Randomized Proof-Labeling Schemes*

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

Pierre Fraigniaud†
Dept. of Computer Science
CNRS and Univ. Paris Diderot
France
Pierre.Fraigniaud@liafa.univ-paris-diderot.fr

Boaz Patt-Shamir‡
School of Electrical Engineering
Tel Aviv University
Tel Aviv, Israel
boaz@tau.ac.il

ABSTRACT

Proof-labeling schemes, introduced by Korman, Kutten and Peleg [PODC 2005], are a mechanism to certify that a network configuration satisfies a given boolean predicate. Such mechanisms find applications in many contexts, e.g., the design of fault-tolerant distributed algorithms. In a proof-labeling scheme, predicate verification consists of neighbors exchanging labels, whose contents depends on the predicate. In this paper, we introduce the notion of *randomized* proof-labeling schemes where messages are randomized and correctness is probabilistic. We show that randomization reduces label size exponentially while guaranteeing probability of correctness arbitrarily close to one. In addition, we present a novel label-size lower bound technique that applies to both deterministic and randomized proof-labeling schemes. Using this technique, we establish several tight bounds on the verification complexity of MST, acyclicity, connectivity, and longest cycle size.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.2 [**Discrete Mathematics**]: Graph Theory; B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance

General Terms

Algorithms, Reliability, Theory

Keywords

distributed verfification, communication complexity

*Research supported in part by the Ministry of Science Technology and Space, Israel, French-Israeli project MAIMONIDE 31768XL, and by the French-Israeli Laboratory FILOFOCS.

†Additional support from the ANR project DISPLEXITY, and from the INRIA project GANG.

‡Supported in part by the Israel Science Foundation (grant No. 1444/14).

PODC'15, July 21–23, 2015, Donostia-San Sebastián, Spain
 ISBN 978-1-4503-3617-8 /15/07 $15.00
DOI:http://dx.doi.org/10.1145/2767386.2767421.

1. INTRODUCTION

Context and Objective.

Deciding the validity of a predicate over a distributed system (e.g., whether the nodes are properly colored, or whether the nodes have reached consensus), in a decentralized fashion, has received much attention over the years due to its applications to various domains, including checking the results obtained from the execution of a distributed program [7, 15, 30], establishing lower bounds on the time required for distributed approximation [10], estimating the complexity of logics required for distributed run-time verification [16], and general distributed complexity theory [14]. In the context of local computing in networks, a distributed decision is typically performed by letting each node inspect its state and the state of its neighbors, and return either TRUE or FALSE depending on whether this local configuration is consistent with a legal (global) state of the network. The decision is correct if all nodes return TRUE on legal states, and if at least one node returns FALSE on every illegal state. (A node returning FALSE could, e.g., launch a recovery procedure). For instance, deciding the correctness of the predicate asserting that the nodes are properly colored is straightforward: every node collects the colors of its neighbors, and returns TRUE if and only if none of these colors is the same as its own color.

Not all distributed network predicates can be decided locally directly. Consider, for example, deciding whether a set of edges constitute a spanning tree: locally, nodes cannot even distinguish between a path and a cycle [14]. To overcome this difficulty, system state is sometimes augmented with some additional information that allows nodes to decide locally the correctness of a global predicate. This is typically the case when checking the correctness of the output of a distributed program: in addition to its required output, each node can compute a local *label* used to verify the correctness of the global output. For instance, in an algorithm computing a spanning tree (i.e., the output at each node v is the identity of its parent in the tree), it is sufficient that every node additionally computes the identity of the root $r(v)$ and the distance from v to $r(v)$ in the tree [7, 22].

The notion of distributed verification as outlined above is formalized by the concept of *proof-labeling scheme*, introduced in [30]. A proof-labeling scheme for a predicate $\mathcal{P}$ consists of a *prover* and a *verifier*. The prover is an oracle which, for every legal state of the network, assigns a *label* $\ell(v)$ to every node v. The verifier is a distributed algorithm whose input, at each node v, consists of the local state of v, its label $\ell(v)$, and the label $\ell(w)$ of each of its neighbors w. The output of the verifier, at each node, is a a boolean value. The proof-labeling scheme is called *correct for predicate* $\mathcal{P}$ if the following two conditions hold: (1) For every legal state, the prover assigns labels to the nodes such that the verifier returns TRUE at ev-

ery node; (2) For every illegal state, and for every label assignment to the nodes, the verifier returns FALSE in at least one node.

The complexity measure used in evaluating the quality of a proof-labeling scheme is the *label size*. This measure captures the ongoing communication complexity of the scheme: In order to verify that the predicate holds, it is assumed that periodically, nodes transmit their labels to their neighbors, and each node runs the local verification procedure on the complete neighborhood's label set. Some predicates can be verified using labels whose sizes are of the same order of magnitude as the size of a node ID, like, e.g., $O(\log n)$-bit labels for spanning tree in n-node networks. However, some predicates require labels whose sizes are significantly larger than the size of an ID, like, e.g., $\Omega(\log^2 n)$ bits for minimum-weight spanning tree (MST) [28], and even $\Omega(n^2)$ bits for Symmetry (i.e., the existence of a non-trivial automorphism) [20]. Such a high overhead may be considered prohibitively expensive.

The main objective of the paper is to study the effect of randomization on the communication complexity of verification. For this purpose, we present and investigate *randomized* proof-labeling schemes. The idea is as follows. Every node v has a label $\ell(v)$. Using $\ell(v)$ and some coin tosses, node v computes a randomized certificate for each of its neighbors and sends it over. Given the certificates received from all neighbors, and its local label $\ell(v)$, node v executes a local deterministic verification procedure that returns TRUE or FALSE such that the following holds, for some arbitrarily small constant $\epsilon \in (0, \frac{1}{2})$ fixed a priori: (1) For every legal state, the prover assigns labels to the nodes such that the probability that the verifier returns TRUE at all nodes is at least $1 - \epsilon$, and (2) For every illegal state, and for every label assignment to the nodes, the probability that the verifier returns FALSE in at least one node is at least $1 - \epsilon$. We also consider the stronger 1-sided error scenario, in which the scheme is not allowed to err on legal instances, that is, in one-sided error schemes (2) remains the same, but we replace (1) with the requirement that for every legal state, the prover assigns labels to the nodes such that the verifier returns TRUE at all nodes with probability 1. We note that schemes with one-sided error may be preferable to those with two-sided errors if false alarms (i.e., a detection of error in a perfectly legal state) have very high cost. (All schemes we present in this paper have one-sided error.)

Intuitively, in the framework of checking the validity of outputs produced by an algorithm that is not completely trustworthy, or whose outputs may be corrupted somehow, a randomized proof-labeling scheme can be used to make sure that if the output is incorrect, then collectively, the nodes will be able to detect it. For that purpose, in addition to the output required by some problem specification (e.g., select an edge subset which forms an MST), the algorithms must also produce a label at each node, from which certificates are generated randomly. After exchanging the certificates between all pairs of neighbors, the randomized proof-labeling scheme is guaranteed (probabilistically) to return TRUE everywhere if and only if the outputs are correct w.r.t. the predicate describing the problem specification. This guarantee holds even in the face of adversarial labels in the following sense: if the outputs are incorrect, then there is no label assignment that guarantees (probabilistically) that the certificates will be accepted everywhere, while if the outputs are correct, and the labels are according to the specification, then with good probability, all local verification procedures will accept. If some node v returns FALSE (which occurs with small—0 in the one-sided case—probability on a legal instance, but with high probability on an illegal instance), then v may launch a recovery procedure or restart the algorithm.

Let us make a few remarks before summarizing our main results. First, observe that a randomized proof-labeling scheme does not exchange the labels between the nodes, but only the randomized certificates. It is thus expected that the communication complexity of randomized proof-labeling schemes be significantly reduced compared to the communication complexity of deterministic ones. Second, the exact value of the probability parameter ϵ is typically not very important. In particular, the correctness of all the schemes in this paper is oblivious to ϵ, in the sense that ϵ can be chosen as close to 0 as desired by straightforward adjustments of our schemes. In term of complexity, when ϵ is constant, the dependence on ϵ is confined to constant factors hidden by our asymptotic notation. Therefore, for the sake of concreteness, we present our schemes with success guarantee at least $\frac{2}{3}$ (i.e., for $\epsilon = \frac{1}{3}$).

Our Contributions.

In this paper we introduce and formalize the concept of *randomized* proof-labeling as outlined above. For this model we give the following universal result. Consider an n-node system, where each node has a k-bit state (a node state includes its identity, input, output and maybe more). We show that every predicate over such system has 1-sided error randomized proof-labeling scheme with certificates of $O(\log n + \log k)$ bits. Hence, assuming that states of nodes require $\text{poly}(n)$ bits to describe, our scheme insures that by exchanging only $O(\log n)$ bits, every (sequentially decidable) property can be probabilistically verified with success probability at least $\frac{2}{3}$. (In contrast, there are natural properties that require the exchange of $\omega(\log n)$ bits to be verified deterministically). Our universal logarithmic bound is tight, as we show by exhibiting a property for which any randomized proof-labeling scheme requires certificates of $\Omega(\log n + \log k)$ bits to verify. Our next generic constructions provides, by using randomization, an exponential improvement over deterministic schemes. More precisely, we prove that for any property that can be verified by a deterministic proof-labeling scheme using κ-bit messages, there is a randomized proof-labeling scheme using only $O(\log \kappa)$-bit messages. A nice corollary of this result is that there exists a randomized proof-labeling scheme for MST using $O(\log \log n)$-bit certificates.

