Locally Checkable Problems in Rooted Trees*

Alkida Balliu
alkida.balliu@cs.uni-freiburg.de
University of Freiburg
Freiburg, Germany

Sebastian Brandt
brandts@ethz.ch
ETH Zurich
Zurich, Switzerland

Dennis Olivetti
dennis.olivetti@cs.uni-freiburg.de
University of Freiburg
Freiburg, Germany

Jan Studený
jan.studeny@aalto.fi
Aalto University
Espoo, Finland

Jukka Suomela
jukka.suomela@aalto.fi
Aalto University
Espoo, Finland

Aleksandr Tereshchenko
aleksandr.tereshchenko@aalto.fi
Aalto University
Espoo, Finland

ABSTRACT

Consider any locally checkable labeling problem Π in *rooted regular trees*: there is a finite set of labels Σ, and for each label $x \in \Sigma$ we specify what are permitted label combinations of the children for an internal node of label x (the leaf nodes are unconstrained). This formalism is expressive enough to capture many classic problems studied in distributed computing, including vertex coloring, edge coloring, and maximal independent set.

We show that the distributed computational complexity of any such problem Π falls in one of the following classes: it is $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, or $n^{\Theta(1)}$ rounds in trees with n nodes (and all of these classes are nonempty). We show that the complexity of any given problem is the same in all four standard models of distributed graph algorithms: deterministic LOCAL, randomized LOCAL, deterministic CONGEST, and randomized CONGEST model. In particular, we show that randomness does not help in this setting, and the complexity class $\Theta(\log \log n)$ does not exist (while it does exist in the broader setting of general trees).

We also show how to systematically determine the complexity class of any such problem Π, i.e., whether Π takes $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, or $n^{\Theta(1)}$ rounds. While the algorithm may take exponential time in the size of the description of Π, it is nevertheless practical: we provide a freely available implementation of the classifier algorithm, and it is fast enough to classify many problems of interest.

CCS CONCEPTS

• **Theory of computation → Distributed computing models**; **Distributed algorithms**.

KEYWORDS

Distributed algorithms; LOCAL model; CONGEST model; Rooted trees; LCL problems

*The extended version of this work is available in [2].

PODC '21, July 26–30, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8548-0/21/07…$15.00
https://doi.org/10.1145/3465084.3467934

ACM Reference Format:
Alkida Balliu, Sebastian Brandt, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. 2021. Locally Checkable Problems in Rooted Trees. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21), July 26–30, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3465084.3467934

1 INTRODUCTION

We aim at *systematizing* and *automating* the study of computational complexity in the field of distributed graph algorithms. Many key problems of interest in the field are *locally checkable*. While it is known that questions related to the distributed computational complexity of locally checkable problems are *undecidable* in general graphs [12, 22], there is no known obstacle that would prevent one from completely automating the study of locally checkable problems in *trees*. Achieving this is one of the major open problems in the field: currently only parts of the complexity landscape are known to be decidable [15], and the general decidability results are primarily of theoretical interest; *practical* automatic techniques are only known for specific families of problems [3, 12, 16].

In this work we show that the study of locally checkable graph problems can be completely automated in *regular rooted trees*. We not only give a full classification of the distributed complexity of any such problem (in all the usual models of distributed computing: deterministic and randomized LOCAL and CONGEST), but we also present an algorithm that can automatically determine the complexity class of any given problem (with one caveat: our algorithm determines if the complexity is $n^{\Theta(1)}$, but not the precise exponent in this case). Even though the algorithm takes in the worst case exponential time in the size of the problem description, it is nevertheless practical: we have implemented it for the case of binary trees, and it is in practice very fast, classifying e.g. the sample problems that we present here in a matter of milliseconds [25].

1.1 Setting

In this work we study locally checkable problems defined in regular, unlabeled, rooted trees. For our purposes, such a problem Π is specified as a triple (δ, Σ, C), where $\delta \in \mathbb{N}$ is to the number of children for the internal nodes, Σ is a finite set of *labels*, and C is the set of permitted *configurations*. Each configuration looks like $x : y_1 y_2 \cdots y_\delta$, indicating that if the label of an internal node is x, then one of the possible labelings for its δ children is $y_1, y_2, \ldots, y_\delta$, in some order. The leaf nodes are unconstrained.

1.2 Example: 3-coloring

Consider the problem of 3-coloring binary trees, i.e., trees in which internal nodes have $\delta = 2$ children. The possible labels of the nodes are $\Sigma = \{1, 2, 3\}$. The color of a node has to be different from the colors of any of its children; hence we can write down the set of configurations e.g. as follows:

$$C = \big\{1 : 22,\ 1 : 23,\ 1 : 33,\ 2 : 11,\ 2 : 13,\ 2 : 33,\ 3 : 11,\ 3 : 12,\ 3 : 22\big\}. \tag{1}$$

We emphasize that the ordering of the children is irrelevant here; hence $1 : 23$ and $1 : 32$ are the same configuration. It is easy to verify that this is a straightforward correct encoding of the 3-coloring problem in binary trees.

It is well-known that this problem can be solved in the LOCAL model of distributed computing in $O(\log^* n)$ rounds in rooted trees, using the technique by Cole and Vishkin [18], and this is also known to be tight, both for deterministic and randomized algorithms [20, 21].

One can also in a similar way define the problem of 2-coloring binary trees; it is easy to check that this is a global problem, with complexity $\Theta(n)$ rounds:

$$C = \big\{1 : 22,\ 2 : 11\big\}. \tag{2}$$

1.3 Example: maximal independent set

Let us now look at a bit more interesting problem: maximal independent sets (MIS). Let us again stick to binary trees, i.e., $\delta = 2$ children. The first natural idea would be to try to use only two labels, 0 and 1, with 1 indicating that a node is in the independent set, but this is not sufficient to express both the notion of independence and the notion of maximality. However, three labels will be sufficient to correctly capture the problem. We set $\Sigma = \{1, a, b\}$, with 1 indicating that a node is in the independent set, and choose the following configurations:

$$C = \big\{1 : aa,\ 1 : ab,\ 1 : bb,\ a : bb,\ b : b1,\ b : 11\big\}. \tag{3}$$

Now it takes a bit more effort to convince oneself that this indeed correctly captures the idea of maximal independent sets. The key observations are these: a node with label 1 cannot be adjacent to another node with label 1, a node with label a has to have 1 above it, and a node with label b has to have 1 below it, so nodes with label 1 clearly form a maximal independent set. Conversely, given any maximal independent set X we can find a corresponding label assignment if we first assign labels 1 to nodes in X, then assign labels b to the parents of the nodes in X, and finally label the remaining nodes with label a. The only minor technicality is that the configurations ensure a correct solution for the internal parts of the tree, but as is often the case, once the internal parts are solved correctly, one can locally fix the labels near the root and the leaves.

