

Shared Randomness Helps with Local Distributed Problems

Alkida Balliu 


Gran Sasso Science Institute, L'Aquila, Italy

Fabian Kuhn 

University of Freiburg, Germany

Dennis Olivetti 

Gran Sasso Science Institute, L'Aquila, Italy

Jukka Suomela 

Aalto University, Espoo, Finland

Mohsen Ghaffari 

MIT, Cambridge, MA, USA

Augusto Modanese 

Aalto University, Espoo, Finland

Mikaël Rabie 

Université Paris Cité, CNRS, IRIF, France

Jara Uitto 

Aalto University, Espoo, Finland

Abstract

By prior work, we have many wonderful results related to distributed graph algorithms for problems that can be defined with local constraints; the formal framework used in prior work is *locally checkable labeling problems* (LCLs), introduced by Naor and Stockmeyer in the 1990s. It is known, for example, that if we have a deterministic algorithm that solves an LCL in $o(\log n)$ rounds, we can speed it up to $O(\log^* n)$ rounds, and if we have a randomized algorithm that solves an LCL in $O(\log^* n)$ rounds, we can derandomize it for free.

It is also known that randomness helps with some LCL problems: there are LCL problems with randomized complexity $\Theta(\log \log n)$ and deterministic complexity $\Theta(\log n)$. However, so far there have not been any LCL problems in which the use of *shared randomness* has been necessary; in all prior algorithms it has been enough that the nodes have access to their own private sources of randomness.

Could it be the case that shared randomness never helps with LCLs? Could we have a general technique that takes any distributed graph algorithm for any LCL that uses shared randomness, and turns it into an equally fast algorithm where private randomness is enough?

In this work we show that the answer is *no*. We present an LCL problem Π such that the round complexity of Π is $\Omega(\sqrt{n})$ in the usual randomized LOCAL model (with private randomness), but if the nodes have access to a source of shared randomness, then the complexity drops to $O(\log n)$.

As corollaries, we also resolve several other open questions related to the landscape of distributed computing in the context of LCL problems. In particular, problem Π demonstrates that distributed *quantum* algorithms for LCL problems strictly benefit from a shared quantum state. Problem Π also gives a separation between *finitely dependent distributions* and *non-signaling distributions*.

2012 ACM Subject Classification Theory of computation \rightarrow Distributed algorithms

Keywords and phrases Distributed computing, locally checkable labelings, shared randomness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.16

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: <https://arxiv.org/abs/2407.05445> [15]

Funding This work was supported in part by MUR (Italy) Department of Excellence 2023–2027, the PNRR MIUR research project GAMING “Graph Algorithms and MinING for Green agents” (PE0000013, CUP D13C24000430001), ANR project ENEDISC (ANR-24-CE48-7768-01), the Helsinki Institute for Information Technology (HIIT), and the Research Council of Finland, Grant 359104.

Acknowledgements This work was started in the *Research Workshop on Distributed Algorithms* (RW-DIST 2024) in L'Aquila, Italy. We would like to thank all workshop participants for discussions, and in particular Henrik Lievonon and Amirreza Akbari for helping us with this research project.



© Alkida Balliu, Mohsen Ghaffari, Fabian Kuhn, Augusto Modanese, Dennis Olivetti, Mikaël Rabie, Jukka Suomela, and Jara Uitto;
licensed under Creative Commons License CC-BY 4.0

52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025).

Editors: Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis

Article No. 16; pp. 16:1–16:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

In this work we present a graph problem that is solely defined with local constraints, yet distributed algorithms for solving it benefit from shared randomness. More formally, we present a locally checkable labeling problem (LCL) Π such that any randomized distributed algorithm that solves Π in the usual LOCAL model of distributed computing requires $\Omega(\sqrt{n})$ communication rounds, but if we have access to shared randomness, then we can exponentially improve the round complexity, down to $O(\log n)$ rounds.

Context: LCL problems and the LOCAL model. LCL problems were originally introduced by Naor and Stockmeyer [40] in the 1990s, and in the recent years they have formed one of the cornerstones of the modern theory of distributed graph algorithms. An LCL problem is simply a graph problem that can be specified by giving a finite set of valid labeled neighborhoods. For example, the task of coloring vertices with 10 colors in graphs of maximum degree at most 20 is an LCL problem. (It can be defined by listing all possible radius-1 neighborhoods of degree at most 20, and by listing for each of them all valid 10-colorings.) Numerous problems that have been studied in distributed graph algorithms over the decades are LCL problems (at least when restricted to bounded-degree graphs); examples include maximal independent sets, maximal matching, various problems related to vertex and edge coloring, and various tasks related to orienting edges or partitioning of edges subject to local constraints. In fact even 3SAT can be interpreted as an LCL problem (with the bounded-degree assumption corresponding to the case in which each variable occurs in a bounded number of clauses).

While LCL problems are meaningful in any model of computing, they have been studied in particular from the perspective of distributed graph algorithms, and the most prominent model there is the LOCAL model of computing [38, 41]. In brief, an algorithm A with running time T in the LOCAL model is simply a function that maps radius- T neighborhoods to local outputs. That is, to apply A in a given graph G , each node v looks at all information in its radius- T neighborhood and uses A to determine its own local output (for example, the color of v if the task is to find a graph coloring). It turns out that we could also equivalently interpret G as a computer network, and then A can be interpreted as a distributed message-passing algorithm in which all nodes stop after T synchronous communication rounds. We will hence interchangeably refer to T as the running time, locality, or round complexity of A .

A comprehensive theory of LCL problems in the LOCAL model has been developed in the past 10 years. There are numerous theorems that apply to all LCL problems, or all LCL problems in some graph family, such as trees or grids [5–7, 10–12, 16–21, 24, 25, 27, 28, 30, 43]. To give some flavor of the power of these results, here is one example: if there is a *randomized* LOCAL algorithm A that solves some LCL problem Π in $T(n) = o(\log \log n)$ rounds in n -node graphs, we can also construct a *deterministic* LOCAL algorithm A' that solves the same problem Π in $T'(n) = O(\log^* n)$ rounds [19]. That is, we can for free derandomize algorithms and speed them up.

In general, the relation between randomized and deterministic algorithms in the context of LCL problems is now well understood, and we also have a clear view of the landscape of all possible round complexities that we may have for LCL problems [44]. However, more care is needed here: what exactly do we mean by *randomized* LOCAL algorithms?

Question: shared vs. private randomness. Essentially all work on LCLs in the randomized LOCAL model assumes that each node has its own *private* source of randomness. More precisely, nodes are initially labeled independently and uniformly at random with strings of bits, and then a T -round algorithm can make use of all such bit strings within radius T .

However, there is another notion of randomized algorithms that has been studied for instance in the context of communication complexity: *shared* randomness (e.g., [1, 36]). That is, there is one global random bit string that all nodes can see. There are many contexts in which access to shared randomness helps [23, 39, 42], but does it help with any LCL problem?

