Balliu, Alkida; Brandt, Sebastian; Chang, Yi-Jun; Olivetti, Dennis; Studený, Jan; Suomela, Jukka

Efficient Classification of Locally Checkable Problems in Regular Trees

Published in:
36th International Symposium on Distributed Computing (DISC 2022)

DOI:
10.4230/LIPIcs.DISC.2022.8

Published: 01/01/2022

Document Version
Publisher's PDF, also known as Version of record

Efficient Classification of Locally Checkable Problems in Regular Trees

Alkida Balliu
Gran Sasso Science Institute, L'Aquila, Italy

Sebastian Brandt
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Yi-Jun Chang
National University of Singapore, Singapore

Dennis Olivetti
Gran Sasso Science Institute, L'Aquila, Italy

Jan Studený
Aalto University, Espoo, Finland

Jukka Suomela
Aalto University, Espoo, Finland

Abstract

We give practical, efficient algorithms that automatically determine the asymptotic distributed round complexity of a given locally checkable graph problem in the $[\Theta(\log n), \Theta(n)]$ region, in two settings. We present one algorithm for unrooted regular trees and another algorithm for rooted regular trees. The algorithms take the description of a locally checkable labeling problem as input, and the running time is polynomial in the size of the problem description. The algorithms decide if the problem is solvable in $O(\log n)$ rounds. If not, it is known that the complexity has to be $\Theta(n^{1/k})$ for some $k = 1, 2, \ldots$, and in this case the algorithms also output the right value of the exponent k.

In rooted trees in the $O(\log n)$ case we can then further determine the exact complexity class by using algorithms from prior work; for unrooted trees the more fine-grained classification in the $O(\log n)$ region remains an open question.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases locally checkable labeling, locality, distributed computational complexity

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.8

Related Version *Full Version*: https://arxiv.org/abs/2202.08544

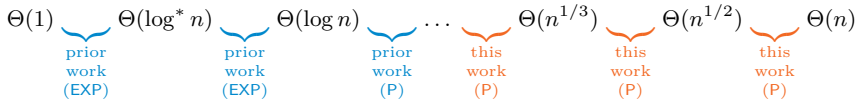
1 Introduction

We give practical, efficient algorithms that automatically determine the asymptotic distributed round complexity of a given *locally checkable* graph problem in *rooted or unrooted regular trees* in the $[\Theta(\log n), \Theta(n)]$ region, for both LOCAL and CONGEST models, see Section 3 for the precise definitions. In these cases, the distributed round complexity of any locally checkable problem is known to fall in one of the classes shown in Figure 1 [22, 21, 11, 20, 14, 31, 12]. Our algorithms can distinguish between all higher complexity classes from $\Theta(\log n)$ to $\Theta(n)$.

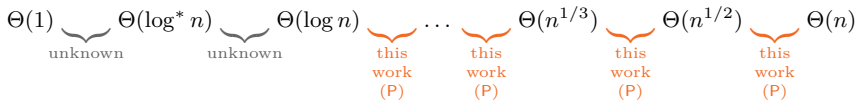
1.1 State of the art

Since 2016, there has been a large body of work studying the possible complexities of LCL problems. After an impressive sequence of works, the complexity landscape of LCL problems on bounded-degree general graphs, trees, and paths is now well-understood. For example,

(a) Rooted regular trees in deterministic and randomized CONGEST and LOCAL:



(b) Unrooted regular trees in deterministic CONGEST and LOCAL:



(c) Unrooted regular trees in randomized CONGEST and LOCAL:

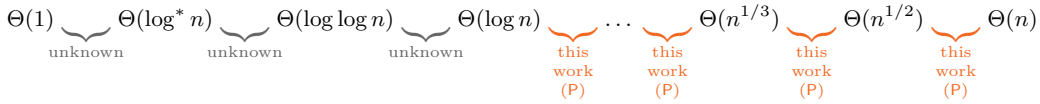


Figure 1 The most efficient algorithms for the classification of distributed round complexities. In the figure we show all possible complexity classes. Each gap between two classes corresponds to a natural decision problem: given a locally checkable problem, determine on which side of the gap its complexity is. For each gap we indicate whether a practical algorithm was provided already by prior work [9], whether it is first presented in this work, or whether the existence of such a routine is still an open question. The figure also indicates whether the algorithms are in P (polynomial time in the size of the problem description) or in EXP (exponential time in the size of the problem description).

it is known that there are no LCLs with deterministic complexity between $\omega(\log^* n)$ and $o(\log n)$. The proofs of some of the complexity gaps implies that the design of asymptotically optimal distributed algorithms can be *automated* in certain settings, leading to a series of research studying the computational complexity of automated design of asymptotically optimal distributed algorithms. See Section 2 for more details.

The most recent paper [9] in this line of research presented an algorithm that takes as input the description of an LCL problem defined in *rooted regular trees* and classifies the problem into one of the four complexity classes $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, and $n^{\Theta(1)}$. The classification applies to both the LOCAL and CONGEST models of distributed computing, both for randomized and deterministic algorithms.

To illustrate the setting of locally checkable problems in rooted regular trees, consider, for example, the following problem, which is meaningful for rooted binary trees:

> Each node is labeled with 1 or 2. If the label of an internal node is 1, exactly one of its two children must have label 1, and if the label of an internal node is 2, both of its children must have label 1.

We can represent it in a concise manner as a problem $\mathcal{C} = \{1 : 12, 2 : 11\}$, where $a : bc$ indicates that a node of label a can have its two children labeled with b and c, in some order. We can take such a description, feed it to the algorithm from [9], and it will output that this problem requires $\Theta(\log n)$ rounds in order to be solved in a rooted tree with n nodes.

1.2 What was missing

What the prior algorithm from [9] can do is classifying a given problem into one of the four main complexity classes $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, and $n^{\Theta(1)}$. However, if the complexity is $n^{\Theta(1)}$, we do not learn whether its complexity is, say, $\Theta(n)$ or $\Theta(\sqrt{n})$ or maybe $\Theta(n^{1/10})$.

There are locally checkable problems of complexity $\Theta(n^{1/k})$ for every $k = 1, 2, \ldots$, and there have not been any *practical* algorithm that would determine the value of the exponent k for any given problem.

Furthermore, the algorithm from [9] is only applicable in rooted regular trees, while the case of unrooted trees is perhaps even more interesting.

It has been known that the problem of distinguishing between e.g. $\Theta(n)$ and $\Theta(\sqrt{n})$ is *in principle* decidable, due to the algorithm of [20]. This algorithm is, however, best seen as a theoretical construction. To the best of our knowledge, nobody has implemented it, there are no plans of implementing it, and it seems unlikely that one could classify any nontrivial problem with it using any real-world computer, due to its doubly exponential time complexity. This is the missing piece that we provide in this work.

1.3 Contributions and motivations

We present polynomial-time algorithms that determine not only whether the round complexity of a given LCL problem is $\Theta(n^{1/k})$ for some k, but they also determine the exact value of k. We give one algorithm for the case of unrooted trees and one algorithm for the case of rooted trees.

Our algorithms not only determine the asymptotic round complexity, but they also output a description of a distributed algorithm attaining this complexity. If the given LCL problem Π has optimal complexity $\Theta(n^{1/k})$, then our algorithms will output a description of a deterministic distributed algorithm that solves Π in $O(n^{1/k})$ rounds in the CONGEST model. Similarly, if the given LCL problem Π has optimal complexity $O(\log n)$, then our algorithms will output a description of a deterministic distributed algorithm that solves Π in $O(\log n)$ rounds in the CONGEST model.

We have implemented both algorithms for the case of 3-regular trees, the proof-of-concept implementations are freely available online,[1] and they work fast also in practice.

From a practical point of view, together with prior work from [9], there is now a practical algorithm that is able to *completely determine* the complexity of any LCL problem in rooted regular trees.[2] In the case of unrooted regular trees deciding between the lower complexity classes below $o(\log n)$ remains an open question.

From a theoretical point of view, this work *significantly expands* the class of LCL problems whose optimal complexity is known to be decidable in polynomial time. See Figure 1 for a summary of the current state of the art on the classification of LCL complexities for regular trees, showing where the new algorithms are applicable and where the state of the art is given by existing results.

We note that the problem of determining the optimal complexity of an LCL problem is computationally hard in general: It is undecidable in general [37], EXPTIME-hard even for bounded-degree trees [20], and PSPACE-hard even for paths and cycles with input labels [2]. Hence, in order to understand whether polynomial-time algorithms are even possible, we must restrict our consideration to restricted cases, such as LCLs with no inputs defined on regular trees. In fact, it is known that it is possible to use LCLs with no inputs defined on non-regular trees to encode LCLs with inputs, and hence, by allowing inputs, or constraints that depend on the degree of the nodes, we would make decidability at least PSPACE-hard.

