Locally Checkable Labelings with Small Messages

Alkida Balliu
Universität Freiburg, Germany

Keren Censor-Hillel
Technion, Haifa, Israel

Yannic Maus
Technion, Haifa, Israel

Dennis Olivetti
Universität Freiburg, Germany

Jukka Suomela
Aalto University, Finland

Abstract

A rich line of work has been addressing the computational complexity of locally checkable labelings (LCLs), illustrating the landscape of possible complexities. In this paper, we study the landscape of LCL complexities under bandwidth restrictions. Our main results are twofold. First, we show that on trees, the CONGEST complexity of an LCL problem is asymptotically equal to its complexity in the LOCAL model. An analog statement for non-LCL problems is known to be false. Second, we show that for general graphs this equivalence does not hold, by providing an LCL problem for which we show that it can be solved in $O(\log n)$ rounds in the LOCAL model, but requires $\tilde{\Omega}(n^{1/2})$ rounds in the CONGEST model.

2012 ACM Subject Classification Networks → Network algorithms; Theory of computation → Distributed algorithms

Keywords and phrases distributed graph algorithms, CONGEST, locally checkable labelings

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.8

Related Version *Full Version*: https://arxiv.org/abs/2105.05574

Funding This project was partially supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 755839 (Keren Censor-Hillel, Yannic Maus).

1 Introduction

Two standard models of computing that have been already used for decades to study distributed graph algorithms are the LOCAL model and the CONGEST model [36]. In the LOCAL model, each node in the network can send *arbitrarily large messages* to each neighbor in each round, while in the CONGEST model the nodes can only send *small messages* (we will define the models in Section 2). In general, being able to send arbitrarily large messages can help a lot: there are graph problems that are trivial to solve in the LOCAL model and very challenging in the CONGEST model, and this also holds in trees.

Nevertheless, we show that there is a broad family of graph problems – *locally checkable labelings* or LCLs in short – in which the two models of computing have exactly the same expressive power *in trees* (up to constant factors): if a locally checkable labeling problem Π can be solved in trees in $T(n)$ communication rounds in the LOCAL model, it can be solved in $O(T(n))$ rounds also in the CONGEST model. We also show that this is no longer the case if we switch from trees to general graphs:

	LCL *problems (our work)*	*General problems (prior work)*
Trees:	CONGEST = LOCAL	CONGEST ≠ LOCAL
General graphs:	CONGEST ≠ LOCAL	CONGEST ≠ LOCAL

Locally Checkable Labelings. The study of the distributed computational complexity of locally checkable labelings (LCLs) in the LOCAL model was initiated by Naor and Stockmeyer [34] in the 1990s, but this line of research really took off only in the recent years [3–12, 14, 15, 17, 21–23, 35].

LCLs are a family of graph problems: an LCL problem Π is defined by listing a *finite set of valid labeled local neighborhoods.* This means that Π is defined on graphs of some finite maximum degree Δ, and the task is to label the vertices and/or edges with labels from some finite set so that the labeling satisfies some set of local constraints (see Section 2 for the precise definition).

A simple example of an LCL problem is the task of coloring a graph of maximum degree Δ with $\Delta+1$ colors (here valid local neighborhoods are all properly colored local neighborhoods). LCLs are a broad family of problems, and they contain many key problems studied in the field of distributed graph algorithms, including graph coloring, maximal independent set and maximal matching.

Classification of LCL problems. One of the key questions related to the LOCAL model has been this: given an arbitrary LCL problem Π, what can we say about its computational complexity in the LOCAL model (i.e., how many rounds are needed to solve Π)? It turns out that we can say quite a lot. There are infinitely many distinct complexity classes, but there are also some wide *gaps* between the classes – for example, if Π can be solved with a deterministic algorithm in $o(\log n)$ rounds, it can also be solved in $O(\log^* n)$ rounds [22].

Furthermore, some parts of the classification are *decidable*: for example, in the case of rooted trees, we can feed the description of Π to a computer program that can determine the complexity of Π in the LOCAL model [8]. The left part of Figure 1 gives a glimpse of what is known about the landscape of possible complexities of LCL problems in the LOCAL model.

However, this entire line of research has been largely confined to the LOCAL model. Little is known about the general structure of the landscape of LCL problems in the CONGEST model. In some simple settings (in particular, paths, cycles, and rooted trees) it is known that the complexity classes are the same between the two models [8, 24], but this has been a straightforward byproduct of work that has aimed at classifying the problems in the LOCAL model. What happens in the more general case has been wide open – the most interesting case for us is LCL problems in (unrooted) trees.

Prior work on LCLs in trees. In the case of trees, LCL problems are known to exhibit a broad variety of different complexities in the LOCAL model. For example, for every $k = 1, 2, 3, \ldots$ there exists an LCL problem whose complexity in the LOCAL model is exactly $\Theta(n^{1/k})$ [23]. There are also problems in which randomness helps exponentially: for example, the sinkless orientation problem belongs to the class of problems that requires $\Theta(\log n)$ rounds for deterministic algorithms and only $\Theta(\log \log n)$ rounds for randomized algorithms in the LOCAL model [17, 22]. Until very recently, one open question related to LCLs in trees remained: whether there are any problems in the region between $\omega(1)$ and $o(\log^* n)$; there is now a (currently unpublished) result [16] that shows that no such problems exist, and this completed the classification of LCLs in trees in the LOCAL model.

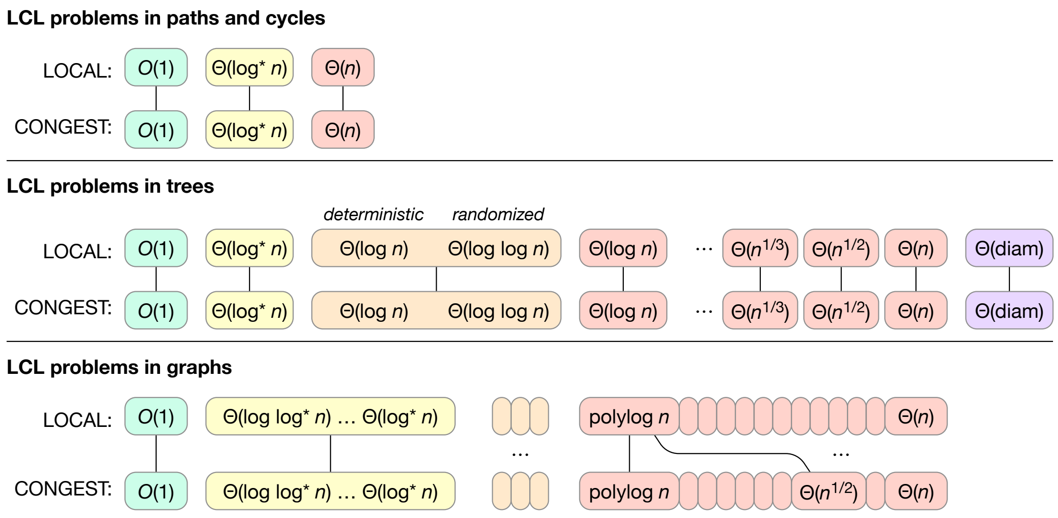

Figure 1 The landscape of LCL problems in the LOCAL and CONGEST models.

