


Local Problems in Trees Across a Wide Range of Distributed Models

Anubhav Dhar ✉ 

Aalto University, Espoo, Finland
Indian Institute of Technology Kharagpur,
India

Eli Kujawa ✉ 

Aalto University, Espoo, Finland
University of Illinois Urbana-Champaign,
IL, USA

Henrik Lievonon ✉ 


Aalto University, Espoo, Finland

Augusto Modanese ✉ 

Aalto University, Espoo, Finland

Mikail Muftuoglu ✉ 

Aalto University, Espoo, Finland

Jan Studený ✉ 

Aalto University, Espoo, Finland

Jukka Suomela ✉ 

Aalto University, Espoo, Finland

Abstract

The *randomized online-LOCAL* model captures a number of models of computing; it is at least as strong as all of these models:

- the classical LOCAL model of distributed graph algorithms,
- the quantum version of the LOCAL model,
- finitely dependent distributions [e.g. Holroyd 2016],
- any model that does not violate physical causality [Gavoille, Kosowski, Markiewicz, DISC 2009],
- the SLOCAL model [Ghaffari, Kuhn, Maus, STOC 2017], and
- the dynamic-LOCAL and online-LOCAL models [Akbari et al., ICALP 2023].

In general, the online-LOCAL model can be much stronger than the LOCAL model. For example, there are *locally checkable labeling problems* (LCLs) that can be solved with logarithmic locality in the online-LOCAL model but that require polynomial locality in the LOCAL model.

However, in this work we show that in *trees*, many classes of LCL problems have the same locality in deterministic LOCAL and randomized online-LOCAL (and as a corollary across all the above-mentioned models). In particular, these classes of problems do not admit any distributed quantum advantage.

We present a near-complete classification for the case of *rooted regular trees*. We also fully classify the super-logarithmic region in *unrooted regular trees*. Finally, we show that in general trees (rooted or unrooted, possibly irregular, possibly with input labels) problems that are global in deterministic LOCAL remain global also in the randomized online-LOCAL model.

2012 ACM Subject Classification Computing methodologies → Distributed algorithms

Keywords and phrases Distributed algorithms, quantum-LOCAL model, randomized online-LOCAL model, locally checkable labeling problems, trees

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2024.27

Related Version *Full Version*: <https://arxiv.org/abs/2409.13795> [13]

Funding This work was supported in part by the Research Council of Finland, Grants 333837 and 359104, the Aalto Science Institute (ASCI), and the Helsinki Institute for Information Technology (HIIT).



© Anubhav Dhar, Eli Kujawa, Henrik Lievonon, Augusto Modanese, Mikail Muftuoglu, Jan Studený, and Jukka Suomela;

licensed under Creative Commons License CC-BY 4.0

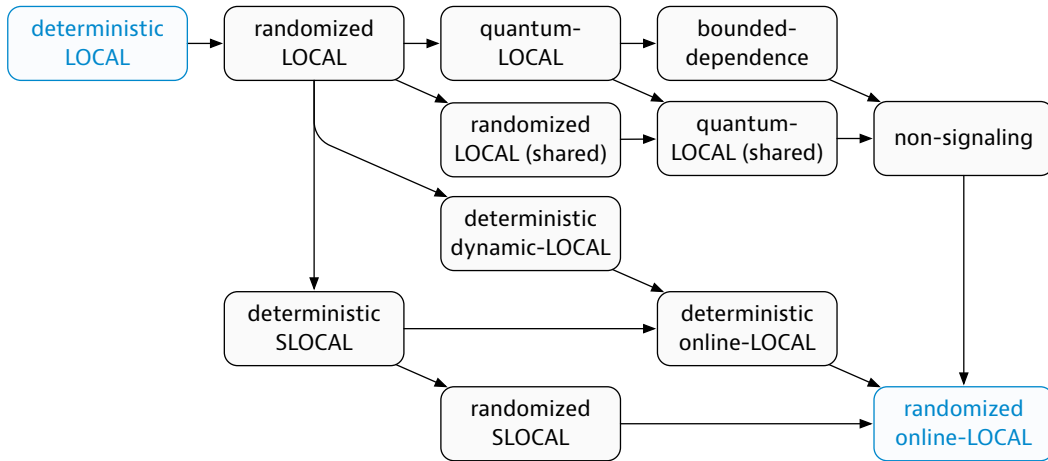
28th International Conference on Principles of Distributed Systems (OPODIS 2024).

Editors: Silvia Bonomi, Letterio Galletta, Etienne Rivière, and Valerio Schiavoni; Article No. 27; pp. 27:1–27:17



Leibniz International Proceedings in Informatics

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



■ **Figure 1** Landscape of models, based on [1]. In this work we show that for many families of LCL problems, the two extreme models – *deterministic LOCAL* and *randomized online-LOCAL* – are equally strong, and hence the same holds for *all* intermediate models in this diagram.

1 Introduction

The *randomized online-LOCAL model* was recently introduced in [1]; this is a model of computing that is at least as strong as many other models that have been widely studied in the theory of distributed computing, as well as a number of emerging models; see Figure 1. In particular, different variants of the *quantum-LOCAL* and *SLOCAL* models are sandwiched between the classical *deterministic LOCAL* model and the *randomized online-LOCAL* model.

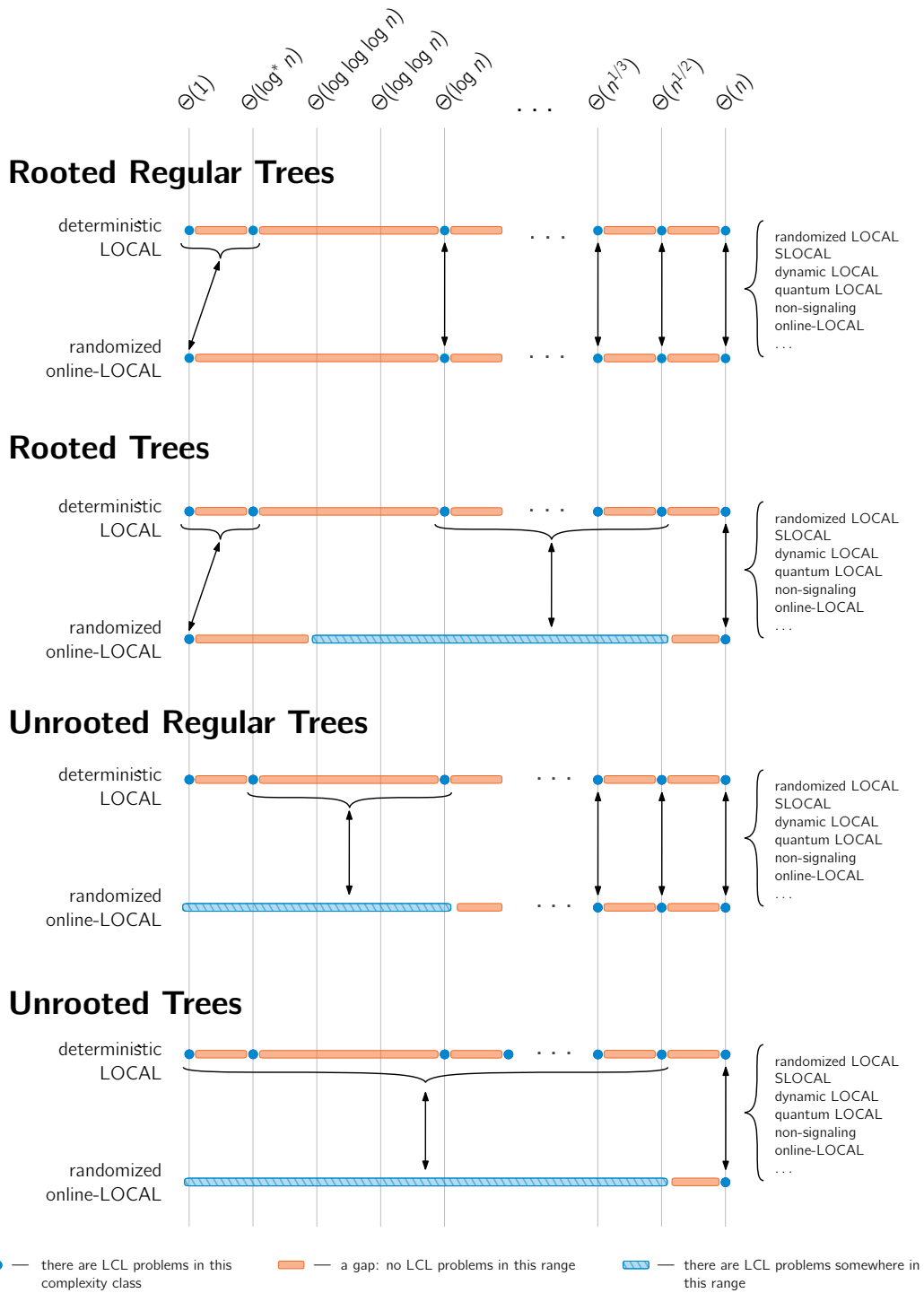
While the *randomized online-LOCAL* model is in general much stronger than the *deterministic LOCAL* model, in this work we show that in *trees*, for many families of *local* problems these two models (and hence all models in between) are asymptotically equally strong. Figure 2 summarizes the relations that we have thanks to this work; for comparison, see Figure 3 for the state of the art before this work.

