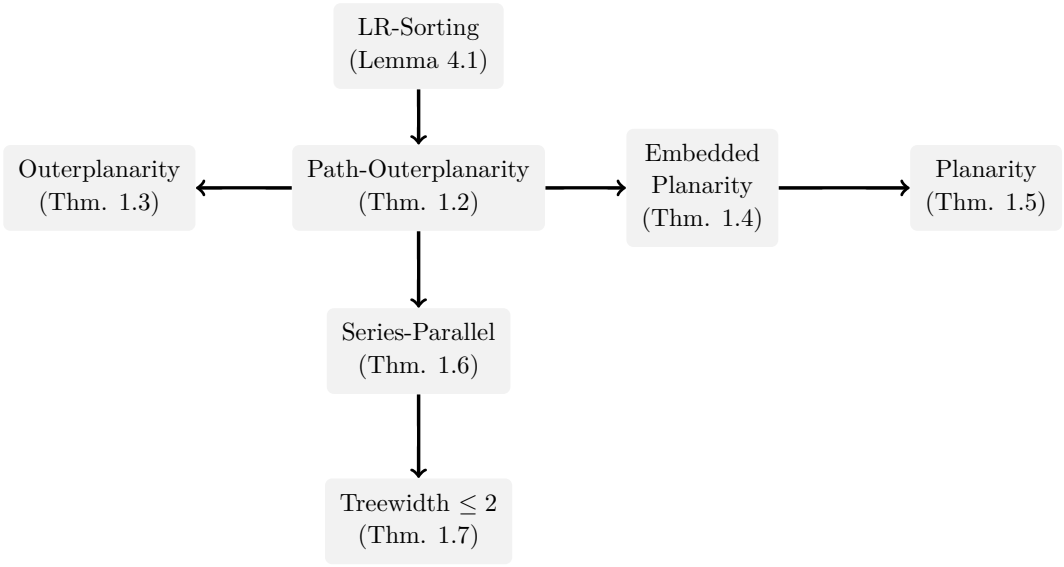

Figure 2 High-level description of the main results and their connections.

Notice that even if somehow we were able to prevent the prover from constructing an adversarial partition, it is possible to construct a no-instance in which the verifier is likely to accept for *any* partition. For example, it is possible to create an instance where each edge of H is subdivided such that its endpoints are at distance $\Omega(n)$ in G (and thus, are separated in any partition). Hence, we conclude that the clustering approach is doomed to fail for the planarity task.

The inherent failure of the clustering approach compels us to come up with a different approach. Similarly to the approach of [6], we seek to reduce planarity tasks to some other well-structured task for which we are able to design an efficient protocol. Perhaps surprisingly, we show that efficient protocols for all tasks considered in the current paper can be obtained based on a protocol for the seemingly unrelated task of LR-sorting. Indeed, starting from LR-sorting we present a sequence of reductions leading to new protocols for outerplanarity, planar embedding, planarity, series-parallel, and graphs of treewidth ≤ 2. Refer to Figure 2 for a chart depicting the dependencies between our different constructions. Notice that an advantage of LR-sorting is that unlike the planarity task, it can benefit from the clustering approach. As we explain below in more detail, this becomes useful in our protocol.

LR-sorting. To give an intuition for the LR-sorting protocol, let us first sketch a simple one-round protocol for LR-sorting with a proof size of $O(\log n)$. The prover assigns each node with its position on the path. Then, each node $v \in V$ that is located at the i-th position can verify that: (1) its path-neighbors are located at positions $i \pm 1$; and (2) all of its outgoing edges go towards nodes of larger positions.

To obtain a distributed interactive proof with a proof size of $O(\log \log n)$, the idea is to divide the nodes into node-disjoint *blocks* where each block is made up of $\lceil \log n \rceil$ consecutive path-nodes. This allows the prover to distribute the position of block b, denoted by $\texttt{pos}(b)$, such that each node receives: (1) its *index* $i \in [\lceil \log n \rceil]$ within b; and (2) $\texttt{pos}(b)[i]$, i.e., the i-th most significant bit of b's position. Ideally, as in the clustering approach, we would like for each block to act as a single node in the trivial protocol. In this short description, let us assume for simplicity that the prover encodes the position of the blocks correctly according

to the P-ordering and let $\texttt{pos}(b)$ denote the position of block b. The main challenge of the protocol is then captured by the following question: suppose that there is an edge (u, v) where u and v belong to different blocks b_u, b_v, how can the prover prove to u and v that $\texttt{pos}(b_u) < \texttt{pos}(b_v)$?

Towards answering this question, let us first consider the simpler case where (u, v) is the *only* edge leaving the block for both b_u and b_v. Furthermore, we will describe the protocol under the assumption that the prover can assign edge-labels. Recall that Lemma 2.4 implies that the edge-labels assumption can be simulated in planar graphs while incurring only a constant overhead to the proof size. Let i be the index of the most significant bit in which $\texttt{pos}(b_u)$ and $\texttt{pos}(b_v)$ differ. Notice that by definition, if $\texttt{pos}(b_u) < \texttt{pos}(b_v)$, then $\texttt{pos}(b_u)[i] = 0$ and $\texttt{pos}(b_v)[i] = 1$. In addition, i is in the range $[\lceil \log n \rceil]$ and thus, can be encoded using $O(\log \log n)$ bits. In the first interaction, the prover encodes i to the label of (u, v). The prover then needs to prove that (1) $\texttt{pos}(b_u)[i] = 0$; (2) $\texttt{pos}(b_v)[i] = 1$; and (3) $\texttt{pos}(b_u)$ and $\texttt{pos}(b_v)$ agree on their $i - 1$ most significant bits.

Let us focus on how (3) is proved. A straightforward way to obtain a proof for (3) is to assign the edge (u, v) with the substring containing the $i - 1$ most significant bits in their blocks. The main problem with this approach is that the proof size could be as large as $\Theta(\log n)$. To avoid this large label size, we can use polynomial identity testing. That is, we interpret the substrings containing the $i - 1$ most significant bits in $\texttt{pos}(b_u)$ and $\texttt{pos}(b_v)$ as polynomials of degree $O(\log n)$ and seek to have the prover and verifier evaluate them at a random point over a field of size $\operatorname{poly} \log n$. The prover then assigns the outcome of the polynomial evaluation to the label of (u, v). This introduces the following *locality* problem: the verification that the polynomials were computed correctly is done at the $(i - 1)$-th leftmost node in each block (using the standard aggregation technique of [20]). Since u and v might be far from the $(i - 1)$-th leftmost node in their respective blocks, they cannot locally detect that the prover is not lying about the outcome. We note that in the simplified case where (u, v) is the only edge leaving b_u and b_v, the problem is easy to solve. Indeed, since (u, v) is the only edge considered by blocks b_u and b_v, the prover can assign the outcome of the polynomial evaluation at the $(i - 1)$-th node to *all* nodes of b_u and b_v. The nodes can then check the correctness of this assignment locally based on the assignment to their block-neighbors.