Prior work separating CONGEST and LOCAL. In the LOCAL model, all natural graph problems can be trivially solved in $O(n)$ rounds and also in $O(D)$ rounds, where D is the diameter of the input graph: in $O(D)$ rounds all nodes can gather the full information on the entire input graph.

However, there are many natural problems that do not admit $O(D)$-round algorithms in the CONGEST model. Some of the best-known examples include the task of finding an (approximate) minimum spanning tree, which requires $\tilde{\Omega}(\sqrt{n} + D)$ rounds [26, 37, 38], and the task of computing the diameter, which requires $\tilde{\Omega}(n)$ rounds [1, 30]. There are also natural problems that do not even admit $O(n)$-round algorithms in the CONGEST model. For example, finding an exact minimum vertex cover or dominating set requires $\tilde{\Omega}(n^2)$ rounds [2, 19].

Moreover, separations also hold in some cases where the LOCAL complexity is constant: One family of such problems is that of detecting subgraphs, for which an extreme example is that for any k there exists a subgraph of diameter 3 and size $O(k)$, which requires $\Omega(n^{2-1/k})$ rounds to detect in CONGEST, even when the network diameter is also 3 [29]. Another example is spanner approximations, for which there is a constant-round $O(n^\epsilon)$-approximation algorithm in the LOCAL model [13], but $\tilde{\Omega}(n^{1/2-\epsilon/2})$ rounds are needed in the CONGEST model (and even $\tilde{\Omega}(n^{1-\epsilon/2})$ rounds for deterministic algorithms) [18]. Separations hold also in trees: all-pairs shortest-paths on a star can be solved in 2 LOCAL rounds, but requires $\tilde{\Omega}(n)$ CONGEST rounds [19, 33].

While these lower bound results do not have direct implications in the context of LCL problems, they show that there are many natural settings in which the LOCAL model is much stronger than the CONGEST model.

1.1 Our Contributions

We show that

LOCAL = CONGEST for LCL problems in trees.

Not only do we have the same round complexity classes, but every LCL problem has the same asymptotic complexity in the two models. In particular, our result implies that *all* prior results related to LCLs in trees hold also in the CONGEST model. For example all decidability results for LCLs in trees in the LOCAL model hold also in the CONGEST model; this includes the decidable gaps in [23] and [21]. We also show that the equivalence holds not only if we study the complexity as a function of n, but also for problems with complexity $\Theta(D)$.

Given the above equality, one could conjecture that the LOCAL model does not have any advantage over the CONGEST model for any LCL problem. We show that this is not the case: as soon as we step outside the family of trees, we can construct an example of an LCL problem that is solvable in polylog n rounds in the LOCAL model but requires $\Omega(\sqrt{n})$ rounds in the CONGEST model. In summary, we show that

LOCAL $\neq$ CONGEST for LCL problems in general graphs.

Here "general graphs" refers to general *bounded-degree* graphs, as each LCL problem comes with some finite maximum degree Δ. We summarize our main results in Figure 1.

Open questions. The main open question after the present work is how wide the gap between CONGEST and LOCAL can be made in general graphs. More concretely, it is an open question whether there exists an LCL problem that is solvable in $O(\log n)$ rounds in the LOCAL model but requires $\Omega(n)$ rounds in the CONGEST model.

1.2 Road Map, Key Techniques, and New Ideas

We prove the equivalence of LOCAL and CONGEST in trees in Sections 3–5, and we show the separation between LOCAL and CONGEST in general graphs in Section 6. We start in Section 3 with some basic facts in the $O(\log n)$ regime that directly follow from prior work. The key new ideas are in Sections 4–6.

Equivalence in trees: superlogarithmic region (Section 4). The first major challenge is to prove that any LCL with some sufficiently high complexity $T(n)$ in the LOCAL model in trees has exactly the same asymptotic complexity $\Theta(T(n))$ also in the CONGEST model. A natural idea would be to show that a given LOCAL model algorithm A can be simulated efficiently in the CONGEST model. However, this is *not possible* in general – the proof has to rely somehow on the property that A solves an LCL problem.

Instead of a direct simulation approach, we use as a starting point prior *gap results* related to LCLs in the LOCAL model. A typical gap result in trees can be phrased as follows, for some $T_2 \ll T_1$:
- Given: a $T_1(n)$-round algorithm A_1 that solves some LCL Π in trees in the LOCAL model.
- We can construct: a $T_2(n)$-round algorithm A_2 that solves it in the LOCAL model.

We *amplify* the gap results so that we arrive at theorems of the following form – note that we not only speed up algorithms but also switch to a weaker model:
- Given: a $T_1(n)$-round algorithm A_1 that solves some LCL Π in trees in the **LOCAL** model.
- We can construct: a $T_2(n)$-round algorithm A_2' that solves it in the **CONGEST** model.

At first, the entire approach may seem hopeless: clearly A_2' has to somehow depend on A_1, but how could a fast CONGEST-model algorithm A_2' possibly make any use of a slow LOCAL-model algorithm A_1 as a black box? Any attempts of simulating (even a small number of rounds) of A_1 seem to be doomed, as A_1 might use large messages.

We build on the strategy that Chang and Pettie [23] and Chang [21] developed in the context of the LOCAL model. In their proofs, A_2 does not make direct use of A_1 as a black box, but mere *existence* of A_1 guarantees that Π is sufficiently well-behaved in the sense that long path-like structures can be labeled without long-distance coordination.

This observation served as a starting point in [21, 23] for the development of an efficient LOCAL model algorithm A_2 that finds a solution that is possibly different from the solution produced by A_1, but it nevertheless satisfies the constraints of problem Π. However, A_2 obtained with this strategy abuses the full power of large messages in the LOCAL model.

Hence while our aim is at understanding the landscape of computational complexity, we arrive at a *concrete algorithm design challenge*: we need to design a CONGEST-model algorithm A'_2 that solves essentially the same task as the LOCAL-model algorithm A_2 from prior work, with the same asymptotic round complexity. We present the new CONGEST-model algorithm A'_2 in Section 4 (more precisely, we develop a family of such algorithms, one for each gap in the complexity landscape).

Equivalence in trees: sublogarithmic region (Section 5). The preliminary observations in Section 3 cover the lowest parts of the complexity spectrum, and Section 4 covers the higher parts. To complete the proof of the equivalence of CONGEST and LOCAL in trees we still need to show the following result:

- Given: a randomized $o(\log n)$-round algorithm A_1 that solves some LCL Π in trees in the LOCAL model.
- We can construct: a randomized $O(\log \log n)$-round algorithm A'_2 that solves it in the CONGEST model.

If we only needed to construct a LOCAL-model algorithm, we could directly apply the strategy from prior work [20, 23]: replace A_1 with a faster algorithm A_0 that has got a higher failure probability, use the Lovász local lemma (LLL) to show that A_0 nevertheless succeeds at least for some assignment of random bits, and then plug in an efficient distributed LLL algorithm [20] to find such an assignment of random bits.

