Redundancy in Distributed Proofs

Laurent Feuilloley, Pierre Fraigniaud, Juho Hirvonen, Ami Paz, Mor Perry

Redundancy in Distributed Proofs

Laurent Feuilloley
IRIF, CNRS and University Paris Diderot, France
feuilloley@irif.fr

Pierre Fraigniaud
IRIF, CNRS and University Paris Diderot, France
pierref@irif.fr

Juho Hirvonen
University of Freiburg, Germany
juho.hirvonen@cs.uni-freiburg.de

Ami Paz
IRIF, CNRS and University Paris Diderot, France
amipaz@irif.fr

Mor Perry
School of Electrical Engineering, Tel-Aviv University, Israel
mor@eng.tau.ac.il

Abstract

Distributed proofs are mechanisms enabling the nodes of a network to collectively and efficiently check the correctness of Boolean predicates on the structure of the network (e.g. having a specific diameter), or on data structures distributed over the nodes (e.g. a spanning tree). We consider well known mechanisms consisting of two components: a *prover* that assigns a *certificate* to each node, and a distributed algorithm called *verifier* that is in charge of verifying the distributed proof formed by the collection of all certificates. We show that many network predicates have distributed proofs offering a high level of redundancy, explicitly or implicitly. We use this remarkable property of distributed proofs to establish perfect tradeoffs between the *size of the certificate* stored at every node, and the *number of rounds* of the verification protocol.

2012 ACM Subject Classification D.1.3 Concurrent Programming (Distributed programming); F.2.2 Nonnumerical Algorithms and Problems.

Keywords and phrases Distributed verification, Distributed graph algorithms, Proof-labeling schemes, Space-time tradeoffs, Non-determinism

Digital Object Identifier 10.4230/LIPIcs.DISC.2018.24

Related Version https://arxiv.org/abs/1803.03031

Funding Research supported by the French-Israeli Laboratory on Foundations of Computer Science (FILOFOCS). The first four authors supported by the ANR project DESCARTES. The first two authors receive additional support from INRIA project GANG. Third author supported by the Ulla Tuominen Foundation. Fourth author supported by the Fondation Sciences Mathématiques de Paris (FSMP)

32nd International Symposium on Distributed Computing (DISC 2018).
Editors: Ulrich Schmid and Josef Widder; Article No. 24; pp. 24:1–24:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

1.1 Context and Objective

In the context of distributed fault-tolerant computing in large scale networks, it is of the utmost importance that the computing nodes can perpetually check the correctness of distributed data structures (e.g., spanning trees) encoded distributedly over the network. Indeed, such data structures can be the outcome of an algorithm that might be subject to failures, or be a-priori correctly given data structures but subject to later corruption. Several mechanisms exist enabling checking the correctness of distributed data structures (see, e.g., [2,3,6,10–12]). For its simplicity and versatility, we shall focus on one classical mechanism known as *proof-labeling schemes* [31], a.k.a. *locally checkable proofs* [25][1].

Roughly, a proof-labeling scheme assigns *certificates* to each node of the network. These certificates can be viewed as forming a distributed proof of the actual data structure (e.g., for a spanning tree, the identity of a root, and the distance to this root in the tree). The nodes are then in charge of collectively verifying the correctness of this proof. The requirements are in a way similar to those imposed on non-deterministic algorithms (e.g., the class NP), namely: (1) on correct structures, the assigned certificates must be accepted, in the sense that every node must accept its given certificate; (2) on corrupted structures, whatever certificates are given to the nodes, they must be rejected, in the sense that at least one node must reject its given certificate. (The rejecting node(s) can raise an alarm, or launch a recovery procedure). Proof-labeling schemes and locally checkable proofs can be viewed as a form of non-deterministic distributed computing (see also [19]).

The main measure of quality for a proof-labeling scheme is the *size* of the certificates assigned to *correct* (a.k.a. *legal*) data structures. Indeed, these certificates are verified using protocols that exchange them between neighboring nodes. Thus using large certificates may result in significant overheads in term of communication. Also, proof-labeling schemes might be combined with other mechanisms enforcing fault-tolerance, including replication. Large certificates may prevent replication, or at the least result in significant overheads in term of space complexity if using replication.

Proof-labeling schemes are extremely versatile, in the sense that they can be used to certify *any* distributed data structure or graph property. For instance, to certify a spanning tree, there are several proof-labeling schemes, each using certificates of logarithmic size [26,31]. Similarly, certifying a minimum-weight spanning tree (MST) can be achieved with certificates of size $\Theta(\log^2 n)$ bits in n-node networks [29,31]. Moreover, proof-labeling schemes are very *local*, in the sense that the verification procedure performs in just one round of communication, each node accepting or rejecting based solely on its certificate and the certificates of its neighbors. However, this versatility and locality comes with a cost. For instance, certifying rather simple graph property, such as certifying that each node holds the value of the diameter of the network, requires certificates of $\widetilde{\Omega}(n)$ bits [13][2]. There are properties that require even larger certificates. For instance, certifying that the network is non 3-colorable, or certifying that the network has a non-trivial automorphism both require certificates of $\widetilde{\Omega}(n^2)$ bits [25]. The good news though is that all distributed data structures (and graph properties) can be certified using certificates of $O(n^2 + kn)$ bits, where k is the

[1] These two mechanisms slightly differ: the latter assumes that every node can have access to the whole state of each of its neighbors, while the former assumes that only part of this state is visible from neighboring nodes; nevertheless, the two mechanisms share the same essential features.

[2] The tilde-notation is similar to the big-O notation, but also ignores poly-logarithmic factors.

size of the part of the data structure stored at each node – see [25, 31].

Several attempts have been made to make proof-labeling schemes more efficient. For instance, it was shown in [9] that randomization helps a lot in terms of *communication* costs, typically by hashing the certificates, but this might actually come at the price of dramatically increasing the certificate size. Sophisticated deterministic and efficient solutions have also been provided for reducing the size of the certificates, but they are targeting specific structures only, such as MST [30]. Another direction for reducing the size of the certificates consists of relaxing the decision mechanism, by allowing each node to output more than just a single bit (accept or reject) [4, 5]. For instance, certifying cycle-freeness simply requires certificates of $O(1)$ bits with just 2-bit output, while certifying cycle-freeness requires certificates of $\Omega(\log n)$ bits with 1-bit output [31]. However, this relaxation assumes the existence of a centralized entity gathering the outputs from the nodes, and there are still network predicates that require certificates of $\widetilde{\Omega}(n^2)$ bits even under this relaxation. Another notable approach is using approximation [13], which reduces, e.g., the certificate size for certifying the diameter of the graph from $\Omega(n)$ down to $O(\log n)$, but at the cost of only determining if the given value is up to two times the real diameter.

In this paper, we aim at designing deterministic and generic ways for reducing the certificate size of proof-labeling schemes. This is achieved by following the guidelines of [33], that is, trading time for space by exploiting the inherent redundancy in distributed proofs.

1.2 Our Results

As mentioned above, proof-labeling schemes include a verification procedure consisting of a single round of communication. In a nutshell, we prove that using more rounds of communication for verifying the certificates enables to reduce significantly the size of these certificates, often by a factor super-linear in the number of rounds, and sometimes even exponential.

More specifically, a proof-labeling scheme of radius t (where t can depend on the size of the input graph) is a proof-labeling scheme where the verification procedure performs t rounds, instead of just one round as in classical proof-labeling schemes. We may expect that proof-labeling schemes of radius t should help reduce the size of the certificates. This expectation is based on the intuition that the verification of classical (radius-1) proof-labeling schemes is done by comparing certificates of neighboring nodes or computing some function of them, and accept only if they are consistent with one another (in a sense that depends on the scheme). If the certificates are poorly correlated, then allowing more rounds for the verification should not be of much help as, with a k-bit certificate per node, the global proof has kn bits in total in n-node graphs, leaving little freedom for reorganizing the assignment of these kn bits to the n nodes. Perhaps surprisingly, we show that distributed proofs do not only involve partially redundant certificates, but inherently *highly redundant certificates*, which enables reducing their size a lot when more rounds are allowed. To capture this phenomenon, we say that a proof-labeling scheme *scales* with scaling factor $f(t)$ if its size can be reduced by a factor $\Omega(f(t))$ when using a t-round verification procedure; we say that the scheme *weakly* scales with scaling factor $f(t)$ if the scaling factor is $\widetilde{\Omega}(f(t))$, i.e., $\Omega(f(t)/\text{polylog}\, n)$ in n-node networks.