1.1 Models

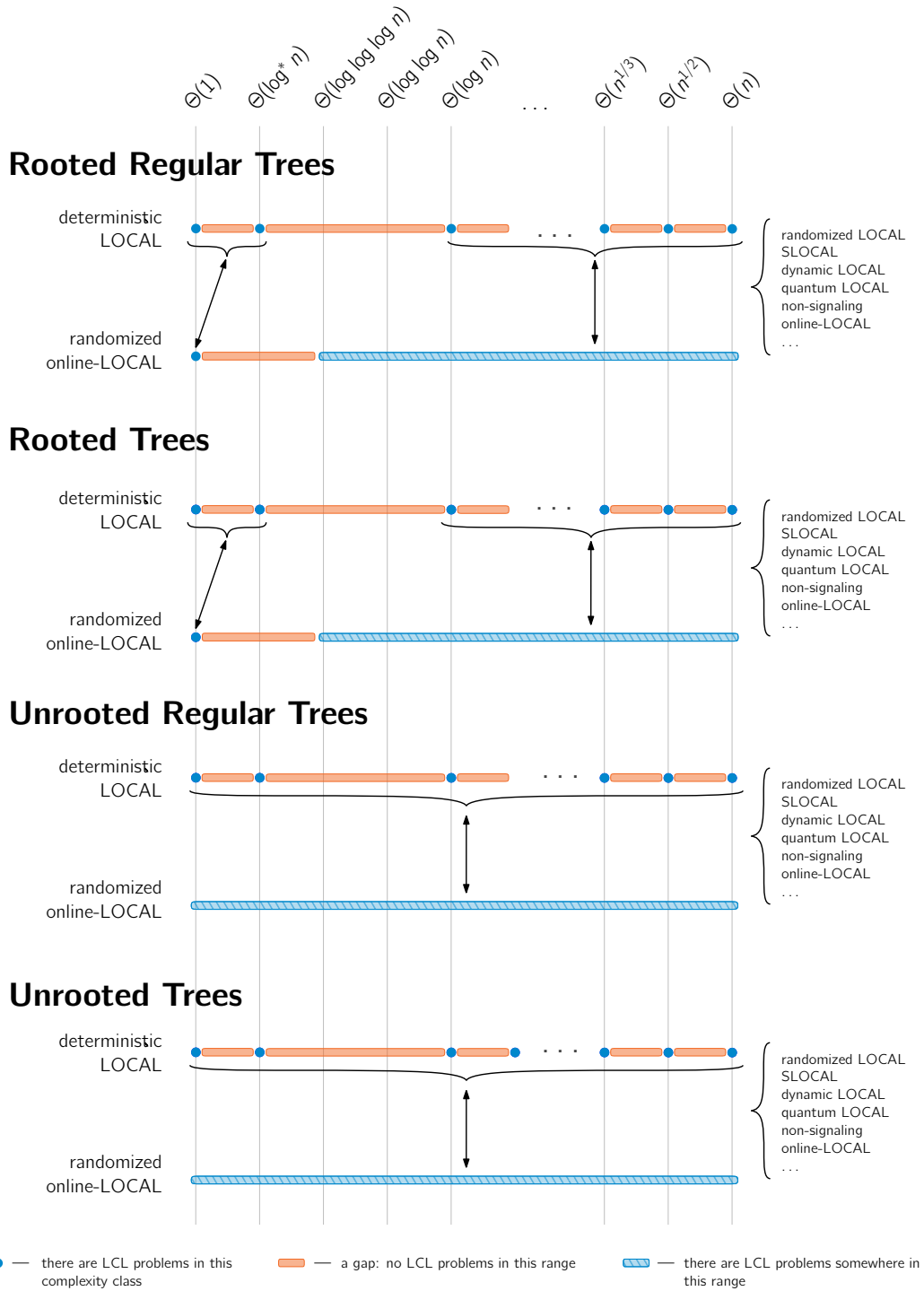
We will define all relevant models formally in Section 3, but for now the following brief definitions suffice:

- In the **deterministic LOCAL** model, an algorithm A with locality T works as follows: The adversary chooses a graph G and an assignment of polynomially-sized unique identifiers. Algorithm A is applied to all nodes *simultaneously in parallel*. When we apply A to node v , algorithm A gets to see the radius- T neighborhood of v and, using this information, it has to choose the output label of node v .
- In the **randomized online-LOCAL** model, an algorithm A with locality T works as follows: The adversary chooses a graph G and a processing order σ . Then the adversary presents nodes *sequentially* following the order σ . Whenever a node v is presented, algorithm A gets to see the radius- T neighborhood of v and, using this information *as well as* all information it has seen previously and a global source of random bits, it has to choose the output label of node v .

This means that *randomized online-LOCAL* is stronger than *deterministic LOCAL* in at least three different ways: (1) we have access to shared randomness, (2) the sequential processing order can be used to break symmetry, and (3) there is global memory thanks to which we



■ **Figure 2** Landscape of LCL problems in trees after this work – compare with Figure 3 that shows the state of the art before this work.



■ **Figure 3** Landscape of LCL problems in trees before this work – compare with Figure 2 to see the impact of our new contributions.

can remember everything we have seen so far. A reader familiar with the SLOCAL model can interpret it as randomized SLOCAL augmented with global memory. Note that the adversary is oblivious; it cannot adapt G and σ based on the actions of A .

