Brief Announcement: Space-Time Tradeoffs for Distributed Verification*

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

Rafail Ostrovsky†
Departments of Computer Science and Mathematics
UCLA
Los Angeles, CA 90095
rafail@cs.ucla.edu

Will Rosenbaum
Department of Mathematics
UCLA
Los Angeles, CA 90095-1555
wrosenbaum@math.ucla.edu

ABSTRACT

Verifying that a network configuration satisfies a given boolean predicate is a fundamental problem in distributed computing. Many variations of this problem have been studied, for example, in the context of proof labeling schemes (PLS) [12], locally checkable proofs (LCP) [10], and non-deterministic local decision (NLD) [8]. In all of these contexts, verification time is assumed to be constant. Korman, Kutten and Masuzawa [11] presented a proof-labeling scheme for MST, with poly-logarithmic verification time, and logarithmic memory at each vertex.

In this paper we introduce the notion of a t-PLS, which allows the verification procedure to run for super-constant time. Our work analyzes the tradeoffs of t-PLS between time, label size, message length, and computation space. We construct a universal t-PLS and prove that it uses the same amount of total communication as a known one-round universal PLS, and t factor smaller labels. In addition, we provide a general technique to prove lower bounds for space-time tradeoffs of t-PLS. We use this technique to show an optimal tradeoff for testing that a network is acyclic (cycle free). Our optimal t-PLS for acyclicity uses label size and computation space $O((\log n)/t)$. We further describe a recursive $O(\log^* n)$ space verifier for acyclicity which does not assume previous knowledge of the run-time t.

Keywords

Distributed algorithms, proof-labeling schemes, space-time tradeoffs.

*A full version of this manuscript can be found on the arXiv [5].

†Supported in part by NSF grants 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award.

PODC'16 July 25–28, 2016, Chicago, IL, USA.

ACM ISBN 978-1-4503-3964-3/16/07.

DOI: http://dx.doi.org/10.1145/2933057.2933071

1. INTRODUCTION

The problem of determining if a network configuration satisfies some predicate has been studied in the context of checking the ouptut of a distributed algorithm [3, 9], designing self-stabilizing algorithms [1, 2, 6, 11], and complexity theory [7, 8, 9]. A network configuration is represented by an underlying graph, where each vertex represents a processor, edges represent communication links between processors, and each vertex has a state. For example, the state of every vertex can be a color, and the predicate signifies that the coloring is proper. Processors learn about the network by exchanging messages along the edges. Some properties are local by nature and easy to verify, yet many natural problems—for example, testing if the network contains cycles—cannot be tested in less than diameter time, even if message size and local computational power are unbounded.

In order to cope with strong time lower bounds, Korman, Kutten, and Peleg introduced in [12] a computational model, called ***proof-labeling schemes*** (PLS), where vertices are given auxiliary global information in the form of ***labels***. This auxiliary information may allow vertices to verify that a property is satisfied more efficiently than could be achieved without the aid of labels. Specifically, a PLS consists of two components: a ***prover*** and a ***verifier***. The prover is an oracle which assigns labels to vertices. The verifier is a distributed algorithm which runs on the labeled configuration and outputs TRUE or FALSE at each vertex as a function of its state, its label, and the labels it receives. A PLS is ***complete*** if for every legal configuration (satisfying the predicate), prover can assign labels such that all vertices output TRUE. The PLS is ***sound*** if for every illegal configuration (which does not satisfy the predicate) for every labeling, some vertex outputs FALSE.

1.1 Our Contributions

In this brief announcement we consider proof-labeling schemes with super-constant verification time t, and analyze tradeoffs between time, label size, message size, and computation space. Our main contributions are as follows:

- A ***universal scheme*** which can verify any predicate $\mathcal{P}$ on a network.
- A general technique for proving label size lower bounds for t-round schemes.
- Tight lower and upper bounds for testing if a network is cycle free.
- A cycle-free verifier which uses space $O(\log^* n)$.