Prior to this work, there was no evidence that shared randomness might help with LCL problems. On the contrary, all numerous LCL problems that we have encountered in prior work seem to be such that either (1) randomness does not help at all, or (2) randomness helps but private randomness is sufficient. There have even been systematic studies of infinite families of LCL problems [9, 21], as well as computer-assisted explorations of the space of LCL problems [45], yet there is no known candidate problem that might benefit from shared randomness. Intuitively, the key obstacle seems to be the combination that LCL problems are defined using local constraints and the set of input and output labels is finite. Shared randomness could be used to e.g. select a globally consistent random label from the set of finite output labels, but if that succeeds w.h.p. in arbitrarily large graphs, there also has to exist a deterministic choice that succeeds.

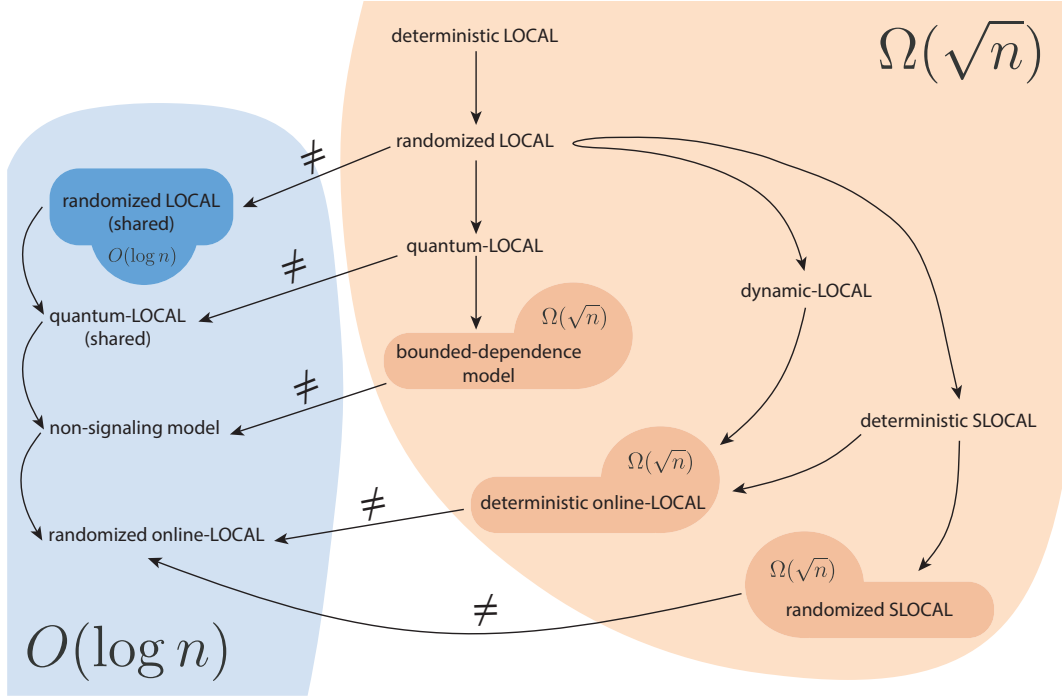
Hence, for all that we know, we might very well be living in a world in which the following conjecture is true: if an LCL problem Π can be solved in $T(n)$ rounds with the help of shared randomness, it can also be solved in $O(T(n))$ rounds with only private randomness. Were this to be true, it would considerably simplify the landscape of models, as discussed further below. It would also give a helpful algorithm design tool: we could design algorithms that exploit shared randomness, and then for free turn them into genuine distributed algorithms that only use private randomness. Conversely, it would allow us to strengthen all existing lower bounds that hold for private randomness into lower bounds that extend all the way to shared randomness.

What we show in this work is that this result cannot be true. Indeed, conversion from shared to private randomness for some LCL problems may lead to an *exponential* increase in the round complexity.

Main contribution. In this work we present an LCL problem Π such that the round complexity of Π is $\Omega(\sqrt{n})$ in the usual randomized LOCAL model (with private randomness), but if the nodes have access to a source of shared randomness, then the complexity drops to $O(\log n)$. This is the first known LCL that separates these two models.

Our problem Π is an LCL exactly in the strict sense originally defined by Naor and Stockmeyer [40], and we do not exploit any promise on the graph family or input. Being promise-free is important, as the entire theory of LCL problems is fundamentally promise-free (for example, the known gap results would disappear if we can have arbitrary promises on the input structure), and hence also any interesting separations or counterexamples have to be promise-free. We refer to the full version [15] for the formal theorem statements of our lower bound and upper bound.

Corollary 1: distributed quantum computing. One of the major open questions at the intersection of distributed computing and quantum information theory is which distributed problems admit quantum advantage. A key model for studying this question is the quantum-LOCAL model, which is essentially what one gets if we imagine that nodes of the input graph are quantum computers and communication channels can be used to exchange qubits. It is known that there are some (artificial) graph problems that can be solved in constant time in quantum-LOCAL yet for which classical LOCAL requires linear time [37]. Nevertheless, at the time of writing, it was still wide open whether there is any LCL problem that admits distributed quantum advantage; see, for instance, [2, 22] (but see Section 3.6 for the latest developments).



■ **Figure 1** Landscape of models from [2], and the new separations between them. We present a new LCL problem Π that is easy in the blue-shaded region (models in which we have access to shared randomness), but hard in the red-shaded region (all other models), and we will get separations for all pairs of models that cross the cut. We give an upper bound in randomized LOCAL with shared randomness, and all upper bounds in the blue region follow, and we give lower bounds in randomized SLOCAL, deterministic online-LOCAL, and the bounded-dependence model, and all lower bounds in the red region follow.

So far, there have been two major variants of quantum-LOCAL that have been studied in the literature: quantum-LOCAL with a shared quantum state (i.e., nodes are configured in advance and share entangled qubits) and quantum-LOCAL without any shared quantum state [2, 4, 26]. It was not known whether either of these is (strictly) stronger than randomized LOCAL for any LCL problem. There are no known examples of LCL problems that would potentially benefit from the shared quantum state, and it seemed reasonable to conjecture that, even if quantum-LOCAL turns out to be stronger than randomized LOCAL, the shared quantum state does not give it any additional power. Indeed, there were (unsuccessful) attempts at unifying the two variants of quantum-LOCAL.

One unexpected corollary of our work is that the two variants of quantum-LOCAL are indeed distinct, though for mundane reasons that have little to do with quantum physics. It simply happens to be the case that a shared quantum state gives the nodes access to shared randomness. As we show, the problem that we construct in this work is hard in quantum-LOCAL without shared quantum state, but it becomes easy in quantum-LOCAL with shared quantum state (since it is easy already in randomized LOCAL with shared randomness).