[1] `https://github.com/jendas1/poly-classifier`

[2] Even though some algorithms in [9] are exponential in the size of the description of the problem, they are nevertheless very efficient in practice. In fact, the authors of [9] have implemented them for the case of binary rooted trees and they are indeed very fast in practice [41].

Motivations. Studying LCLs is interesting because, on the one hand, this class of problems is large enough to contain a significant fraction of problems that are commonly studied in the context of the LOCAL model (e.g., $(\Delta+1)$-coloring, $(2\Delta-1)$-edge coloring, Δ-coloring, weak 2-coloring, maximal matching, maximal independent set, sinkless orientation, many other orientation problems, edge splitting problems, locally maximal cut, defective colorings, ...), but, on the other hand, it is restricted enough so that we can prove interesting results about them, such as decidability and complexity gaps. Moreover, techniques used to prove results on LCLs have been already shown to be extremely useful outside the LCL context: for example, all recent results about lower bounds for locally checkable problems in the unbounded degree case – e.g., for MIS, maximal matching, ruling sets, and other fundamental problems – use techniques that originally were introduced in the context of LCLs [5, 19, 8, 6, 7].

In this work, we restrict our attention to the case of regular trees. The study of LCLs on trees is related with our understanding of graph problems in the general setting. Actually, for many problems of interest, unrooted regular trees are hard instances, and hence understanding the complexity of LCLs on trees could help us in understanding the complexity of problems in general unbounded-degree graphs. In fact, a relatively new and promising technique called *round elimination* has been used to prove tight lower bounds for interesting graph problems such as maximal matchings, maximal independent sets, and ruling sets, even if, for now, we are only able to apply this technique for proving lower bounds on trees [15, 38, 5, 8, 4, 19, 6, 7].

As for the more restrictive setting of *regular* trees, we would like to point out that many natural LCL problems have the same optimal complexity in both bounded-degree trees and regular trees. This includes, for example, the k-coloring problem. For any tree T whose maximum degree is at most Δ, we may consider the Δ-regular tree T^* which is the result of appending degree-1 nodes to all nodes v in T with $1 < \deg(v) < \Delta$ to increase the degree of v to Δ. We may locally simulate T^* in the network T. As any proper k-coloring of T^* restricting to T is also a proper k-coloring, this reduces the k-coloring problem on bounded-degree trees to the same problem on regular trees, showing that the k-coloring problem has the same optimal complexity in both graph classes. More generally, if an LCL problem Π has the property that removing degree-1 nodes preserves the correctness of a solution, then Π has the same optimal complexity in both bounded-degree trees and regular trees, so our results in this work also apply to these LCLs on bounded-degree trees.

2 Related work

Locally Checkable Labeling problems have been introduced by Naor and Stockmeyer [37], but the class of locally checkable problems has been studied in the distributed setting even before (e.g., in the context of self-stabilisation [1]). For many locally checkable problems, researchers have been trying to understand the exact time complexity, and while in many cases upper bounds have been known since the 80s, matching lower bound have been discovered only recently. Examples of this line of research relate to the problems of colorings, matchings, and independent sets, see e.g. [25, 32, 33, 39, 28, 26, 5, 40, 34, 7, 29].

In parallel, there have been many works that tried to understand these problems from a *complexity theory* point of view, trying to develop general techniques for classifying problems, understanding which complexities can actually exist, and developing generic algorithmic techniques to solve whole classes of problems at once. In particular, a broad class[3] of locally checkable problems, called *Locally Checkable Labelings* (LCLs), has been studied in the LOCAL model of distributed computing, which will be formally defined later.

[3] For example, our definition of LCL does not allow an infinite number of labels, so it does not capture some locally checkable problems such as fractional matching.

Paths and cycles. The first graph topologies on which promising results have been proved are paths and cycles. In these graphs, we now know that there are problems with the following *three* possible time complexities:

- $O(1)$: this class contains, among others, trivial problems, e.g. problems that require every node to output the same label.
- $\Theta(\log^* n)$: this class contains, for example, the 3-coloring problem [25, 32, 36].
- $\Theta(n)$: this class contains hard problems, for example the problem of consistently orient the edges of a cycle, or the 2-coloring problem.

For LCLs in paths and cycles, we know that there are no other possible complexities, that is, there are *gaps* between the above classes. In other words, there are no LCLs with a time complexity that lies between $\omega(1)$ and $o(\log^* n)$ [37], and no LCLs with a time complexity that lies between $\omega(\log^* n)$ and $o(n)$ [21]. These results hold also for *randomized* algorithms, and they are constructive: if for example we find a way to design an $O(\log n)$-rounds randomized algorithm for a problem, then we can automatically convert it into an $O(\log^* n)$-round deterministic algorithm.

Moreover, in paths and cycles, given an LCL problem, we can *decide* its time complexity. In particular, it turns out that for problems with no inputs defined on directed cycles, deciding the complexity of an LCL is as easy as drawing a diagram and staring at it for few seconds [18]. This result has later been extended to undirected cycles with no inputs [23]. Unfortunately, as soon as we consider LCLs where the constraints of the problem may depend on the given inputs, decidability becomes much harder, and it is now known to be PSPACE-hard [2], even for paths and cycles.

Trees. Another class of graphs that has been studied quite a lot is the one containing *trees*. While there are still problems with complexities $O(1)$, $\Theta(\log^* n)$, and $\Theta(n)$, there are also additional complexity classes, and sometimes here randomness can help. For example, there are problems that require $\Theta(\log n)$ rounds for both deterministic and randomized algorithms, while there are problems, like *sinkless orientation*, that require $\Theta(\log n)$ rounds for deterministic algorithms and $\Theta(\log \log n)$ rounds for randomized ones [17, 21, 30]. Moreover, there are problems with complexity $\Theta(n^{1/k})$, for any natural number $k \geq 1$ [22]. It is known that these are the only possible time complexities in trees [22, 21, 11, 20, 14, 31]. In [12], it has been shown that the same results hold also in a more restrictive model of distributed computing, called CONGEST model, and that for any given problem, its complexities in the LOCAL and in the CONGEST model, on trees, are actually the same.

Concerning decidability, the picture is not as clear as in the case of paths and cycles. As discussed in the introduction, it is decidable, *in theory*, if a problem requires $n^{\Omega(1)}$ rounds, and, in that case, it is also decidable to determine the exact exponent [22, 20], but the algorithm is very far from being practical, and in this work we address exactly this issue. Moreover, for *lower* complexities, the problem is still open. Different works tried to tackle this issue by considering restricted cases. In [4], authors showed that it is indeed possible to achieve decidability in some cases, that is, when problems are restricted to the case of unrooted regular trees, where leaves are unconstrained, and the problem uses only *two* labels. Then, promising results have been achieved in [9], where it has been shown that, if we consider *rooted* trees, then we can decide the complexity of LCLs even for $n^{o(1)}$ complexities. Unfortunately, it is very unclear if such techniques can be used to solve the problem in the general case. In fact, we still do not know if it is decidable whether a problem can be solved in $O(1)$ rounds or it requires $\Omega(\log^* n)$ rounds, and it is not known if it is decidable whether a problem can be solved in $O(\log^* n)$ rounds or it requires $\Omega(\log n)$ for

deterministic algorithms and $\Omega(\log \log n)$ for randomized ones. These two questions are very important, and understanding them may also help in understanding problems that are not restricted to regular trees of bounded degree. This is because, as already mentioned before, for many problems it happens that unrooted regular trees are hard instances, and studying the complexity of problems in these instances may give insights for understanding problems in the general setting.

General graphs. In general graphs, many more LCL complexities are possible. For example, there is a gap similar to the one between $\omega(1)$ and $o(\log^* n)$ of trees, but now it holds only up to $o(\log \log^* n)$, and we know that there are problems in the region between $\Omega(\log \log^* n)$ and $o(\log^* n)$. In fact, for any rational $\alpha \geq 1$, it is possible to construct problems with complexity $\Theta(\log^\alpha \log^* n)$ [13]. A similar statement holds for complexities between $\Omega(\log n)$ and $O(n)$ [13, 11].

There are still complexity regions in which we do not know if there are problems or not. For example, while it is known that any problem that has randomized complexity $o(\log n)$ can be sped up to $O(T_{\mathrm{LLL}})$ [22], where T_{LLL} is the distributed complexity of the constructive version of the Lovász LOCAL Lemma, the exact value of T_{LLL} is unknown, and we only known that it lies between $\Omega(\log \log n)$ and $O(\mathrm{poly} \log \log n)$ [17, 24, 27, 40]. Another problem that falls in this region is the Δ-coloring problem, for which we still do not know the exact complexity.

Another open question regards the role of randomness. In general graphs, we know that randomness can also help outside the $O(\log n)$ region [10], but we still do not know exactly when it can help and how much.

In general graphs, unfortunately, determining the complexity of a given LCL problem is undecidable. In fact, we know that this question is undecidable even on grids [37].