Moving on to the general case where each block may have *many* edges leaving it, it is not clear how to solve the locality problem described above. Of course, if the prover assigns each node of the block with the outcomes of polynomial evaluations for every relevant index, this could require assigning $\Theta(\log n)$ values to every node which completely defeats the purpose. The main observation that we make is that one can formulate the locality problem as an instance of *multiset equality* between two multisets that are carefully defined within each block. Then, to check whether the multisets are indeed equal, a multiset equality protocol is executed within the blocks. An important property of the constructed multisets is that they are of size $\operatorname{poly} \log n$ which means that this final step can be done while maintaining a proof size of $O(\log \log n)$.

For the correctness, it turns out that the protocol's completeness becomes straightforward from the multisets construction, whereas the soundness argument requires a bit more care. Essentially, we show that in a no-instance the prover is likely to *commit* to a pair of unequal multisets within some block. Then, conditioning on this event, it is likely that the verifier rejects the instance due to the soundness of the multiset equality protocol.

From LR-sorting to path-outerplanarity. In Section B, we devise a protocol for path-outerplanarity. To get intuition of how the protocol works, we first recall the labels assigned in the non-interactive proof of [6]. Essentially, the prover assigns each node $v \in V$ with its position in the path P along with the position of the nodes u and u' for which $(u, u') \in E$ is the first edge that is drawn above v. The authors of [6] show that this labeling allows the (deterministic) verifier to verify that indeed G is path-outerplanar.

Of course, assigning the positions of nodes in P is far too costly for our purposes, so we seek to avoid it by using interaction and randomization. To that end, we first note that the assignment of positions in [6] serves the following purposes: (1) each node learns its P-neighbors; (2) each edge can be identified based on the positions of its endpoints; and (3) each node learns a clockwise orientation of its incident edges. Note that since P is a spanning tree, (1) can be obtained using only constant-sized labels based on Lemmas 2.3 and 2.5. For (2), we show that the edge identifiers can be replaced by random bits. As for (3), we design a protocol in which it is sufficient for every node to only distinguish between its left and right edges with respect to the path P. In fact, we show that under the assumption that every node knows its left and right edges, path-outerplanarity can be solved in 3 rounds and with a *constant* proof size (refer to Lemma B.1 for the full details of the reduction). To lift this assumption, we can apply our LR-sorting protocol which leads to the stated complexity bound of Theorem 1.2.

From path-outerplanarity to outerplanarity and planarity. We handle outerplanarity and planarity through reductions to path-outerplanarity. We note that while such reductions are presented in previous works ([6] for planarity; [2] for outerplanarity), we cannot use them as-is in our setting. This is because these reductions incur an additive $\Theta(\log n)$ overhead to the proof size. Nevertheless, our results rely on some modifications of the existing reductions. For outerplanarity, our reduction is white-box and it avoids the $\Theta(\log n)$ overhead based on the tools of Lemma 2.3 and Lemma 2.5 as well as some observations regarding the path-outerplanarity protocol.

For planarity, we start from the planar embedding task as an intermediate point. To reduce planar embedding to path-outerplanarity, we revisit some of the constructive proofs presented in [6] and show that they can be used to obtain such a reduction. Once again, Lemma 2.3 and Lemma 2.5 are used for the sake of efficient implementation. Then, we show that planarity reduces to planar embedding while incurring only an additive $O(\log \Delta)$ overhead to the proof size, thus obtaining the stated result.

Lower bounds. The starting point for our lower bounds is the lower bound of [6] which applies to proof labeling schemes (in fact, it holds also for the more general locally checkable proofs [16]). The result of [6] shows that an $\Omega(\log n)$ proof size is required even for the task of deciding whether a graph is outerplanar or non-planar. We start by adjusting the lower bound's details so that it would apply for the task of deciding whether a graph is *biconnected* outerplanar or non-planar.[11] This adjustment leads to a bound for all the graph families considered in the current paper since they are planar and contain all biconnected outerplanar graphs. To extend the lower bound to one-round protocols in which the verifier is randomized, we use a framework presented in [11].

[11] Recall that a graph is biconnected if the removal of any node leaves the resulting graph connected. A biconnected component of a graph G is a maximal biconnected subgraph of G.

4 LR-Sorting Protocol

In this section, we present a protocol for the task of LR-sorting on a given directed graph $G = (V, E)$ with Hamiltonian path P. The protocol is implemented under the assumption that the prover is able to assign labels to the nodes and the edges of G. If a label $L(u, v)$ is assigned to the edge $(u, v) \in E$, then both endpoints u and v can view it. The main result of the current section is the following lemma.

▶ **Lemma 4.1.** *There exists a distributed interactive proof for LR-sorting running in* 5 *interaction rounds. The proof admits perfect completeness, a soundness error of* $1/\text{poly}\log n$*, and a proof size of* $O(\log\log n)$ *where labels are assigned to both nodes and edges.*

Recall that by Lemma 2.4, if the given graph is planar, we can lift the edge-labels assumption of Lemma 4.1 to get the following.

▶ **Lemma 4.2.** *There exists a distributed interactive proof for LR-sorting in planar graphs running in* 5 *interaction rounds. The proof admits perfect completeness, a soundness error of* $1/\text{poly}\log n$*, and a proof size of* $O(\log\log n)$*.*

The rest of the section is dedicated to the protocol's description. Recall that our goal is to show how the prover proves that $u \prec v$ for every non-path edge (u, v) directed from u to v. This is described in two stages. First, a division of the path into node-disjoint *blocks* is described. Then, we explain how the block construction allows the nodes to compare their relative position on the path. For clarity, the stages are described without regard for the number of interaction rounds. Then, in Appendix A, we explain how the protocol can be implemented in 5 interaction rounds. Throughout, $c > 0$ is defined as a positive constant that can be made large enough to support the protocol's soundness guarantee.