Hence, the entire question was wrong: shared quantum state does help, but for the wrong reasons. The present work highlights that the right question is whether shared quantum state provides further advantage beyond shared randomness.

Corollary 2: finitely dependent vs. non-signaling distributions. There is a line of research in mathematics that aims at capturing which problems admit *finitely-dependent distributions* [32–34, 46]; these are distributions over nodes such that their restriction to a set X of nodes is independent of their restriction to another set Y of nodes if the shortest-path distance between X and Y is greater than some constant. For example, the output distribution of any constant-time randomized LOCAL algorithm is a finitely-dependent distribution. A key question in this context has been whether finitely-dependent distributions are strictly stronger than constant-time randomized LOCAL algorithms, which indeed is the case [34]. A natural generalization of finitely-dependent distributions to arbitrary (not necessarily constant) distance is called a *bounded-dependence distribution* [2].

Another closely related definition arises from quantum information theory and the study of distributed quantum advantage: *non-signaling distributions* [2, 4, 22, 26]. Informally, a family of output distributions is non-signaling with locality T if the distribution restricted to some set of nodes X does not change if we modify the input graph more than T hops away from X . A bounded-dependence distribution with locality T is non-signaling with locality $O(T)$, but the converse is not necessarily true.

Prior to this work, there were no known examples of LCL problems that admit a non-signaling distribution with locality T but do not admit a bounded-dependence distribution with locality $O(T)$. Indeed, it was again reasonable to conjecture that no such problem exists. Our construction gives a separation also between these two models.

Other corollaries. Our construction also gives an exponential separation between deterministic and randomized versions of the online-LOCAL model (see [2, 3]). Previously, there were no known examples of LCLs that separate these models. For our problem we can prove a lower bound in the deterministic online-LOCAL model, and the upper bound for randomized LOCAL with shared randomness directly works also in randomized online-LOCAL.

The big picture. All of our results are summarized in Figure 1. All separations between the two regions are new, in the sense that there was previously no (promise-free) LCL that would separate these pairs of models. The results that lead to this landscape are:

- In the LOCAL model with shared randomness, the problem Π can be solved in $O(\log n)$ rounds, with success probability $1 - 1/n^c$ for any constant c .
- In the SLOCAL model with private randomness, solving the problem Π requires time $\Omega(\sqrt{n})$.
- In the deterministic online-LOCAL model, solving the problem Π requires $\Omega(\sqrt{n})$ rounds.
- In the bounded-dependence model, solving the problem Π requires locality $\Omega(\sqrt{n})$.

However, to keep this work easy to follow also for those who are not interested in models beyond randomized LOCAL, we also prove the following result that is technically redundant, but serves as a warm-up for the other results:

- In the LOCAL model with private randomness, solving the problem Π with success probability at least $1 - 1/n$ requires $\Omega(\sqrt{n})$ rounds.

Discussion and open questions. We conjecture that our problem Π also exhibits a doubly-exponential separation between the randomized LOCAL model and the *massively parallel computing* (MPC) model [35]. More precisely, we conjecture that our problem Π can be solved in $O(\log \log n)$ rounds in the MPC model, while it is known to require $\Omega(\sqrt{n})$ rounds in randomized LOCAL. Proving this is deferred for future work.

Our problem Π is artificial, and we conjecture that any LCL problem where shared randomness helps is necessarily somewhat artificial. Indeed, what we see as one of our main contributions is this: even though the study of LCL problems has been highly successful, and there are many theorems that hold for any LCL problem, we show that the family of all LCL problems is *too broad*, as it still makes it possible to define artificial problems like this that exhibit unexpected and counterintuitive properties. One of the main open questions is coming up with a meaningful restriction of LCLs that still captures all natural problems but excludes artificial construction of this flavor.

One property that our problem Π fundamentally exploits is the existence of short cycles, in the sense that the problem is trivial in trees and interesting only in graphs with short cycles. A key open question is *whether shared randomness helps with any LCL in trees*, or more generally high-girth graphs. We have preliminary evidence suggesting that shared randomness never helps in rooted regular trees, but the case of general trees remains open.

2 Definitions

► **Definition 1** (Labeled graph). Let \mathcal{V} and \mathcal{E} be sets of labels. A graph $G = (V, E)$ is called $(\mathcal{V}, \mathcal{E})$ -labeled if:

- Each node $u \in V$ is assigned a label from \mathcal{V} ;
- Each node-edge pair $(u, e) \in V \times E$, satisfying $u \in e$, is assigned a label from \mathcal{E} . A node-edge pair (u, e) that satisfies $u \in e$ is also called half-edge incident to u .

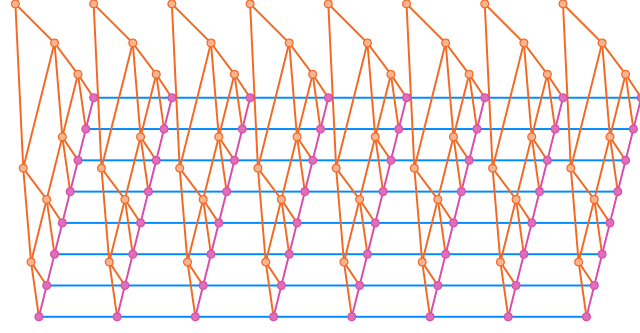
► **Definition 2** (Labeled graph satisfying some constraints). Let G be a graph, and let \mathcal{C} be a set of constraints over the labels \mathcal{V} and \mathcal{E} . The graph G satisfies \mathcal{C} if and only if:

- G is $(\mathcal{V}, \mathcal{E})$ -labeled, and
- the constraints of \mathcal{C} are satisfied over all nodes of G .

► **Definition 3** (Locally checkable labeling (LCL) problem). A locally checkable labeling (LCL) problem Π is defined by a tuple $(\mathcal{V}_{\text{input}}, \mathcal{E}_{\text{input}}, \mathcal{V}_{\text{output}}, \mathcal{E}_{\text{output}}, \mathcal{C})$ where \mathcal{C} is a set of constraints over $\mathcal{V}_{\text{output}}$ and $\mathcal{E}_{\text{output}}$. Given a $(\mathcal{V}_{\text{input}}, \mathcal{E}_{\text{input}})$ -labeled graph G , one is asked to label G so that it satisfies \mathcal{C} .