2. MODEL AND DEFINITIONS

2.1 Computational Framework

A ***graph configuration*** G_s consists of an underling graph $G = (V, E)$, and a state assignment function $\varphi : V \to S$, where S is a state space. The state of a vertex includes all of its local information. It may include the vertex's identity (in an ID based configuration), the weight of its adjacent edges (in a weighted configuration), or the result of an algorithm executed on the graph.

In a proof-labeling scheme, an oracle assigns labels $\ell : V \to L$. Verification is performed by a distributed algorithm on the labeled configuration in synchronous rounds. In each round every vertex receives messages from all of its neighbors, performs local computations, and sends a message to all of its neighbors. At the beginning of each round, a vertex scans its messages in a streaming fashion. The ***computational space*** is the maximum space required by a vertex in its local computation. Each vertex may send different messages to different neighbors in a round. When a vertex halts, it outputs TRUE or FALSE. If the vertex labels contain unique identifiers, then we require that an algorithm has the same output for all legal assignments of unique IDs.

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 a ***prover*** $\mathbf{p}$, and a ***verifier*** $\mathbf{v}$. The prover is an oracle which, given any configuration $G_s \in \mathcal{F}$, assigns a bit string $\ell(v)$ to each vertex v, called the ***label*** of v. The verifier $\mathbf{v}$ at each vertex outputs a boolean value. If the outputs are TRUE at all vertices, $\mathbf{v}$ is said to ***accept*** the configuration. If $\mathbf{v}$ outputs FALSE in at least one vertex, $\mathbf{v}$ is said to ***reject*** the configuration. For correctness, a proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ for $(\mathcal{F}, \mathcal{P})$ must be (1) ***complete*** and (2) ***sound***. Formally, $(\mathbf{p}, \mathbf{v})$ is ***complete*** if $\mathcal{P}(G_s) =$ TRUE then, using the labels assigned by $\mathbf{p}$, the verifier $\mathbf{v}$ accepts G_s. The scheme is ***sound*** if $\mathcal{P}(G_s) =$ FALSE then, for every label assignment, the verifier $\mathbf{v}$ rejects G_s.

The ***verification complexity*** of a proof-labeling scheme $(\mathbf{p}, \mathbf{v})$ is the maximal label size—the maximal length of a label assigned by the prover $\mathbf{p}$ on a legal configuration (satisfying $\mathcal{P}$). Here, we introduce proof-labeling schemes with more than one verification round. In particular the runtime t of $\mathbf{v}$ can be super-constant. We define the ***message size*** of the scheme $(\mathbf{p}, \mathbf{v})$ to be the largest message a vertex sends during the execution of $\mathbf{v}$ on a legal configuration with the labels assigned by $\mathbf{p}$. We denote a proof-labeling scheme with t-round verification by t-PLS.

3. UNIVERSAL SCHEME

A ***universal scheme*** is a PLS that verifies any sequentially decidable property. In this section we assume that each vertex has a unique identifier. A single round 1-PLS can be obtained by having the labels encode a proposed description of the network (along with the states of each vertex). The labeling can be verified in a single round by each vertex ensuring that each of its neighbors' labels are consistent with its label [12, 10, 4]. For $t > 1$, we obtain the following gereneralization a universal scheme.

THEOREM 1. *Let $\mathcal{F}$ be a family of configurations with state set S and diameter at least D. Let $\mathcal{P}$ be a boolean predicate over $\mathcal{F}$. Suppose that every state in S can be represented using s bits. For every $t \in \Omega(D)$ there exists a t-PLS for $(\mathcal{F}, \mathcal{P})$ with label and message size $O((ns + \min\{n^2, m \log n\})/t)$ where n is the number of vertices, and m is the number of edges in the graph.*

For $t = 1$ Theorem 1 reduces to the well-known universal scheme. For larger values of t, labels are significantly smaller, and the total communication is the same as the one round scheme. The idea of the proof of Theorem 1 is to break the labels of the 1-round scheme into $O(t)$ "shares." In t communication rounds, the verifier $\mathbf{v}$ reconstructs the original 1-round labels, which are then accepted or rejected as in the 1-round PLS. We refer to this technique as ***label sharing***.