3 Preliminaries

Graphs. Let $G = (V, E)$ be a graph. We denote with $n = |V|$ the number of nodes of G, with Δ the maximum degree of G, and with $\deg(v)$, for $v \in V$, the degree of v. If G is a directed graph, we denote with $\deg_{\mathsf{in}}(v)$ and $\deg_{\mathsf{out}}(v)$, the indegree and the outdegree of v, respectively. The radius-r neighborhood of a node v is defined to be the subgraph of G induced by the nodes at distance at most r from v.

Model of computing. In the LOCAL model of distributed computing, the network is represented with a graph $G = (V, E)$, where the nodes correspond to computational entities, and the edges correspond to communication links. In this model, the computational power of the nodes is unrestricted, and nodes can send arbitrarily large messages to each other.

This model is synchronous, and computation proceeds in rounds. Nodes all start the computation at the same time, and at the beginning they know n (the total number of nodes), Δ (the maximum degree of the graph), and a unique ID in $\{1, \ldots, n^c\}$, for some constant $c \geq 1$, assigned to them. Then, the computation proceeds in rounds, and at each round nodes can send (possibly different) messages to each neighbor, receive messages, and perform some LOCAL computation.

At the end of the computation, each node must produce its own part of the solution. For example, in the case of the $(\Delta + 1)$-coloring problem, each node must output its own color, that must be different from the ones of its neighbors. The time complexity is measured as the worst case number of rounds required to terminate, and it is typically expressed as a function of n, Δ, and c.

4 Technical overview

Our new results build on several techniques developed in previous works [9, 23] designing polynomial-time algorithms that determine the distributed complexity of LCL problems. In this section, we first give a brief overview of these techniques, then we discuss how in this paper we build upon them and obtain our new results. The aim of this section is to present the intuition behind the results. To keep the discussion at a high level, the presentation here will be a bit imprecise.

4.1 The high-level framework

Existing algorithms for deciding the complexity of a given LCL problem are often based on the following approach.

1. Define some combinatorial property P of LCL problems.
2. Show that computing $P(\Pi)$ for a given problem Π can be done efficiently.
3. Show that Π is in a certain complexity class if and only if $P(\Pi)$ holds.

As discussed in [18, 23], any LCL Π on directed paths can be viewed as a regular language. Taking the corresponding non-deterministic automaton, we obtain a directed graph $G(\Pi)$ that represents Π on directed paths.

For example, the maximal independent set problem can be described as the automaton with states $V = \{00, 01, 10\}$ and transitions $E = \{00 \to 01, 01 \to 10, 10 \to 00, 10 \to 01\}$. Each state corresponds to a possible labeling of the two endpoints u and v of a directed edge $u \to v$. Each transition describes a valid configuration of two neighboring directed edges $u \to v$ and $v \to w$.

It has been shown [9, 18, 16, 23] that in several cases the distributed complexity of an LCL can be characterized by simple graph properties of $G(\Pi)$, even if the underlying graph class is much more complicated than directed paths. The precise definition of $G(\Pi)$ will depend on the choice of the LCL formalism.

4.2 Paths and cycles

It was shown in [18, 23] that the distributed complexity and solvability of Π on paths and cycles can be characterized by simple graph properties of $G(\Pi)$. In particular, Π on directed cycles is solvable in $O(\log^* n)$ rounds if and only if $G(\Pi)$ contains a node v that is *path-flexible*, in the sense that there exists a number K such that, in $G(\Pi)$, there is a length-k returning walk for v, for each $k \geq K$. If such a path-flexible node v exists in $G(\Pi)$, then Π on directed cycles can be solved in $O(\log^* n)$ rounds in the following manner.

1. In $O(\log^* n)$ rounds, compute an independent set I such that the distance between the nodes in I is at least K and at most $2K$.
2. Fix the labels for the nodes in I according to the path-flexible node v in $G(\Pi)$.
3. By the path-flexibility of v, this partial labeling can be completed into a correct complete labeling.

For example, in the automaton for maximal independent set described above, the state 01 is flexible, as for each $k \geq 5$, there is a length-k walk starting and ending at 01, so a maximal independent set can be found in $O(\log^* n)$ rounds on directed cycles via the above algorithm.

The above characterization can be generalized to both paths and cycles, undirected and directed, after some minor modifications, see [23] for the details. For further examples of representing LCLs as automata and how the round complexity of an LCL can be inferred from basic properties of its associated automaton, see [18, Fig. 3] and [23, Fig. 1 and 3].

4.3 The $O(\log n)$ complexity class in regular trees

Subsequently, it was shown in [9, 16] that the class of $O(\log n)$-round solvable LCL problems on rooted and unrooted regular trees can be characterized in a similar way, based on the notion of path-flexibility in the directed graph $G(\Pi)$. To keep the discussion at a high level, we do not discuss the difference between rooted and unrooted trees here. Roughly speaking, Π can be solved in $O(\log n)$ rounds on rooted or unrooted regular trees if and only if there exists a subset of labels S such that, if we restrict Π to S, then its corresponding directed graph is strongly connected and contains a path-flexible node. Such a set S of labels is also called a *certificate* for $O(\log n)$-round solvability.[4]

A key property of such a directed graph is that there exists a number K such that, for each pair of nodes (u, v), and for each integer $k \geq K$, there is a length-k walk from u to v (here we allow the possibility of $u = v$). The property can be described in the following more intuitive manner. For any path of length at least K, regardless of how we fix the labels of its two endpoints using S, it is always possible to complete the partial labeling into a correct labeling w.r.t. Π of the entire path using only labels in S.

The intuition behind such a characterization is the fact [22] that all LCLs solvable in $O(\log n)$ rounds on bounded-degree trees can be solved in a canonical way based on *rake-and-compress decompositions*. Roughly speaking, a rake-and-compress process is a procedure that decomposes a tree by iteratively removing degree-1 nodes (rake) and removing degree-2 nodes (compress). This process partitions the set of nodes into several parts: $V = V_1^{\mathsf{R}} \cup V_1^{\mathsf{C}} \cup V_2^{\mathsf{R}} \cup V_2^{\mathsf{C}} \cup \cdots \cup V_L^{\mathsf{R}}$, where V_i^{R} is the set of nodes removed by the rake operation in the ith iteration and V_i^{C} is the set of nodes removed by the compress operation in the ith iteration. It can be shown that $L = O(\log n)$ [35].

There are several variants of a rake-and-compress process. Here the considered variant is such that, in the compress operation, a degree-2 node v is removed if v belongs to a path whose length is at least ℓ, so we may assume that the connected components in the subgraph induced by V_i^{C} are paths with length at least ℓ.

Let Π be any LCL problem satisfying the combinatorial characterization for $O(\log n)$-round solvability discussed above, and let the set of labels S be a certificate for $O(\log n)$-round solvability. By setting $\ell = K$ in the property of the combinatorial characterization, we may obtain an $O(\log n)$-round algorithm solving the given LCL problem Π using only the labels in S. The high-level idea is that we can label the tree in an order that is the reverse of the one of the rake-and-compress procedure: $V_L^{\mathsf{R}}, \ldots, V_2^{\mathsf{C}}, V_2^{\mathsf{R}}, V_1^{\mathsf{C}}, V_1^{\mathsf{R}}$, as we observe that the property of the combinatorial characterization discussed above ensures that any correct labeling of $V_L^{\mathsf{R}} \cup \cdots \cup V_i^{\mathsf{R}}$ can be extended to a correct labeling of $V_L^{\mathsf{R}} \cup \cdots \cup V_i^{\mathsf{R}} \cup V_{i-1}^{\mathsf{C}}$ and similarly any correct labeling of $V_L^{\mathsf{R}} \cup \cdots \cup V_i^{\mathsf{C}}$ can be extended to a correct labeling of $V_L^{\mathsf{R}} \cup \cdots \cup V_i^{\mathsf{C}} \cup V_i^{\mathsf{R}}$.

The requirement that Π is an LCL problem defined on *regular* trees is *critical* in the above approach, as this requirement ensures that for each non-leaf node, the set of constraints is the same, so we do not need to worry about the possibility for different nodes in the tree to have different sets of constraints in Π. Indeed, if we allow nodes of different degrees to have different sets of constraints, then the problem of determining the distributed complexity of an LCL in bounded-degree trees becomes EXPTIME-hard [20].

[4] Although the certificate described in [9] also includes the steps in the construction of S, the set S alone suffices to certify that Π can be solved in $O(\log n)$ rounds, as the $O(\log n)$-round algorithm described in [9] uses only S.

4.4 The polynomial complexity region in regular trees

In this work, we will extend the above approach to cover all complexity classes in the $[\Theta(\log n), \Theta(n)]$ region. By [11, 20, 22], we know that the possible complexity classes in this region are $\Theta(\log n)$ and $\Theta(n^{1/k})$ for all positive integers k. Similar to the complexity class $O(\log n)$, any LCL problem Π solvable in $O(n^{1/k})$ rounds can be solved in a canonical way in $O(n^{1/k})$ rounds using a variant of rake-and-compress decomposition [20].