4.1 The Block Construction

The block construction is defined so that the first block consists of the $\lceil \log n \rceil$ leftmost nodes in the path, the second block consists of the next $\lceil \log n \rceil$ nodes and so on. For ease of presentation, we assume that all blocks are of size exactly $\lceil \log n \rceil$. One can easily adjust the protocol's details to handle the general case in which (only) the rightmost block may have more than $\lceil \log n \rceil$ (but less than $2\lceil \log n \rceil$) nodes.

The purpose of the block construction is to allow the nodes to receive information regarding their position on the path. The position of a block b, denoted by $\texttt{pos}(b)$, is defined to be $i - 1$ if b is the i-th leftmost block. Notice that due to the block size, it is possible to encode an integer $x \in \{0, \ldots, n-1\}$ through the nodes of a block using only $O(\log\log n)$ bits per node. To do so, assign the j-th leftmost node of the block with the number j as well as the j-th most significant bit of x (leading zeros are added if necessary). Using this mechanism, the prover assigns the values $\texttt{pos}(b)$ and $\texttt{pos}(b) + 1$ to each block b.

In addition, the prover provides a proof that the two numbers assigned to each block are consecutive. To explain how this is done, suppose that x is a nonnegative integer in binary representation and let j be its least significant bit valued 0. Notice that x and $x + 1$ differ (only) in their j least significant bits. For a block b, define j_b to be the least significant bit in $\texttt{pos}(b)$ whose value is 0 and let v_b be the node associated with the index j_b. To prove that the numbers assigned to b are consecutive, the prover marks v_b and informs every other node in the block whether it is to the right/left of v_b.

To present the verification process at block b, let us denote by $x_1(b)$ and $x_2(b)$ the bitstrings assigned to b under the claim $x_1(b) + 1 = x_2(b)$. If a node $v \in b$ was labeled to be to the right of v_b, then it checks that its bit in $x_1(b)$ is 1, its bit in $x_2(b)$ is 0, and its right

neighbor in the block (if such neighbor exists) is also labeled as right of v_b; if v was marked as v_b, then it checks that its bit in $x_1(b)$ is 1, its bit in $x_2(b)$ is 0, its right neighbor is labeled as right of v_b, and its left neighbor is labeled as left of v_b; and if v is labeled as left of v_b, then it checks that it received the same bit in both bitstrings and that its left neighbor is labeled as left of v_b.

To complete the block construction stage, the verifier checks that the position assignment is consistent between adjacent blocks. Let b and b' be two adjacent blocks where b' is to the right of b. The verifier seeks to check that $\texttt{pos}(b) + 1 = \texttt{pos}(b')$. To that end, the multiset equality protocol is used between $x_2(b)$ and $x_1(b')$, where a bitstring is interpreted as the subset of $[\lceil \log n \rceil]$ that contains the indices whose bit is 1. Notice that the sets (and so, the degree of the multiset equality polynomials) are of size at most $\lceil \log n \rceil$. The verifier and prover run the multiset equality protocol over the field $\mathbb{F}_p$ where p is the smallest prime satisfying $p > \log^c n$. The polynomials are computed at a random point $r \in \{0, \ldots, p-1\}$ which is the same for all blocks. To that end, the variable r is sampled by the leftmost node in the path and passed to all nodes in the graph by the prover. Each block b computes (with the prover's assistance) the values of the two polynomials associated with its encoded bitstrings $x_1(b), x_2(b)$. This allows every pair of adjacent blocks to check that the positions assigned to them are indeed consecutive.

Correctness. If the position assignment given by the prover is valid, then $x_2(b) = x_1(b')$ for every pair of adjacent blocks b, b' and by the completeness of the multiset equality protocol, the verifier does not reject in this case. On the other hand, if the position assignment is not valid, then at least one pair b, b' of adjacent blocks satisfies $x_2(b) \neq x_1(b')$ and by the soundness of the multiset equality protocol, the verifier rejects with probability $1 - \lceil \log n \rceil / p = 1 - 1/\text{poly} \log n$.

Remark. An alternative approach to verifying the validity of the block construction is to use the RAM compiler of [21] concurrently on pairs of consecutive blocks. Nevertheless, the approach and notations presented above will be useful in the presentation of the next stage. We also note that it might be plausible to implement the block construction stage with proof size of $o(\log \log n)$, we avoided these optimizations as the key "communication bottleneck" lies in the next stage.

4.2 Comparing Relative Positions

We now describe how the prover uses the block construction to prove claims of the form $u \prec v$ for all non-path edges (u, v). To that end, we divide the edges into two types as follows. The *inner-block* edges are defined as the edges (u, v) in which u and v belong to the same block, and the *outer-block* edges are defined as the edges (u, v) in which u and v belong to different blocks.

Inner-block edges. Suppose that (u, v) is an inner-block edge. To show that $u \prec v$, the prover first assigns a bit to the edge (u, v) indicating that it is an inner-block edge. Let us denote the indices of u and v within their block by i_u and i_v, respectively (recall that these indices were assigned to the nodes during the block construction stage). The nodes u and v check that $i_u < i_v$ and if not, reject immediately. If $i_u < i_v$, then it is left to check that u and v are indeed on the same block. To that end, the leftmost node of each block b (i.e., the node associated with the most significant bit of $\texttt{pos}(b)$) samples a number $r_b \in [\log^c n]$ and sends it to the prover which in response, sends the value r_b to all nodes of the block b.

Each node checks that the number it received is consistent with its block neighbors and the leftmost node in the block checks that it received the same number it sampled. Then, for every edge (u, v) that was labeled as inner-block, u and v check that they both received the same r_b value and reject otherwise.

Correctness for inner-block edges. For completeness, observe that if $u \prec v$, then all checks succeed and the verifier accepts. For soundness, if $v \prec u$ and u and v are on the same block, then it must hold that $i_v < i_u$ and the verifier rejects. So, suppose that v and u are on different blocks $b \neq b'$ but the prover labels (u, v) as an inner-block edge. Then, the verifier rejects unless $r_b = r_{b'}$ which happens with probability $1/\text{poly}\log n$.

Outer-block edges. We complete the protocol's description by addressing the case of outer-block edges. Consider an outer-block edge (u, v), i.e., u and v belong to different blocks b_u and b_v, respectively. The prover's goal is to show that $\texttt{pos}(b_u) < \texttt{pos}(b_v)$. We divide the proof into two parts referred to as the *commitment* scheme and the *verification* scheme.