Maximal independent set is a well-known symmetry-breaking problem, and e.g. in the case of a directed path ($\delta = 1$) it is known to be as hard as e.g. 3-coloring. Hence one might expect that the above problem for binary trees also has got the complexity of $\Theta(\log^* n)$ rounds in the LOCAL model. *This is not the case—maximal independent set in rooted binary trees can be solved in constant time!* Indeed, this is a good example of a non-trivial constant-time-solvable problem. It can be solved in exactly 4 rounds, using the following idea (again, omitting some minor details related to what happens e.g. near the root).

First, we need to pick some consistent way of referring to your "left" child and the "right" child (we can use a port numbering if available, or simply order the children by their unique identifiers). Label all nodes first with an empty string. Then we repeat the following step for 4 times: add 0 to your string and send it to your left child, and add 1 to your string and send it to your right child. Your new label is the label that you received from your parent. This way all nodes get labeled with a 4-bit string. A key property is this: if my string is $xyzw$, the string of my parent is $0xyz$ or $1xyz$. Finally, interpret the binary string as a number between 0 and 15, and output the corresponding element of the following string (using 0-based indexing):

$$b\,1\,a\,b\,b\,b\,1\,b\,b\,1\,1\,b\,b\,b\,1\,b. \tag{4}$$

One can verify the correctness of the algorithm by checking all 2^3 possible cases: for example, if a node is labeled with $x010$, it will output either symbol 2 of (4), which is a, or symbol 10, which is 1. Its two children will have labels 0100 and 0101, so they will output symbols 4 and 5 of (4), which are b and b. This results in a configuration $a : bb$ or $1 : bb$, both of which are valid in (4).

The key point of the example is this: even though the algorithm is somewhat involved, we can use the computer program accompanying in this work to *automatically* discover this algorithm and to determine that this problem is indeed constant-time solvable! Also, this problem demonstrates that there are $O(1)$-round-solvable locally checkable problems in rooted regular trees that require strictly more than zero rounds, while e.g. in the previously-studied family of binary labeling problems [3] all $O(1)$-round-solvable problems are known to be zero round solvable.

1.4 Example: branch 2-coloring

As the final example, let us consider the following problem, with $\delta = 2$ and $\Sigma = \{1, 2\}$:

$$C = \big\{1 : 12,\ 2 : 11\big\}. \tag{5}$$

This problem is, in essence, 2-coloring with a choice: starting with a node of label 1 and going downwards, there is always a monochromatic path labeled with $1, 1, 1, 1, \ldots$, and a properly colored path labeled with $1, 2, 1, 2, \ldots$. It turns out that the choice makes enough of a difference: the complexity of this problem is $\Theta(\log n)$ rounds. We encourage the reader to come up with an algorithm and a matching lower bound—with our techniques we get a tight result immediately.

1.5 Contributions

As we have seen, the family of locally checkable problems in regular rooted trees is rich and expressive. Using auxiliary labels similar to what we saw in the MIS example in Section 1.3, we can encode, in essence, any locally checkable labeling problem (LCL problem) [22] in the classic sense, as long as the problem is such that the interesting part is related to what happens in the internal parts of regular trees. We have already seen that there are problems with at least four distinct complexity classes: $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, and

Table 1: An overview of the landscape and decidability of the round complexity of LCL problems in the LOCAL model. The case studied in the present work (unlabeled, rooted, regular trees) is highlighted with shading, and the darker shade indicates the key new results. The decidability is given assuming P ≠ PSPACE ≠ EXPTIME. We have listed a few key references for each column, focusing on decidability aspects; the overall picture of the complexity landscape is the result of a long sequence of papers, including [5, 6, 8, 11, 14, 18, 20, 21].

Setting		*Paths and cycles*				*Trees*					*General*
	unlabeled input	✓	✓	✓		✓	✓	✓	✓		
	regular	✓	✓			✓	✓	✓	✓		
	directed or rooted	✓	✓					✓			
	binary output	✓					✓				
	homogeneous					✓					
Complexity classes	$O(1)$	D+R	D+R	D+R	D+R	D+R	D+R	D+R	D+R	D+R	D+R
	⋯	—	—	—	—	—	—	—	—	—	—
	$\Theta(\log \log^* n)$	—	—	—	—	—	—	—	?	?	D+R
	⋯	—	—	—	—	—	—	—	?	?	D+R
	$\Theta(\log^* n)$	—	D+R	D+R	D+R	D+R	—	D+R	D+R	D+R	D+R
	⋯	—	—	—	—	—	—	—	—	—	—
	$\Theta(\log \log n)$	—	—	—	—	R	R	—	R	R	R
	$\Theta(\log^\alpha \log n)$	—	—	—	—	—	—	—	—	—	?
	⋯	—	—	—	—	—	—	—	—	—	—
	$\Theta(\log n)$	—	—	—	—	D+R	D+R	D+R	D+R	D+R	D+R
	⋯	—	—	—	—	—	—	—	—	—	D+R
	$\Theta(n^{1/k})$	—	—	—	—	—	—	(n)	?	(n)	D+R
	⋯	—	—	—	—	—	—	—	—	—	D+R
	$\Theta(n)$	D+R	D+R	D+R	D+R	—	D+R	D+R	D+R	D+R	D+R
Decidability	P	✓	✓	✓	—	?	(D)	?	?	—	—
	PSPACE	✓	✓	✓	?	?	(D)	?	?	—	—
	EXPTIME	✓	✓	✓	?	?	(D)	(k)	?	?	—
	decidable	✓	✓	✓	✓	?	(D)	✓	(H)	(H)	—
References		[3, 12, 22]	[12, 22]	[16]	[1]	[9]	[3]	this work	[13, 15]	[13, 15]	[12, 22]

Legend
✓ = yes, ? = unknown, — = not possible, $\alpha > 1$, $k = 2, 3, \ldots$.
D = class exists for deterministic algorithms, R = class exists for randomized algorithms.
(n) = the current construction assumes the knowledge of n; unknown without this information.
(k) = does not determine the value of k for the class $\Theta(n^{1/k})$.
(D) = known only for deterministic complexities, unknown for randomized.
(H) = known only for classes between $\Omega(\log n)$ and $O(n)$.

$\Theta(n)$. In the extended version of this work [2] we also show how to generate problems of complexity $\Theta(n^{1/k})$ for any $k = 1, 2, 3, \ldots$

We prove in this work that *this list is exhaustive*: any problem that can be represented in our formalism has got the complexity $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, or $\Theta(n^{1/k})$ in rooted regular trees with n nodes. This is a *robust* result that does not depend on the specific choice of the model of computing: the complexity of a given problem is the same, regardless of whether we are looking at the LOCAL model or the CONGEST model, and regardless of whether we are using deterministic or randomized algorithms.

One of the surprising consequences is that *randomness does not help* in rooted regular trees. In unrooted regular trees there are problems (the canonical example being the *sinkless orientation* problem) that can be solved with the help of randomness in $\Theta(\log \log n)$ rounds, while the deterministic complexity is $\Theta(\log n)$ [12]. This class of problems disappears in rooted trees.