In general, online-LOCAL (with or without randomness) is much stronger than the classical LOCAL model (with or without randomness). For example, leader election is trivial in online-LOCAL, even with locality $T = 0$ (the first node that the adversary presents is marked as the leader). However, what is much more interesting is whether online-LOCAL has advantage over LOCAL for problems defined using local constraints, such as graph coloring.

1.2 Prior Work on LCL Problems

We will study here locally checkable labeling problems (LCLs) [21]; these are problems that can be specified by giving a finite set of valid local neighborhoods. By prior work, we know a number of LCL problems that can separate the models in Figure 1, for example:

- Sinkless orientation has locality $\Theta(\log n)$ in deterministic LOCAL, locality $\Theta(\log \log n)$ in randomized LOCAL, and locality $\Theta(\log \log \log n)$ in randomized SLOCAL [7, 8, 10, 14, 16].
- One can construct an (artificial) LCL problem that shows that having access to shared randomness helps exponentially in comparison with private randomness, and this also gives a separation between e.g. randomized LOCAL and randomized online-LOCAL [6].
- In 2-dimensional grids, 3-coloring has polynomial locality in e.g. randomized SLOCAL and non-signaling models, while it can be solved with logarithmic locality in deterministic online-LOCAL [2, 12].
- In paths, 3-coloring requires $\Theta(\log^* n)$ locality in randomized LOCAL [19, 20], while it can be solved with $O(1)$ locality in the bounded-dependence model [18].

However, we do not have any such separations in rooted trees outside the $O(\log^* n)$ region. Moreover, in general unrooted trees, all separations are in the sub-logarithmic region. In this work we give a justification for this phenomenon.

1.3 Contributions

We study in this work LCL problems in the following three main settings, all familiar from prior work:

1. LCL problems in trees in general,
2. LCL problems in unrooted regular trees (there are no inputs and we only care about nodes with exactly d neighbors),
3. LCL problems in rooted regular trees (there are no inputs, edges are oriented, and we only care about nodes with exactly 1 successor and d predecessors).

In all these settings, it is known that the locality of any LCL problem in deterministic LOCAL falls in one of the following classes: $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, or $\Theta(n^{1/k})$ for some $k = 1, 2, 3, \dots$ [3, 4, 9, 11, 17]. Furthermore, in the case of rooted regular trees, we can (relatively efficiently) also decide which of these classes any given LCL problem belongs to [3].

Our main contribution is showing that, in many of these complexity classes, **deterministic LOCAL and randomized online-LOCAL are asymptotically equally strong**:

1. In general trees, the localities in randomized online-LOCAL and deterministic LOCAL are asymptotically equal in the region $\omega(\sqrt{n})$.
2. In unrooted regular trees, the localities in randomized online-LOCAL and deterministic LOCAL are asymptotically equal in the region $\omega(\log n)$.
3. In rooted regular trees, the localities in randomized online-LOCAL and deterministic LOCAL are asymptotically equal in the region $\omega(\log^* n)$.

By prior work, the relation between deterministic LOCAL and randomized online-LOCAL was well-understood in the $o(\log \log \log n)$ region for rooted (not necessarily regular) trees [1]; see Figure 3 for the state of the art before this work. Putting together prior results and new results, the landscape shown in Figure 2 emerges.

1.4 Roadmap

We give an overview of key ideas in Section 2. We formally define LCL problems and the models of computing in Section 3, and then we are ready to analyze the sub-logarithmic region in rooted regular trees in Section 4. For the remaining results, please refer to the full version of the paper [13].

2 Key Ideas, Technical Overview, and Comparison with Prior Work

We keep the discussion at a high level but invite the interested reader to consult the formal definitions in Section 3 as needed.

Our results heavily build on prior work that has studied LCL problems in deterministic and randomized LOCAL models. In essence, our goal is to extend its scope across the entire landscape of models in Figure 1.

For example, by prior work we know that any LCL problem Π in trees has locality either $O(\sqrt{n})$ or $\Omega(n)$ in the deterministic LOCAL model [5]. Results of this type are known as *gap results*. For our purposes, it will be helpful to interpret such a gap result as a *speedup theorem* that allows us to speed up deterministic LOCAL algorithms:

If one can solve Π with locality $o(n)$ in **deterministic LOCAL**, then one can solve the same problem Π with locality $O(\sqrt{n})$ in **deterministic LOCAL**.

In essence our objective is to strengthen the statement as follows:

If one can solve Π with locality $o(n)$ in **randomized online-LOCAL**, then one can solve the same problem Π with locality $O(\sqrt{n})$ in **deterministic LOCAL**.

The critical implication would be that a faster randomized online-LOCAL algorithm results not only in a faster randomized online-LOCAL algorithm, but also in a faster deterministic LOCAL algorithm – we could not only reduce locality “for free” but even switch to a weaker model. As a consequence, the complexity class $\Theta(n)$ would contain the same problems across all models – since $\Omega(n)$ in randomized online-LOCAL trivially implies $\Omega(n)$ in deterministic LOCAL (which is a weaker model), the above would give us that $o(n)$ in randomized online-LOCAL implies $o(n)$ in deterministic LOCAL.

If we could prove a similar statement for *every* gap result for LCLs in trees, we would then have a similar implication for each complexity class: the complexities across all models in Figure 1 would be the same for every LCL problem in trees. Alas, such a statement cannot be true in full generality (as we discussed above, there are some known separations between the models), but in this work we take major steps in many cases where that seems to hold. To understand how we achieve this, it is useful to make a distinction between two flavors of prior work: classification- and speedup-style arguments; we will make use of both techniques.

2.1 Classification Arguments for Regular Trees

First, we have prior work that is based on the idea of classification of LCLs, see e.g. [3]. The high-level strategy is as follows:

1. Define some property P so that, for any given LCL Π , we can decide whether Π has property P .
2. Show that, if Π can be solved with locality $o(n)$ in **deterministic LOCAL**, then this implies that Π must have property P .
3. Show that any problem with property P can be solved with locality $O(\sqrt{n})$ in **deterministic LOCAL**.

Such a strategy not only shows that there is a gap in the complexity landscape, but it also usually gives an efficient strategy for determining what is the complexity of a given LCL (as we will need to check some set of properties P_1, P_2, \dots and see which of them holds for a given Π in order to determine where we are in the complexity landscape). For such results our key idea is to modify the second step as follows:

- 2'. Show that, if Π can be solved with locality $o(n)$ in **randomized online-LOCAL**, then this implies that Π must have property P .

Note that we do not need to change the third step; as soon as we establish property P , the pre-existing deterministic LOCAL algorithm kicks in. This is, in essence, what we do in the technical section: we take the prior classification from [3] for rooted regular trees in the sub-logarithmic region and show that it can be extended to cover the entire range of models all the way from deterministic LOCAL to randomized online-LOCAL.