In addition, we provide a general lower bound technique for the certificate size of a proof-labeling scheme. This technique is based on the novel notion called *graph crosses*. It applies to both deterministic and probabilistic schemes, generalizing known lower bounds for specific proof-labeling schemes. The probabilistic version applies to any 1-sided error randomized proof-labeling scheme. It also applies to 2-sided error schemes, under some additional constraints regarding the way the certificates are randomly generated by the nodes. Under these assumptions, the upper bound $O(\log \log n)$ bits on the certificate size for MST is tight.

Finally, we consider a few natural problems and properties and provide randomized proof-labeling schemes for them, with optimal or close-to-optimal certificate sizes. In particular, we show that the randomized verification complexity of acyclicity (and hence also a lower bound for MST), as well as the verification complexity of biconnectivity, is $\Theta(\log \log n)$. We also consider the randomized verification complexity of two versions of the longest cycle problem. For the question of deciding whether there is a cycle of length at least c, we give an upper bound of $O(\log \log n)$ and a lower bound of $\Omega(\log \log c)$. For the complementary question ("yes" if all cycles are smaller than c), we show that $\Omega(\log \frac{n}{c})$ and of $\Omega(\log \log \frac{n}{c})$ bits are required for certificates in deterministic and randomized proof-labeling schemes, respectively.

Related Work.

Labeling schemes were studied extensively in the past, in two directions: *informative labeling schemes*, and *proof-labeling schemes*.

In the framework of informative labeling schemes, one is given a function f on pairs (or sets) of nodes, and $(\mathcal{M}, \mathcal{D})$ is an f-labeling scheme for a graph family $\mathcal{F}$ if $\mathcal{M}$, the *marker*, is an algorithm that, given a graph $G = (V, E)$ in $\mathcal{F}$, assigns a label $\ell(v)$ to every $v \in V$, and $\mathcal{D}$, the *decoder*, is an algorithm that satisfies $\mathcal{D}(\ell(u), \ell(v)) = f(u, v)$ for every pair of nodes in G. The main performance criterion is the size of the labels, which should be as small as possible. Since the seminal work of Kannan, Naor, and Rudich [23] on adjacency-labeling, there have been quite a lot of investigations, for a large set of functions f, including the following: adjacency [4, 23], distance [2, 9, 18, 17, 19, 24, 26, 34, 36], connectivity and flow [25, 27], nearest common ancestor [3, 35], etc. In particular, the notion of universal f-matrices for several functions f was introduced in [31], and used to construct upper and lower bounds on the sizes of the corresponding f-labeling schemes. Most investigations related to the design of compact routing tables can also be placed in the framework of informative labeling. This includes, e.g., the papers [11, 12, 37] on routing in trees.

Proof-labeling schemes are not dealing with computing a function, but with verifying a proof that the given instance satisfies some given boolean predicate. This proof is distributed among the nodes under the form of labels assigned to the node by a *prover* which assigns a label to every node. The *verifier* is a distributed algorithm in charge of verifying the distributed proof. As for informative labeling scheme, the main performance criterion is the size of the labels. This concept was introduced by Korman, Kutten, and Peleg in [30]. Among the results that were presented in this paper, it is worth mentioning the $\Theta(\log n)$ bit bound on the verification complexity of acyclicity and the upper bound $O(\log^2 n + \log n \log W)$ bits for MST, where W is the maximal possible weight of an edge. This bound was improved to $O(\log n \log W)$ bits in [28], where a matching lower bound of $\Omega(\log n \log W)$ bits is established for $W > \log n$. It is worth noticing that proof-labeling schemes are closely related to self stabilizing algorithms, that is, algorithms which have to periodically verify the correctness of the system state. See, e.g., [1] where the notion of local detection was introduced and used for designing a self stabilizing protocol constructing a spanning tree, and [29] for another example of using distributed local verification of proofs for the design of self stabilizing algorithms. The reader interested in the tight connections between proof-labeling schemes and self-stabilization is referred to the recent paper [8]. Proof-labeling schemes, where nodes may communicate at distance greater than 1, i.e., may take their individual decision based on the labels of the nodes in their vicinity at distance $t > 1$, was recently studied in [20]. Finally, distributed decision and verification processes in which the global interpretation of the collection of individual outputs is not restricted to be the logical conjunction of these outputs has been studied in [5, 6].

To our knowledge, there are very few papers dealing with randomization in the framework of informative labeling schemes, or proof-labeling schemes. Randomized informative schemes for trees, including randomized schemes for adjacency and ancestry, were presented in [13]. The crucial difference between our work and this latter work is the following. In our approach, a label is stored at every node, based on which each node produces a random certificate, and we are interested in minimizing the size of the certificates. Instead, the approach in [13] is more restrictive, as each node label is randomly computed and stored at each node, and the measure is the size of these labels. More recently, [14] provided a framework that could be used for setting up a complexity theory for local distributed computing. This framework includes several complexity classes, including NLD (for non-deterministic local decision) and BPNLD (for bounded probability NLD). The former is a generalization of proof-labeling schemes (with slight differences, including the fact that certificates should be independent of the IDs), and the latter is a randomized version of the former. Nevertheless, in both cases, the emphasis was put on the existence of a proof, and not on its size. In fact, it is proved that all (decidable) languages are in BPNLD, but the proof of this result involves labels as large as $\text{poly}(n)$ bits. In contrast, the randomized proof-labeling schemes described in this paper involve labels of poly-logarithmic size.

The conceptual difference between our approach and the approach of [13, 14] is significant. In [13, 14], the prover is randomized, while the verifier is deterministic, and the measure is *space complexity* (the size of the labels). Instead, in this paper, the randomization part is also delegated to the verifier, and the measure is *communication complexity* (the size of the exchanged certificates).

2. MODEL AND DEFINITIONS

2.1 Computational Framework

A network is modeled as a connected graph $G = (V, E)$, without self-loops or multiple edges. Recall that two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a bijection $\sigma : V_1 \to V_2$ such that: $\{u, v\} \in E_1 \iff \{\sigma(u), \sigma(v)\} \in E_2$. We assume that the edges incident to a node v are numbered in sequence $1, \ldots, \deg(v)$, where $\deg(v)$ is the *degree* of v. The number of e at v is the *port number* of e at v. An edge may have different port numbers on its two endpoints.

In a *configuration* G_s, we are given a graph $G = (V, E)$, a *state* space S, and a state assignment function $s : V \to S$. The state of a node v, denoted $s(v)$, includes all local input to v. In particular, the state may include the node identity $\text{Id}(v)$ (if the network is not anonymous) and weights of its incident edges (for edge-weighted networks). The state of v may also include other data like, e.g., the result of an algorithm.

Mechanisms such as proof-labeling schemes involve simple distributed algorithms, acting in one synchronous communication round and computation, in which every node sends a value to each of its neighbors, and, upon reception of the values from all its neighbors, every node computes an output. In the context of proof-labeling schemes, this output is either TRUE or FALSE.

Unless specified otherwise, we always assume non-anonymous networks, i.e., every node v is provided with an identity $\text{Id}(v)$, that is part of the state of v. All identities in the same network are pairwise distinct. Nevertheless, the definition of proof-labeling schemes does not need the presence of identities.

2.2 Deterministic and Randomized Proof-Labeling Schemes

We first recall the definition of deterministic proof-labeling schemes (abbreviated PLS henceforth), as introduced in [30]. Given a family $\mathcal{F}$ of configurations, and a boolean predicate $\mathcal{P}$ over $\mathcal{F}$, a PLS for $(\mathcal{F}, \mathcal{P})$ is a mechanism for deciding $\mathcal{P}(G_s)$ for every $G_s \in \mathcal{F}$. A PLS consists of two components: a *prover* **p**, and a *verifier* **v**. The prover is an oracle which, given any configuration $G_s \in \mathcal{F}$, assigns a bit string $\ell(v)$ to every node v, called the *label* of v. The verifier is a decentralized algorithm running concurrently at every node. At each node v, it takes as input the state $s(v)$ of v, its label $\ell(v)$ and the labels of all its neighbors, i.e., the ordered set $\{\ell(w_i) \mid i = 1, \ldots, \deg(v)\}$ where w_i is the neighbor reachable from v through the edge with port number i. The verifier **v** at each node outputs a boolean. If the outputs are TRUE at all nodes, **v** is said to *accept* the configuration, and otherwise (i.e., **v** outputs FALSE in at least one node) **v** is said to *reject* the configuration. For

correctness, a proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ for $(\mathcal{F}, \mathcal{P})$ must satisfy the following requirements, for every $G_s \in \mathcal{F}$:

- If $\mathcal{P}(G_s) = \text{TRUE}$ then, using the labels assigned by $\mathbf{p}$, the verifier $\mathbf{v}$ accepts G_s.
- If $\mathcal{P}(G_s) = \text{FALSE}$ then, for every label assignment, the verifier $\mathbf{v}$ rejects G_s.