Specifically, Π is $O(n^{1/k})$-round solvable if and only if it can be solved in a canonical way using a rake-and-compress decomposition, where in each iteration, we perform $\gamma = O(n^{1/k})$ rake operations and one compress operation. Similar to the case of complexity class $O(\log n)$, in the compress operation, a degree-2 node v is removed if v belongs to a path whose length is at least ℓ, where $\ell = O(1)$ is some sufficiently large number depending only on the LCL problem Π. It can be shown [20] that by selecting $\gamma = O(n^{1/k})$ to be large enough, the number of layers L in the decomposition $V = V_1^{\mathsf{R}} \cup V_1^{\mathsf{C}} \cup V_2^{\mathsf{R}} \cup V_2^{\mathsf{C}} \cup \cdots \cup V_L^{\mathsf{R}}$ is k, and such a decomposition can be computed in $O(n^{1/k})$ rounds.

To derive a certificate for $O(n^{1/k})$-round solvability based on the result of [20], we will need to take into consideration the following properties about the variant of the rake-and-compress decomposition described above.

- The number of layers $L = k$ is now a *finite* number independent of the size of the graph n. For technical reasons, this means that the certificate for $O(n^{1/k})$-round solvability cannot be based on a single set of labels S, as the certificate for $O(\log n)$-round solvability [9, 16]. We need to consider the possibility that different sets of labels are used for different layers in the design of the certificate for $O(n^{1/k})$-round solvability.
- The number of rake operations for a layer can be unbounded as n goes to infinity. That is, V_i^{R} is no longer an independent set, and each connected component in the subgraph induced by V_i^{R} can be a very large tree.

The certificate. Our certificate for $O(n^{1/k})$-round solvability will be based on the notion of a *good* sequence of sets of labels. The definition of a good sequence relies on two functions on a set of labels: trim and flexible-SCC. As we will later see, these two functions correspond to rake and compress, respectively. Given an LCL problem Π and a set of labels S, $\mathsf{trim}(S)$ and flexible-SCC are defined as follows.

- $\mathsf{trim}(S)$ is the subset of S resulting from removing all labels $\sigma \in S$ meeting the following conditions: There exists some number i such that if the root of the complete regular tree T of height i is labeled by σ, then we are not able to complete the labeling of T using only labels in S such that the overall labeling is correct w.r.t. Π.
- flexible-SCC(S) is a collection of disjoint subsets of S defined as follows. Consider the directed graph representing the LCL problem Π restricted to S. Let flexible-SCC(S) be the set of strongly connected components that have a path-flexible node. The intuition behind this definition is similar to the intuition behind the certificate for $O(\log n)$-round solvability.

We briefly explain the connection between trim and rake. Suppose we want to find a correct labeling of a regular tree T using only the labels in S. If a label σ is in $\mathsf{trim}(S)$, then σ can only be used in places that are sufficiently close to a leaf. To put it another way, if we do a large number of rakes to T, then the labels in $\mathsf{trim}(S)$ can only be used to label the nodes that removed due to a rake operation.

The connection between flexible-SCC to compress is due to the fact that the nodes removed due to a compress operation form long paths, and we know that in order to label long paths efficiently in $O(\log^* n)$ rounds, it is necessary to use labels corresponding to path-flexible nodes, due to the existing automata-theoretic characterization [18, 23] of round complexity of LCLs on paths and cycles.

We say that a sequence $(\Sigma_1^{\mathsf{R}}, \Sigma_1^{\mathsf{C}}, \Sigma_2^{\mathsf{R}}, \Sigma_2^{\mathsf{C}}, \ldots, \Sigma_k^{\mathsf{R}})$ is *good* if it satisfies the following rules, where Σ is the set of all labels of Π.

$$\Sigma_i^{\mathsf{R}} = \begin{cases} \mathsf{trim}(\Sigma) & \text{if } i = 1, \\ \mathsf{trim}(\Sigma_{i-1}^{\mathsf{C}}) & \text{if } i > 1. \end{cases}$$

$\Sigma_i^{\mathsf{C}} \in \mathsf{flexible\text{-}SCC}(\Sigma_i^{\mathsf{R}})$.

$\Sigma_k^{\mathsf{R}} \neq \emptyset$.

The only nondeterminism in the above rules is the choice of $\Sigma_i^{\mathsf{C}} \in \mathsf{flexible\text{-}SCC}(\Sigma_i^{\mathsf{R}})$ for each i. We will show that such a sequence exists if and only if the underlying LCL problem can be solved in $O(n^{1/k})$ rounds. Intuitively, Σ_i^{R} represents the set of labels that are eligible to label the nodes in V_i^{R}, and similarly Σ_i^{C} represents the set of labels that are eligible to label the nodes in V_i^{C}.

The classification. The notion of a good sequence allows us to classify the complexity classes in the region $[\Theta(\log n), \Theta(n)]$. Specifically, we define the *depth* d_Π of an LCL problem Π as the largest k such that a good sequence $(\Sigma_1^{\mathsf{R}}, \Sigma_1^{\mathsf{C}}, \Sigma_2^{\mathsf{R}}, \Sigma_2^{\mathsf{C}}, \ldots, \Sigma_k^{\mathsf{R}})$ exists. If there is no good sequence, then we set $d_\Pi = 0$. If there is a good sequence $(\Sigma_1^{\mathsf{R}}, \Sigma_1^{\mathsf{C}}, \Sigma_2^{\mathsf{R}}, \Sigma_2^{\mathsf{C}}, \ldots, \Sigma_k^{\mathsf{R}})$ for each positive integer k, then we set $d_\Pi = \infty$. We will show that d_Π characterizes the distributed complexity of Π in the following manner.

- If $d_\Pi = 0$, then Π is unsolvable in the sense that there exists a regular tree such that there is no correct solution of Π on this rooted tree. This follows from the definition of trim and the observation that $d_\Pi = 0$ if $\mathsf{trim}(\Sigma) = \emptyset$.
- If $d_\Pi = k$ is a positive integer, then the distributed complexity of Π is $\Theta(n^{1/k})$.
- If $d_\Pi = \infty$, then Π can be solved in $O(\log n)$ rounds. If we can have a good sequence that is arbitrarily long, then there must be a *fixed point* S in the sequence such that $\mathsf{trim}(S) = S$ and $\mathsf{flexible\text{-}SCC}(S) = \{S\}$, because $\Sigma_1^{\mathsf{R}} \supseteq \Sigma_1^{\mathsf{C}} \supseteq \cdots \supseteq \Sigma_k^{\mathsf{R}}$. We will show that the fixed point S qualifies to be a certificate for $O(\log n)$-round solvability.

The fixed point phenomenon explains why the notion of good sequence was not needed in [9, 16], as the existence of a fixed point for the case Π is $O(\log n)$-round solvable implies that we may apply the same strategy according to the fixed point to label each layer of the rake-and-compress decomposition to solve Π in $O(\log n)$ rounds.

To show correctness and efficiency of our characterization, we do the following.

Upper bound: Given a good sequence $(\Sigma_1^{\mathsf{R}}, \Sigma_1^{\mathsf{C}}, \Sigma_2^{\mathsf{R}}, \Sigma_2^{\mathsf{C}}, \ldots, \Sigma_k^{\mathsf{R}})$, show that there exists an $O(n^{1/k})$-round algorithm solving Π. Therefore, $d_\Pi = k$ implies $O(n^{1/k})$-round solvability.

Lower bound: Given an $o(n^{1/k})$-round algorithm solving Π, show that a good sequence $(\Sigma_1^{\mathsf{R}}, \Sigma_1^{\mathsf{C}}, \Sigma_2^{\mathsf{R}}, \Sigma_2^{\mathsf{C}}, \ldots, \Sigma_{k+1}^{\mathsf{R}})$ exists. Therefore, $d_\Pi = k$ implies $\Omega(n^{1/k})$-round solvability.

Efficiency: Design a polynomial-time algorithm that computes d_Π for any given description of an LCL problem Π.

The upper bound proof is relatively simple. Similar to the certificate $O(\log n)$-round solvability, we just need to show that Π can be solved in $O(n^{1/k})$ rounds using rake-and-compress decompositions given that a good sequence $(\Sigma_1^{\mathsf{R}}, \Sigma_1^{\mathsf{C}}, \Sigma_2^{\mathsf{R}}, \Sigma_2^{\mathsf{C}}, \ldots, \Sigma_k^{\mathsf{R}})$ exists.