Log-star certificates. One key property that we make use of are *certificates of $O(\log^* n)$ solvability* for rooted regular trees, introduced in [4]. In Section 4, we show that the existence of an $o(\log n)$ -locality randomized online-LOCAL algorithm implies that the LCL problem has such a certificate; in turn, as the name suggests, the certificate of solvability implies that the problem can be solved with $O(\log^* n)$ locality in both CONGEST and LOCAL. This then extends to an $O(1)$ upper bound in SLOCAL [15] and in deterministic online-LOCAL [2], which then directly implies the same upper bound also for randomized online-LOCAL model. Hence we obtain the following:

► **Theorem 2.1.** *Let Π be an LCL problem on rooted regular trees. If Π can be solved with $o(\log n)$ locality in randomized online-LOCAL, then it can be solved in LOCAL with $O(\log^* n)$ locality. Consequently, Π can be solved with $O(1)$ locality in SLOCAL, and thus also with the same locality in online-LOCAL, even deterministically.*

Depth. Another key property that we make use of is the *depth* of an LCL (see the full version of the paper [13]): to every LCL problem Π on *regular* trees there is an associated quantity d_Π called its depth that depends only on the description of Π and which allows us to classify its complexity in the deterministic LOCAL model [3]. We show that depth also captures the complexity in randomized online-LOCAL. We analyze the case of rooted regular trees and show the following:

► **Theorem 2.2.** *Let Π be an LCL problem on rooted regular trees with finite depth $k = d_\Pi > 0$. Then any algorithm \mathcal{A} solving Π in randomized online-LOCAL must have locality $\Omega(n^{1/k})$.*

The high-level strategy is as follows: Assume we have an algorithm with locality $o(n^{1/k})$. We construct a large experiment graph and run the algorithm on that. If the algorithm fails, then it contradicts the assumption that we have such algorithm. If the algorithm succeeds, then it also produces a certificate showing that $d_\Pi > k$, which again is a contradiction. We prove an analogous statement for unrooted regular trees:

► **Theorem 2.3.** *Let Π be an LCL problem on unrooted regular trees with finite depth $k = d_\Pi > 0$. Then any randomized online-LOCAL algorithm for Π has locality $\Omega(n^{1/k})$.*

Putting things together. Combining the new results with results from prior work [3, 4, 9, 11, 17], we extend the characterization of LCLs in both unrooted and rooted regular trees all the way up to randomized online-LOCAL:

► **Corollary 2.4.** *Let Π be an LCL problem on unrooted regular trees. Then the following holds:*

- *If $d_\Pi = 0$, the problem is unsolvable.*
- *If $0 < d_\Pi < \infty$, then Π has locality $\Theta(n^{1/d_\Pi})$ in both deterministic LOCAL and randomized online-LOCAL.*
- *If $d_\Pi = \infty$, then Π can be solved in CONGEST – and hence also in online-LOCAL, even deterministically – with locality $O(\log n)$.*

► **Corollary 2.5.** *If Π is a solvable LCL problem in rooted regular trees, then it belongs to one of the following four classes:*

1. $O(1)$ in both deterministic LOCAL and randomized online-LOCAL
2. $\Theta(\log^* n)$ in deterministic LOCAL and $O(1)$ in randomized online-LOCAL
3. $\Theta(\log n)$ in both deterministic LOCAL and randomized online-LOCAL
4. $\Theta(n^{1/k})$ in both deterministic LOCAL and randomized online-LOCAL where $k = d_\Pi > 0$

Comparison with prior work. There is prior work [2] that took first steps towards classifying LCL problems in rooted regular trees; the main differences are as follows:

1. We extend the classification all the way to randomized online-LOCAL, while [2] only discusses deterministic online-LOCAL. While this may at first seem like a technicality, Figure 1 highlights the importance of this distinction: randomized online-LOCAL captures *all* models in this diagram. This includes in particular the quantum-LOCAL model and the non-signaling model. Note that it is currently open if deterministic online-LOCAL captures either of these models.
2. We build on the classification of [3], while [2] builds on the (much simpler and weaker) classification of [4]. This has two direct implications:
 - a. Our work extends to *unrooted trees*; the results in [2] only apply to rooted trees.
 - b. We get an *exact classification* also for complexities $\Theta(n^{1/k})$. Meanwhile [2] essentially only shows that, if the complexity is $\Theta(n^{1/k})$ in deterministic LOCAL, then it is $\Theta(n^{1/\ell})$ in online-LOCAL for some $\ell \leq k$; hence it leaves open the possibility that these problems could be solved much faster (i.e., $\ell \gg k$) in online-LOCAL.

2.2 Speedup Arguments for General Trees

Second, we have also prior work based on speedup simulation arguments, see e.g. [5]. The high-level strategy there is as follows:

Assume we are given some algorithm \mathcal{A} that solves Π with locality $o(n)$ in **deterministic LOCAL**. Then we can construct a new algorithm \mathcal{A}' that uses \mathcal{A} as a black box to solve Π with locality $O(\sqrt{n})$ in **deterministic LOCAL**.

Unlike classification-style arguments, this does not (directly) yield an algorithm for determining the complexity of a given LCL problem. In essence, this can be seen as a black-box simulation of \mathcal{A} . For speedup-style arguments, the key idea for us to proceed is as follows:

Assume we are given some algorithm \mathcal{A} that solves Π with locality $o(n)$ in **randomized online-LOCAL**. Then we can construct a new algorithm \mathcal{A}' that uses \mathcal{A} as a black box to solve Π with locality $O(\sqrt{n})$. Moreover, we can simulate \mathcal{A}' efficiently in the **deterministic LOCAL** model.

Here a key challenge is that we need to explicitly change models; while *in general* deterministic LOCAL is not strong enough to simulate randomized online-LOCAL, we show that *in this case* such a simulation is possible. Starting with the argument from [5], we show that sublinear-locality randomized online-LOCAL algorithms not only admit speedup inside randomized online-LOCAL, but also a simulation in deterministic LOCAL. Formally, we prove the following in the full version of this paper [13]:

► **Theorem 2.6.** *Suppose Π is an LCL problem that can be solved with $o(n)$ locality in online-LOCAL (with or without randomness) in unrooted trees. Then Π can be solved in unrooted trees in LOCAL with $O(\sqrt{n})$ locality (with or without randomness, respectively).*

3 Preliminaries

We denote the set of natural numbers, including zero, by \mathbb{N} . For positive integers, excluding zero, we use \mathbb{N}_+ . For a set or multiset X and $k \in \mathbb{N}_+$, we write $\binom{X}{k}$ for the set of multisets over X of cardinality k .

All graphs are simple. Unless stated otherwise, n always denotes the number of nodes in the graph. An algorithm solving some graph problem is said to succeed with high probability if, for all graphs except finitely many, it fails with probability at most $1/n$.

3.1 Locally Checkable Labeling Problems

In this section we define locally checkable labeling (LCL) problems. We first recall the most general definition due to Naor and Stockmeyer [21]:

► **Definition 3.1** (LCL problem in general graphs). *A locally checkable labeling (LCL) problem $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, \mathcal{C}, r)$ is defined as follows:*

- Σ_{in} and Σ_{out} are finite, non-empty sets of input and output labels, respectively.
- $r \in \mathbb{N}_+$ is the checkability radius.
- \mathcal{C} is the set of constraints, namely a finite set of graphs where:
 - Each graph $H \in \mathcal{C}$ is centered at some node v .
 - The distance of v from all other nodes in H is at most r .
 - Each node $u \in H$ is labeled with a pair $(i(u), o(u)) \in \Sigma_{\text{in}} \times \Sigma_{\text{out}}$.