The *verification complexity* of a deterministic proof-labeling scheme $(\mathbf{p}, \mathbf{v})$, denoted by κ, is the maximal length of the labels assigned by the prover $\mathbf{p}$ on a legal configuration $G_s \in \mathcal{F}$ (i.e., a configuration satisfying $\mathcal{P}$).

In this paper we extend the above definition to *randomized proof labeling schemes* (RPLS). The idea is to allow randomization in the verification part of the scheme. Specifically, an RPLS is defined as follows. The goal and the prover in an RPLS remain exactly as defined for PLS. However, in an RPLS the verifier $\mathbf{v}$ has access to a source of independent random bits at each node. At node v, using the label $\ell(v)$ and the private random bits available at v, the verifier produces a random bit string, called *certificate*, for each of its neighbors. The random certificate of v for w_i, the neighbor reachable through port i, is denoted by $c_i(v)$. In an RPLS, *only the certificates are communicated for verification*. More precisely, the input of the verifier at node v consists of its state $s(v)$, its label $\ell(v)$, and all the certificates received from its neighbors, i.e., the collection $\{c_{p_i}(w_i),\ i = 1, \ldots, \deg(v)\}$ where w_i is the neighbor reachable from v through the edge e_i with port number i at v, and p_i is the port number of e_i at w_i. A randomized scheme $(\mathbf{p}, \mathbf{v})$ for $(\mathcal{F}, \mathcal{P})$ must satisfy the following requirements, for every $G_s \in \mathcal{F}$:

- If $\mathcal{P}(G_s) = \text{TRUE}$ then, using the labels assigned by $\mathbf{p}$, $\Pr[\mathbf{v} \text{ accepts } G_s] \geq p_{\text{accept}}$.
- If $\mathcal{P}(G_s) = \text{FALSE}$ then, for every label assignment, $\Pr[\mathbf{v} \text{ rejects } G_s] \geq p_{\text{reject}}$.

Following the sequential complexity classes RP and BPP (see, e.g., [33]), we define two flavors of RPLSs: in *one-sided error* RPLS, we have $p_{\text{accept}} = 1$ and $p_{\text{reject}} = \frac{1}{2}$; and in *two-sided error* RPLS, we have $p_{\text{accept}} = p_{\text{reject}} = \frac{2}{3}$.[1] Unless explicitly stated otherwise, in this paper we refer by RPLS to two-sided RPLS.

Clearly, RPLSs give weaker guarantees than deterministic PLSs. The main reason to prefer an RPLS over a PLS is the possible saving in verification complexity, defined next.

DEFINITION 2.1. *The* verification complexity *of a randomized proof-labeling scheme $(\mathbf{p}, \mathbf{v})$, denoted by κ, is the maximal length of the (random) certificates generated by the (randomized) verifier $\mathbf{v}$ based on the labels assigned to the nodes by the prover $\mathbf{p}$ on a legal configuration $G_s \in \mathcal{F}$ (i.e., a configuration satisfying $\mathcal{P}$).*

3. UNIVERSAL SCHEMES

In this section we show that in RPLS, one can save exponentially in the verification complexity w.r.t. PLS. Using this result, we derive a universal RPLS and prove that there is no better one. We start with a reduction from RPLS to PLS.

THEOREM 3.1. *Let $\mathcal{F}$ be a family of configurations, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. If there is a PLS for $(\mathcal{F}, \mathcal{P})$ with verification complexity κ, then there is a one-sided RPLS for $(\mathcal{F}, \mathcal{P})$ with verification complexity $O(\log \kappa)$.*

The proof of this theorem uses a result about (2-party) communication complexity. In a 2-party communication complexity problem there are two players, Alice and Bob. Alice receives as input a λ-bit string x and Bob receives another λ-bit string y. The goal is for Alice and Bob to compute a certain function $f(x, y)$ by exchanging the smallest possible number of bits. For any two λ-bit strings x and y, let $\text{EQ}(x, y)$ denote the equality predicate. The following fact is well known (see [32]).

LEMMA 3.2. *The randomized communication complexity of deciding EQ over λ-bit strings is $\Theta(\log \lambda)$.*

This lemma implies the following upper bound.

LEMMA 3.3 (BASED ON [32]). *There exists a 2-party randomized protocol π for EQ over λ-bit strings, in which one side sends $O(\log \lambda)$ bits, and the other side then decides the outcome, such that for any input strings a, b we have that $\Pr[\pi(a,b)=\text{TRUE} \mid a=b] = 1$ and $\Pr[\pi(a,b)=\text{FALSE} \mid a \neq b] > 2/3$.*

Proof of Theorem 3.1: Let $(\mathbf{p}, \mathbf{v})$ be a proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$ with verification complexity κ. We construct a randomized proof-labeling scheme $(\mathbf{p}', \mathbf{v}')$ for $(\mathcal{F}, \mathcal{P})$ as follows. Let $G_s \in \mathcal{F}$ be a configuration satisfying predicate $\mathcal{P}$. For every node v, the prover $\mathbf{p}'$ sets the label of v as the vector of labels

$$\ell'(v) = \big(\ell(v), \ell(w_1), \ldots, \ell(w_d)\big)$$

where $w_1, \ldots, w_d$ are the $d = \deg(v)$ neighbors of v in G ordered by port number, and $\ell(\cdot)$ is the label assignment for G_s as provided by $\mathbf{p}$. Next, let π be the protocol for EQ from Lemma 3.3. Recall that in π, there is a *sender* and a *decider*. The certificate sent by node v to each of its neighbors under $(\mathbf{p}', \mathbf{v}')$ is defined to be exactly the message sent by the "sender" under π on input $\ell(v)$. The decision of the local verifier $\mathbf{v}'$ is as follows. If the format of the local label is not as expected, the local verifier outputs FALSE immediately. Otherwise, given the received certificates, the local verifier runs the "decider" part of π for each neighbor. If any of these instances of π returns FALSE, then the local verifier returns FALSE. Otherwise, the complete label $\ell'(v)$ (which is a vector) is fed into the local deterministic verifier $\mathbf{v}$ (assumed to be given), and the output of $\mathbf{v}'$ will be in this case the output of $\mathbf{v}(\ell'(v))$. This concludes the construction of $(\mathbf{p}', \mathbf{v}')$.

The logarithmic verification complexity of $(\mathbf{p}', \mathbf{v}')$ follows directly from the construction and from Lemma 3.3. We now show correctness of the scheme. Suppose first that G_s satisfies the predicate. Then, with the labels assigned by $\mathbf{p}'$, we have that, for every node v, $\ell'(v) = \big(\ell(v), \ell(w_1), \ldots, \ell(w_d)\big)$, with the labels $\ell(\cdot)$ assigned by $\mathbf{p}$. Therefore, by Lemma 3.3 and by construction, π returns TRUE with probability 1 in all d instances run by v. It follows that $\mathbf{v}'$ applies $\mathbf{v}$ at node v on $\ell'(v)$, for which $\mathbf{v}$ returns TRUE at v. Hence, $\mathbf{v}'$ returns TRUE at v as well.

Suppose now that G_s does not satisfy the predicate. If any node v has a label that does not comply with the expected format, the verifier $\mathbf{v}'$ returns FALSE at v and we are done. So assume that every node v has a label of the form $\big(\ell_0^v, \ell_1^v, \ldots, \ell_d^v\big)$ where $d = \deg(v)$. We first note that if the labels are *consistent* in the sense that, for every node v and its ith neighbor w, $\ell_i^v = \ell_0^w$, then $\mathbf{v}'$ returns the value returned by $\mathbf{v}$, which must be FALSE at some node by correctness of $\mathbf{v}$. If the labels are not consistent in the above sense, then there is at least one pair of adjacent nodes $\{v, w\}$ such that w is the ith neighbor of v, and $\ell_i^v \neq \ell_0^w$. In this case, by Lemma 3.3, the local verifier at v outputs FALSE with probability at least $2/3$, as required. ∎

Next, we note the existence of a universal PLS construction, where the label of every node is a representation of the graph configuration, summarized in the following lemma.

[1] The choice of $1/2$ and $2/3$ is somewhat arbitrary. The idea is that we can boost the probability of correctness to $1 - \delta$ by repeating the verification procedure $O(\log(1/\delta))$ times independently and outputting the majority of outcomes.

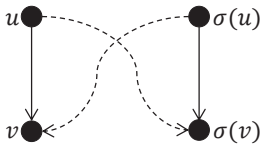

Figure 1: *Crossing two edges under an isomorphism σ. Solid edges are the edges of G, and dashed edged are the edges of $\sigma_{\bowtie}(G)$.*

LEMMA 3.4 ([20, 30]). *Let $\mathcal{F}$ be a family of configurations with states in S, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Assume that every state in S can be represented using $k = k(n)$ bits in n-node networks. There exists a PLS for $(\mathcal{F}, \mathcal{P})$ with verification complexity $O(nk + \min\{n^2, m \log n\})$, where m is the number of edges in the network.*

Combining Theorem 3.1 and Lemma 3.4 we obtain the following universal result for RPLSs.

COROLLARY 3.5. *Let $\mathcal{F}$ be a family of configurations with states in S, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Assume that every state in S can be represented using $k = k(n)$ bits in n-node networks. There exists a randomized proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$ with verification complexity $O(\log n + \log k)$.*

The upper bound in Corollary 3.5 is tight, as stated below.