However, there is one key component missing if we try to do the same in the CONGEST model: a sufficiently fast LLL algorithm. Hence we again arrive at a concrete algorithm design challenge: we need to develop an efficient CONGEST-model algorithm that solves (at least) the specific LLL instances that capture the task of finding a good random bit assignments for A_0.

We present our new algorithm in Section 5. We make use of the shattering framework of [28], but one of the key new ideas is that we can use the equivalence results for the superlogarithmic region that we already proved in Section 4 as a tool to design fast CONGEST model algorithms also in the sublogarithmic region.

Separation in general graphs (Section 6). Our third major contribution is the separation result between CONGEST and LOCAL for general graphs – and as we are interested in proving such a separation for LCLs, we need to prove the separation for bounded-degree graphs.

Our separation result is constructive – we show how to design an LCL problem Π with the following properties:

(a) There is a deterministic algorithm that solves Π in the LOCAL model in $O(\log n)$ rounds.
(b) Any algorithm (deterministic or randomized) that solves Π in the CONGEST model requires $\Omega(\sqrt{n}/\log^2 n)$ rounds.

To define Π, we first construct a graph family $\mathcal{G}$ and an LCL problem Π^{real} such that Π^{real} would satisfy properties (a) and (b) *if we promised that the input comes from family* $\mathcal{G}$. Here we use a bounded-degree version of the lower-bound construction by [38]: the graph has a small diameter, making all problems easy in LOCAL, but all short paths from one end to the other pass through the top of the structure, making it hard to pass a large amount of information across the graph in the CONGEST model.

However, the existence of LCL problems that have a specific complexity given some arbitrary promise is not yet interesting – in particular, LCLs with an arbitrary promise do not lead to any meaningful complexity classes or useful structural theorems. Hence the key challenge is *eliminating the promise* related to the structure of the input graph. To do that, we introduce the following LCL problems:

- Π^{proof} is a distributed proof for the fact $G \in \mathcal{G}$. That is, for every $G \in \mathcal{G}$, there exists a feasible solution X to Π^{proof}, and for every $G \notin \mathcal{G}$, there is no solution to Π^{proof}. This problem can be hard to solve.
- Π^{bad} is a distributed proof for the fact that a given labeling X is *not* a valid solution to Π^{proof}. This problem has to be sufficiently easy to solve in the LOCAL model whenever X is an invalid solution (and impossible to solve whenever X is a valid solution).

Finally, the LCL problem Π captures the following task:

- Given a graph G and a labeling X, solve either Π^{real} or Π^{bad}.

Now given an arbitrary graph G (that may or may not be from $\mathcal{G}$) and an arbitrary labeling X (that may or may not be a solution to Π^{proof}), we can solve Π efficiently in the LOCAL model as follows:

- If X is not a valid solution to Π^{proof}, we will detect it and we can solve Π^{bad}.
- Otherwise X proves that we must have $G \in \mathcal{G}$, and hence we can solve Π^{real}.

Particular care is needed to ensure to that even for an adversarial G and X, at least one of Π^{real} and Π^{bad} is always sufficiently easy to solve in the LOCAL model. A similar high-level strategy has been used in prior work to e.g. construct LCL problems with a particular complexity in the LOCAL model, but to our knowledge the specific constructions of $\mathcal{G}$, Π^{real}, Π^{proof}, and Π^{bad} are all new – we give the details of the construction in Section 6.

2 Preliminaries & Definitions

Formal LCL definition. An LCL problem Π is a tuple $(\Sigma_{\text{in}}, \Sigma_{\text{out}}, C, r)$ satisfying the following.

- Both Σ_{in} and Σ_{out} are constant-size sets of labels;
- The parameter r is an arbitrary constant, called *checkability radius* of Π;
- C is a finite set of pairs $(H = (V^H, E^H), v)$, where:
  - H is a graph, v is a node of H, and the radius of v in H is at most r;
  - Every pair $(v, e) \in V^H \times E^H$ is labeled with a label in Σ_{in} and a label in Σ_{out}.

Solving a problem Π means that we are given a graph where every node-edge pair is labeled with a label in Σ_{in}, and we need to assign a label in Σ_{out} to each node-edge pair, such that every r-radius ball around each node is isomorphic to a (labeled) graph contained in C. We may use the term *half-edge* to refer to a node-edge pair.

LOCAL model. In the LOCAL model [31], we have a connected input graph G with n nodes, which communicate unbounded-size messages in synchronous rounds according to the links that connect them, initially known only to their endpoints. A trivial and well-known observation is that a T-round algorithm can be seen as a mapping from radius-T neighborhoods to local outputs.

CONGEST model. The CONGEST model is in all other aspects identical to the LOCAL model, but we limit the message size: in a network with n nodes, the size of each message is limited to at most $O(\log n)$ bits [36].

Randomized algorithms. We start by formally defining what is a randomized algorithm in our context. We consider randomized Monte Carlo algorithms, that is, the bound on their running time holds deterministically, but they are only required to produce a valid solution with high probability of success.

▶ **Definition 1** (Randomized Algorithm). *A* randomized algorithm *$\mathcal{A}$ run with parameter n, written as $\mathcal{A}(n)$, (known to all nodes) has runtime $t_{\mathcal{A}}(n)$ and is correct with probability $1/n$ on any graph with at most n nodes. There are no unique IDs. Further, we assume that there is a finite upper bound $h_{\mathcal{A}}(n) \in \mathbb{N}$ on the number of random bits that a node uses on a graph with at most n vertices.*

The local failure probability *of a randomized algorithm $\mathcal{A}$ at a node v when solving an* LCL *is the probability that the* LCL *constraint of v is violated when executing $\mathcal{A}$.*

The assumption that the number of random bits used by a node is bounded by some (arbitrarily fast growing) function $h(n)$ is made in other gap results in the LOCAL model as well (see e.g. "the necessity of graph shattering" in [22]). Our results do not care about the growth rate of $h(n)$, e.g., it could be doubly exponential in n or even growing faster. Its growth rate only increases the leading constant in our runtime.

The assumption that randomized algorithms are not provided with unique IDs is made to keep our proofs simpler, but it is not a restriction. In fact, any randomized algorithm can, in 0 rounds, generate an ID assignment, where IDs are unique with high probability. Hence, any algorithm that requires unique IDs can be converted into an algorithm that does not require them, by first generating them and then executing the original algorithm. The ID generation phase may fail, and the algorithm may not even be able to detect it and try to recover from it, but this failure probability can be made arbitrarily small, by making the ID space large enough. Hence, we observe the following.

▶ **Observation 2.** *Let $c \geq 1$ be a constant. For any randomized Monte Carlo algorithm with failure probability at most $1/n^c$ on any graph with at most n nodes that relies on IDs from an ID space of size n^{c+2} there is a randomized Monte Carlo algorithm with failure probability $2/n^c$ that does not use unique IDs.*

Deterministic algorithms. Differently from randomized algorithms, deterministic ones are not allowed to use randomness, and nodes are instead equipped with unique identifiers.