We denote with $L_u(e)$ the label on the half-edge (u, e) . Something that will be very useful throughout the paper is to define a way to denote the node that we can reach from a node u by following some specific chain of labels assigned to half-edges. Let $G = (V, E)$ be (Σ_V, Σ_E) -labeled. Let L_1, L_2, \dots, L_k be labels in Σ_E . We define a function $f(u, L_1, L_2, \dots, L_k)$ that takes as input a node $u \in V$ and labels L_1, \dots, L_k in Σ_E , and returns the node v reachable from u by following the unique path whose edges are labeled with L_1, \dots, L_k (in this order); if there is no such path or it is not unique, then the value of f is undefined. More precisely, let $P = (v_1, v_2, \dots, v_{k+1})$ be a path that starts at $v_1 = u$ and such that, for any edge $e = \{v_i, v_{i+1}\}$, the half-edge (v_i, e) is labeled with $L_{v_i}(e) = L_i$. Then we define $f(u, L_1, L_2, \dots, L_k) = v_{k+1}$ if P exists and is unique, and \perp otherwise.

The LOCAL model. In the LOCAL model of computing, we imagine that nodes of a graph $G = (V, E)$ are computers with access to unbounded computational resources (time and space). Each computer is assigned a unique identifier from the set $\{1, \dots, |V|^c\}$, where $c \geq 1$ is some constant. The communication between computers is as defined by E . The computation proceeds in rounds where, in each round, each computer exchanges messages of unbounded size with their neighbors and performs some local computation (which we may perceive as instantaneous).



■ **Figure 2** An example of a hard instance. The grid is composed of blue edges, purple edges, and purple nodes. Each connected component induced by orange nodes, orange edges, purple edges, and purple nodes connected to the orange edges is a tree-like structure.

The above gives the deterministic variant of **LOCAL**. The randomized variant (with private randomness) is the same but where we also give each node access to an infinite string of random bits (that is guaranteed to be independent of the strings of other nodes in the network). In the variant with shared randomness, nodes are given simultaneous access to the *same* infinite string.

3 High-level ideas

The main ingredient of our work is an LCL problem Π with some desirable properties. On a high-level, this problem is promise-free, in the sense that it is defined on any graph. However, it is defined in such a way that there exists a family of graphs \mathcal{G} that we call *hard instances*:

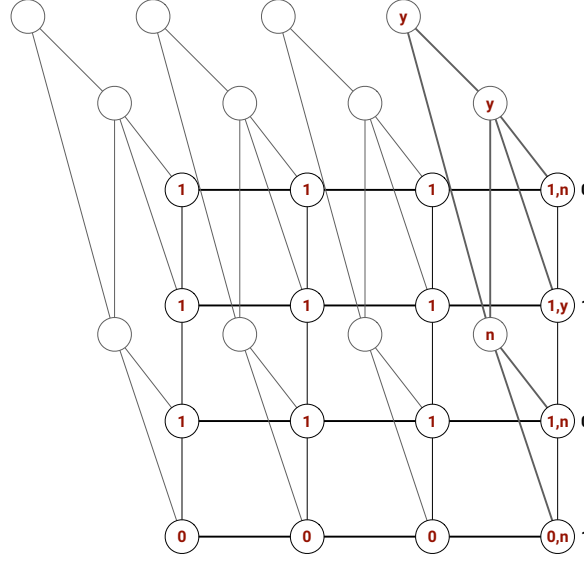
- If the graph G in which the algorithm solving Π is run is not in \mathcal{G} , the LCL is defined in such a way that the nodes of G can produce a locally checkable proof of this fact. Moreover, such a proof can be computed “fast”.
- However, if $G \in \mathcal{G}$, the LCL forces the nodes to solve a problem that can be solved “fast” if they have access to shared random bits, whereas any algorithm working without shared randomness must be “slow”.

Here “fast” and “slow” depend on the precise model considered; in the case of the **LOCAL** model, the problem requires just $O(\log n)$ rounds with shared randomness, but $\Omega(\sqrt{n})$ rounds without it.

In the following, we start by providing an overview of the structure of hard instances. Then we explain what the problem Π is on a hard instance, and later we will explain how the problem is defined to be promise-free.

The hard instances. Consider the graph depicted in Figure 2. It is composed of a square grid, where on top of each column we place a tree-like structure. This kind of graph has a couple of useful properties:

- All nodes that belong to the same column are within distance $O(\log n)$.
- As we will see, this structure can be certified. That is, it is possible to provide a constant-sized certificate to the nodes such that, if the certificate looks good everywhere, then the graph is indeed a hard instance. Conversely, if the graph is not a hard instance, then any assignment of the certificate leads to an error somewhere. We will make use of this fact later when making the problem promise-free.



■ **Figure 3** An example of a solution for Π^{hard} . Black bits represent the inputs of the nodes of the last column. The inputs of the other nodes do not affect the solution and are omitted. Labels in red represent the outputs, where the label y represents a happy node, and the label n represents an unhappy node. All nodes that are not labeled either y or n output y , which is omitted in the figure.

The LCL problem Π^{hard} defined on hard instances. Consider the following problem Π^{hard} , defined on hard instances. Each node in the right-most column of the grid receives a bit as input. Then, all nodes of the grid must output a bit such that the two following constraints are satisfied:

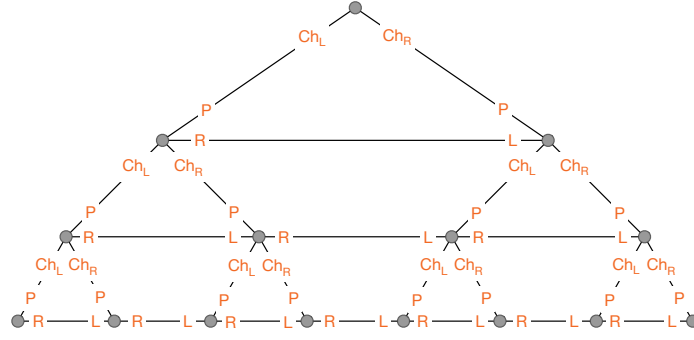
- C1:** For each row of the grid, all nodes must output the same bit.
- C2:** There must exist at least one row such that the output bit is the same as the input bit of the right-most node of that row.

It is possible to present this problem as an LCL as follows (see Figure 3):

- Assume that the grid has an input labeling encoding its orientation; that is, each node knows which of its neighbors is on its left, right, above, and below. Then constraint C1 can be encoded in the LCL by requiring every node to have the same output as its left and right neighbors.
- In order to enforce constraint C2, we use the tree-like structure on top of the last column. Say a grid node of the last column is *happy* if its output agrees with its input. Meanwhile an inner node of the tree is happy if at least one of their children is happy. All nodes of the tree output whether they are happy, and we require the root to be happy. These constraints are local, and in fact they can be encoded as an LCL.

Now, given such a problem, we can obtain a separation between shared and private randomness in the LOCAL model:

- If the nodes have access to shared randomness, then each node only needs $O(\log n)$ rounds to determine its vertical position y in the grid. Once it has figured out the value of y , the node simply outputs the y th shared random bit. In this way, all nodes in the same row output the same bit, and for each row with probability $1/2$ we have that this bit agrees with the input given to the node on the right-most column. Since there are $\Theta(\sqrt{n})$ rows, we get that there is at least one good row with high probability.