We prove that, in trees and other graph classes including e.g. grids, *all* proof-labeling schemes scale, with scaling factor t for t-round verification procedures. In other words, for every boolean predicate $\mathcal{P}$ on labeled trees (that is, trees whose every node is assigned a label, i.e., a binary string), if $\mathcal{P}$ has a proof-labeling scheme with certificates of k bits, for some $k \geq 0$, then $\mathcal{P}$ has a proof-labeling scheme of radius t with certificates of $O(k/t)$ bits, for all $t \geq 1$.

In addition, we prove that, in any graph, uniform parts of proof-labeling schemes weakly scale optimally. That is, for every boolean predicate $\mathcal{P}$ on labeled graphs, if $\mathcal{P}$ has a proof-labeling scheme such that k bits are identical in all certificates, then the part with these k bits weakly scales in an optimal manner: it can be reduced into $\widetilde{O}(k/b(t))$ bits by using a proof-labeling scheme of radius t, where $b(t)$ denotes the size of the smallest ball of radius t in the actual graph. Therefore, in graphs whose neighborhoods increase polynomially, or even exponentially with their radius, the benefit in terms of space-complexity of using a proof-labeling scheme with radius t can be enormous. This result is of particular interest for the so-called *universal* proof-labeling scheme, in which every node is given the full n^2-bit adjacency matrix of the graph as part of its certificate, along with the $O(\log n)$-bit index of that node in the matrix.

We complement these general results by a collection of concrete results, regarding scaling classical boolean predicates on labeled graphs, including spanning tree, minimum-weight spanning tree, diameter, and additive spanners. For each of these predicates we prove tight upper and lower bounds on the certificate size of proof-labeling schemes of radius t on general graphs.

1.3 Our Techniques

Our proof-labeling schemes demonstrate that if we allow t rounds of verification, it is enough to keep only a small portion of the certificates, while all the rest are redundant. In a path, it is enough to keep only two consecutive certificates out of every t: two nodes with $t-2$ missing certificates between them can try all the possible assignments for the missing certificates, and accept only if such an assignment exists. This reduces the *average* certificate size; to reduce the *maximal* size, we split the remaining certificates equally among the certificate-less nodes. This idea is extended to trees and grids, and is at the heart of the proof-labeling schemes presented in Section 3.

On general graphs, we cannot omit certificates from some nodes and let the others check all the options for missing certificates in a similar manner. This is because, for our approach to apply, the parts of missing certificates must be isolated by nodes with certificates. However, if all the certificates are essentially the same, as in the case of the universal scheme, we can simply keep each part of the certificate at some random node[3], making sure that each node has all parts in its t-radius neighborhood. A similar, yet more involved idea, applies when the certificates are distances, e.g., when the predicate to check is the diameter, and the (optimal) certificate of a node contains in a distance-1 proof-labeling scheme its distances to all other nodes. While the certificates are not universal in this latter case, we show that it still suffices to randomly keep parts of the distances, such that on each path between two nodes, the distance between two certificates kept is at most t. These ideas are applied in Sections 4 and 5.

In order to prove lower bounds on the certificate size of proof-labeling schemes and on their scaling, we combine several known techniques in an innovative way. A classic lower bound technique for proof-labeling schemes is called *crossing*, but this cannot be used for lower bounds higher than logarithmic, and is not suitable for our model. A more powerful technique is the use of nondeterministic communication complexity [13, 25], which extends the technique used for the CONGEST model [1, 23]. In these bounds, the nodes are

[3] All our proof-labeling schemes are deterministic, but we use the probabilistic method for proving the existence of some of them.

partitioned between two players, who simulate the verification procedure in order to solve a communication complexity problem, and communicate whenever a message is sent over the edges of the cut between their nodes. When proving lower bounds for proof-labeling schemes, the nondeterminism is used to define the certificates: a nondeterministic string for a communication complexity problem can be understood as a certificate, and, when the players simulate verification on a graph, they interpret their nondeterministic strings as node certificates. However, this technique does not seem to be powerful enough to prove lower bounds for our model of multiple rounds verification. When splitting the nodes between the two players, the first round of verification only depends on the certificates of the nodes touching the cut, but arguing about the other verification rounds seems much harder. To overcome this problem, we use a different style of simulation argument, where the node partition is not fixed but evolves over time [14, 36]. More specifically, while there are sets of nodes which are simulated explicitly by either of the two players during the t rounds, the nodes in the paths connecting these sets are simulated in a decremental manner: both players start by simulating all these nodes, and then simulate less and less nodes as time passes. After the players communicate the certificates of the nodes along the paths at the beginning, they can simulate the verification process without any further communication. In this way, we are able to adapt some techniques used for the CONGEST model to our model, even though proof-labeling schemes are a computing model that is much more similar to the LOCAL model [35].

1.4 Previous Work

The mechanism considered in this paper for certifying distributed data structures and predicates on labeled graphs has at least three variants. The original *proof-labeling schemes*, as defined in [31], assume that nodes exchange solely their certificates between neighbors during the verification procedure. Instead, the variant called *locally checkable proofs* [25] imposes no restrictions on the type of information that can be exchanged between neighbors during the verification procedure. In fact, they can exchange their full individual states, which makes the design of lower bounds far more complex. This latter model is the one actually considered in this paper. There is a third variant, called *non-deterministic local decision* [19], which prevents using the actual identities of the nodes in the certificates. That is, the certificate must be oblivious to the actual identity assignment to the nodes. This latter mechanism is weaker than proof-labeling schemes and locally checkable proofs, as there are graph predicates that cannot be certified in this manner. However, all predicates on labeled graphs can be certified by allowing randomization [19], or by allowing just one alternation of quantifiers (the analog of Π_2 in the polynomial hierarchy) [7]. A distributed variant of the centralized interactive proofs was recently introduced by Kol et al. [27].

Our work was inspired by [33], which aims at reducing the size of the certificates by trading time for space, i.e., allowing the verification procedure to take t rounds, for a non-constant t, in order to reduce the certificate size. They show a trade-off of this kind for example for proving the acyclicity of the input graph. The results in [30] were another source of inspiration, as it is shown that, by allowing $O(\log^2 n)$ rounds of communication, one can verify MST using certificates of $O(\log n)$ bits. In fact, [30] even describe an entire (non-silent) self-stabilizing algorithm for MST construction based on this mechanism for verifying MST.

In [17], the authors generalized the study of the class log-LCP introduced in [25], consisting of network properties verifiable with certificates of $O(\log n)$ bits, to a whole local hierarchy inspired by the polynomial hierarchy. For instance, it is shown that MST is at the second level of that hierarchy, and that there are network properties outside the hierarchy. In [34], the effect of sending different messages to different neighbors on the communication complexity of verification is analyzed. The impact of the number of errors on the ability to detect the illegality of a data structure w.r.t. a given predicate is studied in [16]. The notion of approximate proof-labeling schemes was investigated in [13], and the impact of randomization on communication complexity of verification has been studied in [9].

Finally, verification mechanisms a la proof-labeling schemes were used in other contexts, including the congested clique [28], wait-free computing [21], failure detectors [22], anonymous networks [18], and mobile computing [8, 20]. For more references to work related to distributed verification, or distributed decision in general, see the survey [15]. To our knowledge, in addition to the aforementioned works [30, 33], there is no prior work where verification time and certificate size are traded.

2 Model and Notations

A labeled graph is a pair (G, x) where $G = (V, E)$ is a connected simple graph, and $x : V \to \{0,1\}^*$ is a function assigning a bit-string, called *label*, to every node of G. When discussing a weighted n-nodes graph G, we assume $G = (V, E, w)$, where $w : E \to [1, n^c]$ for a fixed $c \geq 1$, and so $w(e)$ can be encoded on $O(\log n)$ bits. An identity-assignment to a graph G is an assignment $\mathrm{ID} : V \to [1, n^c]$, for some fixed $c \geq 1$, of distinct identities to the nodes.

A distributed decision algorithm is an algorithm in which every node outputs accept or reject. We say that such an algorithm accepts if and only if every node outputs accept.