▶ **Definition 3** (Deterministic Algorithm). *A* deterministic algorithm *$\mathcal{A}$ run with parameter n, written as $\mathcal{A}(n)$, (known to all nodes) has runtime $t_{\mathcal{A}}(n)$ and is always correct on any graph with at most n nodes. We assume that vertices are equipped with unique IDs from a space of size $\mathcal{S}$. We require that a single ID can be sent in a* CONGEST *message. If the parameter $\mathcal{S}$ is omitted, then it is assumed to be n^c, for some constant $c \geq 1$.*

3 Warm-Up: The $O(\log n)$ Region

As a warm-up, we consider the regime of sublogarithmic complexities. In this region, we can use simple observations to show that gaps known for LCL complexities in the LOCAL model directly extend to the CONGEST model as well. Note that the results in this sections are immediate corollaries of previous work. In the following, we assume that the size of the ID space $\mathcal{S}$ is polynomial in n.

We start by noticing that a constant time (possibly randomized) algorithm for the LOCAL model implies a constant time deterministic algorithm for the CONGEST model as well.

▶ **Theorem 4** (follows from [23, 34])**.** *Let Π be an* LCL *problem. Assume that there is a randomized algorithm for the* LOCAL *model that solves Π in $O(1)$ rounds with failure probability at most $1/n$. Then, there is a deterministic algorithm for the* CONGEST *model that solves Π in $O(1)$ rounds.*

Proof. It is known that, the existence of a randomized $O(1)$-round algorithm solving Π in the LOCAL model with failure probability at most $1/n$ implies the existence of a deterministic algorithm solving Π in the LOCAL model in $O(1)$ rounds [23, 34].

Also, it is known that any algorithm $\mathcal{A}$ running in T rounds in the LOCAL model can be normalized, obtaining a new algorithm $\mathcal{A}'$ that works as follows: first gather the T-radius ball neighborhood, and then, without additional communication, produce an output. Since $\Delta = O(1)$, algorithm $\mathcal{A}'$ can be simulated in the CONGEST model. ◀

We can use Theorem 4 to show that, if we have algorithms that lie inside know complexity gaps of the LOCAL model, we can obtain fast algorithms that work in the CONGEST model as well.

▶ **Corollary 5.** *Let Π be an* LCL *problem. Assume that there is a randomized algorithm for the* LOCAL *model that solves Π in $o(\log\log^* n)$ rounds with failure probability at most $1/n$. Then, there is a deterministic algorithm for the* CONGEST *model that solves Π in $O(1)$ rounds.*

Proof. In the LOCAL model, it is known that an $o(\log\log^* n)$-round randomized algorithm implies an $O(1)$-round deterministic algorithm [23, 34]. Then, by applying Theorem 4 the claim follows. ◀

While Corollary 5 holds for any graph topology, in trees, paths and cycles we obtain a better result.

▶ **Corollary 6.** *Let Π be an* LCL *problem on trees, paths, or cycles. Assume that there is a randomized algorithm for the* LOCAL *model that solves Π in $o(\log^* n)$ rounds with failure probability at most $1/n$. Then, there is a deterministic algorithm for the* CONGEST *model that solves Π in $O(1)$ rounds.*

Proof. In the LOCAL model, it is known that, on trees, paths, or cycles, an $o(\log^* n)$-round randomized algorithm implies an $O(1)$-round deterministic algorithm [16, 34]. Then, by applying Theorem 4 the claim follows. ◀

We now show that similar results hold even in the case where the obtained algorithm does not run in constant time.

▶ **Theorem 7** (follows from [22, 32])**.** *Let Π be an* LCL *problem. Assume that there is a deterministic algorithm for the* LOCAL *model that solves Π in $o(\log n)$ rounds, or a randomized algorithm for the* LOCAL *model that solves Π in $o(\log\log n)$ rounds with failure probability at most $1/n$. Then, there is a deterministic algorithm for the* CONGEST *model that solves Π in $O(\log^* n)$ rounds.*

Proof. It is known that for solving LCLs in the LOCAL model, any deterministic $o(\log n)$-round algorithm or randomized $o(\log\log n)$-round algorithm can be converted into a deterministic $O(\log^* n)$-round algorithm [22]. We exploit the fact that the speedup result of [22] produces algorithms that are structured in a normal form. In particular, all problems solvable in $O(\log^* n)$ can also be solved as follows:

1. Find a distance-k $O(\Delta^{2k})$-coloring, for some constant k.
2. Run a deterministic k rounds algorithm.

The first step can be implemented in the CONGEST model by using e.g. Linial's coloring algorithm [32]. Then, similarly as discussed in the proof of Theorem 4, any T rounds algorithm can be normalized into an algorithm that first gathers a T-radius neighborhood and then produces an output without additional communication. Hence, also the second step can be implemented in the CONGEST model in $O(k) = O(1)$ rounds. ◀

We use this result to show stronger results for paths and cycles.

▶ **Corollary 8.** *Let Π be an* LCL *problem on paths or cycles. Assume that there is a randomized algorithm for the* LOCAL *model that solves Π in $o(n)$ rounds with failure probability at most $1/n$. Then, there is a deterministic algorithm for the* CONGEST *model that solves Π in $O(\log^* n)$ rounds.*

Proof. In the LOCAL model, it is known that, on paths and cycles, an $o(n)$-round randomized algorithm implies an $O(\log^* n)$-round deterministic algorithm [22]. The claim follows by applying Theorem 7. ◀

4 Trees: The $\Omega(\log n)$ Region

In this section we prove that in the regime of complexities that are at least logarithmic, the asymptotic complexity to solve any LCL on trees is the same in the LOCAL and in the CONGEST model, when expressed as a function of n. Combined with the results proved in Section 3 and Section 5, which hold in the sublogarithmic region, we obtain that the asymptotic complexity of any LCL on trees is identical in LOCAL and CONGEST.

Polynomial and subpolynomial gaps on trees. Informally, the following theorem states that there are no LCLs on trees with complexity between $\omega(\log n)$ and $n^{o(1)}$, and for any constant integer $k \geq 1$, between $\omega(n^{1/(k+1)})$ and $o(n^{1/k})$, in both the CONGEST and LOCAL models, and that the complexity of any problem is the same in both models.

▶ **Theorem 9** (superlogarithmic gaps). *Let $T^{\text{slow}} = \{n^{o(1)}\} \cup \{o(n^{1/k}) \mid k \in \mathbb{N}^+\}$. For $k \geq 1$, let $f(o(n^{1/k})) := O(n^{1/(k+1)})$. Also, let $f(n^{o(1)}) := O(\log n)$.*

Let $T \in T^{\text{slow}}$. Let Π be any LCL *problem on trees that can be solved with a T-round randomized* LOCAL *algorithm that succeeds with probability at least $1 - 1/n$ on graphs of at most n nodes. The problem Π can be solved with a deterministic $f(T)$-round* CONGEST *algorithm.*

Given the description of Π it is decidable whether there is an $f(T)$-round deterministic CONGEST *algorithm, and if it is the case then it can be obtained from the description of Π.*

▶ Remark 10. The deterministic complexity in Theorem 9 suppresses an $O(\log^* |\mathcal{S}|)$ dependency on the size $|\mathcal{S}|$ of the ID space. Also, there is an absolute constant $l_\Pi = O(1)$ such that the CONGEST algorithm works even if, instead of unique IDs, a distance-l_Π input coloring from a color space of size $|\mathcal{S}|$ is provided.