■ **Figure 4** An example of a properly labeled tree-like structure. The labels L, R, P, Ch_L, and Ch_R, stand, respectively, for left, right, parent, left child, and right child.

- Conversely, if only private randomness is allowed, nodes in the same row do not have any way to coordinate their output and communication across the whole row is expensive ($\Omega(\sqrt{n})$ rounds). The only alternative is for nodes in the same row to deterministically fix their output as a function of their vertical position. In this case we can adversarially pick the input to the rightmost node so that the row does not succeed. Since this cannot hold for every row (as otherwise the algorithm would not be solving Π), we get the lower bound of $\Omega(\sqrt{n})$ rounds.

Although the idea of the problem is simple, the main challenge is making the problem promise-free. That is, we also have to account for all the cases where the input graph is not as in Figure 2. Next we provide an overview of the process that we follow in the full version of this work [15].

3.1 A tree-like structure

A useful property that our problem satisfies is the following: nodes that belong to the same column of the grid should either be able to see the whole column by inspecting their $O(\log n)$ -radius neighborhood, or the nodes can *prove that there is some error* within distance $O(\log n)$. In the full version, we describe *tree-like structures* and how they give us exactly this property. This kind of structure has already been used in [14], and we can borrow some useful properties they proved.

Definition of the tree-like structure. Informally, a tree-like structure is a perfect binary tree in which nodes at the same depth are also connected via a path. Such a structure can be certified by assigning a label to each node-edge pair. An example of this structure as well as an assignment of a certificate for it are depicted in Figure 4. For example, the certificate ensures that all leaves are at the same depth by requiring that, starting from a node not having any incident edge labeled Ch_L or Ch_R (i.e., it does not have any children), and following the edge labeled R, we must reach a node that also does not have any children.

Local certification of tree-like structures. Next we describe tree-like structures from a local perspective. In particular, we define a set of input half-edge labels $\mathcal{E}^{\text{tree}}$ of constant size and a set of constraints $\mathcal{C}^{\text{tree}}$ over constant distance that satisfy the following:

- For all tree-like structures G , there exists an assignment of labels of $\mathcal{E}^{\text{tree}}$ to the half-edges of the graph such that the constraints of $\mathcal{C}^{\text{tree}}$ are satisfied on all nodes of G .

- Let G be a graph where half-edges are labeled with labels from $\mathcal{E}^{\text{tree}}$ and such that the constraints of $\mathcal{C}^{\text{tree}}$ are satisfied on all nodes of G . Then, G is a tree-like structure.

In other words, we show that there exists a locally checkable proof of constant size for the fact that the graph is tree-like.

The tree-like structure as an LCL. Building on the previously defined locally checkable proof, we define an LCL problem Π^{badTree} that satisfies the following:

- For all tree-like graphs G , there exists an input for Π^{badTree} that can be assigned to G such that the only valid solution for Π^{badTree} is the one assigning \perp to all nodes of G .
- Let G be a graph where half-edges are labeled with labels from $\mathcal{E}^{\text{tree}}$ such that the constraints of $\mathcal{C}^{\text{tree}}$ are not satisfied on at least one node of G . Then there exists a solution for Π^{badTree} where all nodes produce an output different from \perp . Moreover, such a solution can be computed in $O(\log n)$ rounds in the LOCAL model.

In other words, we define an LCL problem where the inputs are from $\mathcal{E}^{\text{tree}}$ and where the nodes have two options: they can either prove that the graph is not tree-like, or they can do nothing (by outputting \perp). This problem is defined in such a way that an output different from \perp can be used only on structures that are not tree-like (or on tree-like structures whose input labels are incorrect) whereas, if an output different from \perp can be used, then this can be done relatively fast ($O(\log n)$ rounds in the LOCAL model).

How we will use Π^{badTree} . We now describe how the problem Π^{badTree} is used when defining our main LCL problem Π . The problem Π is defined in such a way that all nodes receive an input indicating whether they are part of a grid, or whether they are part of a tree-like structure. Then Π is defined so that:

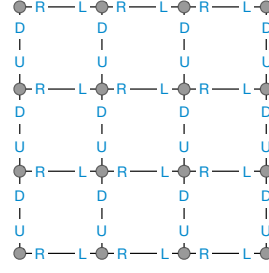
- Nodes can *mark* areas of the graph that do not look like valid hard instances. In particular, nodes having an input indicating that they are part of a tree-like structure are considered marked if they solve Π^{badTree} by giving an output different from \perp .
- We prove that nodes can efficiently mark bad parts of the graph such that the remaining connected components *almost* look like hard instances. Although these remaining parts do not exactly look like hard instances (we provide more details about this later), there is an efficient algorithm for solving them.

3.2 A grid structure

As already mentioned, our hard instances are grids in which we connect a tree-like structure on top of each column. Our next step is to describe a structure that we call *grid structure*. A similar structure has already been used in [14] and in fact some useful properties follow directly from that work.

Definition of the grid structure. Informally, a grid structure denotes a two-dimensional grid that does not wrap around (i.e., it is not a torus). Unlike the case of tree-like structures, we cannot achieve full certification of grids; in particular our scheme for certifying grids is defective and also allows one to label invalid grid structures (and in particular tori) so that no node sees any error.

Figure 5 illustrates a grid structure and its corresponding certificate. An example of a constraint that needs to be checked is that, if a node has R (i.e., “right”) on an incident half-edge, then the corresponding half-edge must be labeled L (i.e., “left”).



■ **Figure 5** An example of a properly labeled grid structure. The labels L, R, U, and D stand, respectively, for left, right, up, and down.

Local certification of grid structures. Next we describe grid structures from a local perspective. In particular, we define a set of input half-edge labels $\mathcal{E}^{\text{grid}}$ of constant size as well as a set of constraints $\mathcal{C}^{\text{grid}}$ over constant distance that satisfy the following:

- For any grid structure G , there exists an assignment of labels of $\mathcal{E}^{\text{grid}}$ to the half-edges of the graph such that the constraints of $\mathcal{C}^{\text{grid}}$ are satisfied on all nodes of G .
- Let G be a graph where half-edges are labeled with labels from $\mathcal{E}^{\text{grid}}$ and such that the constraints of $\mathcal{C}^{\text{grid}}$ are satisfied on all nodes of G . Moreover, suppose that there exists at least one node that has no incident half-edge labeled D (or U) and that there exists at least one node that has no incident half-edge labeled L (or R). Then G is a grid structure.

In other words, if we assume that all nodes satisfy the constraints of $\mathcal{C}^{\text{grid}}$ and we additionally assume that there is at least one node satisfying some additional constraints (which essentially guarantee that there is some “corner” of the grid, thus preventing the graph from being a torus), then the graph is indeed a grid.