The main idea behind the commitment scheme relies on the following simple fact. Suppose that x and y are two nonnegative integers represented by binary strings of same length (leading zeros are added if necessary). Then, $x < y$ if and only if there exists an index i such that the $i-1$ most significant bits of x and y are identical, the i-th most significant bit of x is 0, and the i-th most significant bit of y is 1. We shall refer to this index as the (x, y)-*distinguishing* index and denote it by $I(x, y)$.

Consider a non-path edge (u, v) whose endpoints belong to different blocks b_u and b_v, respectively. The commitment scheme starts by having the prover write the value $I_{u,v} = I(\texttt{pos}(b_u), \texttt{pos}(b_v))$ to the label of edge (u, v). Then, for each block b, the multiset equality polynomial that is associated with $\texttt{pos}(b)$ is computed at a random point $r' \in \{0, \ldots, p-1\}$, where p is the prime number defined above. Similarly to the block construction stage, the computation is done over the finite field $\mathbb{F}_p$ and the variable r' is the same for all blocks. For an index $i \in [\lceil \log n \rceil]$, let $\texttt{pos}(b)[1, \ldots, i]$ denote the substring of $\texttt{pos}(b)$ consisting of its i most significant bits and let us denote by φ_i^b the multiset equality polynomial that is identified with the substring $\texttt{pos}(b)[1, \ldots, i]$. We note that $\varphi_i^b(r')$ is exactly the value computed at the i-th leftmost bit of block b. In addition to computing the multiset equality polynomial values within the blocks, the prover writes the value $\varphi^{b_u}_{I_{u,v}-1}(r')$ (which is equal to $\varphi^{b_v}_{I_{u,v}-1}(r')$ by the definition of the distinguishing index) to the label of each non-path edge (u, v).

For an edge $e = (u, v)$ which is classified as an outer-block edge, let $\rho(e) = (i, j)$ be the pair of values assigned by the prover on the label of e during the commitment scheme. That is, here i is claimed by the prover to be the distinguishing index between u and v's block positions, and j is claimed to be the multiset equality polynomial value computed at the $(i-1)$-th index of both blocks. To complete the commitment scheme, the verifier at each node $v \in V$ makes some consistency checks. First, if the same index i appears as the first element in two pairs $\rho(u, v)$ and $\rho(v, u')$ associated with edges $(u, v), (v, u')$, then the verifier rejects. To see why this condition is imposed, notice that $u \prec v$ requires that the i-th bit of v's block is 1, whereas $v \prec u'$ requires that the i-th bit of v's block is 0. For the second consistency check, the verifier checks that if two of v's incident edges agree on the first element of $\rho(\cdot)$ (and did not fail the first check), then they agree on the second element of $\rho(\cdot)$. For each node $v \in V$, let us define $C_0(v)$ (resp., $C_1(v)$) as the set of pairs $\rho(u, v)$ (resp., $\rho(v, u)$) assigned by the prover to the edges (u, v) (resp., (v, u)) during the commitment scheme. Notice that we define $C_0(v), C_1(v)$ as sets and not multisets. In particular, this means that $|C_0(v)| + |C_1(v)| \leq \lceil \log n \rceil$.

The purpose of the verification scheme is to verify the validity of the values in $C_0(v)$ and $C_1(v)$ for each node $v \in V$. Notice that this cannot be achieved locally in a trivial manner since the indices that appear in $C_0(v)$ and $C_1(v)$ may be associated with nodes on v's block that are not adjacent to v. We describe the verification of $C_1(v)$ values and then explain the small change required for the $C_0(v)$ verification. For a block b, define $C_1(b)$ as the *multiset* $C_1(b) = \bigcup_{v \in b} C_1(v)$. Define $F(b)$ as the set of indices whose bit in $\texttt{pos}(b)$ is equal to 1 and let $D_1(b) = \bigcup_{i \in F(b)} \{(i, \varphi^b_{i-1}(r'))\}$. The main idea behind the verification scheme is that in yes-instances, for each node $v \in b$ and pair $(i,j) \in C_1(v)$, it holds by construction that $(i,j) \in D_1(b)$. Thus, one can construct a multiset which is equal to $C_1(b)$ by taking every element of $D_1(b)$ with some multiplicity between 0 and $\lceil \log n \rceil$ (notice that it is not guaranteed that every element of $D_1(b)$ is in $C_1(b)$, so we allow a "multiplicity" of 0). Following this idea, the validity of $C_1(b)$ can be verified by means of another multiset equality protocol.

To make things more concrete, consider a node $v \in b$ which is associated with an index $i_v \in F(b)$. The prover provides v with a value $M_v \in \{0, \ldots, \lceil \log n \rceil\}$ that counts the number of times the pair $(i_v, \varphi^b_{i_v - 1}(r'))$ appears in $C_1(b)$. Then, the prover and verifier execute a multiset equality protocol to compare between $C_1(b)$ and the multiset obtained by taking M_v copies of the pair $(i_v, \varphi^b_{i_v-1}(r'))$ for each $i_v \in F(b)$. Here, notice that node v gets the value i_v from its own label and the value $\varphi^b_{i_v-1}(r')$ from the label of its left neighbor on the path. When computing the multiset equality polynomials, each pair $(i,j) \in [\lceil \log n \rceil] \times \{0, \ldots, p-1\}$ is mapped to an element from the set $[p \cdot \lceil \log n \rceil]$ by means of a fixed bijection known in advance to all nodes. To accommodate this range of field elements, it suffices to execute the multiset equality protocol over the field $\mathbb{F}_{p'}$ such that p' is the smallest prime that satisfies $p' > p \cdot \lceil \log n \rceil$. To verify the validity of $C_0(b) = \bigcup_{v \in b} C_0(v)$, we apply a similar idea with respect to the set $D_0(b) = \bigcup_{i \notin F(b)} \{(i, \varphi^b_{i-1}(r'))\}$. Observe that all the multisets that are involved in the equality protocols (and thus, the degrees of all polynomials) are of size $O(\log^2 n)$.

Correctness for outer-block edges. The completeness follows directly from the definition of the distinguishing index and the completeness of the multiset equality protocol. Regarding soundness, suppose that for some edge (u,v) directed from u to v, it holds that $v \prec u$. Denote by (i,j) the pair assigned to the edge (u,v) by the prover in the commitment scheme.