The lower bound proof is much more complicated. Given an algorithm $\mathcal{A}$ solving Π in $t = o(n^{1/k})$ rounds, we will consider a tree G that is a result of a hierarchical combination of complete trees and paths of length greater than t. Intuitively, G is chosen to be the fullest possible tree that can be partitioned into $V = V_1^{\mathsf{R}} \cup V_1^{\mathsf{C}} \cup V_2^{\mathsf{R}} \cup V_2^{\mathsf{C}} \cup \cdots \cup V_{k+1}^{\mathsf{R}}$ with a rake-and-compress decomposition of [20] with $L = k + 1$ layers. We will prove by induction that if we take Σ_i^{R} to be the set of possible output labels of $\mathcal{A}$ for $V_i^{\mathsf{R}} \cup V_i^{\mathsf{C}} \cup \cdots \cup V_{k+1}^{\mathsf{R}}$ and take Σ_i^{C} to be the set of possible output labels of $\mathcal{A}$ for $V_i^{\mathsf{C}} \cup V_{i+1}^{\mathsf{R}} \cup \cdots \cup V_{k+1}^{\mathsf{R}}$, then $(\Sigma_1^{\mathsf{R}}, \Sigma_1^{\mathsf{C}}, \Sigma_2^{\mathsf{R}}, \Sigma_2^{\mathsf{C}}, \ldots, \Sigma_{k+1}^{\mathsf{R}})$ must be a good sequence. In particular, the non-emptiness of $\Sigma_{k+1}^{\mathsf{R}}$ follows from the correctness of $\mathcal{A}$.

To design a polynomial-time algorithm computing d_Π, we recall that the only nondeterminism in the rules for a good sequence is the choice of $\Sigma_i^{\mathsf{C}} \in \mathsf{flexible\text{-}SCC}(\Sigma_i^{\mathsf{R}})$, so we will just do a brute-force search for all possibilities. Although this seems inefficient, we recall that $\mathsf{flexible\text{-}SCC}(\Sigma_i^{\mathsf{R}})$ is a collection of disjoint subsets of Σ_i^{R}, so the sum of the size of all sets of labels considered in each level is at most the total number of labels $|\Sigma|$ in Π. The number of levels we need to explore is also bounded, as $\Sigma_1^{\mathsf{R}} \supseteq \Sigma_1^{\mathsf{C}} \supseteq \cdots \supseteq \Sigma_k^{\mathsf{R}}$. If k exceeds $|\Sigma|$, then we know that there is a fixed point Σ_i^{R} such that $\Sigma_i^{\mathsf{R}} = \Sigma_i^{\mathsf{C}} = \Sigma_{i+1}^{\mathsf{R}} = \Sigma_{i+1}^{\mathsf{C}} = \cdots$, so $d_\Pi = \infty$.

The differences between rooted and unrooted trees. The high-level proof strategy presented in this technical overview applies to both rooted and unrooted regular trees, showing that these two graph classes behave very similarly in the complexity region $[\Theta(\log n), \Theta(n)]$. There are still some technical differences between rooted and unrooted trees.

- The formalisms for representing LCL problems are different for rooted and unrooted trees. In the case of rooted trees, the problem can refer to orientations. For example, what is permitted for a parent can be different from what is permitted for a child. Instead of specifying node and edge configurations, we follow [9] and specify what are permitted multisets of child labels for each node label.
- For the upper bound, we need to generalize the rake-and-compress decomposition of [20] so that it is applicable in rooted trees.
- For the lower bound, the lower bound graph for unrooted trees does not work for the rooted trees. Roughly speaking, this is because the presence of edge orientation increases the symmetry breaking capability of nodes, so some indistinguishability arguments in the lower bound proof for unrooted trees do not work for rooted trees. Therefore, we will need to consider a different approach for crafting the lower bound graph for rooted trees.

5 Unrooted trees

In this section, we give a polynomial-time-computable characterization of LCL problems for regular unrooted trees with complexity $O(\log n)$ or $\Theta(n^{1/k})$ for any positive integer k. Due to the page limit, our results in regular rooted trees are left to the full version [3] of the paper. A Δ-regular tree is a tree where the degree of each node is either 1 or Δ. An LCL problem for Δ-regular unrooted trees is defined as follows.

▶ **Definition 1** (LCL problems for regular unrooted trees). *For unrooted trees, an* LCL *problem* $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ *is defined by the following components.*

- Δ *is a positive integer specifying the maximum degree.*
- Σ *is a finite set of labels.*
- $\mathcal{V}$ *is a set of size-*Δ *multisets of labels in* Σ *specifying the node constraint.*
- $\mathcal{E}$ *is a set of size-*2 *multisets of labels in* Σ *specifying the edge constraint.*

We call a size-Δ multiset C of labels in Σ a *node configuration*. A node configuration C is correct with respect to $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ if $C \in \mathcal{V}$. We call a size-2 multiset D of labels in Σ an *edge configuration*. An edge configuration D is correct with respect to $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ if $D \in \mathcal{E}$. We define the correctness criteria for a labeling of a Δ-regular tree in Definition 2.

▶ **Definition 2** (Correctness criteria). *Let $G = (V, E)$ be a tree whose maximum degree is at most Δ. For each edge $e = \{u, v\} \in E$, there are two half-edges (u, e) and (v, e). A solution of $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ on G is a labeling that assigns a label in Σ to each half-edge in G.*

- *For each node $v \in V$ with $\deg(v) = \Delta$ its node configuration C is the multiset of Δ half-edge labels of (v, e_1), (v, e_2), …, (v, e_Δ), where $e_1, e_2, \ldots, e_\Delta$ are the Δ edges incident to v. We say that the labeling is locally-consistent on v if $C \in \mathcal{V}$.*
- *For each edge $e = \{u, v\} \in E$, its edge configuration D is the multiset of two half-edge labels of (u, e) and (v, e). We say that the labeling is locally-consistent on e if $D \in \mathcal{E}$.*

The labeling is a correct solution if it is locally-consistent on all $v \in V$ with $\deg(v) = \Delta$ and all $e \in E$.

In other words, a labeling of $G = (V, E)$ is correct if the edge configuration for each $e \in E$ is correct and the node configuration for each $v \in V$ with $\deg(v) = \Delta$ is correct. All nodes whose degree is not Δ are unconstrained.

Although $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ is defined for Δ-regular unrooted trees, Definition 2 applies to all trees whose maximum degree is at most Δ. We emphasize that all nodes v whose degree is not Δ are *unconstrained* in that there is no requirement about the node configuration of v. Nevertheless, we may focus on Δ-regular unrooted trees without loss of generality. The reason is that for any unrooted tree G whose maximum degree is at most Δ, we may consider the unrooted tree G^* which is the result of appending degree-1 nodes to all nodes v in G with $1 < \deg(v) < \Delta$ to increase the degree of v to Δ. This only blows up the number of nodes by at most a Δ factor. We claim that the asymptotic optimal round complexity of Π is the same in both G and G^*. Any correct solution of Π on G^* restricted to G is a correct solution of Π on G, as all nodes whose degree is not Δ are unconstrained. Therefore, if we have an algorithm for Π in Δ-regular unrooted trees, then the same algorithm also allows us to solve Π in unrooted trees with maximum degree Δ in the same asymptotic round complexity.

▶ **Definition 3** (Complete trees of height i). *We define the rooted trees T_i and T_i^* recursively as follows.*

- *T_0 is the trivial tree with only one node.*
- *T_i is the result of appending $\Delta - 1$ trees T_{i-1} to the root r.*
- *T_i^* is the result of appending Δ trees T_{i-1} to the root r.*

Observe that T_i^* is the unique maximum-size tree of maximum degree Δ and height i. All nodes within distance $i - 1$ to the root r in T_i^* have degree Δ. All nodes whose distance to r is exactly i are degree-1 nodes. Although T_i and T_i^* are defined as rooted trees, they can also be viewed as unrooted trees.

▶ **Definition 4** (Trimming). *Given an* LCL *problem $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ and a subset $\mathcal{S} \subseteq \mathcal{V}$ of node configurations, we define $\mathsf{trim}(\mathcal{S})$ as the set of all node configurations $C \in \mathcal{S}$ such that for each $i \geq 1$ it is possible to find a correct labeling of T_i^* such that the node configuration of the root is C and the node configurations of the remaining degree-Δ nodes are in $\mathcal{S}$.*

In the definition, note that if for some $i \geq 1$ it is not possible to find such a labeling of T_i^*, then it is also not possible for any larger i. The reason is that if such a labeling for larger i exists, then by taking subgraph, we obtain such a labeling for of T_i^*. Here we use the fact that nodes by taking subgraph, and using the fact that all nodes whose degree is not Δ are unconstrained.

Intuitively, $\mathsf{trim}(\mathcal{S})$ is the subset of $\mathcal{S}$ resulting from removing all node configurations in $\mathcal{S}$ that are not usable in a correct labeling of a sufficiently large Δ-regular tree using only node configurations in $\mathcal{S}$.

In fact, given any tree G of maximum degree Δ and a node v of degree Δ in G, after labeling the half-edges surrounding v using a node configuration in $\mathsf{trim}(\mathcal{S})$, it is always possible to extend this labeling to a complete correct labeling of G using only node configurations in $\mathsf{trim}(\mathcal{S})$. Such a labeling extension is possible due to Lemma 5.