Given a finite collection $\mathcal{G}$ of labeled graphs, we consider a boolean predicate $\mathcal{P}$ on every labeled graph in $\mathcal{G}$ (which may even depend on the identities assigned to the nodes). For instance, AUT is the predicate on graphs stating that there exists a non-trivial automorphism in the graph. Similarly, for any weighted graph with identity-assignment ID, the predicate MST on (G, x, ID) states whether $x(v) = \mathrm{ID}(v')$ for some $v' \in N[v]$[4] for every $v \in V(G)$, and whether the collection of edges $\{\{v, x(v)\}, v \in V(G)\}$ forms a minimum-weight spanning tree of G. A proof-labeling scheme for a predicate $\mathcal{P}$ is a pair $(\mathbf{p}, \mathbf{v})$, where

- $\mathbf{p}$, called *prover*, is an oracle that assigns a bit-string called *certificate* to every node of every labeled graph $(G, x) \in \mathcal{G}$, potentially using the identities assigned to the nodes, and
- $\mathbf{v}$, called *verifier*, is a distributed decision algorithm such that, for every $(G, x) \in \mathcal{G}$, and for every identity assignment ID to the nodes of G,

$$\begin{cases} (G, x, \mathrm{ID}) \text{ satisfies } \mathcal{P} & \Longrightarrow \quad \mathbf{v} \circ \mathbf{p}(G, x, \mathrm{ID}) = \text{accept;} \\ (G, x, \mathrm{ID}) \text{ does not satisfy } \mathcal{P} & \Longrightarrow \quad \text{for every prover } \mathbf{p}',\ \mathbf{v} \circ \mathbf{p}'(G, x, \mathrm{ID}) = \text{reject;} \end{cases}$$

here, $\mathbf{v} \circ \mathbf{p}$ is the output of the verifier $\mathbf{v}$ on the certificates assigned to the nodes by $\mathbf{p}$. That is, if (G, x, ID) satisfies $\mathcal{P}$, then, with the certificates assigned to the nodes by the prover $\mathbf{p}$, the verifier accepts at all nodes. Instead, if (G, x, ID) does not satisfy $\mathcal{P}$, then, whatever certificates are assigned to the nodes, the verifier rejects in at least one node.

The *radius* of a proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ is defined as the maximum number of rounds of the verifier $\mathbf{v}$ in the LOCAL model [35], over all identity-assignments to all the instances in $\mathcal{G}$, and all arbitrary certificates. It is denoted by $\mathsf{radius}(\mathbf{p}, \mathbf{v})$. Often in this paper, the phrase proof-labeling scheme is abbreviated into PLS, while a proof-labeling scheme of

[4] In a graph, $N(v)$ denotes the set of neighbors of node v, and $N[v] = N(v) \cup \{v\}$.

radius $t \geq 1$ is abbreviated into t-PLS. Note that, in a t-PLS, one can assume, w.l.o.g., that the verification procedure, which is given t as input to every node, proceeds at each node in two phases: (1) collecting all the data (i.e., labels and certificates) from nodes at distance at most t, including the structure of the ball of radius t around that node, and (2) processing all the information for producing a verdict, either accept, or reject. Note that, while the examples in this paper are of highly uniform graphs, and thus the structure of the t-balls might be known to the nodes in advance, our scaling mechanisms work for arbitrary graphs.

Given an instance (G, x, ID) satisfying $\mathcal{P}$, we denote by $\mathbf{p}(G, x, \mathrm{ID}, v)$ the certificate assigned by the prover $\mathbf{p}$ to node $v \in V$, and by $|\mathbf{p}(G, x, \mathrm{ID}, v)|$ its size. We also let $|\mathbf{p}(G, x, \mathrm{ID})| = \max_{v \in V(G)} |\mathbf{p}(G, x, \mathrm{ID}, v)|$. The *certificate-size* of a proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ for $\mathcal{P}$ in $\mathcal{G}$, denoted $\mathsf{size}(\mathbf{p}, \mathbf{v})$, is defined as the maximum of $|\mathbf{p}(G, x, \mathrm{ID})|$, taken over all instances (G, x, ID) satisfying $\mathcal{P}$, where $(G, x) \in \mathcal{G}$. In the following, we focus on the graph families $\mathcal{G}_n$ of connected simple graphs with n nodes, $n \geq 1$. That is, the size of a proof-labeling scheme is systematically expressed as a function of the number n of nodes. For the sake of simplifying the presentation, the graph family $\mathcal{G}_n$ is omitted from the notations.

The minimum certificate size of a t-PLS for the predicate $\mathcal{P}$ on n-node labeled graphs is denoted by $\mathsf{size\text{-}pls}(\mathcal{P}, t)$, that is,

$$\mathsf{size\text{-}pls}(\mathcal{P}, t) = \min_{\mathsf{radius}(\mathbf{p},\mathbf{v}) \leq t} \mathsf{size}(\mathbf{p}, \mathbf{v}).$$

We also denote by $\mathsf{size\text{-}pls}(\mathcal{P})$ the size of a standard (radius-1) proof-labeling scheme for $\mathcal{P}$, that is, $\mathsf{size\text{-}pls}(\mathcal{P}) = \mathsf{size\text{-}pls}(\mathcal{P}, 1)$. For instance, it is known that $\mathsf{size\text{-}pls}(\textsc{mst}) = \Theta(\log^2 n)$ bits [29, 31], and that $\mathsf{size\text{-}pls}(\textsc{aut}) = \widetilde{\Omega}(n^2)$ bits [25]. More generally, for every decidable predicate $\mathcal{P}$, we have $\mathsf{size\text{-}pls}(\mathcal{P}) = O(n^2 + nk)$ bits [25] whenever the labels produced by x are of k bits, and $\mathsf{size\text{-}pls}(\mathcal{P}, D) = 0$ for graphs of diameter D because the verifier can gather all labels, and all edges at every node in $D + 1$ rounds.

▶ **Definition 1.** Let $\mathcal{I} \subseteq \mathbb{N}^+$, and let $f : \mathcal{I} \to \mathbb{N}^+$. Let $\mathcal{P}$ be a boolean predicate on labeled graphs. A set $(\mathbf{p}_t, \mathbf{v}_t)_{t \in \mathcal{I}}$ of proof-labeling schemes for $\mathcal{P}$, with respective radius $t \geq 1$, *scales* with scaling factor f on $\mathcal{I}$ if $\mathsf{size}(\mathbf{p}_t, \mathbf{v}_t) = O\big(\frac{1}{f(t)} \cdot \mathsf{size\text{-}pls}(\mathcal{P})\big)$ bits for every $t \in \mathcal{I}$. Also, $(\mathbf{p}_t, \mathbf{v}_t)_{t \in \mathcal{I}}$ *weakly scales* with scaling factor f on $\mathcal{I}$ if $\mathsf{size}(\mathbf{p}_t, \mathbf{v}_t) = \widetilde{O}\big(\frac{1}{f(t)} \cdot \mathsf{size\text{-}pls}(\mathcal{P})\big)$ bits for every $t \in \mathcal{I}$.

In the following, somewhat abusing terminology, we shall say that a proof-labeling scheme (weakly) scales while, formally, it should be a set of proof-labeling schemes that scales.

Remark. At first glance, it may seem that no proof-labeling schemes can scale more than linearly, i.e., one may be tempted to claim that for every predicate $\mathcal{P}$ we have $\mathsf{size\text{-}pls}(\mathcal{P}, t) = \Omega\left(\frac{1}{t} \cdot \mathsf{size\text{-}pls}(\mathcal{P})\right)$. The rationale for such a claim is that, given a proof-labeling scheme $(\mathbf{p}_t, \mathbf{v}_t)$ for $\mathcal{P}$, with radius t and $\mathsf{size\text{-}pls}(\mathcal{P}, t)$, one can construct a proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ for $\mathcal{P}$ with radius 1 as follows: the certificate of every node v is the collection of certificates assigned by $\mathbf{p}_t$ to the nodes in the ball of radius t centered at v; the verifier $\mathbf{v}$ then simulates the execution of $\mathbf{v}_t$ on these certificates. In paths or cycles, the certificates resulting from this construction are of size $O(t \cdot \mathsf{size\text{-}pls}(\mathcal{P}, t))$, from which it follows that no proof-labeling scheme can scale more than linearly. There are several flaws in this reasoning, which make it actually erroneous. First, it might be the case that degree-2 graphs are not the worst case graphs for the predicate $\mathcal{P}$; that is, the fact that $(\mathbf{p}, \mathbf{v})$ induces certificates of size $O(t)$ times the certificate size of $(\mathbf{p}_t, \mathbf{v}_t)$ in such graphs may be uncorrelated to the size of the certificates of these proof-labeling schemes in worst case instances. Second, in t rounds of verification every node learns not only the certificates of its t-neighborhood, but also its structure, which may contain valuable information for the verification; this idea stands out when the lower bounds for $\mathsf{size\text{-}pls}(\mathcal{P})$ are established using labeled graphs of constant diameter, in which case there is no room for studying how proof-labeling schemes can scale. The take away message is that establishing lower bounds of the type $\mathsf{size\text{-}pls}(\mathcal{P}, t) = \Omega(\frac{1}{t} \cdot \mathsf{size\text{-}pls}(\mathcal{P}))$ for t within some non-trivial interval requires specific proofs, which often depend on the given predicate $\mathcal{P}$.