First, consider the case that u and v are in the same block b (but (u,v) is labeled as an outer-block edge by the prover). Notice that (i,j) can be in at most one of the sets $D_0(b), D_1(b)$. This is because $i \in F(b)$ implies $(i,j) \notin D_0(b)$ and $i \notin F(b)$ implies $(i,j) \notin D_1(b)$. Assume w.l.o.g. that $(i,j) \notin D_0(b)$. Notice that by construction $(i,j) \in C_0(b)$, which means that the compared multisets cannot be equal. Hence, by the soundness of the multiset equality protocol, the verifier rejects in this case with probability $1 - \log^2 n/(p \log n) = 1 - 1/\text{poly} \log n$.

Now, suppose that u and v are in different blocks $b_u \neq b_v$. Notice that by construction, $(i,j) \in C_0(b_u)$ and $(i,j) \in C_1(b_v)$. If $i \in F(b_u)$ or $i \notin F(b_v)$, then the soundness follows from a similar argument to the former case. Otherwise, by the definition of the distinguishing index and by the soundness of the multiset equality protocol, it follows that $\varphi^{b_u}_{i-1}(r') \neq \varphi^{b_v}_{i-1}(r')$ with probability $1 - 1/\text{poly} \log n$. If this is the case, then it must be that either $j \neq \varphi^{b_u}_{i-1}(r')$ or $j \neq \varphi^{b_v}_{i-1}(r')$. Let us condition on this event and assume w.l.o.g. that $j \neq \varphi^{b_u}_{i-1}(r')$. Then, $(i,j) \notin D_0(b_u)$ and since $(i,j) \in C_0(b_u)$, the soundness of the multiset equality protocol suggests that the verifier rejects with probability $1 - 1/\text{poly} \log n$.

References

1. Aviv Bick, Gillat Kol, and Rotem Oshman. Distributed zero-knowledge proofs over networks. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2426–2458. SIAM, 2022. doi:10.1137/1.9781611977073.97.
2. Nicolas Bousquet, Laurent Feuilloley, and Théo Pierron. Local certification of graph decompositions and applications to minor-free classes. *J. Parallel Distributed Comput.*, 193:104954, 2024. doi:10.1016/J.JPDC.2024.104954.
3. Nicolas Bousquet, Laurent Feuilloley, and Sébastien Zeitoun. Local certification of local properties: Tight bounds, trade-offs and new parameters. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand, France*, volume 289 of *LIPIcs*, pages 21:1–21:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.STACS.2024.21.
4. Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, volume 146 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.DISC.2019.13.
5. Louis Esperet and Benjamin Lévêque. Local certification of graphs on surfaces. *Theor. Comput. Sci.*, 909:68–75, 2022. doi:10.1016/J.TCS.2022.01.023.
6. Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and Ioan Todinca. Compact distributed certification of planar graphs. *Algorithmica*, 83(7):2215–2244, 2021. doi:10.1007/S00453-021-00823-W.
7. Laurent Feuilloley, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, Éric Rémila, and Ioan Todinca. Local certification of graphs with bounded genus. *Discret. Appl. Math.*, 325:9–36, 2023. doi:10.1016/J.DAM.2022.10.004.
8. Pierre Fraigniaud, François Le Gall, Harumichi Nishimura, and Ami Paz. Distributed quantum proofs for replicated data. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ITCS.2021.28.
9. Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. Communication algorithms with advice. *J. Comput. Syst. Sci.*, 76(3-4):222–232, 2010. doi:10.1016/J.JCSS.2009.07.002.
10. Pierre Fraigniaud, Amos Korman, and Emmanuelle Lebhar. Local MST computation with short advice. *Theory Comput. Syst.*, 47(4):920–933, 2010. doi:10.1007/S00224-010-9280-9.
11. Pierre Fraigniaud, Pedro Montealegre, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. On distributed merlin-arthur decision protocols. In Keren Censor-Hillel and Michele Flammini, editors, *Structural Information and Communication Complexity - 26th International Colloquium, SIROCCO 2019, L'Aquila, Italy, July 1-4, 2019, Proceedings*, volume 11639 of *Lecture Notes in Computer Science*, pages 230–245. Springer, 2019. doi:10.1007/978-3-030-24922-9_16.
12. François Le Gall, Masayuki Miyamoto, and Harumichi Nishimura. Distributed quantum interactive proofs. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 42:1–42:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.STACS.2023.42.
13. Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks I: planar embedding. In George Giakkoupis, editor, *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 29–38. ACM, 2016. doi:10.1145/2933057.2933109.
14. Yuval Gil and Merav Parter. New distributed interactive proofs for planarity: A matter of left and right, 2025. doi:10.48550/arXiv.2505.00338.

15 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. doi:10.1137/0218012.

16 Mika Göös and Jukka Suomela. Locally checkable proofs in distributed computing. *Theory Comput.*, 12(1):1–33, 2016. doi:10.4086/TOC.2016.V012A019.

17 Atsuya Hasegawa, Srijita Kundu, and Harumichi Nishimura. On the power of quantum distributed proofs. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024*, pages 220–230. ACM, 2024. doi:10.1145/3662158.3662788.

18 John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974. doi:10.1145/321850.321852.

19 Gillat Kol, Rotem Oshman, and Raghuvansh R Saxena. Interactive distributed proofs. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 255–264, 2018. URL: https://dl.acm.org/citation.cfm?id=3212771.

20 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Comput.*, 22(4):215–233, 2010. doi:10.1007/S00446-010-0095-3.

21 Moni Naor, Merav Parter, and Eylon Yogev. The power of distributed verifiers in interactive proofs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1096–115. SIAM, 2020. doi:10.1137/1.9781611975994.67.

Appendix

A Complexity of the LR-Sorting Protocol

For ease of presentation, our protocol is described in separate stages. Here, we observe that parts of the stages can be parallelized. First, we observe that the block construction stage can be implemented in three interaction rounds. Indeed, it starts with the prover encoding the block positions along with a proof that each block receives two consecutive numbers. Then, the verifier interacts with the prover to compute two multiset equality polynomials within each block. This can be done in two additional interaction rounds for a total of three. Similarly, the proofs of $u \prec v$ for inner-block edges (u, v), and the commitment scheme of outer-block edges can be completed within three rounds. Moreover, a correct execution of these steps does not depend on the execution of the block construction, thus they can be executed in parallel. We also note that the multiplicity values M_v that are presented in the verification stage of outer-block edges can actually be precomputed by the prover and assigned during the first interaction (they are placed in the verification scheme strictly for the sake of clear presentation). Therefore, after three interaction rounds, it is the verifier's turn to speak and the remaining task is the multiset equality protocol of the verification scheme of outer-block edges (here, notice that the verification scheme cannot be executed sooner as it depends on the values assigned in the commitment scheme). This takes two additional interaction rounds for a total of five rounds. Regarding proof size, a bound of $O(\log \log n)$ is straightforward from the construction.