For a graph $G = (V, E)$ whose vertices are labeled according to Σ_{in} , a (node) labeling of a graph $G = (V, E)$ is a solution to Π if it labels every node $v \in V$ with a label from Σ_{out} such that the r -neighborhood of v in G is identical to some graph of \mathcal{C} (when we place v at the center of the respective graph in \mathcal{C}). Every node for which this holds is said to be correctly labeled. If all nodes of G are labeled with a solution, then G itself is correctly labeled.

We use this definition to refer to problems in the case of general, that is, not necessarily regular trees. For regular trees, we use two other formalisms, one for the case of rooted and one for that of unrooted trees. These are simpler to work with and require checking only labels in the immediate vicinity of a node or half-edge. We note there is precedent in the literature for taking this approach (see, e.g., [3]).

► **Definition 3.2** (LCL problem on regular rooted trees). *An LCL problem on regular rooted trees is a tuple $\Pi = (\Delta, \Sigma, \mathcal{V})$ where:*

- $\Delta \in \mathbb{N}_+$ and Σ is a finite, non-empty set.
- The (node) constraints form a set \mathcal{V} of pairs (σ, S) where $\sigma \in \Sigma$ and $S \subseteq \binom{\Sigma}{\Delta}$.

27:10 Local Problems in Trees Across a Wide Range of Distributed Models

For a rooted tree $T = (V, E)$ with nodes having degree in $\{1, \Delta\}$, a solution to Π is a labeling $\ell: V \rightarrow \Sigma$ such that, for each node v with Δ children u_1, \dots, u_Δ , we have $(\ell(v), \{\ell(u_1), \dots, \ell(u_\Delta)\}) \in \mathcal{V}$. Again, this requirement allows us to speak of correctly labeled nodes and thus also correctly labeled trees.

Under this formalism, leaves may be labeled arbitrarily. When Π is clear from the context, we refer to the elements of $\left(\binom{\Sigma}{\Delta}\right)$ as node configurations.

Meanwhile, in the case of unrooted trees, we work with LCL problems where the labels are placed on half-edges (instead of nodes) and are subject to node and edge constraints.

► **Definition 3.3** (LCL problem on regular unrooted trees). *An LCL problem on unrooted trees is a tuple $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ where:*

- $\Delta \in \mathbb{N}_+$ and Σ is a finite, non-empty set.
- The node constraints form a set $\mathcal{V} \subseteq \left(\binom{\Sigma}{\Delta}\right)$.
- The edge constraints form a set $\mathcal{E} \subseteq \left(\binom{\Sigma}{2}\right)$.

For an unrooted tree $T = (V, E)$ with nodes having degree in $\{1, \Delta\}$, a solution to Π is a labeling of half-edges, that is, a map $\ell: V \times E \rightarrow \Sigma$ such that the following holds:

- For every node $v \in V$ with degree Δ that is incident to the edges e_1, \dots, e_Δ , we have $\{\ell(v, e_1), \dots, \ell(v, e_\Delta)\} \in \mathcal{V}$.
- For every edge $e = \{u, v\} \in E$, $\{\ell(u, e), \ell(v, e)\} \in \mathcal{E}$.

As before, if T is labeled with a solution, then it is correctly labeled. In light of these two types of requirements, it is also natural to speak of correctly labeled nodes and edges.

As in the preceding definition, leaves are unconstrained and may be labeled arbitrarily. As before, if such an LCL problem Π on regular unrooted trees is clear from the context, we refer to the elements of $\left(\binom{\Sigma}{\Delta}\right)$ as node configurations and to those of $\left(\binom{\Sigma}{2}\right)$ as edge configurations.

3.2 Models of Distributed Computing

Next we formally define the LOCAL model and its extensions.

► **Definition 3.4** (Deterministic LOCAL model). *The deterministic LOCAL model of distributed computing runs on a graph $G = (V, E)$ where each node $v \in V$ represents a computer and each edge a connection channel. Each node is labeled with a unique identifier. All nodes run the same distributed algorithm in parallel. Initially, a node is only aware of its own identifier and degree. Computation proceeds in synchronous rounds, and in each round a node can send and receive a message to and from each neighbor and update its state. Eventually each node must stop and announce its local output (its part of the solution, e.g. in graph coloring its own color). The running time, round complexity, or locality of the algorithm is the (worst-case) number of rounds $T(n)$ until the algorithm stops in any n -node graph.*

► **Definition 3.5** (Randomized LOCAL model). *The randomized LOCAL model is defined identically to the (deterministic) LOCAL model, with the addition that each node has access to a private, infinite stream of random bits. Additionally, a randomized LOCAL algorithm is required to succeed with high probability.*

► **Definition 3.6** (Deterministic online-LOCAL model [2]). *In the (deterministic) online-LOCAL model, an adversary chooses a sequence of nodes, $\sigma = (v_1, v_2, \dots, v_n)$, and reveals the nodes to the algorithm one at a time. The algorithm processes each node sequentially. Given an online-LOCAL algorithm with locality $T(n)$, when a node v_i is revealed, the algorithm must choose an output for v_i based on the subgraph induced by the radius- $T(n)$ neighborhoods of v_1, v_2, \dots, v_i . In other words, the algorithm retains global memory.*

► **Definition 3.7** (Randomized online-LOCAL model [1]). *In the randomized online-LOCAL model, an adversary first commits to a sequence of nodes, $\sigma = (v_1, v_2, \dots, v_n)$, and reveals the nodes to the algorithm one at a time, as in the online-LOCAL model. The algorithm runs on each v_i and retains global memory, as in the online-LOCAL model. Additionally, the algorithm has access to an infinite stream of random bits unbeknownst to the adversary; that is, the adversary cannot change the order they present the remaining nodes based on the intermediate output of the algorithm. As in the randomized LOCAL, the algorithm is required to succeed with high probability.*

4 Sub-logarithmic Gap for LCLs in Rooted Regular Trees

In this section, we show that there are no LCL problems on rooted trees with locality in the range $\omega(1) - o(\log n)$ in the randomized online-LOCAL model. Moreover, the problems that are solvable with locality $O(1)$ in randomized online-LOCAL are exactly the same problems that are solvable with locality $O(\log^* n)$ in the LOCAL model.

► **Theorem 2.1.** *Let Π be an LCL problem on rooted regular trees. If Π can be solved with $o(\log n)$ locality in randomized online-LOCAL, then it can be solved in LOCAL with $O(\log^* n)$ locality. Consequently, Π can be solved with $O(1)$ locality in SLOCAL, and thus also with the same locality in online-LOCAL, even deterministically.*

We show this by showing that a locality- $o(\log n)$ randomized online-LOCAL algorithm solving LCL problem Π implies the existence of a *coprime certificate for $O(\log^* n)$ solvability* (see Definition 4.6) that then implies that Π is solvable with locality $O(\log^* n)$ in the LOCAL model [4], and hence with locality $O(1)$ in the randomized online-LOCAL model.