Communication Complexity. In the set-disjointness (DISJ) problem on k bits, each of the two players Alice and Bob is given a k-bit string, denoted S_A and S_B respectively. They aim at deciding whether $S_A \cap S_B = \emptyset$, i.e. whether there does not exist $i \in \{1, \dots, k\}$ such that $S_A[i] = S_B[i] = 1$. We consider nondeterministic protocols for the problem, i.e. protocols where the players also get an auxiliary string from an oracle that knows both inputs, and they may use it in order to verify that their inputs are disjoint. The communication complexity of a nondeterministic protocol for DISJ is the number of bits the players exchange on two input strings that are disjoint, in the worst case, when they are given optimal nondeterministic strings. The nondeterministic communication complexity of DISJ is the minimum, among all nondeterministic protocols for DISJ, of the communication complexity of that protocol. The nondeterministic communication complexity of DISJ is $\Omega(k)$ (e.g., as a consequence of Example 1.23 and Definition 2.3 in [32]).

3 All Proof-Labeling Schemes Scale Linearly in Trees

This section is entirely dedicated to the proof of one of our main results, stating that *every* predicate on labeled trees has a proof that scales linearly. Further in the section, we also show how to extend this result to cycles and to grids, and, more generally, to multi-dimensional grids and toruses.

▶ **Theorem 2.** *Let $\mathcal{P}$ be a predicate on labeled trees, and let us assume that there exists a (distance-1) proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ for $\mathcal{P}$, with $\mathsf{size}(\mathbf{p}, \mathbf{v}) = k$. Then there exists a proof-labeling scheme for $\mathcal{P}$ that scales linearly, that is, $\mathsf{size\text{-}pls}(\mathcal{P}, t) = O\left(\frac{k}{t}\right)$.*

The rest of this subsection is dedicated to the proof of Theorem 2. So, let $\mathcal{P}$ be a predicate on labeled trees, and let $(\mathbf{p}, \mathbf{v})$ be a proof-labeling scheme for $\mathcal{P}$ with $\mathsf{size}(\mathbf{p}, \mathbf{v}) = k$. First, note that we can restrict attention to trees with diameter $> t$. Indeed, predicates on labeled trees with diameter $\leq t$ are easy to verify since every node can gather the input of the entire tree in t rounds. More precisely, if we have a scheme that works for trees with diameter $> t$, then we can trivially design a scheme that applies to all trees, by adding a single bit to the certificates, indicating whether the tree is of diameter at most t or not.

The setting of the certificates in our scaling scheme is based on a specific decomposition of the given tree T. Let T be a tree of diameter $> t$, and let $h = \lfloor t/2 \rfloor$. For assigning the certificates, the tree T is rooted at some node r. A node u such that $\mathrm{dist}_T(r, u) \equiv 0 \pmod{h}$, and u possesses a subtree of depth at least $h - 1$ is called a *border* node. Similarly, a node u such that $\mathrm{dist}_T(r, u) \equiv -1 \pmod{h}$, and u possesses a subtree of depth at least $h - 1$ is called an *extra-border* node. A node that is a border or an extra-border node is called a *special* node. All other nodes are *standard* nodes. For every border node v, we define the *domain* of v as the set of nodes in the subtree rooted at v but not in subtrees rooted at border nodes that are descendants of v. The proof of the following lemma is omitted from this extended abstract.

▶ **Lemma 3.** *The domains form a partition of the nodes in the tree T, every domain forms a tree rooted at a border node, with depth in the range $[h-1, 2h-1]$, and two adjacent nodes of T are in different domains if and only if they are both special.*

The certificates of the distance-t proof-labeling scheme contain a 2-bit field indicating to each node whether it is a root, border, extra-border, or standard node. Let us show that this part of the certificate can be verified in t rounds. The prover orients the edges of the tree towards the root r. It is well-known that such an orientation can be given to the edges of a tree by assigning to each node its distance to the root, modulo 3. These distances can obviously be checked locally, in just one round. So, in the remaining of the proof, we assume that the nodes are given this orientation upward the tree. The following lemma (whose proof is omitted) shows that the decomposition into border, extra-border, and standard nodes can be checked in t rounds.

▶ **Lemma 4.** *Given a set of nodes marked as border, extra-border, or standard in an oriented tree, there is a verification protocol that checks whether that marking corresponds to a tree decomposition such as the one described above, in $2h < t$ rounds.*

We are now ready to describe the distance-t proof-labeling scheme. From the previous discussions, we can assume that the nodes are correctly marked as root, border, extra-border, and standard, with a consistent orientation of the edges towards the root. We are considering the given predicate $\mathcal{P}$ on labeled trees, with its proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ using certificates of size k bits. Before reducing the size of the certificates to $O(k/t)$ by communicating at distance t, we describe a proof-labeling scheme at distance t which still uses large certificates, of size $O(k)$, but stored at a few nodes only, with all other nodes storing no certificates.

▶ **Lemma 5.** *There exists a distance-t proof-labeling scheme for $\mathcal{P}$, in which the prover assigns certificates to special nodes only, and these certificates have size $O(k)$.*

Sketch of proof. On legally labeled trees, the prover provides every special node (i.e., every border or extra-border node) with the same certificate as the one provided by $\mathbf{p}$. All other nodes are provided with no certificates. On arbitrary labeled trees, the verifier is active at border nodes only, and all non-border nodes systematically accept (in zero rounds). At a border node v, the verifier first gathers all information at distance $2h$. This includes all the labels of the nodes in its domain, and of the nodes that are neighbors of some node in its domain. Then v checks whether there exists an assignment of k-bit certificates to the standard nodes in its domain that results in $\mathbf{v}$ accepting at every node in its domain. If this is the case, then v accepts, else it rejects. Since the standard nodes form non-overlapping regions well separated by the border and extra-border nodes, this results in a correct distance-t proof-labeling scheme. ◀

We now show how to spread out the certificates of the border and extra-border nodes to obtain smaller certificates. The following lemma is the main tool for doing so. As this lemma is also used further in the paper, we provide a generalized version of its statement, and we later show how to adapt it to the setting of the current proof.

We say that a local algorithm $\mathcal{A}$ *recovers* an assignment of certificates provided by some prover $\mathbf{q}$ from an assignment of certificates provided by another prover $\mathbf{q}'$ if, given the certificates assigned by $\mathbf{q}'$ as input to the nodes, $\mathcal{A}$ allows every node to reconstruct the certificate that would have been assigned to it by $\mathbf{q}$. We define a *special* prover as a prover that assigns certificates only to the special nodes, while all other nodes are given empty certificates.