▶ **Lemma 5** (Property of trimming). *Let $\mathcal{S} \subseteq \mathcal{V}$ such that $\mathsf{trim}(\mathcal{S}) \neq \emptyset$. For each node configuration $C \in \mathsf{trim}(\mathcal{S})$ and each label $\sigma \in C$, there exist a node configuration $C' \in \mathsf{trim}(\mathcal{S})$ and a label $\sigma' \in C'$ such that the multiset $\{\sigma, \sigma'\}$ is in $\mathcal{E}$.*

Proof. Assuming that such C' and σ' do not exist, we derive a contradiction as follows. We pick s to be the smallest number such that there is no correct labeling of T_s^* where the node configuration of the root r is in $\mathcal{S} \setminus \mathsf{trim}(\mathcal{S})$ and the node configuration of each remaining degree-Δ node of T_s^* is in $\mathcal{S}$. Such a number s exists due to the definition of trim.

Now consider a correct labeling of T_{s+1}^* where the node configuration of the root r is C and the node configuration of each remaining degree-Δ node is in $\mathcal{S}$. Such a correct labeling exists due to the fact that $C \in \mathsf{trim}(\mathcal{S})$. Our assumption on the non-existence of C' and σ' implies that the node configuration $\tilde{C}$ of one child w of the root r of T_{s+1}^* must be in $\mathcal{S} \setminus \mathsf{trim}(\mathcal{S})$. However, the radius-$s$ neighborhood of w in T_{s+1}^* is isomorphic to T_s^* rooted at w. Since the node configuration of w is in $\mathcal{S} \setminus \mathsf{trim}(\mathcal{S})$, our choice of s implies that the labeling of the radius-s neighborhood of w cannot be correct, which is a contradiction. ◀

Path-form of an LCL problem. Given an LCL problem $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ and a subset $\mathcal{S} \subseteq \mathcal{V}$ of node configurations, we define

$$\mathcal{D}_{\mathcal{S}} = \text{the set of all size-2 multisets } D \text{ such that } D \text{ is a sub-multiset of } C \text{ for some } C \in \mathcal{S}.$$

To understand the intuition behind the definition $\mathcal{D}_{\mathcal{S}}$, define the length-$k$ hairy path H_k as the result obtained by starting from a length-k path $P = (v_1, v_2, \ldots, v_{k+1})$ and then adding degree-1 nodes to make $\deg(v_i) = \Delta$ for all $1 \leq i \leq k+1$. If our task is to label hairy paths using node configurations in $\mathcal{S}$, then this task is identical to labeling paths using node configurations in $\mathcal{D}_{\mathcal{S}}$. In other words, the LCL problem $(\Delta, \Sigma, \mathcal{S}, \mathcal{E})$ on hairy paths is equivalent to the LCL problem $(2, \Sigma, \mathcal{D}_{\mathcal{S}}, \mathcal{E})$ on paths. Hence $(2, \Sigma, \mathcal{D}_{\mathcal{S}}, \mathcal{E})$ is the *path-form* of $(\Delta, \Sigma, \mathcal{S}, \mathcal{E})$.

Automaton for the path-form of an LCL problem. Given a set $\mathcal{D}$ of size-2 multisets whose elements are in Σ, we define the directed graph $\mathcal{M}_{\mathcal{D}}$ as follows. The node set $V(\mathcal{M}_{\mathcal{D}})$ of $\mathcal{M}_{\mathcal{D}}$ is the set of all pairs $(a, b) \in \Sigma^2$ such that the multiset $\{a, b\}$ is in $\mathcal{D}$. The edge set $E(\mathcal{M}_{\mathcal{D}})$ of $\mathcal{M}_{\mathcal{D}}$ is defined as follows. For any two pairs $(a, b) \in V(\mathcal{M}_{\mathcal{D}})$ and $(c, d) \in V(\mathcal{M}_{\mathcal{D}})$, we add a directed edge $(a, b) \to (c, d)$ if the multiset $\{b, c\}$ is an edge configuration in $\mathcal{E}$. Note that $\mathcal{M}_{\mathcal{D}}$ could contain self-loops.

The motivation for considering $\mathcal{M}_{\mathcal{D}}$ is that it can be seen as an automaton recognizing the correct solutions for the LCL problem $(2, \Sigma, \mathcal{D}, \mathcal{E})$ on paths, as each length-k walk $(a_1, b_1) \to (a_2, b_2) \to \cdots \to (a_{k+1}, b_{k+1})$ of $\mathcal{M}_{\mathcal{D}}$ corresponds to a correct labeling of a length-k path $(v_1, v_2, \ldots, v_{k+1})$ where the labeling of half-edge $(v_i, \{v_{i-1}, v_i\})$ is a_i and the labeling of half-edge $(v_i, \{v_i, v_{i+1}\})$ is b_i.

Path-flexibility. With respect to the directed graph $\mathcal{M}_{\mathcal{D}}$, we say that $(a,b) \in V(\mathcal{M}_{\mathcal{D}})$ is path-flexible if there exists an integer K such that for each integer $k \geq K$, there exist length-k walks $(a,b) \rightsquigarrow (a,b)$, $(a,b) \rightsquigarrow (b,a)$, $(b,a) \rightsquigarrow (a,b)$, and $(b,a) \rightsquigarrow (b,a)$ in $\mathcal{M}_{\mathcal{D}}$. Throughout this paper, we write $u \rightsquigarrow v$ to denote a walk starting from u and ending at v.

It is clear that (a,b) is path-flexible if and only if (b,a) is path-flexible. Hence we may extend the notion of path-flexibility from $V(\mathcal{M}_{\mathcal{D}})$ to $\mathcal{D}$. That is, we say that a size-2 multiset $\{a,b\} \in \mathcal{D}$ is path-flexible if (a,b) is path-flexible.

The following lemma is useful in lower bound proofs. For any $\{a,b\} \in \mathcal{D}$ that is not path-flexible, the following lemma shows that there are infinitely many path lengths k such that there is no length-k $s \rightsquigarrow t$ walk for some $s \in \{(a,b),(b,a)\}$ and $t \in \{(a,b),(b,a)\}$. As we will later see, this inflexibility in the possible path lengths implies lower bounds for distributed algorithms that may use the configuration $\{a,b\}$.

▶ **Lemma 6** (Property of path-inflexibility). *Suppose that the size-2 multiset $\{a,b\} \in \mathcal{D}$ is not path-flexible. Then one of the following holds.*

- *There is no $s \rightsquigarrow t$ walk for at least one choice of $s \in \{(a,b),(b,a)\}$ and $t \in \{(a,b),(b,a)\}$.*
- *There is an integer $2 \leq x \leq |\Sigma|^2$ such that for any positive integer k that is not an integer multiple of x, there are no length-k walks $(a,b) \rightsquigarrow (a,b)$ and $(b,a) \rightsquigarrow (b,a)$ in $\mathcal{M}_{\mathcal{D}}$.*

Proof. Suppose that $\{a,b\} \in \mathcal{D}$ is not path-flexible. We assume that there are $s \rightsquigarrow t$ walks for all choices of $s \in \{(a,b),(b,a)\}$ and $t \in \{(a,b),(b,a)\}$. To prove this lemma, it suffices to show that there is an integer $2 \leq x \leq |\Sigma|^2$ such that for any positive integer k that is not an integer multiple of x, there are no length-k walks $(a,b) \rightsquigarrow (a,b)$ and $(b,a) \rightsquigarrow (b,a)$.

First of all, we claim that for any integer K there is an integer $k \geq K$ such that there is no length-k walk $(a,b) \rightsquigarrow (a,b)$. If this claim does not hold, then there is an integer K such that there is a length-k walk $(a,b) \rightsquigarrow (a,b)$ for each $k \geq K$. Combining these walks with existing walks $(a,b) \rightsquigarrow (b,a)$ and $(b,a) \rightsquigarrow (a,b)$, we infer that there exists an integer K' such that for each integer $k \geq K'$, there exist length-k walks $(a,b) \rightsquigarrow (a,b)$, $(a,b) \rightsquigarrow (b,a)$, $(b,a) \rightsquigarrow (a,b)$, and $(b,a) \rightsquigarrow (b,a)$ in $\mathcal{M}_{\mathcal{D}}$, contradicting the assumption that $\{a,b\} \in \mathcal{D}$ is not path-flexible.