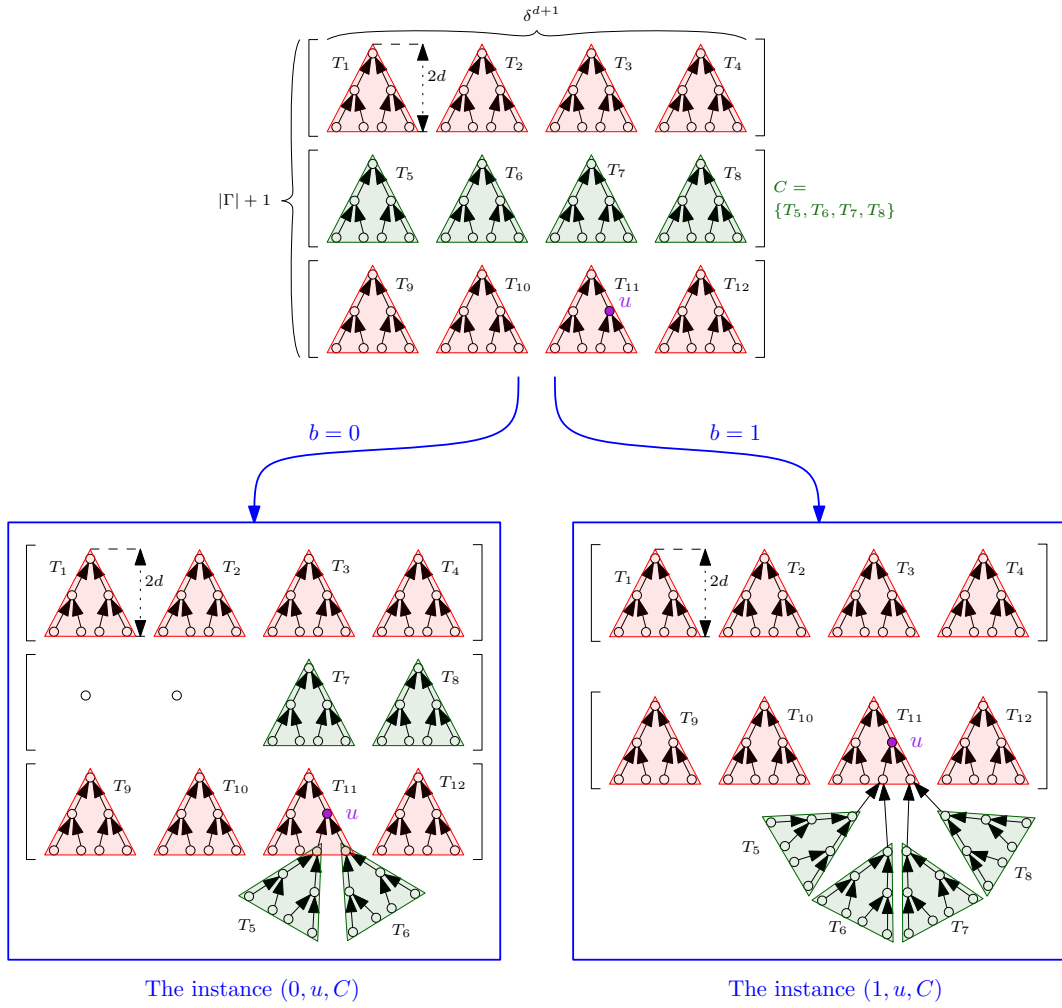
Throughout this section, we consider an LCL problem $\Pi = (\Delta, \Sigma, \mathcal{V})$ and a randomized online-LOCAL algorithm \mathcal{A} that solves Π with locality $T(n) = o(\log n)$ with high probability. We start by constructing a family of input instances and then argue that \mathcal{A} solving these instances produces a canonical labeling regardless of the randomness. Finally, we show how we this canonical labeling yields the coprime certificate for $O(\log^* n)$ solvability.

4.1 Construction of Input Instances

Let d be a depth parameter that we fix later, and let Δ be the number of children of internal nodes. We now construct the input instance as follows:

- **Definition 4.1** (Family of input instances). *To construct the family of input instances:*
1. *Construct $(|\Sigma|+1)$ chunks of trees, each containing Δ^{d+1} complete rooted trees of height $2d$. Give the trees in each chunk an ordering. Let M be the set of nodes in these trees at distance d from the root; we call these nodes middle nodes.*
 2. *Choose a bit b , and choose a node u which is at depth d in any of the trees created in the previous phase. Choose a chunk C such that node u is not contained in chunk C .*
 3. *The subtree rooted at u has Δ^d leaf descendants.*
 - If $b = 0$:** *Identify the roots of the first Δ^d trees of chunk C with the leaf descendants of u in a consistent order (see bottom-left of Figure 4).*
 - If $b = 1$:** *Make the roots of the first Δ^{d+1} trees of chunk C the children of the leaf descendants of u in a consistent order (see bottom-right Figure 4).*

Note that trees constructed in this way have $n = (|\Sigma|+1) \cdot \Delta^{d+1} \cdot \frac{\Delta^{2d+1}-1}{\Delta-1}$ nodes. Moreover, choosing (b, u, C) uniquely fixes the construction. Let \mathcal{P} be the set of choices for (b, u, C) .



■ **Figure 4** Visualization of two trees in the family of input instances. At the top we have $|\Sigma| + 1$ chunks of complete trees of depth $2d$, each containing Δ^{d+1} trees. The green trees in the middle row represent the chosen chunk C . On the bottom-left, we have $b = 0$, in which case we identify the roots of the first Δ^d trees of C with the leaf descendants of node u . On the bottom-right, we have $b = 1$, in which case we make all trees in chunk C the children of the leaf descendants of node u . See Definition 4.1 for full construction.

► **Observation 4.2.** *The number of instances is less than the number of nodes in each instance. In particular, $|\mathcal{P}| = 2|\Sigma|(|\Sigma| + 1)\Delta^{2d+1} < n$.*

We can now fix depth parameter d such that $d > \log_{\Delta}(2|\Sigma|)$ and $d > T(n)$. Recall that $T(n) = o(\log n)$, and hence such d exists.

4.2 Randomness of the Algorithm

Throughout this discussion, we let $\Pr_R(\cdot)$ denote the probability of an event when the randomness is over some source R for which the probability is being defined.

For each instance (b, u, C) , we fix the order middle nodes M such that it is consistent across all instances. We then reveal these middle nodes to \mathcal{A} in this order. Let $S \in \overline{\Sigma} = \Sigma^{\Delta^{2d+1}(|\Sigma|+1)}$ be the random sequence of labels generated by \mathcal{A} for the middle nodes M . Since the locality

of \mathcal{A} is $T(n) < d$, the algorithm does not get to see the roots or the leaves of these height- $2d$ trees. In particular, the algorithm does not get to know which instance (b, u, C) we have chosen. Hence S must be independent of the choice of (b, u, C) .

► **Observation 4.3.** *For every choice $\bar{\sigma} \in \bar{\Sigma}$ of labeling middle nodes M , and every instance $(b, u, C) \in \mathcal{P}$, we must have $\Pr_S(S = \bar{\sigma}) = \Pr_S(S = \bar{\sigma} \mid (b, u, C))$ where $\Pr_S(S = \bar{\sigma})$ denotes the probability that \mathcal{A} produces output $\bar{\sigma}$ for nodes M , and $\Pr_S(S = \bar{\sigma} \mid (b, u, C))$ denotes the same given that the input instance was (b, u, C) .*

Note that while the output S for nodes M is independent of (b, u, C) , the output that \mathcal{A} produces for the rest of the nodes of the input may be dependent of (b, u, C) . We denote this random variable by $\mathcal{A}^+(b, u, C)$.