▶ **Lemma 6.** *There exists a local algorithm $\mathcal{A}$ satisfying the following. For every $s \geq 1$, for every oriented marked tree T of depth at least s, and for every assignment of b-bit certificates provided by some special prover $\mathbf{q}$ to the nodes of T, there exists an assignment of $O(b/s)$-bit certificates provided by a prover $\mathbf{q}'$ to the nodes of T such that $\mathcal{A}$ recovers $\mathbf{q}$ from $\mathbf{q}'$ in s rounds.*

Sketch of proof. The prover $\mathbf{q}'$ spreads the certificate assigned to each border node v along a path starting from v, of length $s-1$, going downward the tree. The algorithm $\mathcal{A}$ gathers the certificates spread along these paths. ◀

Proof of Theorem 2. In the distance-t proof-labeling scheme, the prover chooses a root and an orientation of the tree T, and provides every node with a counter modulo 3 in its certificate allowing the nodes to check the consistency of the orientation. Then the prover constructs a tree decomposition of the rooted tree, and provides every node with its type (root, border, extra-border, or standard) in its certificates. Applying Lemmas 5 and 6, the prover spreads the certificates assigned to the special nodes by $\mathbf{p}$. Every node will get at most two parts, because only the paths associated to a border node and to its parent (an extra-border node) can intersect. Overall, the certificates have size $O(k/h) = O(k/t)$. The verifier checks the orientation and the marking, then recovers the certificates of the special nodes, as in Lemma 6, and performs the simulation as in Lemma 5. This verification can be done with radius $t \leq 2h$, yielding the desired distance-t proof labeling scheme. ◀

Linear scaling in cycles and grids. For the proof techniques of Theorem 2 to apply to other graphs, we need to compute a partition of the nodes into the two categories, special and standard, satisfying three main properties. First, the partition should split the graph into regions formed by standard nodes, separated by special nodes. Second, each region should have a diameter small enough for allowing special nodes at the border of the region to simulate the standard nodes in that region, as in Lemma 5. Third, the regions should have a diameter large enough to allow efficient spreading of certificates assigned to special nodes over the standard nodes, as in Lemma 6. For any graph family in which one can define such a decomposition, an analogue of Theorem 2 holds. We show that this is the case for cycles and grids (the proof is omitted).

▶ **Corollary 7.** *Let $\mathcal{P}$ be a predicate on labeled cycles, and let us assume that there exists a (distance-1) proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ for $\mathcal{P}$ with $\mathsf{size}(\mathbf{p}, \mathbf{v}) = k$. Then there exists a proof-labeling scheme for $\mathcal{P}$ that scales linearly, that is, $\mathsf{size\text{-}pls}(\mathcal{P}, t) = O\left(\frac{k}{t}\right)$. The same holds for predicates on 2-dimensional labeled grids.*

By the same techniques, Corollary 7 can be generalized to toroidal 2-dimensional labeled grids, as well as to d-dimensional labeled grids and toruses, for every $d \geq 2$.

4 Universal Scaling of Uniform Proof-Labeling Schemes

It is known [33] that, for every predicate $\mathcal{P}$ on labeled graphs with $\mathsf{size\text{-}pls}(\mathcal{P}) = \widetilde{\Omega}(n^2)$, there is a proof-labeling scheme that scales linearly on the interval $[1, D]$ in graphs of diameter D. We show that, in fact, the scaling factor can be much larger. We say that a graph $G = (V, E)$ has *growth* $b = b(t)$ if, for every $v \in V$ and $t \in [1, D]$, we have that $|B_G(v, t)| \geq b(t)$. We say that a proof-labeling scheme is *uniform* if the same certificate is assigned to all nodes by the prover.

▶ **Theorem 8.** *Let $\mathcal{P}$ be a predicate on labeled graphs, fix a uniform 1-PLS $(\mathbf{p},\mathbf{v})$ for $\mathcal{P}$ and denote $k = \mathsf{size}(\mathbf{p},\mathbf{v})$. Then there is a proof-labeling scheme for $\mathcal{P}$ that weakly scales with scaling factor $b(t)$ on graphs of growth $b(t)$. More specifically, let G be a graph, let $t_0 = \min\{t \mid b(t) \geq \log n\}$, and $t_1 = \max\{t \mid k \geq b(t)\}$. Then, in G, for every $t \in [t_0, t_1]$, $\mathsf{size\text{-}pls}(\mathcal{P}, t) = \widetilde{O}\left(\frac{k}{b(t)}\right)$.*

Proof. Let $s = (s_1, \dots, s_k)$, where $s_i \in \{0,1\}$ for every $i = 1, \dots, k$, be the k-bit certificate assigned to every node of G. Let $t \geq 1$ be such that $k \geq b(t) \geq c \log n$ for a constant c large enough. For every node $v \in V$, set the certificate of v, denoted $s^{(v)}$, as follows: for every $i = 1, \dots, k$, v stores the pair (i, s_i) in $s^{(v)}$ with probability $\frac{c \log n}{b(t)}$. Recall the following Chernoff bounds: Suppose $Z_1, \dots, Z_m$ are independent random variables taking values in $\{0,1\}$, and let $Z = \sum_{i=1}^m Z_i$. For every $0 \leq \delta \leq 1$, we have $\Pr[Z \leq (1-\delta)\mathbb{E}Z] \leq e^{-\frac{1}{2}\delta^2 \mathbb{E}Z}$, and $\Pr[Z \geq (1+\delta)\mathbb{E}Z] \leq e^{-\frac{1}{3}\delta^2 \mathbb{E}Z}$.

- On the one hand, for every $v \in V$, let X_v be the random variable equal to the number of pairs stored in $s^{(v)}$. By a Chernoff bound, we have $\Pr[X_v \geq \frac{2c\,k \log n}{b(t)}] \leq e^{\frac{c\,k \log n}{3\,b(t)}} = n^{-\frac{c\,k}{3\,b(t)}}$. Therefore, by union bound, the probability that a node v stores more than $\frac{2c\,k \log n}{b(t)}$ pairs (i, s_i) is at most $n^{1-\frac{c\,k}{3\,b(t)}}$, which is less than $\frac{1}{2}$ for c large enough.
- On the other hand, for every $v \in V$, and every $i = 1, \dots, k$, let $Y_{v,i}$ be the number of occurrences of the pair (i, s_i) in the ball of radius t centered at v. By a Chernoff bound, we have $\Pr[Y_{v,i} \leq \frac{1}{2} c \log n] \leq e^{-\frac{c \log n}{8}} = n^{-c/8}$. Therefore, by union bound, the probability that there exists a node $v \in V$, and an index $i \in \{1, \dots, k\}$ such that none of the nodes in the ball of radius t centered at v store the pair (i, s_i) is at most $kn^{1-c/8}$, which is less than $\frac{1}{2}$ for c large enough.

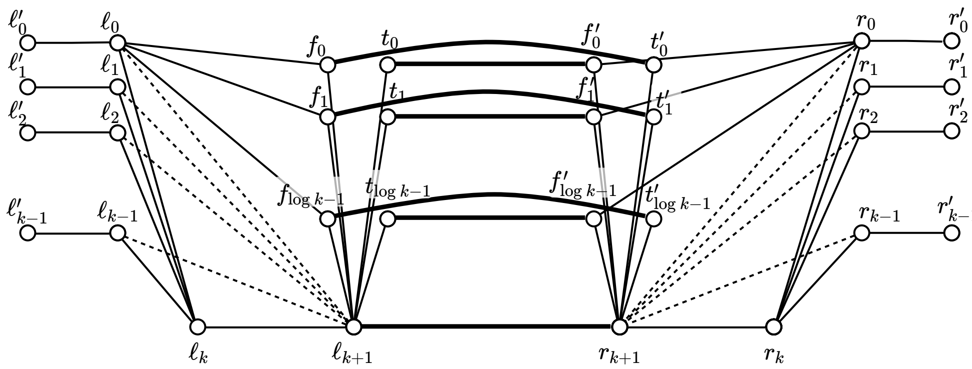
It follows that, for c large enough, the probability that no node stores more than $\widetilde{O}(k/b(t))$ pairs (i, s_i), and every pair (i, s_i) is stored in at least one node of each ball of radius t, is positive. Therefore, there is a way for a prover to distribute the pairs (i, s_i), $i = 1, \dots, k$, to the nodes such that (1) no node stores more than $\widetilde{O}(k/b(t))$ bits, and (2) every pair (i, s_i) appears at least once in every t-neighborhood of each node. At each node v, the verification procedure first collects all pairs (i, s_i) in the t-neighborhood of v, in order to recover s, and then runs the verifier of the original (distance-1) proof-labeling scheme.

Finally, we emphasize that we only use probabilistic arguments as a way to prove the existence of certificate assignment, but the resulting proof-labeling scheme is deterministic and its correctness is not probabilistic. ◀

Theorem 8 finds direct application to the *universal* proof-labeling scheme [25, 31], which uses $O(n^2 + kn)$ bits in n-node graphs labeled with k-bit labels. The certificate of each node consists of the $n \times n$ adjacency matrix of the graph, an array of n entries each equals to the k-bit label at the corresponding node, and an array of n entries listing the identities of the n nodes. It was proved in [33] that the universal proof-labeling scheme can be scaled by a factor t. Theorem 8 significantly improves that result, by showing that the universal proof-labeling scheme can actually be scaled by a factor $b(t)$, which can be exponential in t.