Our main contribution is that the *complexity of any given problem in this formalism is decidable*: there is an algorithm that, given the description of a problem Π as a list of permitted configurations, outputs the computational complexity of problem Π, putting it in one of the four possible classes, i.e., determines whether the complexity is $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, or $\Theta(n^{1/k})$ for some k; in the fourth case our algorithm does not determine the exponent k, but then one could (at least in principle) use the more general decision procedure by Chang [13] to determine the value of k.

While our algorithm takes in the worst case exponential time in the size of the description of Π, the approach is nevertheless *practical*. We have *implemented* the algorithm for the case of $\delta = 2$,

and made it freely available online [25]. Even though it is not at all optimized for performance, it classifies for example all of our sample problems above in a matter of *milliseconds*.

We summarize our key results and compare them with prior work in Table 1.

2 RELATED WORK

2.1 Landscape of LCL problems in the LOCAL model

Paths and cycles. We know that, on graph families such as paths and cycles, there are LCLs with complexities (both deterministic and randomized) of $O(1)$ (e.g., trivial problems), $\Theta(\log^* n)$ [18, 20, 21] (e.g., 3-coloring), and $\Theta(n)$ (e.g., global problems such as properly orienting a path/cycle). In these families of graphs, there are no LCLs with round complexity between $\omega(1)$ and $o(\log^* n)$ [22], and between $\omega(\log^* n)$ and $o(n)$ [14]. These works show that the only possible complexities for LCL problems on paths and cycles are $O(1)$, $\Theta(\log^* n)$, and $\Theta(n)$, and randomness does not help in solving problems faster.

Trees. For the case of the graph family of trees almost everything is understood nowadays. As in the case of paths and cycles, we have LCLs with time complexities (both deterministic and randomized) $O(1)$, $\Theta(\log^* n)$, and $\Theta(n)$. On trees, we know that there is more: there are LCL problems with both deterministic and randomized complexity of $\Theta(\log n)$ (e.g., problems of the form "copy the input of the nearest leaf"), and $\Theta(n^{1/k})$ for any $k \geq 2$ [15]. Moreover, there are cases where randomness helps, in fact there are problems that have $\Theta(\log n)$ deterministic and $\Theta(\log \log n)$ randomized complexity [11]. As far as gaps are concerned, let us first consider the spectrum of time complexities of $\omega(\log^* n)$, and then the one of $o(\log^* n)$. Chang et al. [14] showed that the deterministic complexity of any LCL problem on bounded-degree trees is either $O(\log^* n)$ or $\Omega(\log n)$, while its randomized complexity is either $O(\log^* n)$ or $\Omega(\log \log n)$. Moreover, Chang and Pettie [15] showed that any algorithm that takes $n^{o(1)}$ rounds can be sped up to run in $O(\log n)$ rounds. Balliu et al. [5] showed that there is a gap between $\omega(\sqrt{n})$ and $o(n)$ for deterministic algorithms, and Chang [13] extended these results and showed that there is a gap between $\omega(n^{1/k})$ and $o(n^{1/(k-1)})$, for any $k \geq 2$, for both deterministic and randomized algorithms. The spectrum of time complexities of $o(\log^* n)$ is still not entirely understood. Chang and Pettie [15] showed that ideas similar to Naor and Stockmeyer [22] can be used to prove that there are no LCLs on bounded-degree trees with time complexity between $\omega(1)$ and $o(\log \log^* n)$. Also, in the same paper, the authors conjectured that it should be possible to extend this gap up to $o(\log^* n)$. While this still remains an open question, Balliu et al. [9] showed that such a gap exists for a special subclass of LCLs, called *homogeneous* LCLs.

General graphs. As in the case of trees, also on general bounded-degree graphs we have LCLs with the same time complexities, so the question is if there are also the same gaps, or if in the case of general graphs we have a denser spectrum of complexities. First of all, the gaps of the lower spectrum on trees hold also on general graphs: we still have the $\omega(1) - o(\log \log^* n)$ gap for both deterministic and randomized algorithms, the $\omega(\log^* n) - o(\log n)$ for deterministic algorithms, and the $\omega(\log^* n) - o(\log \log n)$ gap for randomized algorithms. Also, Chang and Pettie [15] showed that any $o(\log n)$-round randomized algorithm can be sped up to run in $O(T_{\mathsf{LLL}})$ rounds, where T_{LLL} is the time required for solving with randomized algorithms the distributed constructive Lovász Local Lemma problem (LLL) [17] under a polynomial criterion. By combining this result with the results on the complexity of LLL by Fischer and Ghaffari [19] and the network decomposition one by Rozhoň and Ghaffari [24], we get a gap for randomized algorithms between $\omega(\text{poly}(\log \log n))$ and $o(\log n)$. Balliu et al. [8] showed that, differently from the case of trees, the regions between $\omega(\log \log^* n)$ and $o(\log^* n)$ and between $\omega(\log n)$ and $o(n)$ are dense. In fact, there are infinitely many LCLs with time complexity (both deterministic and randomized) that fall into these regions. Also, in the case of trees, randomness either helps exponentially or not at all, while in the case of general graphs this is not the case anymore. In fact, Balliu et al. [6] showed that there are LCL problems on general graphs where randomness helps only polynomially by defining LCLs with deterministic complexity $\Theta(\log^k n)$ and randomized complexity $\Theta(\log^{k-1} n \log \log n)$, for any integer $k \geq 1$.

Special settings. Over the years, researchers have investigated the complexity of interesting subclasses of LCLs. We already mentioned homogeneous LCLs on trees [9], that, on a high level, are LCLs for which the hard instances are Δ-regular trees. For this subclass of LCL problems, the spectrum of deterministic complexities consists of $O(1)$, $\Theta(\log^* n)$, and $\Theta(\log n)$. Also, as in the case of trees, there are cases where randomness helps: there are homogeneous LCLs with $\Theta(\log n)$ deterministic and $\Theta(\log \log n)$ randomized complexity. These are the only possible complexities for homogeneous LCLs. Brandt et al. [12] studied LCLs on d-dimensional torus grids, and showed that there are LCLs with complexity (both deterministic and randomized) $O(1)$, $\Theta(\log^* n)$, and $\Theta(n^{1/d})$. The authors showed that these are the only possible complexities, implying that randomness does not help. Balliu et al. [3] studied *binary labeling problems*, that are LCLs that can be expressed with no more than two labels in the *edge labeling* formalism [4, 23] (such LCLs include, for example, sinkless orientation). The authors showed that, in trees, there are no such LCLs with deterministic round complexity between $\omega(1)$ and $o(\log n)$, and between $\omega(\log n)$ and $o(n)$, proving that the spectrum of deterministic complexities of binary labeling problems in bounded-degree trees consists of $O(1)$, $\Theta(\log n)$ and $\Theta(n)$. The authors also studied the randomized complexity of binary labeling problems that have deterministic complexity $\Theta(\log n)$, showing that for some of them randomness does not help, while for some others it does help (note that from previous work we know that, in this case, randomness either helps exponentially or not at all). Determining the tight randomized complexity of all binary labeling problems is still an open question.