THEOREM 3.6. *For any function $k(n)$, there exist a family $\mathcal{F}$ of configurations with states on $k = k(n)$ bits in n-node graphs, and a predicate $\mathcal{P}$ over $\mathcal{F}$ such that any randomized proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$ has verification complexity $\Omega(\log n + \log k)$.*

The proof, which is omitted from this extended abstract, is by reduction from 2-party EQ. The idea is that for any function $k(n)$, we can show a family $\mathcal{F}$ of configurations with states on $k = k(n)$ bits in n-node graphs, and a predicate $\mathcal{P}$ over $\mathcal{F}$ such that the following holds. Any randomized proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$, with verification complexity κ, can be used to construct a 2-party communication protocol for EQ of n-bit strings and of k-bit strings, with error probability of at most $1/3$, using κ bits of communication. Then, from Lemma 3.2, we get the desired lower bound.

4. GENERIC LOWER BOUNDS

In this section we formalize a general tool that can be used to prove lower bounds for both deterministic and randomized proof-labeling schemes. The general idea was used many times in the past, most relevantly in [30].

4.1 A General Tool: Edge Crossing

We start with a technical definition, and then define our main concept.

DEFINITION 4.1. *Let $G = (V, E)$ be a graph and let $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ be two subgraphs of G. H_1 and H_2 are* independent *if and only if $V_1 \cap V_2 = \emptyset$ and $E \cap (V_1 \times V_2) = \emptyset$.*

The following definition is illustrated in Figure 1.

DEFINITION 4.2 (CROSSING). *Let $G = (V, E)$ be a graph, and let $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ be two independent isomorphic subgraphs of G with isomorphism $\sigma : V_1 \to V_2$. The* crossing *of G induced by σ, denoted by $\sigma_{\bowtie}(G)$, is the graph obtained from G by replacing every pair of edges $\{u, v\} \in E_1$ and $\{\sigma(u), \sigma(v)\} \in E_2$, by the pair $\{u, \sigma(v)\}$ and $\{\sigma(u), v\}$.*

Crossing can be very useful in proving lower bounds on the verification complexity of both deterministic and randomized proof labeling schemes. We start by showing the simpler case of deterministic PLSs.

PROPOSITION 4.3. *Let $(\mathbf{p}, \mathbf{v})$ be a deterministic PLS for $(\mathcal{F}, \mathcal{P})$ with verification complexity κ. Suppose that there is a configuration $G_s \in \mathcal{F}$ where G contains r pairwise independent isomorphic subgraphs $H_1, \ldots, H_r$ with s edges each, and let $\sigma_i : H_1 \to H_i$ be a port-preserving isomorphism for $i \in \{1, \ldots, r\}$. If $\kappa < \frac{\log r}{2s}$, then there are $1 \leq i < j \leq r$ such that G_s is accepted by $(\mathbf{p}, \mathbf{v})$ if and only if $(\sigma_j \circ \sigma_i^{-1})_{\bowtie}(G)_s$ is accepted by $(\mathbf{p}, \mathbf{v})$.*

Proof: Let G_s be a configuration as described in the statement. Assume that $\kappa < \frac{\log r}{2s}$, and consider a collection $\{\sigma_i : V(H_1) \to V(H_i), i = 1, \ldots, r\}$ of r port-preserving isomorphisms. Order the nodes of H_1 arbitrarily. This order induces an order on the nodes of H_i thanks to σ_i. For every i, consider the bit-string L_i constructed by concatenating the labels given by $\mathbf{p}$ to the nodes of H_i, in the order induced by σ_i. We have $|L_i| < \log r$ for every i because $|V(H_i)| \leq 2s$, and thus there are less than r possible distinct L_i's in total. Therefore, by the pigeonhole principle, there are $i \neq j$ such that $L_i = L_j$. Define $\sigma = \sigma_j \circ \sigma_i^{-1}$, and consider the output of the verifier $\mathbf{v}$ in G_s and in $\sigma_{\bowtie}(G)_s$.

Suppose G_s is accepted by $(\mathbf{p}, \mathbf{v})$, i.e., with the labels provided by $\mathbf{p}$, the verifier $\mathbf{v}$ outputs TRUE at all nodes of G. Therefore, all nodes outside H_i and H_j also output TRUE in $\sigma_{\bowtie}(G)_s$. Now, let v be a node in H_i. Each neighbor w of v from H_i is replaced in $\sigma_{\bowtie}(G)_s$ by the node $\sigma(w)$ from H_j. Since w and $\sigma(w)$ have the same label, the verifier acts the same at v in both G_s and $\sigma_{\bowtie}(G)_s$. Therefore $\mathbf{v}$ outputs TRUE at every node of H_i in $\sigma_{\bowtie}(G)_s$. For the same reason, the verifier acts the same at every node v of H_j, in both G_s and $\sigma_{\bowtie}(G)_s$. Therefore, the verifier also outputs TRUE at all nodes in $\sigma_{\bowtie}(G)_s$, which implies that $\sigma_{\bowtie}(G)_s$ is accepted by $(\mathbf{p}, \mathbf{v})$.

Similarly, if G_s is rejected by $(\mathbf{p}, \mathbf{v})$ then, for any label-assignment to the nodes, the verifier $\mathbf{v}$ outputs FALSE in at least one node v of G. If this node v is not in H_i or H_j, then $\mathbf{v}$ also outputs FALSE at v in $\sigma_{\bowtie}(G)_s$. If this node v belongs to one of the two subgraph H_i or H_j, then, since the verifier acts the same at v in both G_s and $\sigma_{\bowtie}(G)_s$, $\mathbf{v}$ also outputs FALSE at this node in $\sigma_{\bowtie}(G)_s$, which implies that $\sigma_{\bowtie}(G)_s$ is rejected by $(\mathbf{p}, \mathbf{v})$. The proposition follows. ∎

Proposition 4.3 has the following useful consequence.

THEOREM 4.4. *Let $\mathcal{F}$ be a family of configurations, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Suppose that there is a configuration $G_s \in \mathcal{F}$ satisfying that (1) G contains as subgraphs r pairwise independent isomorphic copies $H_1, \ldots, H_r$ with s edges each, and (2) there exist r port-preserving isomorphisms $\sigma_i : V(H_1) \to V(H_i)$ such that for every $i \neq j$, the isomorphism $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$ satisfies $\mathcal{P}(G_s) \neq \mathcal{P}(\sigma^{ij}{}_{\bowtie}(G)_s)$. Then the verification complexity of any proof-labeling scheme for $(\mathcal{F}, \mathcal{P})$ is $\Omega(\frac{\log r}{s})$.*

Remark: Note that Theorem 4.4 cannot yield lower bounds greater than $\Omega(\log n)$, because $r = O(n)$.

4.2 Generic Lower Bounds for Randomized Proof-Labeling Schemes

We now proceed with a generalization of Theorem 4.4 to randomized proof-labeling schemes. First we define *edge-independent* RPLSs.

DEFINITION 4.5. *An RPLS* $(\mathbf{p},\mathbf{v})$ *is called* edge-independent *if the verifier* $\mathbf{v}$ *uses independent random bits for each certificate* $c_i(v)$*, for all edges* e_i *incident to* v*,* $i=1,\ldots,\deg(v)$.

We can prove the following result for edge-independent two-sided error RPLSs.

PROPOSITION 4.6. *Let* $(\mathbf{p},\mathbf{v})$ *be an edge-independent RPLS for* $(\mathcal{F},\mathcal{P})$ *with verification complexity* κ*. Assume that there is a configuration* $G_s \in \mathcal{F}$ *with* $\mathcal{P}(G_s) =$ TRUE *such that* G *contains* r *pairwise independent isomorphic subgraphs* $H_1,\ldots,H_r$ *with* s *edges each, and let* $\sigma_i : H_1 \to H_i$ *be a port-preserving isomorphism for each* $i \in \{1,\ldots,r\}$*. If* $\kappa < (\frac{1}{2s} - o(1))\log\log r$*, then there are* $1 \le i < j \le r$ *such that* G_s *is accepted by* $(\mathbf{p},\mathbf{v})$ *if and only if* $(\sigma_j \circ \sigma_i^{-1})_{\bowtie}(G)_s$ *is accepted by* $(\mathbf{p},\mathbf{v})$.

Proof: We start the proof by a collection of technical preliminary results. Given a real number x and $0 < \epsilon \le 1$, we denote by $\lfloor x \rfloor_\epsilon$ the value of x rounded down to the closest integer multiple of ϵ, i.e., $\lfloor x \rfloor_\epsilon \stackrel{\text{def}}{=} \left\lfloor \frac{x}{\epsilon} \right\rfloor \epsilon$. Given a real function $f : X \to \mathbb{R}$ over a set X, and $\epsilon > 0$, we define the *ϵ-rounded* $f_\epsilon : X \to \mathbb{R}$ of f by $f_\epsilon(x) \stackrel{\text{def}}{=} \lfloor f(x) \rfloor_\epsilon$ for every $x \in X$.