4. LOWER BOUND TECHNIQUE

The main tool we use for proving lower bounds for t-PLS is Theorem 5. Towards proving Theorem 5, will will find it useful to consider orientated edges in the network. We denote the head and tail of a directed edge e by $H(e)$ and $T(e)$, respectively.

DEFINITION 2 (EDGE CROSSING). *Let $G = (V, E)$ be a graph, and $e_1, e_2 \in E$ be two directed edges. The **edge crossing** of e_1 and e_2 in G, denoted by $C(e_1, e_2, G)$, is the graph obtained from G by replacing e_1 and e_2, by the edges $(T(e_1), H(e_2))$ and $(T(e_2), H(e_1))$.*

Edge crossings were formalized as a tool for proving lower bounds of verification complexity in [4]. We now show how to use edge crossing in order to prove lower bounds for label size of t-PLS.

DEFINITION 3 (EDGE k-NEIGHBORHOOD). *Let $G = (V, E)$ be a graph, and $e = (u, v) \in E$. The **k-neighborhood** of e in G, denoted by $N_k(e, G)$, is the subgraph (V', E') of G satisfying (1) $w \in V'$ if and only if $w \in V$ and $\min(\text{dist}(w, u), \text{dist}(w, v)) \leq k$, and (2) $e' \in E'$ if and only if $e' \in E \cap (V' \times V')$.*

PROPOSITION 4. *Let $(\mathbf{p}, \mathbf{v})$ be a deterministic t-PLS for $(\mathcal{F}, \mathcal{P})$ with label size $|\ell|$. Suppose that there is a configuration $G_s \in \mathcal{F}$ which satisfies $\mathcal{P}$ and contains r directed edges $e_1, \ldots, e_r$, whose t-neighborhoods $N_t(e_1, G_s), \ldots, N_t(e_r, G_s)$ are pairwise disjoint, contain q vertices each, and there exist r state preserving isomorphisms $\sigma_i : V(N_t(e_1, G_s)) \to V(N_t(e_i, G_s))$ for $i = 1, \ldots, r$ such that $\sigma_i(H(e_1)) = H(e_i)$ and $\sigma_i(T(e_1)) = T(e_i)$. If $|\ell| < (\log r)/q$, then there exist i, j with $1 \leq i < j \leq r$ such that every connected component of $C(e_i, e_j, G_s)$ is accepted by $(\mathbf{p}, \mathbf{v})$.*

The main result of this section follows from Proposition 4.

THEOREM 5. *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 $G_s \in \mathcal{F}$ which satisfies*

1. *$\mathcal{P}(G_s) =$ TRUE,*
2. *G_s contains r directed edges $e_1, \ldots, e_r$, whose t-neighborhoods $N_t(e_1, G_s), \ldots, N_t(e_r, G_s)$ are pairwise disjoint, contain q vertices each, and there exist r state preserving isomorphisms $\sigma_i : V(N_t(e_1, G_s)) \to V(N_t(e_i, G_s))$ for $i = 1, \ldots, r$ such that $\sigma_i(H(e_1)) = H(e_i)$ and $\sigma_i(T(e_1)) = T(e_i)$, and*
3. *for every $i \neq j$, there exists a connected component H_s of $C(e_i, e_j, G_s)$ such that $\mathcal{P}(H_s) =$ FALSE.*

Then the label size of any t-PLS for $(\mathcal{F}, \mathcal{P})$ is $\Omega((\log r)/q)$.

5. ACYCLICITY

In this section we focus on the acyclicity property, and give tight t-PLS lower and upper bounds. The lower bounds hold in computational model where vertices have unique identifiers, and the labels are allowed to depend on the ID of a vertex. The upper bounds apply even in a weaker computational model where vertices do not have unique IDs.