B Path-outerplanarity

In this section, we present a protocol that uses LR-sorting as a sub-task to decide whether a given graph is path-outerplanar. The properties of the protocol are specified in the following lemma.

▶ **Lemma B.1.** *Suppose that there exists a distributed interactive proof for LR-sorting verification in planar graphs running in t interaction rounds. Let ℓ be the proof size, ϵ_c be the completeness error, and ϵ_s be the soundness error of the LR-sorting protocol. Then, there is a distributed interactive proof for path-outerplanarity running in $\max\{t, 3\}$ rounds and admitting a proof size of $O(\ell)$, a completeness error of ϵ_c, and a soundness error of $\epsilon_s + 2^{-\ell}$.*

As a consequence, we get Theorem 1.2.

Proof of Theorem 1.2. The protocol is obtained by plugging the LR-sorting protocol of Lemma 4.2 into the statement of Lemma B.1. ◀

The rest of the section is dedicated to the description of the protocol that proves Lemma B.1. For clarity, the protocol is described in separate stages without regard for the number of interaction rounds. Then, by the end of the section, we explain how the protocol can be implemented within the desired amount of interaction. Throughout, $c > 0$ is defined as a positive constant that can be made large enough to support the protocol's soundness guarantee.

Committing to a path. The protocol starts by having the prover commit to a Hamiltonian path P of G. To encode P, the prover uses the labels of Lemma 2.3 where P is rooted at the leftmost node in the path. Each node can verify that it has at most one child in the given tree encoding. Additionally, to verify that the given subgraph is indeed a Hamiltonian path of the graph, the prover and verifier execute the protocol of Lemma 2.5 amplified by means of a $c \cdot \ell$ parallel repetition.

Observe that if the graph is indeed path-outerplanar, then the prover can successfully send the verifier a Hamiltonian path. Consequently, each node knows its path-edges and is able to differentiate between its right and left neighbor on the path. On the other hand, if the graph is not path-outerplanar, then the graph is either not Hamiltonian or not outerplanar. In the former case, the prover is not able to provide a Hamiltonian path which causes the verifier to reject with probability $1 - 2^{-\Theta(\ell)}$; in the latter case, the prover is able to send the verifier a Hamiltonian path but the non-path edges are not properly nested.

LR-sorting. This stage starts by having the prover inform the verifier whether $u \prec v$ or $v \prec u$ for every edge $(u, v) \in E$. To see how this is achieved, recall that in the simulation of edge-labels that proves Lemma 2.4, e's label is written within the label of one of its endpoints. Let us refer to that endpoint as the endpoint *accountable* for e. So, if u is accountable for e, then the prover assigns the bit 1 to u's sub-label associated with e to signify that $u \prec v$, and 0 otherwise.

Following this assignment, the goal of the verifier is to check that all edges were labeled correctly by the prover, i.e., to check that if an edge (u, v) was labeled $u \prec v$, then indeed u appears before v in P. To that end, the prover and verifier execute an LR-sorting protocol. To create an instance for LR-sorting, the edges of the graph are oriented according to the prover's labeling. That is, if edge (u, v) was labeled $u \prec v$, then it is oriented from u to v.

Notice that if the verifier accepts the LR-sorting instance, then this means that the prover labeled all edges correctly (up to a soundness error of ϵ_s). So, for the rest of the protocol, we assume that for every non-path edge $e = (u, v)$, both endpoints know whether $u \prec v$ or $v \prec u$. Notice that in particular, this means that each $v \in V$ can distinguish between its left and right edges.

Nesting verification. In this final stage, the goal is to verify that the non-path edges are properly nested. The stage starts with the prover informing the endpoints of each non-path edge $e = (u, v)$, $u \prec v$, whether it is the longest u-right edge and whether it is the longest v-left edge. This is done by assigning two bits within the label of the endpoint accountable for e similarly to the previous stage.

Upon receiving the edge-labels, the verifier at each node $v \in V$ runs the following checks. If v has any right (resp., left) edges, then the verifier checks that exactly one of them is marked as longest v-right (resp., v-left) edge. In addition, for every right (resp., left) edge (v, u) that was not marked longest v-right (resp., v-left), the verifier checks that it was marked longest u-left (resp., u-right). If one of the checks fail, then the verifier immediately rejects. Otherwise, v samples a bitstring $s_v \in \{0,1\}^{c \cdot \ell}$ uniformly at random and sends it to the prover. For each non-path edge (u, v) such that $u \prec v$, define its *name* to be the pair (s_u, s_v).

After receiving the s_v values from all nodes, the prover assigns to each edge e its name through a sub-label $\texttt{name}(e)$ and its successor's name through a sub-label $\texttt{succ}(e)$ where the name of the virtual edge $e^* = (u^*, v^*)$ is defined by the designated symbol $\perp$ (recall that e^* is the successor of edges with no real successor in the graph). Additionally, if edge $e = (u, v)$ has predecessors $(u_1, v_1), \ldots, (u_k, v_k)$ such that $u \preceq u_1 \prec v_1 \preceq \cdots \preceq u_k \prec v_k \preceq v$, then the prover assigns the label $\texttt{above}(w) = \texttt{name}(e) = (s_u, s_v)$ to every node w such that $(u \prec w \preceq u_1) \vee (v_1 \preceq w \preceq u_2) \vee \cdots \vee (v_k \preceq w \prec v)$. In other words, the prover assigns e's name to all nodes for which e is the first edge drawn entirely above them (including the endpoints of e's predecessors; excluding the endpoints of e). In particular, if $e = (u, v)$ does not have any predecessors, then $\texttt{above}(w) = \texttt{name}(e)$ for all nodes w such that $u \prec w \prec v$. Observe that by definition, each node is associated with only one such edge and thus, receives only one edge name.