▶ **Corollary 9.** *For every predicate $\mathcal{P}$ on labeled graphs, there is a proof-labeling scheme for $\mathcal{P}$ as follows. For every graph G with growth $b(t)$, let $t_0 = \min\{t \mid b(t) \geq \log n\}$. Then, for every $t \geq t_0$ we have $\mathsf{size\text{-}pls}(\mathcal{P}, t) = \widetilde{O}\left(\frac{n^2+kn}{b(t)}\right)$.*

Theorem 8 is also applicable to proof-labeling scheme where the certificates have the same sub-certificate assigned to all nodes; in this case, the size of this common sub-certificate



■ **Figure 1** The lower bound graph construction. Thin lines represent P-paths, thick lines represent $(2t+1)$-paths, and the dashed lines represent edges who's existence depend on the input. The paths connecting ℓ_i and r_i to their binary representations are omitted, except for those of ℓ_0 and r_0.

can be drastically reduced by using a t-round verification procedure. This is particularly interesting when the size of the common sub-certificate is large compared to the size of the rest of the certificates. An example of such a scheme is in essence the one described in [31, Corollary 2.4] for $\textsc{iso}_k$. Given a parameter $k \in \Omega(\log n)$, let $\textsc{iso}_k$ be the predicate on graph stating that there exist two vertex-disjoint isomorphic induced subgraphs of size k in the given graph. The proof of the next corollary appears in the full version of our paper.

▶ **Corollary 10.** *For every $k \in \left[1, \frac{n}{2}\right]$, we have $\mathsf{size\text{-}pls}(\textsc{iso}_k) = \Theta(k^2)$ bits, and, for every $t > 1$, $\mathsf{size\text{-}pls}(\textsc{iso}_k, t) = \widetilde{O}\left(\frac{k^2}{b(t)}\right)$.*

5 Certifying Distance-Related Predicates

For any labeled (weighted) graph (G, x), the predicate $\textsc{diam}$ on (G, x) states whether, for every $v \in V(G)$, $x(v)$ is equal to the (weighted) diameter of G.

▶ **Theorem 11.** *There is a proof-labeling scheme for $\textsc{diam}$ that scales linearly between $[c \log n, n/\log n]$, for some constant c. More specifically, there exists $c > 0$, such that, for every $t \in [c \log n, n/\log n]$, $\mathsf{size\text{-}pls}(\textsc{diam}, t) = \widetilde{O}\left(\frac{n}{t}\right)$. Moreover, no proof-labeling schemes for $\textsc{diam}$ can scale more than linearly on the interval $[1, n/\log n]$, that is, for every $t \in [1, n/\log n]$, $\mathsf{size\text{-}pls}(\textsc{diam}, t) = \widetilde{\Omega}\left(\frac{n}{t}\right)$.*

The upper bound proof follows similar lines to those of Theorem 8: each node keeps only a partial list of distances to other nodes. In the verification process, a node u computes its distance to a node v as follows: first, u finds a node v' in its t-neighborhood that has the distance to v in its certificate; then, u computes its distance to v', which is possible since u knowns all its t-neighborhood; and finally, u deduces its own distance from v. A suitable choice of parameter guarantees the existence of a "good" v', that will indeed allow u to compute the correct distance. The full proof appears in the full version of our paper.

We now describe the construction of the lower bound graph (see Figure 1). Let $k = \Theta(n)$ be a parameter whose exact value will follow from the graph construction. Alice and Bob use the graph in order to decide $\textsc{disj}$ on k-bit strings. Let $P \geq 1$ be a constant, and let t be the parameter of the t-PLS, which may or may not be constant. The graph consists of

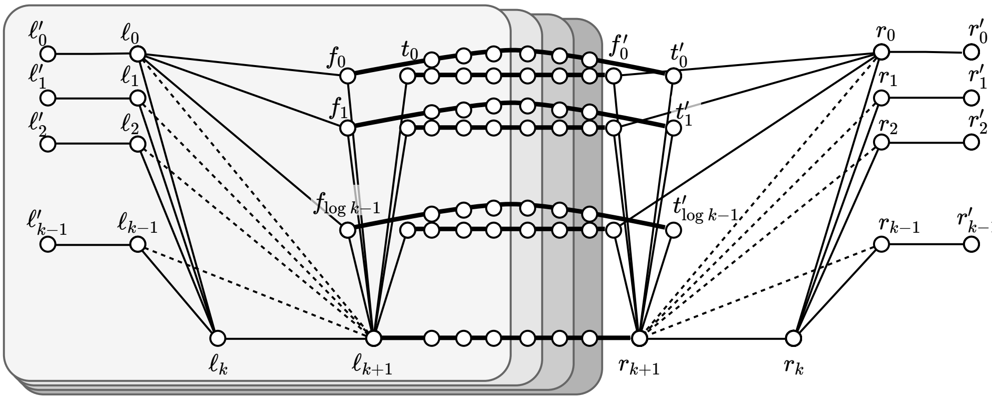


Figure 2 The lower bound graph construction for $t = 3$, and the sets of nodes simulated by Alice in the three rounds of verification (from dark gray to lighter gray). Alice eventually knows the outputs of all the nodes in the light-most gray shaded set.

the following sets of nodes: $L = \{\ell_0, \ldots, \ell_{k-1}\}$, $L' = \left\{\ell'_0, \ldots, \ell'_{k-1}\right\}$, $T = \{t_0, \ldots, t_{\log k-1}\}$, $F = \{f_0, \ldots, f_{\log k-1}\}$, and ℓ_k and ℓ_{k+1}, which will be simulated by Alice, and similarly $R = \{r_0, \ldots, r_{k-1}\}$, $R' = \left\{r'_0, \ldots, r'_{k-1}\right\}$, $T' = \left\{t'_0, \ldots, t'_{\log k-1}\right\}$, $F' = \left\{f'_0, \ldots, f'_{\log k-1}\right\}$, and r_k and r_{k+1}, which will be simulated by Bob.

The nodes are connected by paths, where the paths consist of additional, distinct nodes. For each $0 \le i \le k-1$, connect with P-paths (i.e., paths of P edges and $P-1$ new nodes) the following pairs of nodes: (ℓ_i, ℓ'_i), (ℓ_i, ℓ_k), (ℓ_k, ℓ_{k+1}), (r_i, r'_i), (r_i, r_k), and (r_k, r_{k+1}). Add such paths also between ℓ_{k+1} and all $t_h \in T$ and $f_h \in F$, and between r_{k+1} and all $t'_h \in T'$ and $f'_h \in F'$. Connect by a P-path each $\ell_i \in L$ with the nodes representing its binary encoding, that is, connect ℓ_i to each t_h that satisfies $i[h] = 1$, and to each f_h that satisfies $i[h] = 0$, where $i[h]$ is bit h of the binary encoding of i. Add similar paths between each $r_i \in R$ and its encoding by nodes t'_h and f'_h. In addition, for each $0 \le h \le \log k - 1$, add a $(2t+1)$-path from t_h to f'_h and from f_h to t'_h, and a similar path from ℓ_{k+1} to r_{k+1}.

Assume Alice and Bob want to solve the DISJ problem for two k-bit strings S_A and S_B using a non-deterministic protocol. They build the graph described above, and add the following edges: (ℓ_i, ℓ_{k+1}) whenever $S_A[i] = 0$, and (r_i, r_{k+1}) whenever $S_B[i] = 0$. The next claim is at the heart of our proof.

▶ **Claim 1.** *If S_A and S_B are disjoint then $D = 4P + 2t + 2$, and otherwise $D \ge 6P + 2t + 1$.*

The proof of this claim follows similar lines of the proof of [1, Lemma 2], and appears in the full version of our paper. We can now prove the lower bound from Theorem 11.

Proof of lower bound from Theorem 11. Fix $t \in [1, n/\log n]$, and let S_A and S_B be two input strings for the DISJ problem on k bits. We show how Alice and Bob can solve DISJ on S_A and S_B in a nondeterministic manner, using the graph described above and a t-PLS for DIAM $= 4P + 2t + 2$.