Note that gap theorems do not hold if one has promises on the input of the LCL. For example, consider a path, where some nodes are marked and others are unmarked, and the problem requires to 2-color unmarked nodes. If we have the promise that there is at least one marked node every $\sqrt{n}$ steps, then we obtain a problem with complexity $\Theta(\sqrt{n})$, that does not exist on paths for LCLs without promises on inputs.

Since Theorem 9 shows how to construct CONGEST algorithms starting from LOCAL ones, the existence of CONGEST problems with complexity $\Theta(n^{1/k})$ follows implicitly from the existence of these complexities in the LOCAL model. In particular, Chang and Pettie devised a series of problems that they name $2\frac{1}{2}$-coloring [23]. These problems are parameterized by an integer constant $k > 1$ and have complexity $\Theta(n^{1/k})$ on trees.

Diameter time algorithms. Additionally, we prove that a randomized diameter time LOCAL algorithm is asymptotically not more powerful than a deterministic diameter time CONGEST algorithm, when solving LCLs on trees. This result can be seen as an orthogonal result to the remaining results that we prove for LCLs on trees, because the runtime is not expressed as a function of n, but as a function of a different parameter, that is, the diameter of the graph. While the result might be of independent interest, it mainly deals as a warm-up to explain the proof of the technically more involved Theorem 9.

▶ **Theorem 11** (diameter algorithms). *Let Π be an LCL problem on trees that can be solved with a randomized LOCAL algorithm running in $O(D)$ rounds that succeeds with high probability, where D is the diameter of the tree. The problem Π can be solved with a deterministic CONGEST algorithm running in $O(D)$ rounds. The CONGEST algorithm does not require unique IDs but a means to break symmetry between adjacent nodes, that can be given by unique IDs, an arbitrary input coloring, or an arbitrary orientation of the edges.*

Any solvable LCL problem on trees can trivially be solved in the LOCAL model in $O(D)$ rounds by gathering the whole tree topology at a leader node, who then locally computes a solution and distributes it to all nodes. Using pipelining, the same algorithm can be simulated in the CONGEST model in $O(D + n) = O(n)$ rounds, but this running time can still be much larger than the $O(D)$ running time obtainable in the LOCAL model. On a high level, we show that for LCLs on trees, it is not required to gather the whole topology at a single node and brute force a solution – gathering the whole information at a single node has an $\Omega(n)$ lower bound even if the diameter is small.

Black-white formalism. In order to keep our proofs simple, we consider a simplified variant of LCLs, called *LCLs in the black-white formalism.* The main purpose of this formalism is to reduce the radius required to verify if a solution is correct. We will later show that, on trees, the black-white formalism is in some sense equivalent to the standard LCL definition.

A problem Π is a tuple $(\Sigma_{\text{in}}, \Sigma_{\text{out}}, C_W, C_B)$ satisfying the following.

- Both Σ_{in} and Σ_{out} are constant-size sets of labels;
- C_B and C_W are sets of multisets of pairs of labels, where each pair (i, o) satisfies $i \in \Sigma_{\text{in}}$ and $o \in \Sigma_{\text{out}}$.

Solving a problem Π means that we are given a bipartite two-colored graph where every edge is labeled with a label in Σ_{in}, and we need to assign a label in Σ_{out} to each edge, such that for every black (resp. white) node, the multiset of pairs of input and output labels assigned to the incident edges is in C_B (resp. C_W).

Node-edge formalism. The black-white formalism allows us to define problems also on graphs that are not bipartite and two-colored, as follows. Given a graph G, we define a bipartite graph H, where white nodes correspond to nodes of G, and black nodes correspond to edges of G. A labeling of edges of H corresponds to a labeling of node-edge pairs of G. The constraints C_W of white nodes of H correspond to node constraints of G, and the constraints C_B of black nodes of H corresponds to edge constraints of G.

Equivalence on trees. Clearly, any problem that can be defined with the node-edge formalism can be also expressed as a standard LCL. Claim 12 states that the node-edge formalism and the standard LCL formalism are in some sense equivalent, if we restrict to trees.

▷ Claim 12. For any LCL problem Π with checkability radius r we can define an *equivalent* node-edge checkable problem Π'. In other words, given a solution for Π', we can find, in $O(r)$ rounds, a solution for Π, and vice versa.

By Claim 12, on trees, all LCLs can be converted into an equivalent node-edge checkable LCL, and note that the node-edge formalism is a special case of the black-white formalism where black nodes have degree 2. To make our proofs easier to read, in the rest of the section we prove our results in the black-white formalism (where the degree of black nodes is 2), but via Claim 12 all results also hold for the standard definition of LCLs. We start by proving Theorem 11.

Proof of Theorem 11. Recall that in the black and white formalism nodes are properly 2-colored, input and output labels are on edges (that is, there is only one label for each edge, and not one for each node-edge pair), and the correctness of a solution can be checked independently by black and white nodes by just inspecting the labeling of their incident edges. Assume we are given a tree. We apply the following algorithm, that is split into 3 phases.

1. *Rooting the tree.* By iteratively "removing" nodes with degree one from the tree, nodes can produce an orientation that roots the tree in $O(D)$ CONGEST rounds. This operation can be performed even if, instead of IDs, nodes are provided with an arbitrary edge orientation. In the same number of rounds, nodes can know their distance from the (computed) root in the tree. We say that nodes with the same distance to the root are in the same *layer*, where leaves are in layer 1, and the root is in layer $L = O(D)$.
2. *Propagate label-sets up.* We process nodes layer by layer, starting from layer 1, that is, from the leaves. Each leaf u of the tree tells its parent v which labels for the edge $\{u, v\}$ would make them happy. The set of these labels is what we call a *label-set.* A leaf u is *happy* with a label if the label satisfies u's LCL constraints in Π. Then, in round i, each node v in layer i receives from all children the sets of labels that make them happy, and tells to its parent w which labels for the edge $\{v, w\}$ would make it happy, that is, it also sends a *label-set.* Here v is *happy* with a label for the edge $\{v, w\}$ if it can label all the edges $\{\{v, u\} \mid u \text{ is a child of } v\}$ to its children such that all of its children are happy. In other words, it must hold that for any element in the label-set sent by v to w, there must exist a choice in the sets previously sent by the children of v to v, such that the constraints of Π are satisfied at v. In $O(D)$ rounds, this propagation of sets of label-sets reaches the root of the tree.
3. *Propagate final labels down.* We will later prove that, if Π is solvable, then the root can pick labels that satisfy its own LCL constraints and makes all of its children happy. Then, layer by layer, in $O(D)$ iterations, the vertices pick labels for all of their edges to their children such that their own LCL constraints are satisfied and all their children are happy. More formally, from phase 2 we know that, for any choice made by the parent, there always exists a choice in the label-sets previously sent by the children, such that the LCL constraints are satisfied on the node.