Let U be the set of integers k such that there is a length-k walk $(a,b) \rightsquigarrow (a,b)$. Note that by taking reversal, the existence of a length-k walk $(a,b) \rightsquigarrow (a,b)$ implies the existence of a length-k walk $(b,a) \rightsquigarrow (b,a)$, and vice versa. Our assumption on the existence of a walk $(a,b) \rightsquigarrow (a,b)$ implies $U \neq \emptyset$. We choose $x = \gcd(U)$ to be the greatest common divisor of U, so that for any integer k that is not an integer multiple of x, there are no length-k walks $(a,b) \rightsquigarrow (a,b)$ and $(b,a) \rightsquigarrow (b,a)$ in $\mathcal{M}_{\mathcal{D}}$. We must have $x \geq 2$ because there cannot be two co-prime numbers in U, since otherwise there exists an integer K such that U includes all integers that are at least K, contradicting the claim proved above. Specifically, if the two co-prime numbers are k_1 and k_2, then we may set $K = g(k_1,k_2) + 1 = k_1k_2 - k_1 - k_2 + 1$, where $g(k_1,k_2)$ is the Frobenius number of the set $\{k_1,k_2\}$ [42]. We also have $x \leq |\Sigma|^2$, since the smallest number in U is at most the number of nodes in $\mathcal{M}_{\mathcal{D}}$, which is upper bounded by $|\Sigma|^2$. ◀

For the special case of $|\Sigma| = 1$ and $\mathcal{D} \neq \emptyset$, we must have $a = b$ in Lemma 6. Since there is no integer x satisfying $2 \leq x \leq |\Sigma|^2$ when $|\Sigma| = 1$, Lemma 6 implies that if $\{a,a\}$ is not path-flexible, then there is no walk $(a,a) \rightsquigarrow (a,a)$, where $\{a,a\}$ is the unique element in $\mathcal{D}$.

Path-flexible strongly connected components. Since each $\{a,b\} \in \mathcal{D}$ corresponds to two nodes (a,b) and (b,a) in $\mathcal{M}_{\mathcal{D}}$, we will consider a different notion of a strongly connected component. In Definition 7, we do not require the elements a, b, c, and d to be distinct. For example, we may have $\{a,b\} = \{c,d\}$ or $a = b$.

▶ **Definition 7** (Strongly connected components). *Let $\mathcal{D}$ be a set of size-2 multisets of elements in Σ. For each $\{a,b\} \in \mathcal{D}$ and $\{c,d\} \in \mathcal{D}$, we write $\{a,b\} \sim \{c,d\}$ if there is a walk $s \rightsquigarrow t$ in $\mathcal{M}_{\mathcal{D}}$ for each choice of $s \in \{(a,b),(b,a)\}$ and $t \in \{(c,d),(d,c)\}$.*

Let $\mathcal{D}^{\sim}$ be the set of all $\{a,b\} \in \mathcal{D}$ such that $\{a,b\} \sim \{a,b\}$. Then we define the strongly connected components of $\mathcal{D}$ as the equivalence classes of $\sim$ over $\mathcal{D}^{\sim}$.

By taking reversal, the existence of an $(a,b) \rightsquigarrow (c,d)$ walk implies the existence of a $(d,c) \rightsquigarrow (b,a)$ walk. Therefore, if there is a walk $s \rightsquigarrow t$ in $\mathcal{M}_{\mathcal{D}}$ for each choice of $s \in \{(a,b),(b,a)\}$ and $t \in \{(c,d),(d,c)\}$, then there is also a walk $t \rightsquigarrow s$ in $\mathcal{M}_{\mathcal{D}}$ for each choice of $s \in \{(a,b),(b,a)\}$ and $t \in \{(c,d),(d,c)\}$. Hence the relation $\sim$ in Definition 7 is symmetric over $\mathcal{D}$. It is clear from the definition of $\sim$ in Definition 7 that it is transitive over $\mathcal{D}$ and it is reflexive over $\mathcal{D}^{\sim}$, so $\sim$ is indeed an equivalence relation over $\mathcal{D}^{\sim}$.

For any strongly connected component $\mathcal{D}'$ of $\mathcal{D}$, it is clear that either all $\{a,b\} \in \mathcal{D}'$ are path-flexible or all $\{a,b\} \in \mathcal{D}'$ are not path-flexible. We say that a strongly connected component $\mathcal{D}'$ is path-flexible if all $\{a,b\} \in \mathcal{D}'$ are path-flexible. We define $\mathsf{flexibility}(\mathcal{D}')$ as the minimum number K such that for each integer $k \geq K$ there is an $(a,b) \rightsquigarrow (c,d)$ walk of length k for all choices of a, b, c, and d such that $\{a,b\} \in \mathcal{D}'$ and $\{c,d\} \in \mathcal{D}'$. Such a number K exists given that $\mathcal{D}'$ is a path-flexible strongly connected component. We define

$$\mathsf{flexible\text{-}SCC}(\mathcal{D}) = \text{the set of all subsets of } \mathcal{D} \text{ that are a path-flexible strongly connected component of } \mathcal{D}.$$

Clearly, elements in $\mathsf{flexible\text{-}SCC}(\mathcal{D})$ are disjoint subsets of $\mathcal{D}$. It is possible that $\mathsf{flexible\text{-}SCC}(\mathcal{D})$ is an empty set, and this happens when all nodes in the directed graph $\mathcal{M}_{\mathcal{D}}$ are not path-flexible.

Restriction of a set of node configurations. Given an LCL problem $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$, a subset $\mathcal{S} \subseteq \mathcal{V}$ of node configurations, and a set $\mathcal{D}$ of size-2 multisets whose elements are in Σ, we define the restriction of $\mathcal{S}$ to $\mathcal{D}$ as follows.

$$\mathcal{S} \restriction_{\mathcal{D}} = \{C \in \mathcal{S} \mid \text{all size-2 sub-multisets of } C \text{ are in } \mathcal{D}\}.$$

Lemma 8 shows that if we label the two endpoints of a sufficiently long path using node configurations in $\mathcal{S} \restriction_{\mathcal{D}^*}$, where $\mathcal{D}^* \in \mathsf{flexible\text{-}SCC}(\mathcal{D}_{\mathcal{S}})$, then it is always possible to complete the labeling of the path using only node configurations in $\mathcal{S}$ in such a way that the entire labeling is correct. Specifically, consider a path $P = (v_1, v_2, \ldots, v_{d+1})$ of length $d \geq \mathsf{flexibility}(\mathcal{D}^*)$. Assume that the node configuration of v_1 is already fixed to be $C \in \mathcal{S} \restriction_{\mathcal{D}^*}$ where the half-edge $(v_1, \{v_1, v_2\})$ is labeled by $\beta \in C$ and the node configuration of v_{d+1} is already fixed to be $C' \in \mathcal{S} \restriction_{\mathcal{D}^*}$ where the half-edge $(v_{d+1}, \{v_d, v_{d+1}\})$ is labeled by $\alpha' \in C'$. Lemma 8 shows that it is possible to complete the labeling of P using only node configurations in $\mathcal{S}$, as we may label v_i using the node configuration C_i where the two half-edges $(v_i, \{v_{i-1}, v_i\})$ and $(v_i, \{v_i, v_{i+1}\})$ are labeled by α_i and β_i, for each $2 \leq i \leq d$.

▶ **Lemma 8** (Property of path-flexible strongly connected components). *Let $\mathcal{S} \subseteq \mathcal{V}$ be a set of node configurations, and let $\mathcal{D}^* \in \mathsf{flexible\text{-}SCC}(\mathcal{D}_{\mathcal{S}})$. For any choices of $C \in \mathcal{S} \restriction_{\mathcal{D}^*}$, $C' \in \mathcal{S} \restriction_{\mathcal{D}^*}$, size-2 sub-multisets $\{\alpha, \beta\} \subseteq C$, $\{\alpha', \beta'\} \subseteq C'$, and a number $d \geq \mathsf{flexibility}(\mathcal{D}^*)$, there exists a sequence $\alpha_1, C_1, \beta_1, \alpha_2, C_2, \beta_2, \ldots, \alpha_{d+1}, C_{d+1}, \beta_{d+1}$ satisfying the following conditions.*

- *First endpoint: $\alpha_1 = \alpha$, $\beta_1 = \beta$, and $C_1 = C$.*
- *Last endpoint: $\alpha_{d+1} = \alpha'$, $\beta_{d+1} = \beta'$, and $C_{d+1} = C'$.*
- *Node configurations: for $1 \leq i \leq d+1$, $\{\alpha_i, \beta_i\}$ is a size-2 sub-multiset of C_i, and $C_i \in \mathcal{S}$.*
- *Edge configurations: for $1 \leq i \leq d$, $\{\beta_i, \alpha_{i+1}\} \in \mathcal{E}$.*

Proof. By the path-flexibility of $\mathcal{D}^*$, there exists a length-d walk $(\alpha, \beta) \rightsquigarrow (\alpha', \beta')$ in $\mathcal{M}_{\mathcal{D}_\mathcal{S}}$. We fix $(\alpha_1, \beta_1) \to (\alpha_2, \beta_2) \to \cdots \to (\alpha_{d+1}, \beta_{d+1})$ to be any such walk. This implies that $\{\beta_i, \alpha_{i+1}\} \in \mathcal{E}$ for each $1 \le i \le d$. Since $\{\alpha_i, \beta_i\}$ is a size-2 multiset of $\mathcal{D}_\mathcal{S}$, there exists a choice of $C_i \in \mathcal{S}$ for each $2 \le i \le d$ such that $\{\alpha_i, \beta_i\}$ is a sub-multiset of C_i. ◀

Good sequences. Given an LCL problem $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ on Δ-regular trees, we say that a sequence $(\mathcal{V}_1, \mathcal{D}_1, \mathcal{V}_2, \mathcal{D}_2, \ldots, \mathcal{V}_k)$ is *good* if it satisfies the following requirements.