2.2 Decidability of LCL problems

As we have seen, there are often gaps in the spectrum of distributed complexities of LCLs. Hence, a natural question that arises is the following: given a specific LCL, can we decide on which side of the gap it falls? In other words, are these classifications of LCL problems decidable? We can push this question further and ask whether it is possible to automate the design of distributed algorithms for

optimally solving LCLs. There is a long line of research that has investigated these kind of questions.

For graph families that consist of *unlabeled* paths and cycles (that is, nodes do not have any label in input), the complexity of a given LCL is decidable [12, 16, 22]. The next natural question is whether we have decidability in the case of trees (rooted or not). Because the structure of a tree can be used to encode input labels, researchers had to first understand the role of input labels in decidability. For this purpose, Balliu et al. [1] studied the decidability of *labeled* paths and cycles, showing that the complexity of LCLs in this setting is decidable, but it is PSPACE-hard to decide it, and this PSPACE-hardness result extends also for the case of bounded-degree unlabeled trees (since the structure of the tree may encode input labels). The authors also show how to automate the design of asymptotically optimal distributed algorithms for solving LCLs in this context. Later, Chang [13] improved these results showing that, in this setting, it is EXPTIME-hard to decide the complexity of LCLs. While the decidability on bounded degree trees is still an open question, there are some positive partial results in this direction. In fact, Chang and Pettie [15] along with the $\omega(\log n) - n^{o(1)}$ gap, showed also that we can decide on which side of the gap the complexity of an LCL lies. Moreover, Balliu et al. [3] showed that, the deterministic complexity of binary labeling problems on trees is decidable and we can automatically find optimal algorithms that solve such LCLs. The works of Brandt [10] and Olivetti [23] played a fundamental role in further understanding to which extent we can automate the design of algorithms that optimally solve LCLs.

Unfortunately, in general, the complexity of an LCL is not decidable. In fact, Naor and Stockmeyer showed that, even on unlabeled non-toroidal grid graphs, it is undecidable whether the complexity of a given LCL is $O(1)$ [22]. For unlabeled toroidal grids, Brandt et al. [12] showed that, given an LCL, it is decidable whether its complexity is $O(1)$, but it is undecidable whether its complexity is $\Theta(\log^* n)$ or $\Theta(n)$. On the positive side, the authors showed that, given an LCL with round complexity $O(\log^* n)$, one can automatically find an $O(\log^* n)$ rounds algorithm that solves it.

3 ROAD MAP

We will start by providing some useful definitions in Section 4. Then, in Section 5 we will consider the spectrum of complexities in the $\Omega(\log n)$ region. We will define an object called *certificate for $O(\log n)$ solvability*, for which we will prove, in Theorem 5.3, that we can decide the existence in polynomial time. We will prove in Theorem 5.1 that, if such a certificate for a problem exists, then the problem can be solved in $O(\log n)$ time with a deterministic algorithm, even in the CONGEST model, while if such a certificate does not exist then we will prove in Theorem 5.2 that the problem requires $n^{\Omega(1)}$ rounds, even in the LOCAL model and even for randomized algorithms. By combining these results, we will essentially obtain a *decidable gap* between $\omega(\log n)$ and $n^{o(1)}$ that is robust on the choice of the model.

We will then consider, in Section 6, the spectrum of complexities in the $O(\log n)$ region. We will define the notion of *certificate for $O(\log^* n)$ solvability*, and we will prove, in Theorem 6.5, that we can decide in exponential time if such a certificate exists. We will also prove, in Theorem 6.2, that the existence of such a certificate implies a deterministic $O(\log^* n)$ algorithm for the CONGEST model, while we will prove in Theorem 6.4 that the non-existence of such a certificate implies an $\Omega(\log n)$ randomized lower bound for the LOCAL model. Hence, also in this case we obtain a decidable gap that is robust on the choice of the model.

Finally, we will consider in Section 7 the spectrum of complexities in the $O(\log^* n)$ region. We will define the notion of *certificate for $O(1)$ solvability*, that will be nothing else than a certificate for $O(\log^* n)$ solvability that has some special property. We will show in Theorem 7.4 that, also in this case, we can decide its existence in exponential time, and we will show in Theorem 7.2 that its existence implies a constant time deterministic algorithm for the CONGEST model, while we will show in Theorem 7.3 that the non-existence implies an $\Omega(\log^* n)$ lower bound for the LOCAL model. Hence, we will obtain that there are only four possible complexities, $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, and $n^{\Omega(1)}$, and that for all problems we can decide which of these four complexities is the right one.

For the fine-grained structure inside the $n^{\Omega(1)}$ class we refer to the prior work [7, 13]; while these papers study the case of *unrooted* trees, we note that the orientation can be encoded as a locally checkable input, and the results are also applicable here. It follows that there are only classes $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, and $\Theta(n^{1/k})$ for $k = 1, 2, \ldots$, and the exact class (including the value of k) is decidable.

4 DEFINITIONS

In this section we define some notions that will be used in the following sections.

Input graphs. We assume that all input graphs will be *unlabeled rooted trees* where each node has either exactly δ or 0 children for some positive integer δ. That is, input graphs are *full δ-ary trees*.

For convenience, when not specified otherwise, a tree T is assumed to be a *full δ-ary* tree.

Models of computing. The models that we consider in this work are the classical LOCAL and CONGEST model of distributed computing. Let G be any graph with n nodes and maximum degree Δ. In the LOCAL model, each node of G is equipped with an identifier in $\{1, 2, \ldots, \text{poly}(n)\}$, and the initial knowledge of a node consists of its own identifier, its degree (i.e., the number of incident edges), the total number n of nodes, and Δ (in the case of rooted trees, each node knows also which of its incident edges connects it to its parent). Nodes try to learn more about the input instance by communicating with the neighbors. The computation proceeds in synchronous rounds, and at each round nodes send messages to neighbors, receive messages from them, and perform local computation. Messages can be arbitrarily large and the local computational can be of arbitrary complexity. Each node must terminate its computation at some point and decide on its local output. The running time of a distributed algorithm running at each node in the LOCAL model is determined by the number of rounds needed such that all nodes have produced their local output. In the randomized version of the LOCAL model, each node has access to a stream of random bits. The randomized algorithms considered in this paper are Monte Carlo ones, that is, a randomized algorithm of complexity T that

solves a problem P must terminate at all nodes upon T rounds and this should result in a global solution for P that is correct with probability at least $1 - 1/n$.

There is only one difference between the CONGEST and the LOCAL model, and it lies in the size of the messages. While in the LOCAL model messages can be arbitrarily large, in the CONGEST model the size of the messages is bounded by $O(\log n)$ bits.