We shall consider ϵ-rounded probability distributions. Note that an ϵ-rounded probability distribution is not necessarily a probability distribution, because it may not sum up to 1. However, it has the following probabilistic interpretation. Let X be a set, and let $\Pr$ and $\Pr'$ be two probability distributions over X. Then

$$\Pr_\epsilon = \Pr'_\epsilon \;\Rightarrow\; \forall Y \subseteq X, \left|\Pr[Y] - \Pr'[Y]\right| \le \epsilon|X| \,. \qquad (1)$$

Indeed, if $\Pr_\epsilon[x] = \Pr'_\epsilon[x]$ for every $x \in X$, then we have that $\left|\Pr[x] - \Pr'[x]\right| < \epsilon$ for every x. Using the triangle inequality we thus get $\left|\Pr[Y] - \Pr'[Y]\right| \le \sum_{x\in Y} \left|\Pr[x] - \Pr'[x]\right| < \epsilon|Y| \le \epsilon|X|$.

The advantage of ϵ-rounded distributions is that there are not too many. Indeed, let X be a finite set, and let $\mathcal{D}$ be the set of all probability distributions over X. The number of distinct ϵ-rounded distributions over X is at most $2^{|X|}\epsilon^{-|X|}$, that is,

$$|\{\Pr_\epsilon \mid \Pr \in \mathcal{D}\}| \le (2/\epsilon)^{|X|}. \qquad (2)$$

This is because the set of ϵ-rounded distributions is a subset of the functions from X to $\{\epsilon i, i = 0, 1 \ldots, \lfloor 1/\epsilon \rfloor\}$, which implies that their number is at most $(1+\frac{1}{\epsilon})^{|X|} \le 2^{|X|}\epsilon^{-|X|}$.

We now have all the ingredients we need to prove the proposition. Let G_s be a configuration and $\{\sigma_i\}_{i=1}^r$ be the isomorphisms as described in the proposition. Note that for any $i \ne j$, if G_s is labeled by $\mathbf{p}$ and the crossing with $\sigma = \sigma_j \circ \sigma_i^{-1}$ satisfies

$$\left|\Pr[\mathbf{v} \text{ accepts } G_s] - \Pr[\mathbf{v} \text{ accepts } \sigma_{\bowtie}(G)_s]\right| < \frac{1}{3}, \qquad (3)$$

then G_s is accepted by $(\mathbf{p},\mathbf{v})$ if and only if $\sigma_{\bowtie}(G)_s$ is accepted by $(\mathbf{p},\mathbf{v})$. We prove that if $\kappa < (\frac{1}{2s} - o(1))\log\log r$, then there exist $i \ne j$ that satisfy Eq. (3). We use a counting argument. Order the edges of H_1 arbitrarily, and obtain, using σ_i, an ordering for each H_i. Assume w.l.o.g. that all certificates are exactly κ bits long. Then, using the order we defined, there is a 1-1 correspondence between each $2\kappa s$ bit string and each particular choice of certificates communicated in any H_i. Let $\mathcal{D}$ denote the set of distributions over $2\kappa s$-long bit strings, and define

$$\epsilon = 1/(12s \cdot 2^{2s\kappa}) \,.$$

Consider the set of distributions in $\mathcal{D}$, ϵ-rounded. Since there are at most $2^{2s\kappa}$ bit strings of length $2\kappa s$, from Eq. (2) we conclude that there are no more than $(2/\epsilon)^{(2^{2s\kappa})}$ such ϵ-rounded distributions. Let us now make the following technical observation. Let $\alpha \ge 1$, $\beta \ge 1$ and $\gamma \ge 2$ be such that $\log(\beta + \log\alpha) = o(\log\log\gamma)$. Then

$$\beta < (1 - o(1))\log\log\gamma \;\Rightarrow\; \gamma > (\alpha 2^\beta)^{(2^\beta)}.$$

This follows from the fact that

$$\begin{aligned} \beta < \log\log\gamma - \log(\beta + \log\alpha) \quad &\Longleftrightarrow \quad 2^\beta(\beta + \log\alpha) < \log\gamma \\ &\Longleftrightarrow \quad (\alpha 2^\beta)^{(2^\beta)} < \gamma. \end{aligned}$$

By setting $\alpha = 24s$ and $\beta = 2s\kappa$ we obtain that the number of ϵ-rounded distributions satisfies

$$(2/\epsilon)^{(2^{2s\kappa})} = \left(24s \cdot 2^{2s\kappa}\right)^{(2^{2s\kappa})} < r.$$

Therefore, by the pigeonhole principle, it must be the case that among $H_1,\ldots,H_r$ there are H_i and H_j, where $i \ne j$, with identical ϵ-rounded distributions over the certificates.

Now, let $\sigma = \sigma_j \circ \sigma_i^{-1}$. For any $u \in H_i$, we say that u and $\sigma(u) \in H_j$ are *siblings*. Consider running $\mathbf{v}$ on G_s and on $\sigma_{\bowtie}(G)_s$, where, in both cases, we assume that the correct labels are given to the nodes by prover $\mathbf{p}$ applied to G_s. This means that in both G_s and $\sigma_{\bowtie}(G)_s$, the distributions of certificates sent by sibling nodes are the same. However, the distributions of certificates *received* by sibling nodes may have changed (albeit only slightly, as we show next), due to crossing H_i with H_j.

To analyze $\mathbf{v}$ on G_s and $\sigma_{\bowtie}(G)_s$, we change the certificates sent in G_s to those sent in $\sigma_{\bowtie}(G)_s$ inductively, and show that each such modification results only in a small change in the probability of acceptance. We view $G = (V,E)$ as a symmetric directed graph, i.e., each edge $e = \{u,v\} \in E$ is viewed as two symmetric arcs (u,v) and (v,u). Let us order these arcs in G arbitrarily, and let $\mathcal{C}$ denote the set of certificate vectors $\vec{c}$ for which the verifier $\mathbf{v}$ accepts G_s, with coordinates ordered according to the fixed order of the arcs. Consider a node v in H_i, and one of its incoming arcs (u,v) in H_i. Let u' and v' in H_j be the respective siblings of u and v. Assume, w.l.o.g., that $(u,v) = e_1$. Let $\vec{c} \in \mathcal{C}$. The certificate sent to v by u along e_1 is therefore c_1. Let $\vec{c}_{-1}$ be the vector $\vec{c}$ with the first coordinate c_1 omitted, i.e., $\vec{c}_{-1}$ is one dimension less than $\vec{c}$. Denote by A the event where V sends $\vec{c}_{-1}$ on $E \setminus \{e_1\}$. Using the above notations, we have:

$$\begin{aligned} &\Pr[\mathbf{v} \text{ accepts } G_s] \\ &\quad = \sum_{\vec{c}\in\mathcal{C}} \Pr[\text{nodes in } V \text{ send } \vec{c} \text{ on } E] && (4) \\ &\quad = \sum_{\vec{c}\in\mathcal{C}} \Big(\Pr[u \text{ sends } c_1 \text{ on } e_1] \cdot \Pr[A] \Big) && (5) \\ &\quad > \sum_{\vec{c}\in\mathcal{C}} \Big(\big(\Pr[u' \text{ sends } c_1 \text{ on } (u',v')] - \epsilon \big) \cdot \Pr[A] \Big) && (6) \end{aligned}$$

Eq. (4) is by definitions. Eq. (5) follows from the independence of c_1 from $\vec{c}_{-1}$ by our assumption of edge independence of $\mathbf{v}$. Eq. (6) follows from Eq. (1) since u and u' have the same ϵ-rounded distribution over their certificates. Let G'_s be the virtual labeled configuration consisting of G_s labeled by $\mathbf{p}$ but where the certificate distribution sent by u along e_1 is changed to the distribution of certificates sent by u' along (u',v') in G_s. We get that

$$\begin{aligned} \Pr[\mathbf{v} \text{ accepts } G_s] &> \Pr[\mathbf{v} \text{ accepts } G'_s] - \epsilon \sum_{\vec{c}\in\mathcal{C}} \Pr[A] && (7) \\ &\ge \Pr[\mathbf{v} \text{ accepts } G'_s] - \epsilon\, 2^\kappa && (8) \end{aligned}$$

Eq. (7) is by definition of G'_s, and Eq. (8) follows from the observation that the second sum in Eq. (7) is at most 2^κ since the number

of distinct c_1 values is at most 2^κ, and, for any fixed certificate value γ,

$$\sum_{\vec{c}\in\mathcal{C}, c_1=\gamma} \Pr[V \text{ sends } \vec{c}_{-1} \text{ on } E\setminus\{e_1\}] \leq 1.$$

We repeat the same process for the certificate sent along another arc (a,b) of H_i, resulting in a virtual configuration G_s'' in which we replace the distribution of the certificates sent by a to b by the distribution of the certificates sent by a' to b', where a' and b' are the respective siblings of a and b in H_j. Again, we get

$$\Pr[\mathbf{v} \text{ accepts } G_s'] > \Pr[\mathbf{v} \text{ accepts } G_s''] - \epsilon 2^\kappa.$$

By repeating the process $4s$ times, once for every arc in H_i and in H_j, we eventually get

$$\Pr[\mathbf{v} \text{ accepts } G_s] > \Pr[\mathbf{v} \text{ accepts } \sigma_{\bowtie}(G)_s] - 4\epsilon s 2^\kappa.$$

Moreover, by switching the roles of G_s and $\sigma_{\bowtie}(G)_s$ in the analysis, we also get that,

$$\Pr[\mathbf{v} \text{ accepts } \sigma_{\bowtie}(G)_s] > \Pr[\mathbf{v} \text{ accepts } G_s] - 4\epsilon s 2^\kappa.$$

I.e., $\Pr[\mathbf{v} \text{ accepts } \sigma_{\bowtie}(G)_s]$ and $\Pr[\mathbf{v} \text{ accepts } G_s]$ differ by less than $\pm 4\epsilon s 2^\kappa$. By the choice of ϵ, we conclude that

$$\left|\Pr[\mathbf{v} \text{ accepts } G_s] - \Pr[\mathbf{v} \text{ accepts } \sigma_{\bowtie}(G)_s]\right| < \frac{1}{3},$$

which completes the proof. ∎

The following corollary of Proposition 4.6 is the way we use to bound the verification complexity of two-sided error, edge independent RPLSs.