- $\mathcal{V}_1 = \mathsf{trim}(\mathcal{V})$. That is, we start the sequence from the result of trimming the set $\mathcal{V}$ of all node configurations in the given LCL problem $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$.
- For each $1 \le i \le k-1$, $\mathcal{D}_i \in \mathsf{flexible\text{-}SCC}(\mathcal{D}_{\mathcal{V}_i})$. That is, $\mathcal{D}_i$ is a path-flexible strongly connected component of the automaton associated with the path-form of the LCL problem $(\Delta, \Sigma, \mathcal{V}_i, \mathcal{E})$, which is Π restricted to the set of node configurations $\mathcal{V}_i$.
- For each $2 \le i \le k$, $\mathcal{V}_i = \mathsf{trim}(\mathcal{V}_{i-1} \restriction_{\mathcal{D}_{i-1}})$. That is, $\mathcal{V}_i$ is the result of taking the restriction of the set of node configurations $\mathcal{V}_{i-1}$ to $\mathcal{D}_{i-1}$ and then performing a trimming.
- $\mathcal{V}_k \neq \emptyset$. That is, we require that the last set of node configurations is non-empty.

It is straightforward to see that $\mathcal{V}_1 \supseteq \mathcal{V}_2 \supseteq \cdots \supseteq \mathcal{V}_k$ since $\mathcal{V}_i = \mathsf{trim}(\mathcal{V}_{i-1} \restriction_{\mathcal{D}_{i-1}})$ is always a subset of $\mathcal{V}_{i-1}$. Similarly, we also have $\mathcal{D}_1 \supseteq \mathcal{D}_2 \supseteq \cdots \supseteq \mathcal{D}_{k-1}$, as $\mathcal{D}_i \in \mathsf{flexible\text{-}SCC}(\mathcal{D}_{\mathcal{V}_i})$ is a subset of $\mathcal{D}_{\mathcal{V}_i}$ and $\mathcal{D}_{\mathcal{V}_i}$ is a subset of $\mathcal{D}_{i-1}$ due to the definition $\mathcal{V}_i = \mathsf{trim}(\mathcal{V}_{i-1} \restriction_{\mathcal{D}_{i-1}})$.

Depth of an LCL problem. We define the depth d_Π of an LCL problem $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ on Δ-regular trees as follows. If there is no good sequence, then we set $d_\Pi = 0$. If there is a good sequence $(\mathcal{V}_1, \mathcal{D}_1, \mathcal{V}_2, \mathcal{D}_2, \ldots, \mathcal{V}_k)$ for each positive integer k, then we set $d_\Pi = \infty$. Otherwise, we set d_Π as the largest integer k such that there is a good sequence $(\mathcal{V}_1, \mathcal{D}_1, \mathcal{V}_2, \mathcal{D}_2, \ldots, \mathcal{V}_k)$. In the full version [3] of the paper, we prove the following results.

▶ **Theorem 9** (Characterization of complexity classes). *Let $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ be an* LCL *problem on Δ-regular trees. We have the following.*

- *If $d_\Pi = 0$, then Π is unsolvable in the sense that there exists a tree of maximum degree Δ such that there is no correct solution of Π on this tree.*
- *If $d_\Pi = k$ is a positive integer, then the optimal round complexity of Π is $\Theta(n^{1/k})$.*
- *If $d_\Pi = \infty$, then Π can be solved in $O(\log n)$ rounds.*

▶ **Theorem 10** (Complexity of the characterization). *There is a polynomial-time algorithm $\mathcal{A}$ that computes d_Π for any given* LCL *problem $\Pi = (\Delta, \Sigma, \mathcal{V}, \mathcal{E})$ on Δ-regular trees. If $d_\Pi = k$ is a positive integer, then $\mathcal{A}$ also outputs a description of an $O(n^{1/k})$-round algorithm for Π. If $d_\Pi = \infty$, then $\mathcal{A}$ also outputs a description of an $O(\log n)$-round algorithm for Π.*

In Theorem 9, all upper bounds hold in the CONGEST model, and all lower bounds hold in the LOCAL model. For example, if $d_\Pi = 5$, then Π can be solved in $O(n^{1/5})$ rounds in the CONGEST model, and there is a matching lower bound $\Omega(n^{1/5})$ in the LOCAL model. The distributed algorithms returned by the polynomial-time algorithm $\mathcal{A}$ in Theorem 10 also work in the CONGEST model. We note that there are several natural definitions of unsolvability of an LCL w.r.t. a given graph class that are different from the one in Theorem 9, see [23].

References

1 Yehuda Afek, Shay Kutten, and Moti Yung. The local detection paradigm and its application to self-stabilization. *Theor. Comput. Sci.*, 186(1-2):199–229, 1997. doi:10.1016/S0304-3975(96)00286-1.

2 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.

3 Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studenỳ, and Jukka Suomela. Efficient classification of local problems in regular trees. *arXiv preprint*, 2022. arXiv:2202.08544.

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 *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 283–293. ACM, 2021. doi:10.1145/3465084.3467901.

7 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Deterministic δ-coloring plays hide-and-seek. In *Proc. 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*. ACM, 2022.

8 Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. In *Proc. 61st IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 365–376, 2020. doi:10.1109/FOCS46700.2020.00042.

9 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)*, pages 263–272. ACM Press, 2021. doi:10.1145/3465084.3467934.

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, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the LOCAL model. *Distributed Computing*, 34:259–281, 2021. doi:10.1007/s00446-020-00375-2.

12 Alkida Balliu, Keren Censor-Hillel, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Locally checkable labelings with small messages. In *35th International Symposium on Distributed Computing, DISC 2021*, volume 209 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.DISC.2021.8.

13 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.

14 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.

15 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.

16 Sebastian Brandt, Yi-Jun Chang, Jan Grebík, Christoph Grunau, Václav Rozhoň, and Zoltán Vidnyánszky. Local problems on trees from the perspectives of distributed algorithms, finitary factors, and descriptive combinatorics. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 29:1–29:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.29.

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

18 Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław 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.

19 Sebastian Brandt and Dennis Olivetti. Truly tight-in-Δ bounds for bipartite maximal matching and variants. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 69–78, 2020. doi:10.1145/3382734.3405745.

20 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.

21 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.

22 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.

23 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. arXiv:2002.07659.

24 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.

25 Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control.*, 70(1):32–53, 1986. doi:10.1016/S0019-9958(86)80023-7.

26 Manuela Fischer. Improved deterministic distributed matching via rounding. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC 2017)*, pages 17:1–17:15, 2017. doi:10.4230/LIPIcs.DISC.2017.17.

27 Manuela Fischer and Mohsen Ghaffari. Sublogarithmic distributed algorithms for lovász local lemma, and the complexity hierarchy. In *Proc. 31st International Symposium on Distributed Computing (DISC 2017)*, volume 91 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.DISC.2017.18.

28 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In *Proc. 57th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 625–634, 2016. doi:10.1109/FOCS.2016.73.

29 Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *Proc. 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS 2021)*, 2021.

30 Mohsen Ghaffari and Hsin-Hao Su. Distributed Degree Splitting, Edge Coloring, and Orientations. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2505–2523. Society for Industrial and Applied Mathematics, 2017. doi:10.1137/1.9781611974782.166.

31 Christoph Grunau, Václav Rozhoň, and Sebastian Brandt. The landscape of distributed complexities on trees, 2021. arXiv:2202.04724.

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

33 Michael Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. doi:10.1137/0215074.

34 Yannic Maus and Tigran Tonoyan. Local conflict coloring revisited: Linial for lists. In *34th International Symposium on Distributed Computing, DISC 2020*, volume 179 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.16.

35 Gary L. Miller and John H. Reif. Parallel tree contraction and its application. In *Proc. 26th Annual Symposium on Foundations of Computer Science (FOCS 1985)*, pages 478–489. IEEE, 1985. doi:10.1109/SFCS.1985.43.

36 Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM J. Discret. Math.*, 4(3):409–412, 1991. doi:10.1137/0404036.

37 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM J. Comput.*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

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

39 Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed Computing*, 14(2):97–100, 2001. doi:10.1007/PL00008932.

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

41 Jan Studený and Aleksandr Tereshchenko. Rooted tree classifier, 2021. URL: https://github.com/jendas1/rooted-tree-classifier.

42 J. J. Sylvester. On subvariants, i.e. semi-invariants to binary quantics of an unlimited order. *American Journal of Mathematics*, 5(1):79–136, 1882. URL: http://www.jstor.org/stable/2369536.