LCL *problem.* We define LCL problems as follows.

Definition 4.1 (LCL problem)**.** An LCL problem is a triple $\Pi = (\delta, \Sigma, C)$ where:

- δ is the number of allowed children;
- Σ is a finite set of (output) labels;
- C is a set of tuples of size $\delta + 1$ from $\Sigma^{\delta+1}$ called *allowed configurations*.

We will use different possible notations for the configurations. A configuration $(a, b_1, \ldots, b_\delta)$ will also be written as $(a : b_1, \ldots, b_\delta)$, in order to highlight that the label a is for the parent and $b_1, \ldots, b_\delta$ are the label of its leaves. Sometimes we will omit the commas, and just write $(a : b_1 \ldots b_\delta)$. Sometimes even the parenthesis will be omitted, obtaining $a : b_1 \ldots b_\delta$, that is the notation used in e.g. Section 1.3. As a shorthand notation, for an LCL problem Π, we will also denote the labels and configurations of Π by Σ_Π and C_Π.

Definition 4.2 (solution)**.** A solution to an LCL problem Π for a tree T is a labeling function λ for which:

- every node $v \in T$ is labeled by a label $\lambda(v)$ from Σ_Π;
- every node $v \in T$ with δ children $v_1, \ldots, v_\delta$ satisfies that there exists a permutation $\rho\colon \{1, \ldots, \delta\} \to \{1, \ldots, \delta\}$ such that $(\lambda(v) : \lambda(v_{\rho(1)}), \ldots, \lambda(v_{\rho(\delta)}))$ is in C_Π.

In other words, a solution is a labeling for the nodes that must satisfy some local constraints. Note that only nodes with δ children are constrained, but that such LCL problems could be well-defined even on non-full δ-ary trees (nodes with a number of children different from δ are unconstrained). Full δ-ary trees are the hardest instances for the problems.

Definition 4.3 (restriction)**.** Given an LCL problem $\Pi = (\delta, \Sigma, C)$, a restriction of Π to labels $\Sigma' \subseteq \Sigma$ is a new LCL problem $\Pi' = (\delta, \Sigma', C')$ where C' is obtained by keeping all and only the configurations in C that only use labels in Σ'.

Definition 4.4 (path-form of an LCL problem)**.** Let Π be an LCL problem. The *path-form* of Π, denoted by Π^{path} is a problem on directed paths defined as follows. We transform each configuration $c \in C_\Pi$ of form $c = (a : b_1, b_2, .., b_\delta)$ to δ configurations of Π^{path}, $(a : b_i)$, for all $1 \le i \le \delta$.

Definition 4.5 (automaton associated with path-form of an LCL problem; [16])**.** Let Π^{path} be the path-form of an LCL problem Π. The automaton $\mathcal{M}(\Pi^{\text{path}})$ is a nondeterministic unary semiautomaton such that:

- The set of states is $\Sigma_{\Pi^{\text{path}}}$ $(= \Sigma_\Pi)$.
- There is a transition from state a to state b whenever the configuration $(a : b)$ is in $C_{\Pi^{\text{path}}}$.

Definition 4.6 (flexible state of an automaton; [16])**.** A state a from $\mathcal{M}(\Pi^{\text{path}})$ is flexible with flexibility $K = \text{flexibility}(a)$ if for all $k \ge K$ there is a walk $a \rightsquigarrow a$ of length exactly k in $\mathcal{M}(\Pi^{\text{path}})$.

As the set of states of the automation is the set of labels, we can expand the notion of flexibility of a state to the notion of flexibility of a label.

Definition 4.7 (path-flexibility)**.** Let Π be an LCL problem and Π^{path} its path-flexible form. A label $\sigma \in \Sigma_\Pi$ is *path-flexible* if σ is a flexible state in automaton $\mathcal{M}(\Pi^{\text{path}})$, and *path-inflexible* otherwise.

Definition 4.8 (ruling set)**.** Let G be a graph. A (k, l)-ruling set is a subset S of vertices of G such that the distance between any two vertices in S is at least k, and the distance between any vertex in G and the closest vertex in S is at most l.

5 SUPER-LOGARITHMIC REGION

In this section we prove that there is no LCL problem Π with distributed time complexity between $\omega(\log n)$ and $n^{o(1)}$. Also, we prove that, given a problem Π, we can *decide* if its complexity is $O(\log n)$ or $n^{\Omega(1)}$. Moreover, we prove that randomness cannot help: if a problem has *randomized* complexity $O(\log n)$, then it has the same *deterministic* complexity.

High level idea. The key idea is that we *iteratively prune* the description of problem Π by removing subsets of labels that we call *path-inflexible*—these are sets of labels that require long-distance coordination (cf. 2-coloring). After each such step, we may arrive at a subproblem that contains a new path-inflexible set, but eventually the pruning process will terminate, as there is only a finite number of labels.

Assume the pruning process terminates after k steps. Let X_1, $X_2, \ldots, X_k$ be the sets of labels we removed during the process, and let X' be the set of labels that is left after no path-inflexible labels remain. We have two cases:

(1) Set X' is empty. In this case we can show that the round complexity of the problem Π is at least $\Omega(n^{1/k})$. To prove this, we make use of a k-level construction that generalizes the one used for $2\frac{1}{2}$-coloring in [15]; each level consists of large enough constant-size balanced subtrees and paths of length $\Theta(n^{1/k})$. We argue that, roughly speaking, there must exist a path of length $\Theta(n^{1/k})$ that will be labeled using only labels from set X_i for some i, and we prove that this requires coordination over distance $\Theta(n^{1/k})$.
(2) Set X' is non-empty. But now we are left with a non-empty *path-flexible* subproblem $\Pi' \subseteq \Pi$, and we can make use of the flexibility to solve Π' in $O(\log n)$ rounds. Hence the original problem Π is also solvable in $O(\log n)$ rounds.

We say that problem Π has a *certificate for* $O(\log n)$ *solvability* if and only if the set X' is non-empty.

In the full version of this work [2] we prove the following theorems.

Theorem 5.1. *Let Π be a problem having a certificate for $O(\log n)$ solvability. Then Π is solvable in $O(\log n)$ rounds in the* CONGEST *model.*

Theorem 5.2. *Let Π be an* LCL *problem having no certificate for $O(\log n)$ solvability. Then both the randomized and the deterministic complexity of Π in the* LOCAL *model is $\Omega(n^{1/k})$ for some $k \ge 1$.*

Theorem 5.3. *Whether an* LCL *problem* Π *has round complexity* $O(\log n)$ *or* $n^{\Omega(1)}$ *can be decided in polynomial time.*

6 SUBLOGARITHMIC REGION

In this section we prove that there is no LCL problem Π with distributed time complexity between $\omega(\log^* n)$ and $o(\log n)$. Also, we prove that, given a problem Π, we can *decide* if its complexity is $O(\log^* n)$ or $\Omega(\log n)$. Moreover, we prove that randomness cannot help: if a problem has *randomized* complexity $O(\log^* n)$, then it has the same *deterministic* complexity.