Enforcing the dimensions of the grid. As previously discussed, when defining our main problem Π , we allow nodes to *mark* areas of the graph that do not look like valid hard instances. Recall we cannot guarantee that unmarked areas are exactly hard instances. Nevertheless, we are able to ensure that the unmarked parts of the graphs are grids (with properly attached tree-like structures) that are at least as tall as they are large. We call such grids *vertical*. We later discuss how this property is sufficient to obtain an efficient algorithm that has access to shared randomness.

On a high-level, in order to enforce a grid to be vertical, we define a set of input node labels $\mathcal{V}^{\text{vGrid}}$, and a set of constraints $\mathcal{C}^{\text{vGrid}}$, that satisfy the following.

- For any vertical grid structure G , there exists an assignment of labels of $\mathcal{E}^{\text{grid}}$ to the half-edges of G and an assignment of labels of $\mathcal{V}^{\text{vGrid}}$ to the nodes of G , such that the constraints of $\mathcal{C}^{\text{grid}}$ and $\mathcal{C}^{\text{vGrid}}$ are satisfied on all nodes of G .
- Suppose G is a graph where half-edges are labeled with labels from $\mathcal{E}^{\text{grid}}$, nodes are labeled with labels from $\mathcal{V}^{\text{vGrid}}$, such that the constraints of $\mathcal{C}^{\text{grid}}$ and $\mathcal{C}^{\text{vGrid}}$ are satisfied on all nodes of G . Then, G is a vertical grid structure.

We note that, while the second point provides the intuition of what we will do, the statement, as is, is false. We will provide more details in the full version [15].

How we will use vertical grids. Consider a hard instance in which the grid is more large than tall, and in particular consider the extreme case in which the grid is just a path of linear length. In this case, in the case of shared randomness, if we apply the **LOCAL** algorithm for solving Π^{hard} described at the beginning of the section, we would have a success probability

of just $1/2$, since there is only a single row in the grid. In order to guarantee a large-enough success probability, the problem Π will be defined such that unmarked regions are *vertical* grid structures (possibly of small size). This will guarantee the following.

- If the grid has height less than $\log n$, then its width is also less than $\log n$, which implies that nodes can see the whole grid in $O(\log n)$ rounds and solve the problem Π^{hard} by brute force.
- If the grid has height strictly larger than $\log n$, then the success probability will be at least $1 - 1/2^{\log n} = 1 - 1/n$, and hence the algorithm will succeed with high probability.

By combining the above, we will get that $O(\log n)$ rounds will be an upper bound on the runtime for succeeding in solving Π^{hard} with high probability when using shared randomness.

3.3 Family of hard instances

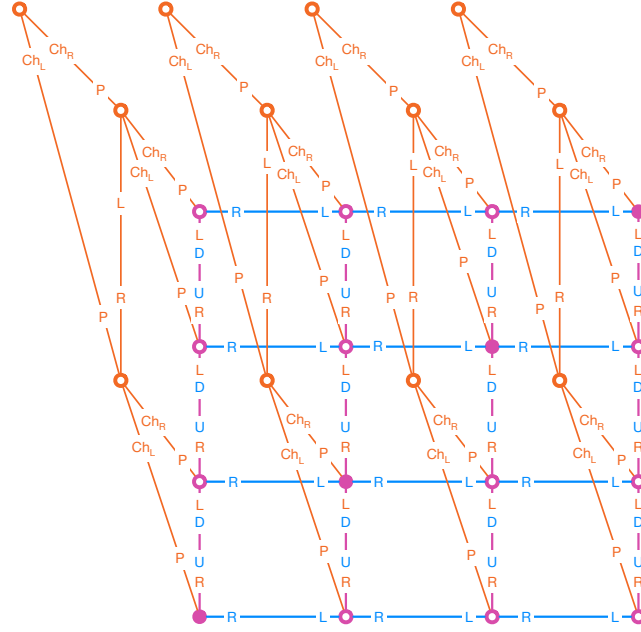
Next we formally define the family \mathcal{G} of hard instances. These graphs are similar to the ones informally explained at the beginning of this section (see Figure 2 for an example), with the only difference that grids do not need to be squares, but they only need to satisfy that their height is at least as large as their width. Then, we define an LCL Π^{badGraph} satisfying the following.

- There are two possible types of output, and different nodes could give outputs of different type.
- One possible output is the empty output, and if a graph G is in \mathcal{G} , the problem Π^{badGraph} is defined such that all nodes must produce the empty output.
- The other possible output is a proof for the fact that $G \notin \mathcal{G}$. More in detail, if the graph is not in the family, nodes can spend $O(\log n)$ time in the LOCAL model to produce a proof of this fact, such that, the subgraph G' induced by nodes producing an empty output satisfies that each connected component of G' is a graph in \mathcal{G} .

Informally, our main problem Π will be defined such that all nodes need to solve Π^{badGraph} , and then, on the subgraph induced by nodes producing an empty output for Π^{badGraph} , nodes need to solve the problem Π^{hard} that we informally defined on hard instances at the beginning of this section. The definition of Π will satisfy that, if we consider some graph $G \in \mathcal{G}$, the only possible way to solve Π is by producing an empty solution for Π^{badGraph} , implying that nodes must then solve Π^{hard} on the whole graph. On the other hand, if a graph G is not in \mathcal{G} , nodes can quickly (i.e., in $O(\log n)$ rounds in the LOCAL model) mark bad parts of the graph, and then solve Π^{hard} on the connected components that are part of \mathcal{G} . In other words, the problem Π^{badGraph} is the one allowing us to remove the promise in the definition of Π^{hard} .

High-level ideas behind the definition of Π^{badGraph} . Observe that, by how hard-instances are constructed, nodes can either be exclusively part of a tree-like structure, or they can belong at the same time to the grid and to some tree-like structure. However, edges can be of three types: they can be exclusively part of the grid, exclusively part of a tree-like structure, or be part of both. In the problem Π^{badGraph} , nodes and edges are input labeled to indicate to which structure(s) they belong to. Then, the possible outputs for Π^{badGraph} are the following.

- If on a node the constraints of $\mathcal{C}^{\text{vGrid}}$ or the constraints of $\mathcal{C}^{\text{tree}}$ are not locally satisfied, or the input labeling of Π^{badGraph} is such that there is some local error in how the two structures are connected, then the node can output an error.
- Consider the subgraph obtained by excluding edges labeled as horizontal edges (i.e., L and R) of the grid. Each connected component is either a valid grid column with a properly-attached tree-like structure on top, or not. In the latter case, we also include the scenario in which some node in the connected component gave error in the previous



■ **Figure 6** An example of a properly input-labeled hard instance. Orange nodes are labeled (treeNode) , purple empty nodes are labeled $(\text{treeNode}, \text{gridNode}, 0)$, purple full nodes are labeled $(\text{treeNode}, \text{gridNode}, 1)$, orange half-edges with label ℓ are labeled $(\text{treeEdge}, \ell)$, blue half-edges with label ℓ are labeled $(\text{gridEdge}, \ell)$, purple half-edges labeled ℓ in blue and ℓ' in orange are labeled $((\text{gridEdge}, \ell), (\text{treeEdge}, \ell'))$.

step. In each connected component, nodes need to solve Π^{badTree} , and thus we get the following two cases:

- The connected component looks good (i.e., it contains no nodes that output error), and the only solution for Π^{badTree} is the one giving \perp (i.e., an empty output) on all nodes;
- The connected component does not look good, and nodes can (efficiently) solve Π^{badTree} such that no node uses the output \perp .