THEOREM 4.7. *Let $\mathcal{F}$ be a family of configurations, and let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. If there is a configuration $G_s \in \mathcal{F}$ satisfying that (1) G contains r pairwise independent isomorphic subgraphs $H_1,\ldots,H_r$ with s edges each, and (2) there exist r port-preserving isomorphisms $\sigma_i : V(H_1) \to V(H_i)$ such that $\mathcal{P}(G_s) \neq \mathcal{P}(\sigma_{\bowtie}(G)_s)$ for every isomorphism $\sigma = \sigma_j \circ \sigma_i^{-1}$ between H_i and H_j, for $1 \leq i \neq j \leq r$, then the verification complexity of any edge-independent RPLS for $(\mathcal{F},\mathcal{P})$ is $\Omega(\frac{\log\log r}{s})$.*

Again, we note that since $r = O(n)$, Theorem 4.7 cannot be used to prove lower bounds greater than $\Omega(\log\log n)$.

Lower bounds for one-sided RPLSs.

We can replace the assumption of edge-independent RPLS in Theorem 4.7 by the requirement that the RPLS is *one sided* and obtain essentially the same lower bound. Recall that in one-sided RPLS we insist that if $\mathcal{P}(G_s) = \text{TRUE}$, then the verifier $\mathbf{v}$ must accept *always*, i.e., with probability 1. For this case we have the following proposition, whose proof is much simpler.

PROPOSITION 4.8. *Let $(\mathbf{p},\mathbf{v})$ be a one-sided RPLS for $(\mathcal{F},\mathcal{P})$ with verification complexity κ. Assume that there is a configuration $G_s \in \mathcal{F}$ with $\mathcal{P}(G_s) = \text{TRUE}$ such that G contains r pairwise independent isomorphic subgraphs $H_1,\ldots,H_r$ with s edges each, and let $\sigma_i : H_1 \to H_i$ be a port-preserving isomorphism for each $1 \leq i \leq r$. If for any isomorphism $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$ we have that $\mathcal{P}(\sigma_{\bowtie}(G)_s) = \text{FALSE}$, then $\kappa \geq \frac{1}{2s}\log\log r$.*

Proof: Fix G_s as in the statement, and let $(\mathbf{p},\mathbf{v})$ be a one-sided RPLS for $(\mathcal{F},\mathcal{P})$. Assume w.l.o.g. that all certificates under $(\mathbf{p},\mathbf{v})$ have length exactly κ. For each edge $e=\{u,v\}$, let $x(u,v)$ denote the support of the certificates sent over e by u, i.e., all bit strings of length κ with positive probability to be sent from u to v under G_s and $(\mathbf{p},\mathbf{v})$. Since there are κ bits in each certificate, the number of distinct certificates is at most 2^κ, and hence the number of distinct supports is $2^{(2^\kappa)}$. Ordering the nodes of H_1 somehow and using the order induced by the port-preserving isomorphisms, we can represent each specific setting of the $2s$ certificates sent over the edges of H_i as a $2s\kappa$-long bit string. Now, if $\kappa < \frac{1}{2s}\log\log r$, then $2^{(2^{2s\kappa})} < r$, and hence, by the pigeonhole principle, there are $1 \leq i < j \leq r$ such that the supports of all the $2s$ respective (directed) edges in H_i and H_j are identical. Define $\sigma = \sigma_j \circ \sigma_i^{-1}$, and let $u' = \sigma(u)$ for some node u in H_i. Let v be a neighbor of u in H_i and let $v' = \sigma(v)$.

Fix an arbitrary global certificate $\underline{c}$ assignment (assigning a certificate to each direction of each edge of G) that can be generated in G_s by $(\mathbf{p},\mathbf{v})$ with positive probability. Note that since $\mathcal{P}(G_s) = \text{TRUE}$ and the RPLS is one sided, and since the probability of generating $\underline{c}$ is strictly positive, it must be the case that under $\underline{c}$, the verifier accepts at all nodes (deterministically). Now, let c_1 denote the coordinate of (u,v) in $\underline{c}$, i.e., the certificate sent by u to v in $\underline{c}$. Similarly let c_2 denote the certificate sent by u' to v' in $\underline{c}$. Let $\underline{c}'$ denote the global certificate obtained from $\underline{c}$ by switching c_1 and c_2, i.e., under $\underline{c}'$, u sends c_1 to v' and u' sends c_2 to v' (cf. Figure 1).

We claim that the verifier $\mathbf{v}$ accepts at all nodes under $\underline{c}'$. To see that, note that since $\mathcal{P}(G_s) = \text{TRUE}$, it must be the case (by the independence of certificates received at a node, given the labels) that *any* certificate in $x(u',v')$ sent to v' results in the verifier $\mathbf{v}$ at v' outputting TRUE: otherwise, there will be a non-zero probability that $\mathbf{v}$ rejects at v', resulting in rejecting a "yes" instance. Therefore, the fact that $x(u,v) = x(u',v')$ (by our choice) necessarily implies that if v' receives $c_1 \in x(u,v) = x(u',v')$, the verifier $\mathbf{v}$ at v' outputs TRUE. Similarly for v accepting c_2.

Continuing inductively in this fashion we switch the certificates edge by edge, and arrive at the conclusion that the verifier $\mathbf{v}$ accepts $\sigma_{\bowtie}(G)_s$. Moreover, since we can apply the switching procedure described above to *any* legal certificate assignment $\underline{c}$ for G_s, we have that if $\kappa < \frac{1}{2s}\log\log r$, then $\sigma_{\bowtie}(G)_s$ is accepted by the RPLS with probability 1, contradicting our assumption that $\mathcal{P}(\sigma_{\bowtie}(G)_s) = \text{FALSE}$ and the requirement that a one-sided RPLS rejects a "no" instance with probability at least $\frac{1}{2}$. ∎

We note that all the upper bounds we derive in Section 5 are both edge-independent and one-sided.

5. VERIFICATION COMPLEXITY OF SOME SPECIFIC PREDICATES

We now bound, from above and from below, the deterministic and randomized verification complexity of a few specific problems using the tools developed in Sections 3 and 4. We study three important problems of independent interest. Each of these problems has received attention in the framework of (deterministic) proof-labeling schemes, as well as in other frameworks like distributed algorithm design, property testing, etc. We note that all RPLSs constructed in this section are edge-independent and one-sided, and that all lower bounds here are for RPLSs which are either edge independent or one-sided.

In the following, let $\mathcal{F}_{con}$ be the family of all connected graphs, and let $G_e\{x,y\}$ denote the graph over the set of nodes $\{x,y\}$ and an edge connecting x and y.

5.1 Minimum-Weight Spanning Tree (MST)

Recall that a Minimum-weight Spanning Tree (MST) of a weighted n-node graph G is a spanning tree of G whose sum of all its edge-weights is minimum among all spanning trees of G. In this setting,

we assume that every node is aware of the weights of its incident edges (i.e., these weights, indexed by port numbers, are part of its state).

THEOREM 5.1. *The randomized verification complexity of* $(\mathcal{F}_{con}, MST)$ *is* $\Theta(\log\log n)$.

Proof: The upper bound follows from combining Theorem 3.1 with the proof-labeling scheme for MST in [30], whose verification complexity is $O(\log^2 n)$ (assuming polynomial edge weights). For the lower bound, we will show that any randomized proof-labeling scheme for the much simpler predicate of *acyclicity* has verification complexities as stated. Let $\mathcal{F}$ be the family of graphs that consist of lines and cycles, i.e., if $G \in \mathcal{F}$ then G is a line or G is a cycle, where all edges have weight 1. Let $P = (u_1, u_2, \ldots, u_n)$ be the n-node path, with port numbers consistently ordered. We define $H_1, \ldots, H_r$, for $r = \lfloor \frac{n}{3} \rfloor - 1$, as follows. Let $H_1 = G_e\{u_1, u_2\}$. For $i = 2, \ldots, \lfloor \frac{n}{3} \rfloor - 1$, let $H_i = G_e\{u_{3i}, u_{3i+1}\}$, and $\sigma_i : V(H_1) \to V(H_i)$ satisfying $\sigma_i(u_1) = u_{3i}$ and $\sigma_i(u_2) = u_{3i+1}$. For any $1 \leq i < j \leq \lfloor \frac{n}{3} \rfloor - 1$, let $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$. We get that ${\sigma^{ij}}_{\bowtie}(P)$ consists of removing edges $\{u_{3i}, u_{3i+1}\}$ and $\{u_{3j}, u_{3j+1}\}$ from P, and replacing them by $\{u_{3i}, u_{3j+1}\}$ and $\{u_{3j}, u_{3i+1}\}$, creating the cycle $C = (u_{3i+1}, u_{3i+2}, \ldots, u_{3j-1}, u_{3j})$. Note that ${\sigma^{ij}}_{\bowtie}(P) \notin \mathcal{F}$, but C is a connected component of ${\sigma^{ij}}_{\bowtie}(P)$ and $C \in \mathcal{F}$. Moreover, C is a cycle, i.e., not satisfying the predicate. Therefore, by Theorem 4.7, the verification complexity of $(\mathcal{F}, acyclicity)$ is $\Omega(\log\log n)$. Hence, since $\mathcal{F} \subset \mathcal{F}_{con}$, the verification complexity of $(\mathcal{F}_{con}, \text{MST})$ is $\Omega(\log\log n)$. ∎

5.2 Vertex Bi-Connectivity

A connected graph G is called vertex-biconnected if the result of removing any node from G is a connected graph.