This algorithm can be implemented in $O(D)$ rounds in CONGEST as the label-sets that are propagated in the second phase are subsets of the constant size alphabet Σ. In the third phase, each vertex only has to send one final label per outgoing edge to its children.

Let $L_{v,u}$ be the label-set received by node v from its child u. We now prove that, if Π is solvable, then there is a choice of labels, in the label-sets received by the root, that makes the root happy. Since Π is solvable, then there exists an assignment ϕ of labels to the edges of the tree such that the constraints of Π are satisfied on all nodes. We prove, by induction on the layer number j, that every edge $\{u,v\}$ such that u is in layer j and v is the parent of u, satisfies that $\phi(\{u,v\}) \in L_{v,u}$. For $j = 1$ the claim trivially holds, since leaves send to their parent the set of all labels that make them happy. For $j > 1$, consider some node v in layer j. Let $u_1, \dots, u_d$ be its children. By the induction hypothesis it holds that $\phi(\{v,u_i\}) \in L_{v,u_i}$. Let p be the parent of v (if it exists). Since ϕ is a valid labeling, and since it is true that if the parent labels the edge $\{v,p\}$ with label $\phi(\{v,p\})$ then v can pick something from its sets L_{v,u_i} and be happy, then the algorithm puts $\phi(\{v,p\})$ into the set $L_{p,v}$. Hence, every set L_{r,u_i} received by the root r contains the label $\phi(\{r,u_i\})$, implying that there is a valid choice for the root. ◀

While this process is extremely simple, its runtime of $O(D)$ rounds is rather slow. In order to obtain the $O(n^{1/k})$ and $O(\log n)$ CONGEST algorithms required for proving Theorem 9, we need a more sophisticated approach. As done by prior work in [21, 23], we decompose the tree into *fewer* layers, and show that, the mere existence of a fast algorithm for the problem implies that, similar to the algorithm of Theorem 11, it is sufficient to propagate constant sized label-sets between the layers; thus we obtain a complexity that only depends on the number of layers and their diameter. We proceed as follows.

5 Trees: $o(\log n)$ Randomized Implies $O(\log\log n)$

In this section we show that, on trees, any randomized algorithm solving an LCL problem Π in $o(\log n)$ rounds can be transformed into a randomized algorithm that solves Π in $O(\log\log n)$ rounds. This implies that in the CONGEST model there is no LCL problem in trees with a randomized complexity that lies between $\omega(\log\log n)$ and $o(\log n)$. Moreover, we show that it is not necessary to start from an algorithm for the CONGEST model, but that a LOCAL model one is sufficient. More formally, we will prove the following theorem.

▶ **Theorem 13** (sublogarithmic gap). *Let $c \geq 1$ be a constant. Given any* LCL *problem Π, if there exists a randomized algorithm for the* LOCAL *model that solves Π on trees in $o(\log n)$ rounds with failure probability at most $1/n$, then there exists a randomized algorithm for the* CONGEST *model that solves Π on trees in $O(\log\log n)$ rounds with failure probability at most $1/n^c$.*

In Section 5 we present the high level idea of the proof of Theorem 13. The proof itself is split into three subsections, for which a road map appears at the end of Section 5.

Proof Idea for Theorem 13

In the LOCAL model it is known that, on trees, there are no LCL problems with randomized complexity between $\omega(\log\log n)$ and $o(\log n)$ [20, 23]. At a high level, we follow a similar approach in our proof. However, while some parts of the proof directly work in the CONGEST model, there are some challenges that need to be tackled in order to obtain an algorithm that runs in $O(\log\log n)$ that is actually bandwidth efficient. We now provide the high level idea of our approach.

The standard approach: expressing the problem as an LLL instance. As in the LOCAL model case, the high level idea is to prove that if a randomized algorithm for an LCL problem Π runs in $o(\log n)$ rounds, then we can make it run faster at the cost of increasing its failure probability. In this way, we can obtain a constant time algorithm $\mathcal{A}_0$ at the cost of a very large failure probability. This partially gives what we want: we need a fast algorithm with small failure probability, and now we have a very fast algorithm with large failure probability. One way to fix the failure probability issue is to derandomize the algorithm $\mathcal{A}_0$, that is, to find a random bit assignment satisfying that if we run the algorithm with this specific assignment of random bits then the algorithm does not fail. Ironically, we use a randomized algorithm to find such a random bit assignment.

▶ **Lemma 14** (informal version)**.** *For any problem Π solvable in $o(\log n)$ rounds with a randomized* LOCAL *algorithm having failure probability at most $1/n$, there exists a constant time* LOCAL *algorithm $\mathcal{A}_0$ that solves Π with constant local failure probability p.*

It turns out that the problem itself of finding a good assignment of random bits such that the constant time algorithm $\mathcal{A}_0$ does not fail can be formulated as a Lovász Local Lemma (LLL) instance. In an LLL instance there are random variables and a set of *bad events* that depend on these variables. The famous Lovász Local Lemma [27] states that if the probability of each bad event is upper bounded by p, each bad event only shares variables with d other events and the *LLL criterion $epd < 1$* holds, then there exists an assignment to the variables that avoids all bad events (a more formal treatment of the Lovász Local Lemma is contained in the full version). In our setting, the random variables are given by the random bits used by the vertices and each vertex v has a bad event $\mathcal{E}_v$ that holds if the random bits are such that v's constraints in Π are violated if $\mathcal{A}_0$ is executed with these random bits. We show that a large polynomial LLL criterion – think of $p(ed)^{30} < 1$ – holds. Thus, the Lovász Local Lemma implies that there exist *good random bits* such the LCL constraints of Π are satisfied for all nodes when using these bits in $\mathcal{A}_0$. In the LOCAL model it is known how to solve an LLL problem with such a strong LLL criterion efficiently. We show that the same holds in the CONGEST model, i.e., $O(\log\log n)$ CONGEST rounds are sufficient to find a good assignment of random bits. We point out that we do not give a general LLL algorithm in the CONGEST model but an algorithm that is tailored for the specific instances that we obtain. The constant time algorithm $\mathcal{A}_0$ executed with these random bits does not fail at any node.

We summarize the high level approach as follows: Given an LCL problem Π defined on trees and an $o(\log n)$-rounds randomized algorithm $\mathcal{A}$ for Π, we obtain a constant time algorithm $\mathcal{A}_0$ for Π and a new problem Π' of finding good random bits for $\mathcal{A}_0$. Problem Π' is defined on the same graph as problem Π. The algorithm $\mathcal{A}_0$ and the problem Π' only depend on Π and $\mathcal{A}$. We show that Π' is also an LCL problem. In problem Π' each node of the tree needs to output a bit string such that if the constant time algorithm $\mathcal{A}_0$ is run with the computed random bits, the problem Π is solved. We will show that Π' can be solved in $O(\log\log n)$ rounds, and note that once Π' is solved, one can run $\mathcal{A}(n_0)$ for $t_0 = t_{\mathcal{A}_0} = O(1)$ rounds to solve Π.