Figure 6 shows an example of an input labeling for Π^{badGraph} that forces all nodes to output \perp . In the full version [15], we will prove that the following properties are satisfied by our construction:

- For any $G \in \mathcal{G}$, i.e., for any hard instance, it is possible to assign an input labeling on G , such that the only valid output for Π^{badGraph} is the one where each node outputs \perp (i.e., empty).
- For any graph G , there exists a solution for Π^{badGraph} , which can be computed in $O(\log n)$ rounds in the LOCAL model, such that the subgraph induced by nodes that output \perp satisfies that each connected component is a graph in \mathcal{G} .

Note that, as discussed earlier, these two properties are exactly the ones that we need in order to make Π^{hard} promise-free.

3.4 Problem Π and its complexity in the LOCAL model

We already discussed the high-level idea behind the definition of Π , and in the full version [15] we define the LCL problem Π more formally. We now give a bit more details about Π . Each node u receives a pair of inputs. One input is exactly the input for Π^{badGraph} , and the other

input is a single bit b_u . The problem Π is defined such that each node needs to solve Π^{badGraph} , and then, on the subgraph induced by nodes producing an empty output for Π^{badGraph} , each node u needs to solve the problem Π^{hard} , where the input of node u for Π^{hard} is b_u . While Π^{hard} was explained on square grids of size $\Theta(\sqrt{n}) \times \Theta(\sqrt{n})$, as we already discussed, hard instances can be a bit different from square grids, which makes it harder to prove an upper bound for the problem Π . Next we will provide lower and upper bounds for the problem Π in different models of computation.

The complexity of Π in the LOCAL model with shared randomness. When we discussed vertical grids, we also provided the high-level idea of the upper bound for solving Π^{hard} with shared randomness in the LOCAL model on hard instances, which we now recap and explain how it is used to solve Π in any graph. At first, the nodes spend $O(\log n)$ rounds to mark the “bad” parts of the graph. After that, each remaining connected component is a hard instance. Then, the problem Π^{hard} is solved by brute force on components of diameter $O(\log n)$, while shared randomness is used to solve components of larger diameter.

The complexity of Π in the LOCAL model with private randomness. In order to prove a lower bound of $\Omega(\sqrt{n})$ in the LOCAL model for private randomness, we essentially show that any algorithm that is too fast cannot deviate too much from being deterministic, or in other words, for each row of a hard instance, the algorithm must fix the output bits almost deterministically (i.e., the output must always be the same, with high probability). Then, we fix the input of the last column in an adversarial way, as a function of the almost-deterministic outputs of the algorithm, and we prove that in such case the failure probability of the algorithm is too large.

3.5 Complexity of Π in other models

Finally, we extend the $\Omega(\sqrt{n})$ lower bound to several other models that are far more powerful than LOCAL:

- **SLOCAL with private randomness.** In the SLOCAL model [29], nodes are revealed in a sequential, potentially adversarial order. Each time a node is revealed, it sees all the states of previously revealed nodes that are at most T hops away from it (in particular also their outputs) and is asked to commit to an output. This model is more powerful than LOCAL since the sequential processing of nodes gives us symmetry breaking essentially for free.
- **Deterministic online-LOCAL model.** The deterministic online-LOCAL model [3] is similar to SLOCAL in that the nodes are also revealed following some sequential order. Nevertheless, it is potentially more powerful than SLOCAL because it is able to maintain *global* knowledge of what has been revealed thus far. We show the lower bound for the deterministic variant of online-LOCAL. A randomized variant of online-LOCAL also exists [2], but it has access to shared randomness by definition, and so the upper bound from LOCAL extends there.
- **Bounded-dependence model.** The bounded-dependence model encompasses all algorithms satisfying the property that the output of any node is independent of outputs that are more than T hops away from it. This includes, for instance, all quantum-LOCAL algorithms without a pre-shared quantum state. The bounded-dependence model has been shown to be less powerful than the randomized version of online-LOCAL [2], but its relation to deterministic online-LOCAL is still unclear.

Throughout this work we assume that the nodes have (implicitly or explicitly) some knowledge on the total number of nodes n . In the full version [15] we show that this is a necessary assumption.

3.6 Latest follow-up work

We briefly mention here two very recent developments. First, it is now known that there are LCL problems that admit a distributed quantum advantage [8, 13]; this was an open question at the time we originally wrote this paper. Second, there is now a follow-up paper by Hadad and Naor [31] that explores shared randomness in a setting in which the choice of the input may depend on the shared random string.

References

- 1 Jayadev Acharya, Clément L. Canonne, and Himanshu Tyagi. Communication-constrained inference and the role of shared randomness. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 30–39. PMLR, 2019. URL: <http://proceedings.mlr.press/v97/acharya19a.html>.
- 2 Amirreza Akbari, Xavier Coiteux-Roy, Francesco D’Amore, François Le Gall, Henrik Lievonon, Darya Melnyk, Augusto Modanese, Shreyas Pai, Marc-Olivier Renou, Václav Rozhon, and Jukka Suomela. Online locality meets distributed quantum computing. *CoRR*, abs/2403.01903, 2024. doi:10.48550/arXiv.2403.01903.
- 3 Amirreza Akbari, Navid Eslami, Henrik Lievonon, Darya Melnyk, Joonas Särkijärvi, and Jukka Suomela. Locality in online, dynamic, sequential, and distributed graph algorithms. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPICs*, pages 10:1–10:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ICALP.2023.10.
- 4 Heger Arfaoui and Pierre Fraigniaud. What can be computed without communications? *SIGACT News*, 45(3):82–104, 2014. doi:10.1145/2670418.2670440.
- 5 Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. The distributed complexity of locally checkable problems on paths is decidable. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 262–271. ACM, 2019. doi:10.1145/3293611.3331606.
- 6 Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studený, and Jukka Suomela. Efficient classification of locally checkable problems in regular trees. In Christian Scheideler, editor, *36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA*, volume 246 of *LIPICs*, pages 8:1–8:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.DISC.2022.8.
- 7 Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. Locally checkable problems in rooted trees. *Distributed Computing*, 36(3):277–311, 2023. doi:10.1007/S00446-022-00435-9.
- 8 Alkida Balliu, Sebastian Brandt, Xavier Coiteux-Roy, Francesco D’Amore, Massimo Equi, François Le Gall, Henrik Lievonon, Augusto Modanese, Dennis Olivetti, Marc-Olivier Renou, Jukka Suomela, Lucas Tendick, and Isadora Veeren. Distributed quantum advantage for local problems. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, CZ, Czech Republic, June 23-27, 2025*. ACM, 2025. doi:10.1145/3717823.3718233.