Consider a label assignment to the nodes and non-path edges. First, for each non-path edge e, its endpoints verify that $\texttt{name}(e)$ is consistent with their sampled values. Then, each node $v \in V$ checks that there exists an ordering $e_1^+, \ldots, e_k^+$ of its right edges, and an ordering $e_1^-, \ldots, e_{k'}^-$ of its left edges such that the following conditions are satisfied:

1. e_k^+ and $e_{k'}^-$ are marked as the longest v-right and v-left edges, respectively.
2. $\texttt{succ}(e_i^+) = \texttt{name}(e_{i+1}^+)$ for all $1 \leq i < k$, and $\texttt{succ}(e_i^-) = \texttt{name}(e_{i+1}^-)$ for all $1 \leq i < k'$.
3. $\texttt{above}(v) = \texttt{succ}(e_k^+) = \texttt{succ}(e_{k'}^-)$.
4. if u is v's right neighbor on the path, then $\texttt{name}(e_1^+) = \texttt{above}(u)$ if the set of v's right edges is non-empty, and $\texttt{above}(v) = \texttt{above}(u)$ otherwise.
5. if u is v's left neighbor on the path, then $\texttt{name}(e_1^-) = \texttt{above}(u)$ if the set of v's left edges is non-empty, and $\texttt{above}(v) = \texttt{above}(u)$ otherwise.

We note that a pair of orderings that satisfies the described conditions does not have to be unique. Also, notice that nodes which are not incident on any non-path edges only need to check that they were assigned the same value as their neighbors on the path (conditions (4) and (5)). This concludes the description of the nesting verification. We go on to establish its correctness.

Correctness of nesting verification. Towards proving the completeness and soundness of the nesting verification, we show the following two observations.

▶ **Observation B.2.** *Fix some node $u \in V$. If the prover marks the longest u-right or the longest u-left edge incorrectly, then the verifier rejects the instance with probability $1 - 2^{-c \cdot \ell}$.*

Proof. Suppose that edge (u, v) is the longest u-right edge but not marked as such. Recall that by the initial verification conditions, (u, v) must be marked as the longest v-left edge (otherwise the verifier rejects). If v has a right edge, then by verification conditions (1)

and (3), the value $\texttt{succ}(u,v)$ should be identical to the value $\texttt{succ}(v,w)$, where (v,w) is the right edge of v which is marked as longest. If v does not have a right edge, then by verification conditions (3) and (4), the value $\texttt{succ}(u,v)$ should be identical to the value the value $\texttt{above}(w')$ where w' is v's right neighbor on the path. In either case, following the verification conditions we get that $\texttt{succ}(u,v)$ should be identical to $\texttt{name}(u',v')$ of some edge (u',v') such that $v \prec v'$. Moreover, the edge (u',v') is fully determined by the marking of longest left and right edges by the prover (and in particular, determined before the sampling of names). Note that since (u,v) is the longest u-right edge and $v \prec v'$, it must hold that $(u,v') \notin E$ and thus, $u' \neq u$. On the other hand, since (u,v) is not marked as the longest u-right edge, by condition (2), the first element of $\texttt{succ}(u,v)$ should be s_u. So, the verifier rejects unless $s_u = s_{u'}$ which happens with probability $2^{-c\cdot\ell}$. The case of longest left edges follows a similar reasoning. ◀

Going forward with the correctness proof, we shall assume that all longest left/right edges are marked correctly. For two nodes $u \prec v$, denote by $P_{u,v}$ the set of nodes on the (u,v)-subpath in G.

▶ **Observation B.3.** *Suppose that for a non-path edge (u,v), it holds that $G(P_{u,v})$ is path-outerplanar w.r.t. $P_{u,v}$ (i.e., the edges of $G(P_{u,v})$ are properly nested within $P_{u,v}$). If the verifier accepts the instance, then $\texttt{succ}(u',v')$ is the name of the successor of (u',v') in $G(P_{u,v})$ for all non-path edges $(u',v') \neq (u,v)$, $u \preceq u' \prec v' \preceq v$.*

Proof. Let (x,y) be a non-path edge in $G(P_{u,v})$ and let (x_ℓ,y_ℓ) and (x_r,y_r) be its leftmost and rightmost predecessors, respectively. First, if $y_r = y$, then it must hold that $x \prec x_r$ which means that $(x_r,y_r) = (x_r,y)$ is not the longest y-left edge. Therefore, by condition (2) it must hold that the second element of $\texttt{succ}(x_r,y_r)$ is s_y. Now, suppose that $y_r \prec y$. Here, since (x_r,y_r) is a predecessor of (x,y), it follows that (x_r,y_r) is the longest y_r-left edge. Applying conditions (1), (3), and (4) along the (y_r,y)-path, we once again get that the second element of $\texttt{succ}(x_r,y_r)$ must be s_y. For similar reasoning, we can deduce that the first element of $\texttt{succ}(x_\ell,y_\ell)$ is s_x. Now, we observe that by conditions (1), (3), (4), and (5), every pair of adjacent siblings must have the same $\texttt{succ}(\cdot)$ field. Therefore, every predecessor (x',y') of (x,y) must satisfy $\texttt{succ}(x',y') = (s_x,s_y) = \texttt{name}(x,y)$ which concludes our proof. ◀

We can now prove the completeness and soundness of our protocol.

▶ **Lemma B.4.** *The described nesting verification admits perfect completeness and a soundness error of $2^{-\Theta(\ell)}$.*

Proof. We start from completeness. First, we note that by Observation 2.1, the honest prover can mark each edge (u,v) as longest u-right/v-left correctly. Furthermore, observe that the feasibility of the $\texttt{name}(\cdot)$, $\texttt{succ}(\cdot)$, and $\texttt{above}(\cdot)$ labels assigned by the honest prover is guaranteed by Observation 2.2. Now, consider some node $v \in V$ and let $e^-_{k'} = (v,u^-_{k'}),\ldots,e^-_1 = (v,u^-_1), e^+_1 = (v,u^+_1),\ldots,e^+_k = (v,u^+_k)$ be its incident non-path edges such that $u^-_{k'} \prec \cdots \prec u^-_1 \prec v \prec u^+_1 \prec \cdots \prec u^+_k$. Given the labels assigned by the honest prover, the orderings $e^-_1,\ldots,e^-_{k'}$ and $e^+_1,\ldots,e^+_k$ defined on the left and right edges of v satisfy all the verification conditions, thus causing the verifier to accept.

We now establish the soundness guarantee. Let us define (u,v) as an edge that admits a crossing edge (u',v') such that $u \prec u' \prec v \prec v'$ but not a crossing edge (u',v') such that $u' \prec u \prec v' \prec v$. That is, (u,v) is not crossed by edges that has an endpoint to the left of u. Moreover, assume that (u,v) is the deepest nested such edge, i.e., every edge $(x,y) \neq (u,v)$

where $u \preceq x \prec y \preceq v$ does not admit a crossing edge. Observe that if there exists a pair of crossing edges in the graph, then there exists an edge (u, v) satisfying the assumptions stated above.