▶ **Lemma 15** (informal version)**.** *The problem Π', that is, the problem of finding a good assignment of random bits that allows us to solve Π in constant time, can be solved in $O(\log\log n)$ rounds in the* CONGEST *model.*

Problem Π' is defined on the same tree as Π but problem Π' has checking radius $r + t_0$. Thus, the *dependency graph* of the LLL instance is a power graph of the tree, or in other words the LLL instance is *tree structured*. Hence, in the LOCAL model, Π' can be solved in $O(\log\log n)$ rounds by using a $O(\log\log n)$ randomized LOCAL algorithm for tree structured LLL instances [20]. We cannot do the same here, as it is not immediate whether this algorithm works in the CONGEST model (it is only clear that its shattering phase works in CONGEST).

The shattering framework. Our main contribution is showing how to solve Π' in $O(\log\log n)$ rounds in a bandwidth-efficient manner. To design an $O(\log\log n)$-round algorithm for Π', we apply the shattering framework for LLL of [28], that works as follows. After a precomputation phase of $O(\log^* n)$ rounds, the shattering process uses $\operatorname{poly}\Delta = O(1)$ rounds (the exponent depends on t_0 and the checking radius of the LCL) to determine the random bits of some of the nodes. The crucial property is that all nodes with unset random bits form *small connected components* of size $N = \operatorname{poly}(\Delta)\cdot\log n = O(\log n)$; in fact, even all nodes that are close to nodes with unset random bits form small connected components $C_1,\dots,C_k$. Furthermore, each connected component can be solved (independently) with an LLL procedure as well, with a slightly tighter polynomial criterion (e.g., $p(ed)^{15} < 1$). Note that, in order to solve these smaller instances, we need to use a deterministic algorithm.This is because, if we try to recursively apply a randomized LLL algorithm on the smaller instances (e.g. by using the randomized LOCAL algorithm of [25]) we get that each component can be solved independently in $O(\log N)$ rounds, but with failure probability $1/\operatorname{poly} N \gg 1/n$.

Our main contribution: solving the small instances in a bandwidth efficient manner. Since it seems that we cannot directly use an LLL algorithm to solve the small remaining instances $C_1,\dots,C_k$, we follow a different route. We devise a deterministic CONGEST algorithm that we can apply on each of the components in parallel: In Theorem 9 we prove that, on trees, any randomized algorithm running in $n^{o(1)}$ rounds (subpolynomial in the number of nodes) in the LOCAL model can be converted into a deterministic algorithm running in $O(\log n)$ in the CONGEST model. We use this result here, to show that, the mere existence of the randomized LOCAL algorithm of [25], that fails with probability at most $1/\operatorname{poly} N$ and runs in $O(\log N)$ rounds in the LOCAL model, which fits the runtime requirement of Theorem 9, implies that Π' can be solved in $O(\log N) = O(\log\log n)$ CONGEST rounds deterministically on the components induced by unset bits. To apply Theorem 9 that only holds for LCL problems, we express the problem of completing the partial random bit assignment as a problem Π'', that intuitively is almost the same problem as Π', but allows some nodes to already receive bit strings as their input. We show that Π'' is a proper LCL.

Formally, there are several technicalities that we need to take care of. In particular, we do not want to provide any promises on the inputs of Π'', as Theorem 9 does not hold for LCLs with promises on the input. For example, we cannot guarantee in the LCL definition that the provided input, that is, the partial assignment of random bits, can actually be completed into a full assignment that is good for solving Π. On the other hand, if we just defined Π'' as the problem of completing a partial given bit string assignment, it might be unsolvable for some given inputs, and this would imply that an $n^{o(1)}$ time algorithm for this problem cannot exist to begin with, thus there would be no way to use Theorem 9.

In order to solve this issue, we define Π'' such that it can be solved fast even if the input is not *nice* (for some technical definition of nice). In particular, we make sure, in the definition of Π'', that if the input is nice then the only way to solve Π'' is to actually complete the partial assignment, while if the input is not nice, and only in this case, nodes are allowed to output *wildcards* $\star$; the constraints of nodes that see wildcards in their checkability radius are automatically satisfied. This way the problem is always solvable. For an efficient algorithm, we make sure that nodes can verify in constant time if a given input assignment is nice or not. We also show that inputs produced for Π'' in the shattering framework are always nice. The definition of Π'' and the provided partial assignment allows us to split the instance of Π' into many independent instances of Π'' of size $N = O(\log n)$. By applying Theorem 9 we get that [25] implies the existence of a deterministic CONGEST algorithm $\mathcal{B}$ for Π'' with complexity $O(\log N) = O(\log\log n)$.

▶ **Lemma 16** (informal version). *There is a deterministic* CONGEST *algorithm to solve* Π'' *on any tree with at most* N *nodes in* $O(\log N)$ *rounds, regardless of the predetermined input.*

We apply algorithm $\mathcal{B}$ on each of the components $C_1, \dots, C_k$ in parallel, and the solution of Π'' on each small components together with the random bit strings from the shattering phase yield a solution for Π'. This can then be transformed into a solution for Π on the whole tree by running the constant time algorithm $\mathcal{A}_0$ with the computed random bits, which completes our task and proves Theorem 13.

6 Separation for General Graphs

In this section we define an LCL problem Π on general bounded-degree graphs, and show that, while Π can be solved deterministically in $O(\log n)$ rounds in the LOCAL model, any randomized CONGEST algorithm requires $\Omega(\sqrt{n}/\log^2 n)$ rounds. On a high level, the section is structured as follows. We start by formally defining a family $\mathcal{G}$ of graphs of interest, and then we present a set $\mathcal{C}^{\mathsf{proof}}$ of local constraints, satisfying that a graph G is in $\mathcal{G}$ if and only if it can be labeled with labels from a constant-size set, such that all nodes satisfy the constraints in $\mathcal{C}^{\mathsf{proof}}$. We then define our LCL Π in the following way:

- there is a problem Π^{real} such that, on any correctly labeled graph $G \in \mathcal{G}$, nodes must solve Π^{real};
- on any labeled graph $G \notin \mathcal{G}$, nodes can either output a locally checkable proof that shows that there exists a node in G that does not satisfy some constraint in $\mathcal{C}^{\mathsf{proof}}$ (by solving some LCL problem that we call Π^{bad}), or solve Π^{real} (if possible).

Finally, we show lower and upper bounds for Π in the CONGEST and LOCAL model respectively. The challenging part is to express all these requirements as a proper LCL, while preventing nodes from "cheating", that is, on all graphs $G \in \mathcal{G}$, it must not be possible for nodes to provide a locally checkable proof showing that G is not in $\mathcal{G}$, while for any graph $G \notin \mathcal{G}$ it should be possible to produce such a proof within the required running time.