DEFINITION 6 (ACYCLICITY). *Let $\mathcal{F}$ be the family of all connected graphs. Given a graph configuration $G_s \in \mathcal{F}$,* ACYCLIC(G_s) = TRUE *if and only if the underling graph G is cycle free.*

THEOREM 7. *Every scheme which verifies* ACYCLIC *in t communication rounds requires labels of size $\Omega((\log n)/t)$.*

The proof of Theorem 7 is an applicaiton of Theorem 5 to a network consisting of a path. The result follows by observing that any edge-crossing of a path contains a cycle.

THEOREM 8. *Suppose $G = (V, E)$ is a graph with diameter D. For every $t \leq \min\{\log n, D\}$, there exists an $O(t)$-*PLS *for* ACYCLIC *with labels and messages of size $O((\log n)/t)$. Further, the verifier $\mathbf{v}$ uses space of size $O((\log n)/t)$.*

The upper bound of Theorem 8 is obtained by applying label sharing to a well-known 1-PLS. In the 1-PLS, the labels consist of an integer between 0 and $n-1$. A unique "root" is assigned label 0, while each other label is a vertex's distance from the root. Each vertex then verifies that either (1) its label is 0 and all its neighbors have label 1, or (2) its label is $\ell > 0$, and it has a unique neighbor with label $\ell - 1$, while all other neighbors have label $\ell + 1$. Theorem 8 follows by breaking the 1-PLS labels into $\Theta(t)$ shares of size $\Theta((\log n)/t)$. Each vertex then verifies that the corresponding 1-PLS labeling is correct by examining the labels in its t-neighborhood.

The scheme from Theorem 8 gives asymptotically optimal label size for $t \leq \log n$. Further, the communication per round and local memory usage is linear in the label size. However, the scheme above crucially requires each vertex to be given a truthful representation of the parameter t, the runtime of the verification. This may be undesirable as for $t \sim \log n$, the labels are of size $O(1)$, yet any representation of t is significantly larger ($\Omega(\log\log n)$). We describe a verifier for ACYCLIC that only assumes that the space provided to each processor is $O(\log^* n)$. The tradeoff is that our algorithm runs in time which may be linear in n in the worst case.

THEOREM 9. *There exists an $O(n)$-*PLS *for* ACYCLIC *which uses labels and space of size $O(\log^* n)$. In each round, the communication per-edge is $O(1)$.*

The $O(\log^* n)$ space scheme of Theorem 9 combines the ideas of label sharing with recursion. At the top level of recursion, we simulate the 1-PLS for acyclicity. Shares of the labels are stored in paths emanating from the proposed root, which we refered to as ***blocks***. Each block has size $O(\log n)$, and stores its distance from the purported root, where each node's share consists of a single bit. The correctness of the block labels are verified recursively using sub-blocks of size $O(\log\log n)$. After $O(\log^* n)$ levels of recursion, the blocks have constant size, and the correctness of the labeling can be trivially verified.

While verification time in Theorem 9 is $O(n)$ in the worst case, the actual time depends on the labels given to the vertices. In particular, for every acyclic graph G, with diameter D, there exists a correct labeling which will be accepted in time $O(\log D)$. Thus there is a tradeoff between the runtime of the algorithm and the amount of truthful information about D given to the vertices.

6. 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] B. Awerbuch and R. Ostrovsky. Memory-efficient and self-stabilizing network reset (extended abstract). In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '94, pages 254–263, New York, NY, USA, 1994. ACM.

[3] 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.

[4] M. Baruch, P. Fraigniaud, and B. Patt-Shamir. Randomized proof-labeling schemes. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC*, pages 315–324, 2015.

[5] M. Baruch, R. Ostrovsky, and W. Rosenbaum. Space-time tradeoffs for local verification. Manuscript, available at www.arxiv.org, 2016.

[6] 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.

[7] 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.

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

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

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

[11] 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.

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