We start from the case where (u, v) is the longest v-left edge. Define $u \prec u' \prec v$ to be the rightmost node incident on a right edge that crosses (u, v) and define (u', v') as the longest u'-right edge (by definition, (u', v') crosses (u, v)). Let $e_1 = (u_1, v), e_2 = (u_2, v), \ldots, e_k = (u_k, v)$ be v's left edges ordered such that $u = u_k \prec \cdots \prec u_2 \prec u_1 \prec v$. Observe that by the assumptions on (u, v), it follows that $u' \preceq u_{k-1}$. Moreover, all edges that are drawn below e_{k-1} are properly nested. Therefore, Observation B.3 implies that if the verifier accepts the instance, then $\texttt{succ}(e_i) = \texttt{name}(e_{i+1})$ for every $1 \leq i < k - 1$. Furthermore, recall that (u, v) is marked as the longest v-left edge. Thus, for condition (2) to be satisfied at node v, it must also hold that $\texttt{succ}(e_{k-1}) = \texttt{name}(e_k) = (s_u, s_v)$.

To show that the verifier is likely to reject in this case, the idea is to define a sequence of edges that must agree with e_{k-1} on their $\texttt{succ}(\cdot)$ value, but also must have $s_{u'}$ as their $\texttt{succ}(\cdot)$ value's first element. This implies that the verifier rejects unless $s_u = s_{u'}$ which happens with probability $1/\text{poly}\log n$. The sequence $(x_1, y_1), \ldots, (x_t, y_t)$ of edges is defined as follows. Start by taking $x_1 \preceq u_{k-1}$ to be the closest node to u_{k-1} incident on a left edge and set (x_1, y_1) as the longest x_1-left edge. Then, take $x_2 \preceq y_1$ to be the closest node to y_1 incident on a left edge and set (x_2, y_2) as the longest x_2-left edge. Continue this process until reaching y_t such that all nodes w such that $u' \prec w \preceq y_t$ are not incident on a left edge. Notice that the sequence construction is feasible since by our assumption on (u, v), no edge within (u, v) crosses (u', v') (and thus, the sequence is entirely to the right of u'). Moreover, since u' is the rightmost node incident on a right edge crossing (u, v), it follows that every edge (x_i, y_i) in the sequence is the longest y_i-right edge. We note that the verification conditions dictate that every pair of adjacent edges in the sequence should have the same $\texttt{succ}(\cdot)$ value and that $\texttt{succ}(x_1, y_1) = \texttt{succ}(e_{k-1})$. On the other hand, for u' to satisfy condition (4), the first element of $\texttt{succ}(x_t, y_t)$ must be $s_{u'}$ which concludes the soundness for this case.

We move on to the case where (u, v) is not the longest v-left edge. If (u, v) is also not the longest u-right edge, then by the construction the verifier rejects. So, assume that (u, v) is the longest u-right edge. Let $u \prec u' \prec v$ be the leftmost node incident on an edge crossing (u, v) and let (u', v') be the longest u'-right edge (by definition, (u', v') crosses (u, v)). By similar measures to the previous case, it is possible to find a sequence $(x_1, y_1), \ldots, (x_t, y_t)$ of edges such that must satisfy $\texttt{succ}(x_1, y_1) = \cdots = \texttt{succ}(x_t, y_t) = \texttt{succ}(u', v')$; and the first element of $\texttt{succ}(x_i, y_i)$ is s_u for each $1 \leq i \leq t$. On the other hand, by similar reasoning to the one presented in the proof of Observation B.2, it must hold that $\texttt{succ}(u', v') = \texttt{name}(w, w')$ for some edge (w, w') such that $v' \preceq w'$. Moreover, this edge is fully determined by the marking of longest left and right edges by the prover (and in particular before the sampling of names). Recall that $v \prec v' \preceq w'$ and that (u, v) is the longest u-right edge. Therefore, it must hold that $w \neq u$ which means that the probability of $s_u = s_w$ is at most $2^{-c \cdot \ell}$. ◀

Analysis of the protocol. By construction, the proof size of the protocol is $O(\ell)$. Moreover, all stages apart from the black-box use of the LR-sorting protocol admit perfect completeness and a soundness error of $2^{-\Theta(\ell)}$. Thus, by union bound arguments, the completeness error of the protocol is ϵ_c and the soundness error is $\epsilon_s + 2^{-\Theta(\ell)}$. Of course, taking a sufficiently large c, we can have a soundness error of $\epsilon_s + 2^{-\ell}$ as desired. Finally, regarding the number of interaction rounds, we note that all stages can be executed in parallel without affecting the correctness of the algorithm. It is straightforward to see that the stages committing to a path and nesting can be implemented in 3 interaction rounds. Since the LR-sorting protocol requires t rounds, we get that in total the protocol runs in $\max\{t, 3\}$ rounds.

C Additional Related Work

Beyond planarity. Following the introduction of efficient distributed proof systems for planarity [21, 6], researchers have become interested in distributed proof systems for other graph families. The aforementioned compiler of [21] implies a three-round distributed interactive protocol with $O(\log n)$ proof size for families of sparse graphs (i.e., $m = O(n)$ edges) that admit a linear-time recognition algorithm. These include, e.g., *bounded genus* graphs and *outerplanar* graphs. Distributed proofs for bounded genus graphs were studied further in [5, 7] where proof labeling schemes with a proof size of $O(\log n)$ are presented. For outerplanar graphs, a proof labeling scheme with a proof size of $O(\log n)$ is presented in [2]. Additionally, the authors show similar results for a myriad of minor-free graphs.

Distributed interactive proofs variants. In [4], trade-offs between different parameters of the DIP model are explored. The parameters considered include the form of randomness, the complexity measures, and the number of interaction rounds. Recently, the notion of *distributed quantum interactive proofs* was introduces by the authors of [12] as a quantum variant of distributed interactive proofs. The main result of [12] is a generic transformation from a k-round "standard" proof into a 5-round quantum proof for any constant $k > 5$. Distributed quantum proofs have also been considered in a non-interactive setting in [8, 17]. Another exciting variant that was introduced recently in [1] is that of a *distributed zero-knowledge proof*. In particular, the authors adapt the classical notion of knowledge from the centralized setting (as defined in [15]) to a distributed setting.