High level idea. Informally, we prove that all problems that are $O(\log^* n)$ solvable can be solved in a normalized way, that is the following:

- Split the rooted tree in constant size rooted subtrees, where each root has some minimum distance from the leaves. Note that each leaf is the root of another subtree.
- In each subtree, assign labels to the leaves, such that for any assignment to the root, the subtree can be completed with a valid labeling.
- Complete the labeling in each subtree independently.

Note that the only part requiring $\Theta(\log^* n)$ is the first one, while the rest requires constant time. We then also prove that we can *decide* if there is a subset of labels, and an assignment for the leaves of the subtrees, that satisfies the second point.

The certificate. We start by defining what is a uniform certificate of $O(\log^* n)$ solvability. Informally, it is a sequence of labeled trees having the same depth and the leaves labeled in the same way, such that for each label used in the trees there is a tree with the root labeled with that label. An example of such a certificate for the 3-coloring problem is depicted in Figure 1.

Definition 6.1 (uniform certificate for $O(\log^* n)$ solvability)**.** Let Π be an LCL problem. A uniform certificate of $O(\log^* n)$ solvability for $\Pi = (C_\Pi, \Sigma_\Pi)$ with labels $\Sigma_\mathcal{T} = \{\sigma_0, \ldots, \sigma_t\} \subseteq \Sigma_\Pi$ and depth d is a sequence $\mathcal{T}$ of t labeled trees (denoted by $\mathcal{T}_i$) such that:

(1) Each tree is a complete δ-ary tree of depth d (d has to be at least one).
(2) Each tree $\mathcal{T}_i$ is labeled by labels from $\Sigma_\mathcal{T}$ and correct w.r.t. configurations C_Π.
(3) Let $\overline{\mathcal{T}}_i$ be the tree obtained by starting from $\mathcal{T}_i$ and removing the labels of all non-leaf nodes. It must hold that all trees $\overline{\mathcal{T}}_i$ are isomorphic, preserving the labeling.
(4) Root of tree $\mathcal{T}_i$ is labeled with label σ_i.

In the full version of this work [2] we prove the following theorems.

Theorem 6.2. *Assume that a uniform certificate of* $O(\log^* n)$ *solvability for* Π *exists. Then* Π *can be solved in* $O(\log^* n)$ *rounds in the* CONGEST *model.*

Theorem 6.3. *If* Π *has deterministic complexity* $o(\log n)$ *then there exists a certificate of* $O(\log^* n)$ *solvability.*

Theorem 6.4. *Let* Π *be an* LCL *problem for which no certificate for* $O(\log^* n)$ *solvability exists. Then, the randomized and deterministic complexity of* Π *in the* LOCAL *model is* $\Omega(\log n)$.

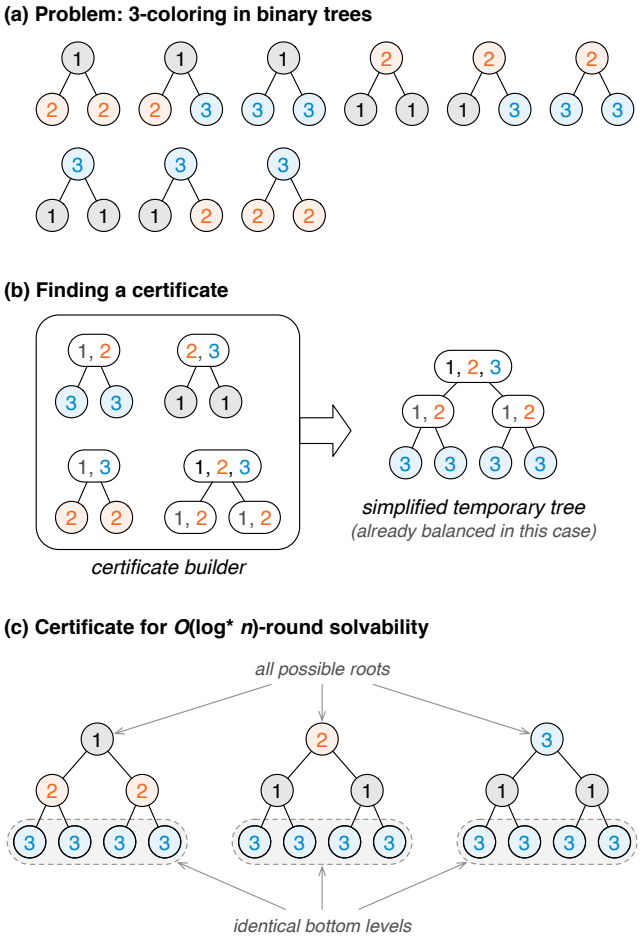

Figure 1: Finding a uniform certificate for $O(\log^* n)$ solvability (Definition 6.1) for the 3-coloring problem (Section 1.2).

Theorem 6.5. *Whether an* LCL *problem* Π *has round complexity* $O(\log^* n)$ *or* $\Omega(\log n)$ *can be decided in time at most exponential in the size of the* LCL *problem.*

7 SUB-LOG-STAR REGION

In this section we prove that there is no LCL problem Π with distributed time complexity between $\omega(1)$ and $o(\log^* n)$. Also, we prove that, given a problem Π, we can *decide* if its complexity is $O(1)$ or $\Omega(\log^* n)$. Moreover, we prove that randomness cannot help: if a problem has *randomized* complexity $O(1)$, then it has the same *deterministic* complexity.

High level idea. We prove that deciding if a problem Π can be solved in constant time is surprisingly simple: a problem is $O(1)$ rounds solvable if and only if it can be solved in $O(\log^* n)$ rounds and Π contains an allowed configuration of a specific form. This configuration must allow a node to have the same label ℓ of one child, the labels used by this configuration should be contained in the ones used by some certificate for $O(\log^* n)$ solvability, and ℓ should be used by at least one leaf of the certificate. If we consider