Alice and Bob simulate the verifier on the labeled graph (see Figure 2). The nodes simulated by Alice, denoted A, are $L \cup L' \cup T \cup F \cup \{\ell_k, \ell_{k+1}\}$ and all the paths between them, and by Bob, denoted B, are $R \cup R' \cup T' \cup F' \cup \{r_k, r_{k+1}\}$ and the paths between them. For each pair of nodes $(a, b) \in A \times B$ that are connected by a $(2t+1)$-path, let P_{ab} be this path, and $\{P_{ab}(i)\}$, $i = 0, \ldots, 2t+1$ be its nodes in consecutive order, where $P_{ab}(0) = a$ and

$P_{ab}(2t+1) = b$. Let C be the set of all $(2t+1)$-path nodes, i.e. $C = V \setminus (A \cup B)$. The nodes in C are simulated by both players, in a decremental way described below.

Alice interprets her nondeterministic string as the certificates given to the nodes in $A \cup C$, and she sends the certificates of C to Bob. Bob interprets his nondeterministic string as the certificates of B, and gets the certificates of C from Alice. They simulate the verifier execution for t rounds, where, in round $r = 1, \ldots, t$, Alice simulates the nodes of A and all nodes $P_{ab}(i)$ with $(a,b) \in A \times B$ and $i \leq 2t+1-r$, while Bob simulates the nodes of B and all nodes $P_{ab}(i)$ with $i \geq r$.

Note that this simulation is possible without further communication. The initial state of nodes in A is determined by S_A, the initial state of the nodes $P_{ab}(i)$ with $i \leq 2t$ is independent of the inputs, and the certificates of both node sets are encoded in the nondeterministic string of Alice. In each round of verification, all nodes whose states may depend on the input of Bob or on his nondeterministic string are omitted from Alice's simulation, and so she can continue the simulation without communication with Bob. Similar arguments apply to the nodes simulated by Bob. Finally, each node is simulated for t rounds by at least one of the players. Thus, if the verifier rejects, that is, at least one node rejects, then at least one of the players knows about this rejection.

Using this simulation, Alice and Bob can determine whether DISJ on (S_A, S_B) is true as, from Claim 1, we know that if it is true then $\text{DIAM} = 4P + 2t + 2$, and the verifier of the PLS accepts, while otherwise it rejects. The nondeterministic communication complexity of the true case of DISJ on k-bit strings is $\Omega(k) = \Omega(n)$, so Alice and Bob must communicate this amount of bits. From the graph definition, $|C| = \Theta(t \log n)$ which implies $\textsf{size-pls}(\text{DIAM}, t) = \Omega\left(\frac{n}{t \log n}\right)$, as desired. ◀

Let k be a non-negative integer. For any labeled graph (G,x), k-SPANNER is the predicate on (G,x) that states whether the collection of edges $E_H = \{\{v,w\}, v \in V(G), w \in x(v)\}$ forms a k-additive spanner of G, i.e., a subgraph H of G such that, for every two nodes s,t, we have $\text{dist}_H(s,t) \leq \text{dist}_G(s,t) + k$. There is a proof-labeling scheme for additive-spanner that weakly scales linearly, or more precisely, $\textsf{size-pls}(k\text{-SPANNER}, t) = \widetilde{\Theta}(\frac{n}{t})$ for any constant k and $t \in [1, n/\log n]$. In the full version of our paper we prove this result, its optimality, as well as slightly weaker results for general spanners.

6 Distributed Proofs for Spanning Trees

In this section, we study two specific problems which are classical in the domain of proof-labeling schemes: the verification of a spanning tree, and of a minimum-weight spanning tree. The predicates ST and MST are the sets of labeled graphs where some edges are marked and these edges form a spanning tree, and a minimum spanning tree, respectively. For these predicates, we present proof-labeling schemes that scale linearly in t. Note that ST and MST are problems on general labeled graphs and not on trees, i.e., the results in this section improve upon Section 4 (for these specific problems), and are incomparable with the results of Section 3.

Formally, let $\mathcal{F}$ be the family of all connected undirected, weighted, labeled graphs (G,x). Each label $x(v)$ contains a (possibly empty) subset of edges adjacent to v, which is consistent with the neighbors of v, and we denote the collection of edges represented in x by T_x. In the ST (respectively, MST) problem, the goal is to decide for every labeled graph $(G,x) \in \mathcal{F}$ whether T_x is a spanning tree of G (respectively, whether T_x is a spanning tree of G with the sum of all its edge-weights minimal among all spanning trees of G). For these problems we have the following results.

▶ **Theorem 12.** *For every $t \in O(\log n)$, we have that* size-pls(ST, t) $= O\left(\frac{\log n}{t}\right)$.

Proof sketch. To prove that a marked subgraph T_x is a spanning tree, we verify it has the following properties: (1) spanning the graph, (2) acyclic, (3) connected. The first property is local—every node verifies that it has at least one incident marked edge. For the second property, we use the t-distance proof-labeling scheme for acyclicity designed by Ostrovsky et al. [33, Theorem 8], where oriented trees are verified and every root knows that it is a root, using $O(\log n/t)$-bit certificates. Finally, we use Theorem 2 within the tree in order to split the root ID; the nodes then verify they all agree on the root, which implies connectivity. ◀

▶ **Theorem 13.** *For every $t \in O(\log n)$, we have that* size-pls(MST, t) $= O\left(\frac{\log^2 n}{t}\right)$.

Our theorem only applies for $t \in O(\log n)$, meaning that we can get from proofs of size $O(\log^2 n)$ to proofs of size $O(\log n)$, but not to a constant. For the specific case $t = \Theta(\log n)$, our upper bound matches the lower bound of Korman et al. [30, Corollary 3]. In the same paper, the authors also present an $O(\log^2 n)$-round verification scheme for MST using $O(\log n)$ bits of memory at each node (both for certificates and for local computation). Removing the restriction of $O(\log n)$-bit memory for local computation, one may derive an $O(\log n)$-round verification scheme with $O(\log n)$ proof size out of the aforementioned $O(\log^2 n)$-round scheme, which matches our result for $t = \Theta(\log n)$. The improvement we present is two-folded: our scheme is scalable for different values of t (as opposed to schemes for only $t = 1$ and $t = \Theta(\log n)$), and our construction is much simpler, as described next.

Our upper bound is based on a famous 1-round PLS for MST [29, 30], which in turn builds upon the algorithm of Gallager, Humblet, and Spira (GHS) [24] for a distributed construction of an MST. The idea behind this scheme is, given a labeled graph (G, x), to verify that T_x is consistent with an execution of the GHS algorithm in G.

The GHS algorithm maintains a spanning forest that is a subgraph of the minimum spanning tree, i.e., the trees of the forest are fragments of the desired minimum spanning tree. The algorithm starts with a spanning forest consisting of all nodes and no edges. At each phase each of the fragments adds the minimum-weight edge going out of it, thus merging several fragments into one. After $O(\log n)$ iterations, all the fragments are merged into a single component, which is the desired minimum-weight spanning tree. We show that each phase can be verified with $O(\log n/t)$ bits, giving a total complexity of $O(\log^2 n/t)$ bits.

The GHS algorithm assumes distinct edge weights, which implies a unique minimum-weight spanning tree and a unique (synchronous) execution of the algorithm. The case of non-unique edge weights is easily resolved in the algorithm by any consistent tie-breaking (e.g., using node IDs); handling non-unique edge weights in verification is not as easy, since the tie-breaking mechanism must result in the specified spanning tree. Theorem 13 is true without the assumption of distinct edge weights, but we prove it here only under this assumption, and leave the proof of the general case to the full version of our paper.

Proof of Theorem 13. Let (G, x) be a labeled graph such that T_x is a minimum-weight spanning tree. If t is greater than the diameter D of G, every node can see the entire labeled graph in the verification process, and we are done; we henceforth assume $t \leq D$. The certificates consist of four parts.

First, we choose a root and orient the edges of T_x towards it. We give each node its distance from the root modulo 3, which allows it to obtain the ID of its parent and the edge pointing to it in one round. Second, we assign the certificate described above for ST (Theorem 12), which certifies that T_x is indeed a spanning tree. This uses $O(\log n/t)$ bits.

The third part of the certificate tells each node the phase in which the edge connecting it to its parent is added to the tree in the GHS algorithm, and which of the edge's endpoints added it to the tree. Note that after one round of verification, each node knows for every incident edge, at which phase it is added to the spanning tree, and by which of its endpoints. This part uses $O(\log\log n)$ bits.