Informally, the graphs contained in the family $\mathcal{G}$ of graphs look like the following: we start from a 2-dimensional grid; we build a binary tree-like structure on top of each column i, and let r_i be the root of the tree-like structure in top of column i; we use another grid to connect all left-most nodes of these trees; finally, we build on top of these r_i nodes another binary tree-like structure, where r_i nodes are its leaves. Note that the graphs in $\mathcal{G}$ are the bounded-degree variant of the lower bound family of graphs of Das Sarma et al. [38], where we also add some edges that are necessary in order to make the construction locally checkable.

More formally, we prove the following theorem.

▶ **Theorem 17.** *There exists an* LCL *problem* Π *that can be solved in* $O(\log n)$ *deterministic rounds in the* LOCAL *model, that requires* $\Omega(\sqrt{n}/\log^2 n)$ *rounds in the* CONGEST *model, even for randomized algorithms.*

References

1 Amir Abboud, Keren Censor-Hillel, and Seri Khoury. Near-linear lower bounds for distributed distance computations, even in sparse networks. In *Proc. 30th International Symposium on Distributed Computing (DISC 2016)*, volume 9888 of *Lecture Notes in Computer Science*, pages 29–42. Springer, 2016. doi:10.1007/978-3-662-53426-7_3.

2 Nir Bachrach, Keren Censor-Hillel, Michal Dory, Yuval Efron, Dean Leitersdorf, and Ami Paz. Hardness of distributed optimization. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 238–247, 2019. doi:10.1145/3293611.3331597.

3 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 *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 262–271. ACM Press, 2019. doi:10.1145/3293611.3331606.

4 Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In *Proc. 34th International Symposium on Distributed Computing (DISC 2020)*, volume 179 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.17.

5 Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2019)*, pages 481–497. IEEE, 2019. doi:10.1109/FOCS.2019.00037.

6 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Improved distributed lower bounds for MIS and bounded (out-)degree dominating sets in trees. In *Proc. 40th ACM Symposium on Principles of Distributed Computing (PODC 2021)*. ACM Press, 2021. doi:10.1145/3465084.3467901.

7 Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 365–376. IEEE, 2020. doi:10.1109/FOCS46700.2020.00042.

8 Alkida Balliu, Sebastian Brandt, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. Locally checkable problems in rooted trees. In *Proc. 40th ACM Symposium on Principles of Distributed Computing (PODC 2021)*. ACM Press, 2021. doi:10.1145/3465084.3467934.

9 Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the LOCAL model. *Distributed Computing*, 2020. doi:10.1007/s00446-020-00375-2.

10 Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. How much does randomness help with locally checkable problems? In *Proc. 39th ACM Symposium on Principles of Distributed Computing (PODC 2020)*, pages 299–308. ACM Press, 2020. doi:10.1145/3382734.3405715.

11 Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In *Proc. 50th ACM Symposium on Theory of Computing (STOC 2018)*, pages 1307–1318. ACM Press, 2018. doi:10.1145/3188745.3188860.

12 Alkida Balliu, Juho Hirvonen, Dennis Olivetti, and Jukka Suomela. Hardness of minimal symmetry breaking in distributed computing. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 369–378. ACM Press, 2019. doi:10.1145/3293611.3331605.

13 Leonid Barenboim, Michael Elkin, and Cyril Gavoille. A fast network-decomposition algorithm and its applications to constant-time distributed computation. *Theor. Comput. Sci.*, 751:2–23, 2018. doi:10.1016/j.tcs.2016.07.005.

14 Sebastian Brandt. An automatic speedup theorem for distributed problems. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 379–388. ACM, 2019. doi:10.1145/3293611.3331611.

15 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 *Proc. 48th ACM Symposium on Theory of Computing (STOC 2016)*, pages 479–488. ACM Press, 2016. doi:10.1145/2897518.2897570.

16 Sebastian Brandt, Jan Grebík, Christoph Grunau, and Václav Rozhoň. The landscape of distributed complexities on trees, 2021. Unpublished manuscript.

17 Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznański. LCL problems on grids. In *Proc. 36th ACM Symposium on Principles of Distributed Computing (PODC 2017)*, pages 101–110. ACM Press, 2017. doi:10.1145/3087801.3087833.

18 Keren Censor-Hillel and Michal Dory. Distributed spanner approximation. In Calvin Newport and Idit Keidar, editors, *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 139–148. ACM, 2018. doi:10.1145/3212734.3212758.

19 Keren Censor-Hillel, Seri Khoury, and Ami Paz. Quadratic and near-quadratic lower bounds for the CONGEST model. In *Proc. 31st International Symposium on Distributed Computing (DISC 2017)*, volume 91 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.DISC.2017.10.

20 Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto. The complexity of distributed edge coloring with small palettes. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018. doi:10.1137/1.9781611975031.168.

21 Yi-Jun Chang. The complexity landscape of distributed locally checkable problems on trees. In *Proc. 34th International Symposium on Distributed Computing (DISC 2020)*, volume 179 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.18.

22 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. *SIAM J. Comput.*, 48(1):122–143, 2019. doi:10.1137/17M1117537.

23 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. *SIAM J. Comput.*, 48(1):33–69, 2019. doi:10.1137/17M1157957.

24 Yi-Jun Chang, Jan Studený, and Jukka Suomela. Distributed graph problems through an automata-theoretic lens. In *Proc. 28th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2021)*, LNCS. Springer, 2021. doi:10.1007/978-3-030-79527-6_3.

25 Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. *Distributed Comput.*, 30(4):261–280, 2017. doi:10.1007/s00446-016-0287-6.

26 Michael Elkin. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM Journal on Computing*, 36(2):433–456, 2006. doi:10.1137/s0097539704441058.

27 P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions, 1973.

28 M. Fischer and M. Ghaffari. Sublogarithmic distributed algorithms for Lovász local lemma, and the complexity hierarchy. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, volume 91 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.DISC.2017.18.

29 Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and impossibilities for distributed subgraph detection. In Christian Scheideler and Jeremy T. Fineman, editors, *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 153–162. ACM, 2018. doi:10.1145/3210377.3210401.

30 Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 1150–1162. SIAM, 2012. doi:10.1137/1.9781611973099.91.

31 Nathan Linial. Distributive graph algorithms – global solutions from local data. In *Proc. Symp. on Foundations of Computer Science (FOCS 1987)*, pages 331–335, 1987. doi:10.1109/SFCS.1987.20.

32 Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992. `doi:10.1137/0221015`.

33 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 565–573, 2014. `doi:10.1145/2591796.2591850`.

34 Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. `doi:10.1137/S0097539793254571`.

35 Dennis Olivetti. Round Eliminator: a tool for automatic speedup simulation, 2020. URL: `https://github.com/olidennis/round-eliminator`.

36 David Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

37 David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM Journal on Computing*, 30(5):1427–1442, 2000. `doi:10.1137/s0097539700369740`.

38 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing*, 41(5):1235–1265, 2012. `doi:10.1137/11085178x`.