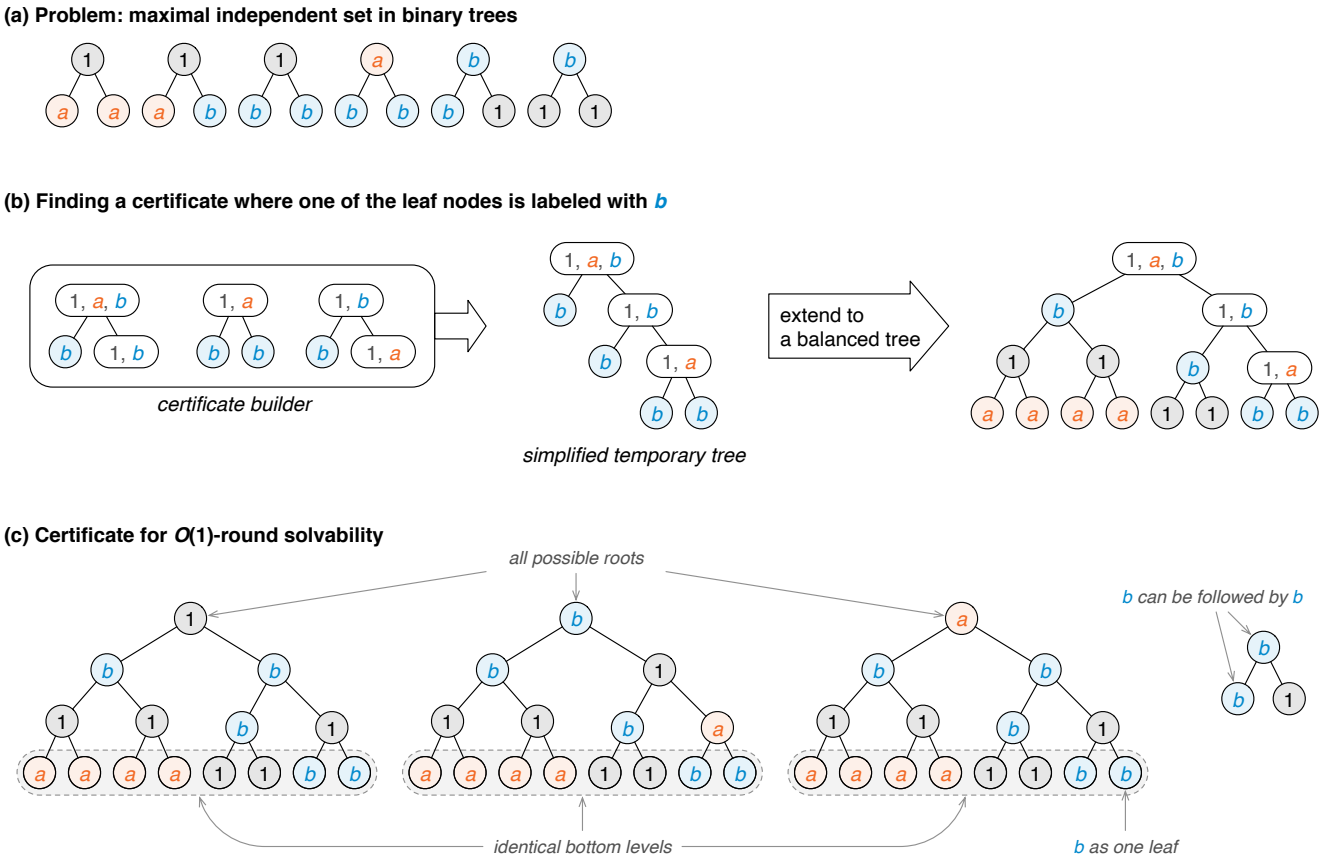

Figure 2: Finding a certificate for $O(1)$ solvability (Definition 7.1) for the maximal independent set problem (Section 1.3).

the definition of the MIS problem given in Section 1.3, we can see that it allows the configuration $(b : b1)$, and informally this configuration is what makes the problem constant time solvable. Note that, however, the algorithm that we can obtain by using this certificate, while still being constant time, may have a worse complexity compared to the one described in Section 1.3. On the other hand, we can see that in the definition of the 3-coloring problem given in Section 1.2 there is no configuration of this form, and this is what makes the problem $\Omega(\log^* n)$.

Informally, the reason is the following. The n that appears in $O(\log^* n)$ complexities does not usually refer to the size of the graph, but to the range of the identifiers assigned to the nodes. In fact, in the proof of Theorem 6.2, $O(\log^* n)$ is spent only to compute some ruling set, while the rest only requires constant time, and in order to compute such a ruling set, a distance-k coloring, for some large enough k, is sufficient. Unfortunately, it is not possible to compute a distance-k coloring in constant time, but as we will show, some defective coloring will be sufficient for our purposes. We show that in constant time we can produce some defective distance-k coloring, for some large enough constant k, such that:

- we can label defective nodes with the special configuration,
- unlabeled nodes are properly colored, and
- labeled nodes that are in different connected components are far enough from each other.

We can then complete the partial labeling in constant time with the help of the certificate, similarly to how we use the certificate of $O(\log^* n)$ solvability to solve problems in $O(\log^* n)$ rounds, but this time we can speed the computation of the ruling set up, and make it run in constant time by exploiting the distance-k coloring. In the other direction, we show that if the special configuration does not exist, or if it does not satisfy the required properties, then any algorithm solving the problem can also be used to solve the coloring problem with a constant size palette, that is known to require $\Omega(\log^* n)$ rounds.

The certificate. We start by defining what is a certificate for $O(1)$ solvability, that is nothing else but a certificate for $O(\log^* n)$ solvability and a configuration of some specific form. An example of such a certificate for the MIS problem is depicted in Figure 2.

Definition 7.1. Let Π be an LCL problem. A certificate for $O(1)$ solvability for problem Π is a pair $\mathcal{S}$ consisting of a certificate for $O(\log^* n)$ solvability $\mathcal{T}$ and a configuration $(a : b_1, \ldots, a, \ldots, b_\delta) \in \mathcal{C}_\Pi$ where $a, b_i \in \Sigma_\mathcal{T}$ and at least one leaf of the trees in $\mathcal{T}$ is labeled a.

$O(1)$ *solvability.* We now prove that we can use a certificate for $O(1)$ solvability to construct an algorithm that solves the problem Π in constant time. Informally, we first spend a constant number of rounds to try to construct some distance-k coloring. This coloring cannot always be correct, since the coloring problem requires $\Omega(\log^* n)$ rounds. We will use the special configuration to label nodes in which the coloring procedure failed. The coloring will also satisfy some desirable property, such as having improperly colored regions that are far enough from each other. This will give us a proper distance-k coloring in the unlabeled regions, and we will use this coloring to complete the labeling in constant time.

In the full version of this work [2] we prove the following theorems.

Theorem 7.2. *Any* LCL *problem* Π *that has a certificate of* $O(1)$ *solvability is constant time solvable with a deterministic* CONGEST *algorithm.*

Theorem 7.3. *Let* Π *be an* LCL *problem for which no certificate for* $O(1)$ *solvability exists. Then, the randomized and deterministic complexity of* Π *in the* LOCAL *model is* $\Omega(\log^* n)$.

Theorem 7.4. *Whether an* LCL *problem* Π *has round complexity* $O(1)$ *or* $\Omega(\log^* n)$ *can be decided in time at most exponential in the size of the* LCL *problem.*

8 FUTURE WORK

While we completely characterize all complexities for LCLs in rooted trees in both LOCAL and CONGEST, for both deterministic and randomized algorithms, and we show that we can decide what is the complexity of a given problem, there are many questions that are left open.