- 9 Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPICs*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.DISC.2020.17.
- 10 Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. *Journal of the ACM*, 68(5):39:1–39:30, 2021. doi:10.1145/3461458.
- 11 Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. How much does randomness help with locally checkable problems? In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, pages 299–308. ACM, 2020. doi:10.1145/3382734.3405715.
- 12 Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the LOCAL model. *Distributed Computing*, 34(4):259–281, 2021. doi:10.1007/S00446-020-00375-2.
- 13 Alkida Balliu, Filippo Casagrande, Francesco d’Amore, Massimo Equi, Barbara Keller, Henrik Lievonen, Dennis Olivetti, Gustav Schmid, and Jukka Suomela. Distributed quantum advantage in locally checkable labeling problems, 2025. doi:10.48550/arXiv.2504.05191.
- 14 Alkida Balliu, Keren Censor-Hillel, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Locally checkable labelings with small messages. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPICs*, pages 8:1–8:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.DISC.2021.8.
- 15 Alkida Balliu, Mohsen Ghaffari, Fabian Kuhn, Augusto Modanese, Dennis Olivetti, Mikaël Rabie, Jukka Suomela, and Jara Uitto. Shared randomness helps with local distributed problems. *CoRR*, abs/2407.05445, 2024. doi:10.48550/arXiv.2407.05445.
- 16 Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1307–1318. ACM, 2018. doi:10.1145/3188745.3188860.
- 17 Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed Lovász local lemma. In Daniel Wachs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 479–488. ACM, 2016. doi:10.1145/2897518.2897570.
- 18 Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznanski. LCL problems on grids. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 101–110. ACM, 2017. doi:10.1145/3087801.3087833.
- 19 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 615–624. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.72.
- 20 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. *SIAM Journal on Computing*, 48(1):33–69, 2019. doi:10.1137/17M1157957.
- 21 Yi-Jun Chang, Jan Studený, and Jukka Suomela. Distributed graph problems through an automata-theoretic lens. *Theoretical Computer Science*, 951:113710, 2023. doi:10.1016/J.TCS.2023.113710.

- 22 Xavier Coiteux-Roy, Francesco D’Amore, Rishikesh Gajjala, Fabian Kuhn, François Le Gall, Henrik Lievonen, Augusto Modanese, Marc-Olivier Renou, Gustav Schmid, and Jukka Suomela. No distributed quantum advantage for approximate graph coloring. In Bojan Mohar, Igor Shinkar, and Ryan O’Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1901–1910. ACM, 2024. doi:10.1145/3618260.3649679.
- 23 Pierluigi Crescenzi, Pierre Fraigniaud, and Ami Paz. Trade-offs in distributed interactive proofs. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, volume 146 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.DISC.2019.13.
- 24 Sameep Dahal, Francesco D’Amore, Henrik Lievonen, Timothé Picavet, and Jukka Suomela. Brief announcement: Distributed derandomization revisited. In Rotem Oshman, editor, *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L’Aquila, Italy*, volume 281 of *LIPIcs*, pages 40:1–40:5. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.DISC.2023.40.
- 25 Manuela Fischer and Mohsen Ghaffari. Sublogarithmic distributed algorithms for Lovász local lemma, and the complexity hierarchy. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.DISC.2017.18.
- 26 Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. What can be observed locally? In Idit Keidar, editor, *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, volume 5805 of *Lecture Notes in Computer Science*, pages 243–257. Springer, 2009. doi:10.1007/978-3-642-04355-0_26.
- 27 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 662–673. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00069.
- 28 Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, Yannic Maus, Jukka Suomela, and Jara Uitto. Improved distributed degree splitting and edge coloring. *Distributed Computing*, 33(3-4):293–310, 2020. doi:10.1007/S00446-018-00346-8.
- 29 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 784–797. ACM, 2017. doi:10.1145/3055399.3055471.
- 30 Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2505–2523. SIAM, 2017. doi:10.1137/1.9781611974782.166.
- 31 Adar Hadad and Moni Naor. Shared randomness in locally checkable problems: The role of computational assumptions, 2025. doi:10.48550/arXiv.2504.17583.
- 32 Alexander E. Holroyd. Symmetrization for finitely dependent colouring. *Electronic Communications in Probability*, 29, 2024. doi:10.1214/24-ecp600.
- 33 Alexander E. Holroyd, Tom Hutchcroft, and Avi Levy. Finitely dependent cycle coloring. *Electronic Communications in Probability*, 23, 2018. doi:10.1214/18-ecp118.
- 34 Alexander E. Holroyd and Thomas M. Liggett. Finitely Dependent Coloring. *Forum of Mathematics, Pi*, 4, 2016. doi:10.1017/fmp.2016.7.
- 35 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 938–948. SIAM, 2010. doi:10.1137/1.9781611973075.76.

- 36 Gowtham R. Kurri, Vinod M. Prabhakaran, and Anand D. Sarwate. Coordination through shared randomness. *IEEE Transactions on Information Theory*, 67(8):4948–4974, 2021. doi:10.1109/TIT.2021.3091604.
- 37 François Le Gall, Harumichi Nishimura, and Ansis Rosmanis. Quantum advantage for the LOCAL model in distributed computing. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 49:1–49:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.STACS.2019.49.
- 38 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.
- 39 Pedro Montealegre, Diego Ramírez-Romero, and Ivan Rapaport. Shared vs private randomness in distributed interactive proofs. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 51:1–51:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ISAAC.2020.51.
- 40 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.
- 41 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, 2000.
- 42 Anup Rao and Amir Yehudayoff. *Communication Complexity and Applications*. Cambridge University Press, 2020.
- 43 Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 350–363. ACM, 2020. doi:10.1145/3357713.3384298.
- 44 Jukka Suomela. Landscape of locality (invited talk). In Susanne Albers, editor, *17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2020, June 22-24, 2020, Tórshavn, Faroe Islands*, volume 162 of *LIPIcs*, pages 2:1–2:1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.SWAT.2020.2.
- 45 Aleksandr Tereshchenko. Automated classification of distributed graph problems. Master’s thesis, Aalto University, 2021. URL: <https://urn.fi/URN:NBN:fi:aalto-202105236941>.
- 46 Ádám Timár. Finitely dependent random colorings of bounded degree graphs. *CoRR*, abs/2402.17068, 2024. doi:10.48550/arXiv.2402.17068.