We now analyze the failure probability of \mathcal{A} to show that there exists a fixed labeling $\bar{\sigma}^*$ for middle nodes M such that the labeling is still completable for each input instance (b, u, C) . Let \mathcal{F} denote the event that algorithm \mathcal{A} fails to solve the problem. By assumption, \mathcal{A} succeeds with high probability; hence $\Pr_{S, \mathcal{A}^+(b, u, C)}(\mathcal{F}) \leq \frac{1}{n}$ for all choices of $(b, u, C) \in \mathcal{P}$. We can now pick a labeling of the middle nodes that minimizes the average failure probability:

► **Definition 4.4.** *Let $\bar{\sigma}^*$ be defined as follows:*

$$\bar{\sigma}^* \in \arg \min_{\bar{\sigma} \in \bar{\Sigma}, \Pr_S(S = \bar{\sigma}) > 0} \sum_{(b, u, C) \in \mathcal{P}} \Pr_{\mathcal{A}^+(b, u, C)}(\mathcal{F} \mid (b, u, C), S = \bar{\sigma})$$

where ties are broken deterministically.

This definition ensures the following: Give that the algorithm labels nodes M with $\bar{\sigma}^*$, the total probability of failure over all instances (b, u, C) is minimized. In some sense, $\bar{\sigma}^*$ is the best labeling for M when the algorithm know nothing about the instance. Note that $\bar{\sigma}^*$ is a concrete element of $\bar{\Sigma}$ and hence does not depend on the randomness of the algorithm. We can formalize this idea:

► **Lemma 4.5.** *Regardless of the instance $(b, u, C) \in \mathcal{P}$, if the labeling S of nodes M is $\bar{\sigma}^*$, there exists a valid way to label the remaining nodes of the instance.*

Proof. We start by noting that $\frac{1}{n} \geq \Pr_{S, \mathcal{A}^+(b, u, C)}(\mathcal{F})$ since \mathcal{A} works correctly with high probability. We can now expand the probability to be conditional over the initial labeling S :

$$\Pr_{S, \mathcal{A}^+(b, u, C)}(\mathcal{F}) = \sum_{\bar{\sigma} \in \bar{\Sigma}} \Pr_{\mathcal{A}^+(b, u, C)}(\mathcal{F} \mid (b, u, C), S = \bar{\sigma}) \Pr_S(S = \bar{\sigma} \mid (b, u, C)).$$

Next, we apply Observation 4.3 to get

$$\frac{1}{n} \geq \sum_{\bar{\sigma} \in \bar{\Sigma}} \Pr_{\mathcal{A}^+(b, u, C)}(\mathcal{F} \mid (b, u, C), S = \bar{\sigma}) \Pr_S(S = \bar{\sigma}),$$

and then we sum over all choices of (b, u, C) on both sides to get

$$\frac{|\mathcal{P}|}{n} = \sum_{(b, u, C) \in \mathcal{P}} \frac{1}{n} \geq \sum_{(b, u, C) \in \mathcal{P}} \sum_{\bar{\sigma} \in \bar{\Sigma}} \Pr_{\mathcal{A}^+(b, u, C)}(\mathcal{F} \mid (b, u, C), S = \bar{\sigma}) \Pr_S(S = \bar{\sigma})$$

Exchanging the order of summation and noting that $\bar{\sigma}^*$ minimizes

$$\sum_{(b, u, C) \in \mathcal{P}} \Pr_{\mathcal{A}^+(b, u, C)}(\mathcal{F} \mid (b, u, C), S = \bar{\sigma}),$$

we get

$$\frac{|\mathcal{P}|}{n} \geq \sum_{(b,u,C) \in \mathcal{P}} \Pr_{\mathcal{A}^+(b,u,C)}(\mathcal{F} \mid (b,u,C), S = \bar{\sigma}^*).$$

Recalling that $\frac{|\mathcal{P}|}{n} < 1$ by Observation 4.2 and that all probabilities are non-negative numbers, we complete our calculation:

$$1 > \Pr_{\mathcal{A}^+(b,u,C)}(\mathcal{F} \mid (b,u,C), S = \bar{\sigma}^*) \quad \text{for each } (b,u,C) \in \mathcal{P}.$$

Hence for each instance (b,u,C) , the algorithm fails with probability strictly less than 1. Thus it succeeds with non-zero probability, and hence a correct labeling must exist. ◀

4.3 Getting a Coprime Certificate as a Subset of All Such Instances

We are now ready to extract from algorithm \mathcal{A} a coprime certificate for $O(\log^* n)$ solvability. The idea is to pick a subset of input instances \mathcal{P} that – along with the labeling $\bar{\sigma}^*$ – form the pairs of sequences of trees of the certificate. We defer the proof of why this implies the existence of a locality- $O(\log^* n)$ LOCAL algorithm to previous work [4].

► **Definition 4.6** (Coprime certificate for $O(\log^* n)$ solvability [4]). *Let $\Pi = (\Delta, \Sigma, C)$ be an LCL problem. A certificate for $O(\log^* n)$ solvability of Π consists of labels $\Sigma_{\mathcal{T}} = \{\sigma_1, \dots, \sigma_t\} \subseteq \Sigma$, a depth pair (d_1, d_2) and a pair of sequences \mathcal{T}^1 and \mathcal{T}^2 of t labeled trees such that*

1. *The depths d_1 and d_2 are coprime.*
2. *Each tree of \mathcal{T}^1 (resp. \mathcal{T}^2) is a complete Δ -ary tree of depth $d_1 \geq 1$ (resp. $d_2 \geq 1$).*
3. *Each tree is labeled by labels from Σ correctly according to problem Π .*
4. *Let $\bar{\mathcal{T}}_i^1$ (resp. $\bar{\mathcal{T}}_i^2$) be the tree obtained by starting from \mathcal{T}_i^1 (resp. \mathcal{T}_i^2) and removing the labels of all non-leaf nodes. It must hold that all trees $\bar{\mathcal{T}}_i^1$ (resp. $\bar{\mathcal{T}}_i^2$) are isomorphic, preserving the labeling. All the labels of the leaves of $\bar{\mathcal{T}}_i^1$ (resp. $\bar{\mathcal{T}}_i^2$) must be from set $\Sigma_{\mathcal{T}}$.*
5. *The root of tree \mathcal{T}_i^1 (resp. \mathcal{T}_i^2) is labeled with label σ_i .*

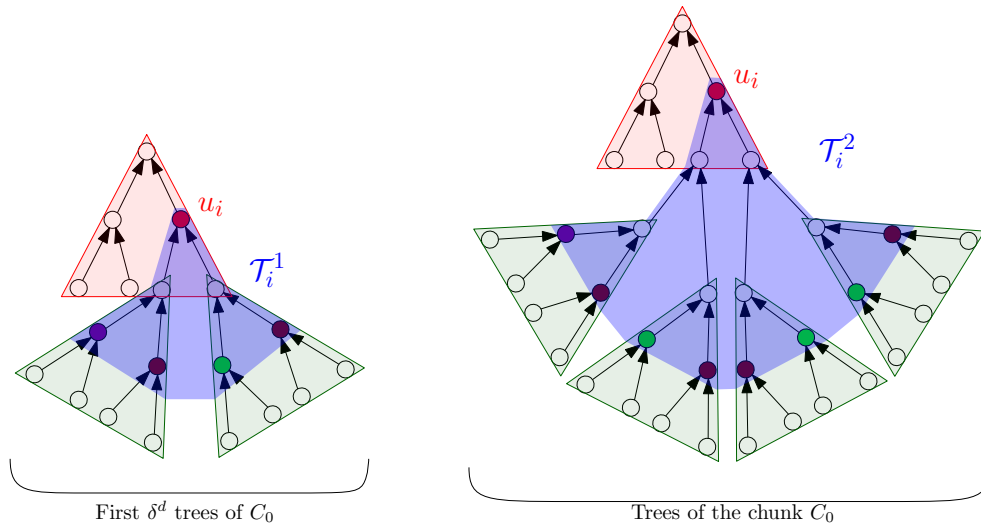
Let $\Sigma_{\mathcal{T}} = \{\sigma_1, \sigma_2, \dots, \sigma_t\} \subseteq \Sigma$ be the set of labels appearing in $\bar{\sigma}^*$, and let (u_1, u_2, \dots, u_t) be some middle nodes in the input instances such that node u_i has label σ_i according to $\bar{\sigma}^*$. Note that there are $|\Sigma| + 1$ chunks whereas $t = |\Sigma_{\mathcal{T}}| < |\Sigma| + 1$. Therefore, we get the following result by the pigeonhole principle:

► **Observation 4.7.** *There is a chunk C_0 which does not contain any node in $\{u_1, \dots, u_t\}$.*

With these definitions of labels $\Sigma_{\mathcal{T}} = (\sigma_1, \dots, \sigma_t)$, chunk C_0 , and nodes (u_1, \dots, u_t) , we are ready to prove our main result:

► **Theorem 4.8.** *For an LCL problem Π without inputs on rooted regular trees, if there is a randomized online-LOCAL algorithm \mathcal{A} with locality $T(n) = o(\log n)$, then there exists a coprime certificate of $O(\log^* n)$ solvability for Π , with $\Sigma_{\mathcal{T}}$ as the subset of labels.*

Proof. For each $i \in \{1, \dots, t\}$, consider the instance $(0, u_i, C_0)$ with $S = \bar{\sigma}^*$ and some valid way to label the remaining nodes; such a labeling exists due to Lemma 4.5. The subtree rooted at u_i contains the first Δ^d trees of the chunk C_0 . Let the depth- $2d$ subgraph rooted at u_i be tree \mathcal{T}_i^1 . See Figure 5a for a visualization.



(a) Trees \mathcal{T}_i^1 of depth $2d$ for certificate from instance $(0, u_i, C_0)$. **(b)** Trees \mathcal{T}_i^2 of depth $2d + 1$ for certificate from instance $(1, u_i, C_0)$.

■ **Figure 5** Construction of trees of coprime certificate for $O(\log^* n)$ solvability. The blue shaded region represents the tree of the sequence and extends from the labeled middle node u_i to the middle nodes of the trees hanging from its leaf descendants.

Note that for each $i \in \{1, \dots, t\}$, tree \mathcal{T}_i^1 has a root labeled with σ_i , and the leaves are labeled with labels from set $\Sigma_{\mathcal{T}}$ in an identical way. Hence \mathcal{T}^1 form the first sequence of the certificate. Similarly, we consider $(1, u_i, C_0)$ for all $i \in \{1, \dots, t\}$. This time we let the depth- $(2d + 1)$ subtree rooted at u_i be tree \mathcal{T}_i^2 . See Figure 5b for a visualization. The sequence \mathcal{T}^2 forms the second sequence of the certificate, again by similar arguments.

It is easy to check that $\Sigma_{\mathcal{T}}$, $2d$ and $2d + 1$ and sequences \mathcal{T}^1 and \mathcal{T}^2 indeed form a coprime certificate for $O(\log^* n)$ solvability for problem Π . ◀

We can now proof the main result of this section:

► **Theorem 2.1.** *Let Π be an LCL problem on rooted regular trees. If Π can be solved with $o(\log n)$ locality in randomized online-LOCAL, then it can be solved in LOCAL with $O(\log^* n)$ locality. Consequently, Π can be solved with $O(1)$ locality in SLOCAL, and thus also with the same locality in online-LOCAL, even deterministically.*

Proof. A locality- $o(\log n)$ randomized online-LOCAL algorithm for problem Π implies the existence of certificate of $O(\log^* n)$ solvability of Π by Theorem 4.8. This further implies existence of LOCAL algorithm solving Π with locality $O(\log^* n)$ [4], which in turn implies the existence of SLOCAL algorithm solving Π with locality $O(1)$ [15]. Since the randomized online-LOCAL is stronger than the SLOCAL model [1], this implies the existence of a locality- $O(1)$ randomized online-LOCAL algorithm that solves Π . Therefore, there can be no LCL problem on rooted regular trees where the optimal algorithm has a locality of $T(n)$ which is both $o(\log n)$ and $\omega(1)$. ◀

References

- 1 Amirreza Akbari, Xavier Coiteux-Roy, Francesco D'Amore, François Le Gall, Henrik Lievonen, Darya Melnyk, Augusto Modanese, Shreyas Pai, Marc-Olivier Renou, Václav Rozhon, and Jukka Suomela. Online locality meets distributed quantum computing, 2024. doi:10.48550/arXiv.2403.01903.
- 2 Amirreza Akbari, Navid Eslami, Henrik Lievonen, 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.
- 3 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.
- 4 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.
- 5 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.
- 6 Alkida Balliu, Mohsen Ghaffari, Fabian Kuhn, Augusto Modanese, Dennis Olivetti, Mikael Rabie, Jukka Suomela, and Jara Uitto. Shared randomness helps with local distributed problems, 2024. doi:10.48550/arXiv.2407.05445.
- 7 Alkida Balliu, Janne H. Korhonen, Fabian Kuhn, Henrik Lievonen, Dennis Olivetti, Shreyas Pai, Ami Paz, Joel Rybicki, Stefan Schmid, Jan Studený, Jukka Suomela, and Jara Uitto. Sinkless orientation made simple. In Telikeyalli Kavitha and Kurt Mehlhorn, editors, *2023 Symposium on Simplicity in Algorithms, SOSA 2023, Florence, Italy, January 23-25, 2023*, pages 175–191. SIAM, 2023. doi:10.1137/1.9781611977585.CH17.
- 8 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 Wichs 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.
- 9 Yi-Jun Chang. The complexity landscape of distributed locally checkable problems on trees. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPICs*, pages 18:1–18:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.DISC.2020.18.
- 10 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. *SIAM Journal on Computing*, 48(1):122–143, 2019. doi:10.1137/17M1117537.
- 11 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.
- 12 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 *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 1901–1910, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3618260.3649679.
- 13 Anubhav Dhar, Eli Kujawa, Henrik Lievonen, Augusto Modanese, Mikail Muftuoglu, Jan Studený, and Jukka Suomela. Local problems in trees across a wide range of distributed models, 2024. doi:10.48550/arXiv.2409.13795.

- 14 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.
- 15 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.
- 16 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.
- 17 Christoph Grunau, Václav Rozhon, and Sebastian Brandt. The landscape of distributed complexities on trees and beyond. In Alessia Milani and Philipp Woelfel, editors, *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 37–47. ACM, 2022. doi:10.1145/3519270.3538452.
- 18 Alexander E. Holroyd and Thomas M. Liggett. Finitely dependent coloring. *Forum of Mathematics, Pi*, 4, 2016. doi:10.1017/fmp.2016.7.
- 19 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.
- 20 Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991. doi:10.1137/0404036.
- 21 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.