In [30], the authors proved a $\Theta(\log n)$ bound on the deterministic verification complexity of the s-t connectivity problem. In this problem, given a connected graph $G = (V, E)$ and two specified nodes $s, t \in V$, the goal is for all nodes to agree on a natural number k, where k is the vertex connectivity between s and t in G. Note that this is not a decision problem as it was presented. Slightly modified, where k is a parameter of the problem, we obtain the problem s-t k-connectivity, where the goal is to decide whether the vertex connectivity between s and t is exactly k, and the $\Theta(\log n)$ bound still holds. This problem is closely related to vertex-biconnectivity, with the main differences that in the latter we consider the connectivity between *all* pairs of nodes and we only check whether it is at least 2. Let v2con denote the predicate of vertex-biconnectivity. We have the following result.

THEOREM 5.2. *The deterministic verification complexity of* $(\mathcal{F}_{con}, v2con)$ *is* $\Theta(\log n)$, *and its randomized verification complexity is* $\Theta(\log\log n)$.

Proof: The upper bounds follow from the observation that we can encode all relevant information used in the biconnectivity algorithm of Tarjan [21] using $O(\log n)$ bits. For the lower bounds, let G be a graph that consists of an n-node cycle, with port numbers consistently ordered, and additional edges from one node to all other nodes. (See Figure 2a for illustration). That is,
$G = (\{v_0, \ldots, v_{n-1}\}, E_c \cup E_0)$ where

$$E_c = \big\{\{v_i, v_{(i+1) \bmod n}\} \mid i = 0, \ldots, n-1\big\} \text{ and}$$
$$E_0 = \big\{\{v_0, v_j\} \mid j = 2, \ldots, n-2\big\}.$$

We have v2con(G) = TRUE. Now, let $H_1 = G_e\{v_1, v_2\}$. For $i = 2, \ldots, \lfloor \frac{n}{3} \rfloor - 1$, let $H_i = G_e\{v_{3i}, v_{3i+1}\}$, and $\sigma_i : V(H_1) \to V(H_i)$ satisfying $\sigma_i(v_1) = v_{3i}$ and $\sigma_i(v_2) = v_{3i+1}$. For any $1 \leq i < j \leq \lfloor \frac{n}{3} \rfloor - 1$, let $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$. We get that ${\sigma^{ij}}_{\bowtie}(G)$ consists of two disjoint cycles with some edges in E_0 between them, and v_0 is an articulation point. Therefore, v2con$({\sigma^{ij}}_{\bowtie}(G))$ = FALSE. Hence, the conditions of Theorems 4.4 and 4.7 are satisfied, and the lower bounds follow. ∎

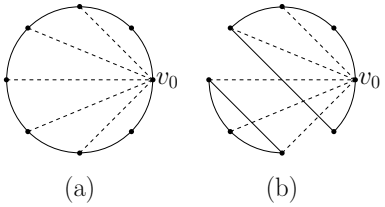

Figure 2: *(a): The graph G in the proof of Theorem 5.2, and restricted to nodes $\{v_0, \ldots, v_{c-1}\}$ in the proof of Theorem 5.4. Dashed edges are E_0. (b): The graph ${\sigma^{ij}}_{\bowtie}(G)$.*

Another result in [30] regarding connectivity is an upper bound of $O(k \log n)$ on the deterministic verification complexity of the k-flow problem. In this problem, the goal is to decide whether the value of the maximum flow between s and t is exactly k. Using Theorem 3.1 on that result, we get an upper bound of $O(\log k + \log\log n)$ on the randomized verification complexity of the k-flow problem.

5.3 Longest Cycle Size

For any positive integer c, we define the predicate "cycle-at-most-c" over graphs, which is true for G if and only if all simples cycle in G contain at most c nodes. We also define the predicate "cycle-at-least-c" over graphs, which is true for G if and only if there is a simple cycle in G with at least c nodes. Note that due to the asymmetry between acceptance and rejection in verification schemes, these are different questions despite their complementary nature. Similarly, note that for $c = n$, cycle-at-least-c is NP-complete and cycle-at-most-c is coNP-complete. We have the following upper bounds for cycle-at-least-c.

THEOREM 5.3. *The deterministic verification complexity of* $(\mathcal{F}_{con}, \text{cycle-at-least-}c)$ *is* $O(\log n)$, *and its randomized verification complexity is* $O(\log\log n)$.

The upper bounds follow by marking the cycle nodes using $O(\log n)$ bits (details omitted). The following theorem states lower bounds for cycle-at-least-c.

THEOREM 5.4. *The deterministic verification complexity of* $(\mathcal{F}_{con}, \text{cycle-at-least-}c)$ *is* $\Omega(\log c)$, *and its randomized verification complexity is* $\Omega(\log\log c)$.

Proof: Let G be a graph that consists of a c-node cycle, with port numbers consistently ordered, and additional edges from one to all other nodes of the graph (see Figure 2). Formally,

$$G = (\{v_0, \ldots, v_{n-1}\}, E_c \cup E_0) \text{ , where}$$
$$E_c = \big\{\{v_i, v_{(i+1) \bmod c}\} \mid i = 0, \ldots, c-1\big\} \text{ , and}$$
$$E_0 = \big\{\{v_0, v_j\} \mid j = 2, \ldots, n-1 \text{ , } j \neq c-1\big\} \text{ .}$$

Clearly, cycle-at-least-$c(G)$ = TRUE. Let $H_1 = G_e\{v_0, v_1\}$. For $i = 2, \ldots, \lfloor \frac{c}{3} \rfloor - 1$, let $H_i = G_e\{v_{3i}, v_{3i+1}\}$, and $\sigma_i : V(H_1) \to V(H_i)$ satisfying $\sigma_i(v_0) = v_{3i}$ and $\sigma_i(v_1) = v_{3i+1}$. For any $1 \leq i < j \leq \lfloor \frac{c}{3} \rfloor - 1$, let $\sigma^{ij} = \sigma_j \circ \sigma_i^{-1}$. We get that

$\sigma^{ij}{}_{\bowtie}(G)$ consists of two disjoint cycles of size strictly less than $c-1$ each, with some edges in E_0 between them. The size of every simple cycle in $\sigma^{ij}{}_{\bowtie}(G)$ is strictly less than c, and therefore, cycle-at-least-$c(\sigma^{ij}{}_{\bowtie}(G))$ = FALSE. Hence, the conditions of Theorems 4.4 and 4.7 are satisfied, and the lower bounds follow. ∎

The lower bound in Theorem 5.4 shows the hardness of distinguishing between graphs which contain a cycle of length c and ones which contain only cycles of length up to $c-1$. We now present an alternative technique, which shows that this lower bound holds also in the case where the question is to distinguish between graphs with a cycle of size n and graphs where all cycles are strictly smaller than c. Formally, let $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$, where $\mathcal{F}_1$ is the family of all connected graphs over $n \geq c$ nodes that contains an n-node cycle, and $\mathcal{F}_2$ is the family of all connected graphs over $n \geq c$ nodes where all cycles have size at most $c-1$.

THEOREM 5.5. *The deterministic verification complexity of* $(\mathcal{F}$, *cycle-at-least-*$c)$ *is* $\Omega(\log c)$.