The fourth part of the certificate consists of $O(\log^2 n/t)$ bits, $O(\log n/t)$ for each of the $O(\log n)$ phases of the GHS algorithm. To define the part of a certificate of every phase, fix a phase, a fragment F in the beginning of this phase, and let $e = (u, v)$ be the minimum-weight edge going out of F, where $u \in F$ and $v \notin F$. Our goal is that the nodes of F verify together that e is the minimum-weight outgoing edge of F, and that no other edge was added by F in this phase. To this end, we first orient the edges of F towards u, i.e. set u as the root of F. If the depth of F is less than t, then in $t - 1$ rounds the root u can see all of F and check that (u, v) is the lightest outgoing edge. All other nodes just have to verify that no other edge is added by the nodes of F in this phase. Otherwise, if the depth of F is at least t, by Theorem 2, the information about $\mathrm{ID}(u)$ and $w(e)$ can be spread on F such that in t rounds it can be collected by all nodes of F. With this information known to all the nodes of F, the root can locally verify that it is named as the node that adds the edge and that it has the named edge with the right weight. The other nodes of F can locally verify that they do not have incident outgoing edges with smaller weights, and that no other edge is added by F.

Overall, our scheme verifies that T_x is a spanning tree, and that it is consistent with every phase of the GHS algorithm. Therefore, the scheme accepts (G, x) if and only if T_x is a minimum spanning tree. ◀

7 Conclusion

We have proved that, for many classical boolean predicates on labeled graphs (including MST), there are proof-labeling schemes that linearly scale with the radius of the scheme, i.e., the number of rounds of the verification procedure. More generally, we have shown that for *every* boolean predicate on labeled trees, cycles and grids, there is a proof-labeling scheme that scales linearly with the radius of the scheme. This yields the following question:

▶ Open Problem 1. Prove or disprove that, for every predicate $\mathcal{P}$ on labeled graphs, there is a proof-labeling scheme for $\mathcal{P}$ that (weakly) scales linearly.

In fact, the scaling factor might even be larger than t, and be as large as $b(t)$ in graphs with ball growth b. We have proved that the uniform part of any proof-labeling scheme can be scaled by such a factor $b(t)$ for t-PLS. This yields the following stronger open problem:

▶ Open Problem 2. Prove or disprove that, for every predicate $\mathcal{P}$ on labeled graphs, there is a proof-labeling scheme for $\mathcal{P}$ that scales with factor $\widetilde{\Omega}(b)$ in graphs with ball growth b.

We are tempted to conjecture that the answer to the first problem is positive (as it holds for trees and cycles). However, we believe that the answer to the second problem might well be negative. In particular, it seems challenging to design a proof-labeling scheme for DIAM that would scale with the size of the balls. Indeed, checking diameter is strongly related to checking shortest paths in the graph, and this restricts significantly the way the certificates can be redistributed among nodes in a ball of radius t. Yet, there might be some other way to certify DIAM, so we let the following as an open problem:

▶ Open Problem 3. Is there a proof-labeling scheme for DIAM that scales by a factor greater than t in all graphs where $b(t) \gg t$?

Acknowledgements: We thank Seri Khoury and Boaz Patt-Shamir for valuable discussions, and the anonymous reviewers of DISC 2018.

References

1 Amir Abboud, Keren Censor-Hillel, and Seri Khoury. Near-linear lower bounds for distributed distance computations, even in sparse networks. In *30th Int. Symposium on Distributed Computing (DISC)*, pages 29–42, 2016. Full version at arXiv:1605.05109.

2 Yehuda Afek and Shlomi Dolev. Local stabilizer. *J. Parallel Distrib. Comput.*, 62(5):745–765, 2002.

3 Yehuda Afek, Shay Kutten, and Moti Yung. The local detection paradigm and its applications to self-stabilization. *Theoretical Computer Science*, 186(1):199 – 229, 1997.

4 Heger Arfaoui, Pierre Fraigniaud, David Ilcinkas, and Fabien Mathieu. Distributedly testing cycle-freeness. In *40th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 8747 of *LNCS*, pages 15–28. Springer, 2014.

5 Heger Arfaoui, Pierre Fraigniaud, and Andrzej Pelc. Local decision and verification with bounded-size outputs. In *15th Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 8255 of *LNCS*, pages 133–147. Springer, 2013.

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

7 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)*, volume 66 of *LIPIcs*, pages 8:1–8:13, 2017.

8 Evangelos Bampas and David Ilcinkas. On mobile agent verifiable problems. In *12th Latin American Symposium on Theoretical Informatics (LATIN)*, LNCS 9644, pages 123–137. Springer, 2016.

9 Mor Baruch, Pierre Fraigniaud, and Boaz Patt-Shamir. Randomized proof-labeling schemes. In *24th Symposium on Principles of Distributed Computing (PODC)*, pages 315–324. ACM, 2015.

10 Joffroy Beauquier, Sylvie Delaët, Shlomi Dolev, and Sébastien Tixeuil. Transient fault detectors. *Distributed Computing*, 20(1):39–51, 2007.

11 Lélia Blin and Pierre Fraigniaud. Space-optimal time-efficient silent self-stabilizing constructions of constrained spanning trees. In *35th Int. Conference on Distributed Computing Systems (ICDCS)*, pages 589–598. IEEE, 2015.

12 Lélia Blin, Pierre Fraigniaud, and Boaz Patt-Shamir. On proof-labeling schemes versus silent self-stabilizing algorithms. In *16th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS, pages 18–32. Springer, 2014.

13 Keren Censor-Hillel, Ami Paz, and Mor Perry. Approximate proof labeling schemes. In *24th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2017.

14 A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

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

16 Laurent Feuilloley and Pierre Fraigniaud. Error-sensitive proof-labeling schemes. In *31st International Symposium on Distributed Computing (DISC)*, pages 16:1–16:15, 2017.

17 Laurent Feuilloley, Pierre Fraigniaud, and Juho Hirvonen. A Hierarchy of Local Decision. In *43rd Int. Colloquium on Automata, Languages, and Programming (ICALP)*, LIPIcs, pages 118:1–118:15, 2016.

18 Klaus-Tycho Foerster, Thomas Luedi, Jochen Seidel, and Roger Wattenhofer. Local checkability, no strings attached: (a)cyclicity, reachability, loop free updates in sdns. *Theor. Comput. Sci.*, 709:48–63, 2018.

19 Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60(5):35, 2013.

20 Pierre Fraigniaud and Andrzej Pelc. Decidability classes for mobile agents computing. *J. Parallel Distrib. Comput.*, 109:117–128, 2017.

21 Pierre Fraigniaud, Sergio Rajsbaum, and Corentin Travers. Minimizing the number of opinions for fault-tolerant distributed decision using well-quasi orderings. In *12th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 497–508. Springer, 2016.

22 Pierre Fraigniaud, Sergio Rajsbaum, Corentin Travers, Petr Kuznetsov, and Thibault Rieutord. Perfect failure detection with very few bits. In *18th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS 10083, pages 154–169. Springer, 2016.

23 Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *23rd Symposium on Discrete Algorithms (SODA)*, pages 1150–1162. ACM-SIAM, 2012.

24 R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 5(1):66–77, 1983.

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

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

27 Gillat Kol, Rotem Oshman, and Raghuvansh R. Saxena. Interactive distributed proofs. In *37th ACM Symposium on Principles of Distributed Computing (PODC 2018)*, to appear.

28 Janne H. Korhonen and Jukka Suomela. Brief announcement: Towards a complexity theory for the congested clique. In *31st International Symposium on Distributed Computing (DISC)*, pages 55:1–55:3, 2017.

29 Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20:253–266, 2007.

30 Amos Korman, Shay Kutten, and Toshimitsu Masuzawa. Fast and compact self-stabilizing verification, computation, and fault detection of an MST. *Distributed Computing*, 28(4):253–295, 2015.

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

32 Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, New York, 1997.

33 Rafail Ostrovsky, Mor Perry, and Will Rosenbaum. Space-time tradeoffs for distributed verification. In *24th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 53–70, 2017.

34 Boaz Patt-Shamir and Mor Perry. Proof-labeling schemes: Broadcast, unicast and in between. In *19th Int. Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS 10616, pages 1–17. Springer, 2017.

35 David Peleg. *Distributed Computing: A Locality-Sensitive Approach.* Discrete Mathematics and Applications. SIAM, Philadelphia, 2000.

36 David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed MST construction. In *40th Symp. on Foundations of Computer Science (FOCS)*, pages 253–261. IEEE, 1999.