The first question regards the running time of the algorithm that tries to find a certificate for $O(\log^* n)$ solvability. The current running time is exponential, and an open question is whether we can find such a certificate in polynomial time, or if we can prove that e.g. deciding the existence of a certificate is an NP-hard problem.

The second question regards the complexity class of $n^{\Theta(1)}$. While we present a practical algorithm that determines if the complexity is $\Theta(n^{1/k})$ for some $k = 1, 2, \ldots$, our algorithm does not determine the precise value of k, it only gives an upper bound. Whether there is an efficient algorithm for finding the value of k remains open.

Another natural question regards extending our results to unrooted trees. While decidability is known in the $\Omega(\log n)$ region, it is known to require exponential time [13]. In our setting, we can decide if a problem is $O(\log n)$ or $n^{\Omega(1)}$ in polynomial time, and it is an open question whether it is possible to obtain an analogous decidability result for regular unrooted trees. Also, deciding if a problem on regular trees requires $O(1)$, $\Theta(\log^* n)$, or $\Omega(\log n)$ rounds is a major open question.

ERRATA

A prior version of this work contained a mistake in the classification in the $n^{\Theta(1)}$ region. It was erroneously claimed that the only possible complexity class in this region is $\Theta(n)$, which is not the case. We are thankful for Yi-Jun Chang for pointing out this mistake; in the extended version of this work [2] (with Yi-Jun Chang as a new coauthor) we will discuss this region in more detail.

ACKNOWLEDGMENTS

We would also like to thank Yannic Maus for helpful feedback on related work, Juho Hirvonen for useful discussions, Mikaël Rabie for pointing out subtle errors, and anonymous reviewers for their helpful feedback on previous versions of this work. The authors wish to acknowledge CSC – IT Center for Science, Finland, for computational resources.

REFERENCES

[1] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. 2019. The distributed complexity of locally checkable problems on paths is decidable. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*. ACM Press, 262–271. https://doi.org/10.1145/3293611.3331606 arXiv:1811.01672

[2] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. 2021. Locally Checkable Problems in Rooted Trees. arXiv:2102.09277

[3] Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. 2020. Classification of distributed binary labeling problems. In *Proc. 34th International Symposium on Distributed Computing (DISC 2020) (LIPIcs, Vol. 179)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 17:1–17:17. https://doi.org/10.4230/LIPIcs.DISC.2020.17 arXiv:1911.13294

[4] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. 2019. Lower bounds for maximal matchings and maximal independent sets. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2019)*. IEEE, 481–497. https://doi.org/10.1109/FOCS.2019.00037 arXiv:1901.02441

[5] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. 2020. Almost global problems in the LOCAL model. *Distributed Computing*. https://doi.org/10.1007/s00446-020-00375-2 arXiv:1805.04776

[6] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. 2020. How much does randomness help with locally checkable problems?. In *Proc. 39th ACM Symposium on Principles of Distributed Computing (PODC 2020)*. ACM Press, 299–308. https://doi.org/10.1145/3382734.3405715 arXiv:1902.06803

[7] Alkida Balliu, Keren Censor-Hillel, Yannic Maus, Dennis Olivetti, and Jukka Suomela. 2021. Locally Checkable Labelings with Small Messages. arXiv:2105.05574

[8] Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. 2018. New classes of distributed time complexity. In *Proc. 50th ACM Symposium on Theory of Computing (STOC 2018)*. ACM Press, 1307–1318. https://doi.org/10.1145/3188745.3188860 arXiv:1711.01871

[9] Alkida Balliu, Juho Hirvonen, Dennis Olivetti, and Jukka Suomela. 2019. Hardness of minimal symmetry breaking in distributed computing. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*. ACM Press, 369–378. https://doi.org/10.1145/3293611.3331605 arXiv:1811.01643

[10] Sebastian Brandt. 2019. An Automatic Speedup Theorem for Distributed Problems. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*. ACM, 379–388. https://doi.org/10.1145/3293611.3331611

[11] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. 2016. A lower bound for the distributed Lovász local lemma. In *Proc. 48th ACM Symposium on Theory of Computing (STOC 2016)*. ACM Press, 479–488. https://doi.org/10.1145/2897518.2897570 arXiv:1511.00900

[12] Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznański. 2017. LCL problems on grids. In *Proc. 36th ACM Symposium on Principles of Distributed Computing (PODC 2017)*. ACM Press, 101–110. https://doi.org/10.1145/3087801.3087833 arXiv:1702.05456

[13] Yi-Jun Chang. 2020. The Complexity Landscape of Distributed Locally Checkable Problems on Trees. In *Proc. 34th International Symposium on Distributed Computing (DISC 2020) (LIPIcs, Vol. 179)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 18:1–18:17. https://doi.org/10.4230/LIPIcs.DISC.2020.18

[14] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. 2019. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. *SIAM J. Comput.* 48, 1 (2019), 122–143. https://doi.org/10.1137/17M1117537

[15] Yi-Jun Chang and Seth Pettie. 2019. A Time Hierarchy Theorem for the LOCAL Model. *SIAM J. Comput.* 48, 1 (2019), 33–69. https://doi.org/10.1137/17M1157957

[16] Yi-Jun Chang, Jan Studený, and Jukka Suomela. 2021. Distributed graph problems through an automata-theoretic lens. In *Proc. 28th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2021) (LNCS)*. Springer. arXiv:2002.07659

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

[18] Richard Cole and Uzi Vishkin. 1986. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Inf. Control.* 70, 1 (1986), 32–53. https://doi.org/10.1016/S0019-9958(86)80023-7

[19] Manuela Fischer and Mohsen Ghaffari. 2017. Sublogarithmic Distributed Algorithms for Lovász Local Lemma, and the Complexity Hierarchy. In *Proc. 31st International Symposium on Distributed Computing (DISC 2017) (LIPIcs, Vol. 91)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 18:1–18:16. https://doi.org/10.4230/LIPIcs.DISC.2017.18

[20] Nathan Linial. 1992. Locality in Distributed Graph Algorithms. *SIAM J. Comput.* 21, 1 (1992), 193–201. https://doi.org/10.1137/0221015

[21] Moni Naor. 1991. A Lower Bound on Probabilistic Algorithms for Distributive Ring Coloring. *SIAM J. Discret. Math.* 4, 3 (1991), 409–412. https://doi.org/10.1137/0404036

[22] Moni Naor and Larry J. Stockmeyer. 1995. What Can be Computed Locally? *SIAM J. Comput.* 24, 6 (1995), 1259–1277. https://doi.org/10.1137/S0097539793254571

[23] Dennis Olivetti. 2020. Round Eliminator: a tool for automatic speedup simulation. https://github.com/olidennis/round-eliminator

[24] Václav Rozhoň and Mohsen Ghaffari. 2020. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proc. 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*. ACM, 350–363. https://doi.org/10.1145/3357713.3384298

[25] Jan Studený and Aleksandr Tereshchenko. 2021. Rooted Tree Classifier. https://github.com/jendas1/rooted-tree-classifier