Proof: Let G be as described in the lower bound part of the proof of Theorem 5.2 (see Figure 2a), where $n \geq c$. This graph satisfies $G \in \mathcal{F}$ and cycle-at-least-$c(G)$ = TRUE. Let $(\mathbf{p}, \mathbf{v})$ be a proof-labeling scheme for cycle-at-least-c over $\mathcal{F}$. Assume, for the purpose of contradiction, that the verification complexity of that scheme is less than $\frac{1}{2}\log(\lfloor \frac{c-1}{3} \rfloor)$. Let $\{H_1, \ldots, H_r\}$, for $r = \lfloor \frac{c-1}{3} \rfloor$, be a set of pairwise independent subgraphs, each consists of exactly one cycle edge and its two endpoints. This set exists since $n \geq c-1$. For each edge, there are less than $\log(\lfloor \frac{c-1}{3} \rfloor)$ bits in the sequence of both labels of the extremities of the edge. Hence, there are less than $\lfloor \frac{c-1}{3} \rfloor$ possible sequences. Therefore, by the pigeonhole principle, there are two independent cycle edges, say e_1 and e_2, that have exactly the same labels. Recall that the port numbers of the cycle edges are consistently ordered. Hence, for the corresponding isomorphism, $\sigma_{\bowtie}(G)$ consists of two disjoint cycles of size strictly less than n each, with only edges in E_0 between them. Note that possibly $\sigma_{\bowtie}(G) \notin \mathcal{F}$. We apply this crossing inductively as long as there is a cycle of size at least $c-1$. This process terminates when we eventually get at a graph G' which consists of a set $\mathcal{C}$ of disjoint cycles, all of size less than $c-1$, with only edges in E_0 connecting them (see Figure 2b). Note that at most two E_0 edges can participate in any simple cycle in G'. Since there are no other connections between the cycles in $\mathcal{C}$, both edges in E_0 have to be connected to the same cycle in $\mathcal{C}$. Therefore, a simple cycle of maximum length in G' is a simple cycle of maximum length in $\mathcal{C}$, with two edges from E_0 instead of just one, and its size is strictly less than c. Hence, $G' \in \mathcal{F}$ and cycle-at-least-$c(G')$ = FALSE. On the other hand, the verifier accepts G', a contradiction. This concludes the proof of the theorem. ∎

Finally, consider now the problem of deciding the predicate cycle-at-most-c. We note that it is co-NP hard, because for $c = n-1$, cycle-at-most-c is the complement of Hamiltonian Cycle. Observe that a proof-labeling scheme for $(\mathcal{F}_{con}$, cycle-at-most-$c)$, with polynomial verification complexity and polynomial computation at each node can be translated into a sequential verifiable proof of polynomial size, and hence the existence of such a PLS would imply that NP = co-NP. Therefore, we do not expect to find an efficient PLS (let alone RPLS) for this problem. The universal scheme, in which the computation complexity at each node is unbounded, is the best scheme we know for this problem from the viewpoint of verification complexity. A lower bound on the verification complexity is presented in the following theorem.

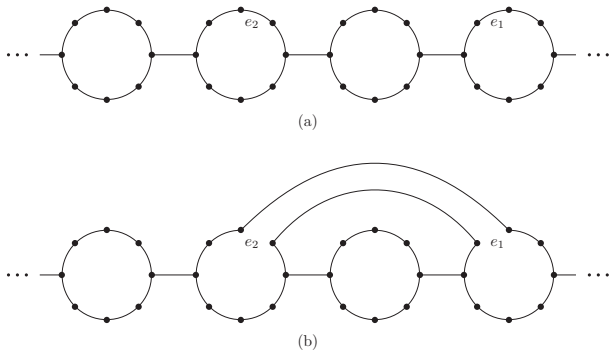

Figure 3: *(a) the graph G in the proof of Theorem 5.6 for $c = 8$. (b) $\sigma_{\bowtie}(G)$.*

THEOREM 5.6. *The deterministic verification complexity of* $(\mathcal{F}_{con}$, *cycle-at-most-*$c)$ *is* $\Omega(\log \frac{n}{c})$, *and its randomized verification complexity is* $\Omega(\log\log \frac{n}{c})$.

Proof: Let G be a chain of $\lceil \frac{n}{c} \rceil$ disjoint cycles of c nodes each (except one of at most c nodes), where every two neighboring cycles are connected by an edge (see Figure 3a). It obviously holds that cycle-at-most-$c(G)$ = TRUE. Let $\{H_1, \ldots, H_r\}$, for $r = \lceil \frac{n}{c} \rceil$, be a set of pairwise independent subgraphs, each consists of an edge from different cycle and its two endpoints. This set exists since there are $\lceil \frac{n}{c} \rceil$ cycles. For every two independent edges from different cycles, say $e_1 = \{u, v\}$ from cycle i and $e_2 = \{u', v'\}$ from cycle j, where $i \neq j$, we have that cycle-at-most-$c(\sigma_{\bowtie}(G))$ = FALSE for any isomorphism σ (see Figure 3). Therefore, the conditions of Theorems 4.4 and 4.7 are satisfied, and the lower bounds follow. ∎

6. CONCLUSION

In this paper we introduced the concept of randomized proof-labeling schemes (RPLS) and derived a few fundamental results and a few concrete results. The main message of the paper is that randomizing proof-labeling schemes reduces communication complexity exponentially. We also formalized the crossing technique which allows one to obtain lower bounds on the communication complexity of proof labeling schemes, both deterministic and randomized. Many interesting questions remain open. We list a few below.

- What is the relation between one-sided and two-sided RPLSs?
- Can the crossing technique (Theorem 4.7) be extended to show lower bounds on the verification complexity of two-sided RPLSs which are not edge independent? Can it be extended to the case where nodes have access to shared randomness?
- What are the relations between RPLS and the different complexity classes in [14], including LD, BPLD, NLD, and BPNLD?
- Can the complexity of the prover **p** be also accounted for in proof-labeling scheme? In particular, what is the effect of randomization on the prover complexity?

7. REFERENCES

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

[2] S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. In *14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 689–698, 2003.

[3] S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe. Nearest common ancestors: a survey and a new distributed algorithm. In *14th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 258–264, 2002.

[4] S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In *43rd Symposium on Foundations of Computer Science (FOCS)*, pages 53–62, 2002.

[5] H. Arfaoui, P. Fraigniaud, D. Ilcinkas, and F. Mathieu. Distributedly testing cycle-freeness. In *40th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, LNCS, pages 15–28. Springer, 2014.

[6] H. Arfaoui, P. Fraigniaud, and A. Pelc. Local decision and verification with bounded-size outputs. In *15th Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS, pages 133–147. Springer, 2013.

[7] B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-stabilization by local checking and correction. In *32nd Symposium on Foundations of Computer Science (FOCS)*, pages 268–277. IEEE, 1991.

[8] L. Blin, P. Fraigniaud, and B. 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.

[9] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and distance queries via 2-hop labels. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 937–946, 2002.

[10] 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.

[11] P. Fraigniaud and C. Gavoille. Routing in trees. In *28th International Colloquium on Automata, Languages and Programming (ICALP)*, LNCS, pages 757–772. Springer, 2001.

[12] P. Fraigniaud and C. Gavoille. A space lower bound for routing in trees. In *19th Symp. on Theoretical Aspects of Computer Science (STACS)*, LNCS, pages 65–75. Springer, 2002.

[13] P. Fraigniaud and A. Korman. On randomized representations of graphs using short labels. In *21st ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 131–137, 2009.

[14] P. Fraigniaud, A. Korman, and D. Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60(5):35, 2013.

[15] P. Fraigniaud, S. Rajsbaum, and C. Travers. Locality and checkability in wait-free computing. *Distributed Computing*, 26(4):223–242, 2013.

[16] P. Fraigniaud, S. Rajsbaum, and C. Travers. On the number of opinions needed for fault-tolerant run-time monitoring in distributed systems. In *5th International Conference on Runtime Verification (RV)*, LNCS, pages 92–107. Springer, 2014.

[17] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg. Approximate distance labeling schemes. In *9th European Symposium on Algorithms (ESA)*, pages 476–487, 2001.

[18] C. Gavoille and C. Paul. Split decomposition and distance labelling: An optimal scheme for distance hereditary graphs. *Electronic Notes in Discrete Mathematics*, 10:117–120, 2001.

[19] C. Gavoille, D. Peleg, S. Perennes, and R. Raz. Distance labeling in graphs. In *12th ACM Symposium on Discrete Algorithms (SODA)*, pages 210–219, 2001.

[20] M. Göös and J. Suomela. Locally checkable proofs. In *30th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 159–168, 2011.

[21] J. E. Hopcroft and R. E. Tarjan. Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, 1973.

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

[23] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. *SIAM J. Discrete Math.*, 5(4):596–603, 1992.

[24] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *7th Int. Workshop on Algorithms and Data Structures (WADS)*, pages 246–257, 2001.

[25] M. Katz, N. A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 927–936, 2002.

[26] M. Katz, N. A. Katz, and D. Peleg. Distance labeling schemes for well-separated graph classes. *Discrete Applied Mathematics*, 145(3):384–402, 2005.

[27] A. Korman. Labeling schemes for vertex connectivity. *ACM Transactions on Algorithms*, 6(2), 2010.

[28] A. Korman and S. Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20:253–266, 2007.

[29] A. Korman, S. Kutten, and T. Masuzawa. Fast and compact self stabilizing verification, computation, and fault detection of an MST. In *30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 311–320, 2011.

[30] A. Korman, S. Kutten, and D. Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.

[31] A. Korman, D. Peleg, and Y. Rodeh. Constructing labeling schemes through universal matrices. *Algorithmica*, 57(4):641–652, 2010.

[32] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[33] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, 1994.

[34] D. Peleg. Proximity-preserving labeling schemes and their applications. In *25th International Workshop Graph-Theoretic Concepts in Computer Science (WG)*, pages 30–41, 1999.

[35] D. Peleg. Informative labeling schemes for graphs. *Theor. Comput. Sci.*, 340(3):577–593, 2005.

[36] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In *42nd Symp. on Foundations of Computer Science (FOCS)*, pages 242–251, 2001.

[37] M. Thorup and U. Zwick. Compact routing schemes. In *13th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 1–10, 2001.