Classification of Distributed Binary Labeling Problems

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

Sebastian Brandt
ETH Zürich, Switzerland
brandts@ethz.ch

Yuval Efron
Technion – Israel Institute of Technology, Haifa, Israel
efronyuv@gmail.com

Juho Hirvonen
Aalto University, Finland
juho.hirvonen@aalto.fi

Yannic Maus
Technion – Israel Institute of Technology, Haifa, Israel
yannic.maus@cs.technion.ac.il

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

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

Abstract

We present a complete classification of the deterministic distributed time complexity for a family of graph problems: *binary labeling problems* in trees. These are locally checkable problems that can be encoded with an alphabet of size two in the edge labeling formalism. Examples of binary labeling problems include sinkless orientation, sinkless and sourceless orientation, 2-vertex coloring, perfect matching, and the task of coloring edges red and blue such that all nodes are incident to at least one red and at least one blue edge. More generally, we can encode e.g. any cardinality constraints on indegrees and outdegrees.

We study the deterministic time complexity of solving a given binary labeling problem in trees, in the usual LOCAL model of distributed computing. We show that the complexity of any such problem is in one of the following classes: $O(1)$, $\Theta(\log n)$, $\Theta(n)$, or unsolvable. In particular, a problem that can be represented in the binary labeling formalism cannot have time complexity $\Theta(\log^* n)$, and hence we know that e.g. any encoding of maximal matchings has to use at least three labels (which is tight).

Furthermore, given the description of any binary labeling problem, we can easily determine in which of the four classes it is and what is an asymptotically optimal algorithm for solving it. Hence the distributed time complexity of binary labeling problems is *decidable*, not only in principle, but also *in practice*: there is a simple and efficient algorithm that takes the description of a binary labeling problem and outputs its distributed time complexity.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Complexity classes

Keywords and phrases LOCAL model, graph problems, locally checkable labeling problems, distributed computational complexity

Digital Object Identifier 10.4230/LIPIcs.DISC.2020.17

Related Version The full version of this work is available at `https://arxiv.org/abs/1911.13294`.

Funding This work was supported in part by the Academy of Finland, Grant 314888 (Juho Hirvonen), and by the European Union's Horizon 2020 Research And Innovation Programme under grant agreement no. 755839 (Yuval Efron, Yannic Maus).

Acknowledgements We thank Jan Studený and anonymous reviewers for helpful comments on earlier versions of this work.

1 Introduction

This work presents a complete classification of the deterministic distributed time complexity for a family of distributed graph problems: *binary labeling problems* in trees. These are a special case of widely-studied locally checkable labeling problems [27]. The defining property of a binary labeling problem is that it can be encoded with an *alphabet of size two* in the *edge labeling formalism*, which is a modern representation for locally checkable graph problems [3, 9, 28]; we will give the precise definition in Section 2.

Contributions. In this work, we focus on *deterministic* distributed algorithms in the LOCAL model of distributed computing [24,29], and we study the computational complexity of solving a binary labeling problem in *trees*. It is easy to see that there are binary labeling problems that fall in each of the following classes:

- Trivial problems, solvable in $O(1)$ rounds.
- Problems similar to sinkless orientation, solvable in $\Theta(\log n)$ rounds [10, 15, 22].
- Global problems, requiring $\Theta(n)$ rounds.
- Unsolvable problems.

We show that this is a *complete* list of all possible complexities. In particular, there are no binary labeling problems of complexities such as $\Theta(\log^* n)$ or $\Theta(\sqrt{n})$. For example, maximal matching is a problem very similar in spirit to binary labeling problems, it has a complexity $\Theta(\log^* n)$ in bounded-degree graphs [17, 24], and it can be encoded in the edge labeling formalism using an alphabet of size three [3] – our work shows that three labels are also necessary for all problems in this complexity class.

Moreover, using our results one can easily determine the complexity class of any given binary labeling problem. We give a simple, concise characterization of all binary labeling problems for classes $O(1)$, $\Theta(n)$, and unsolvable, and we show that all other problems belong to class $\Theta(\log n)$. Hence the deterministic distributed time complexity of a binary labeling problem is *decidable*, not only in theory but also *in practice*: given the description of any binary labeling problem, a human being or a computer can easily find out the distributed computational complexity of the problem, as well as an asymptotically optimal algorithm for solving the problem. Our classification of all binary labeling problems is presented in Table 1, and given any binary labeling problem Π, one can simply do mechanical pattern matching to find its complexity class in this table.

Our work also sheds new light on the *automatic round elimination technique* [9, 28]. Previously, it was known that sinkless orientation is a nontrivial *fixed point* for round elimination [10] – such fixed points are very helpful for lower bound proofs, but little was known about the existence of other nontrivial fixed points. Our classification of binary labeling problems in this work led to the discovery of new nontrivial fixed points – this will hopefully pave the way for the development of a theoretical framework that enables us to understand when round elimination leads to fixed points and why.

This work will also make it *easier to prove lower bounds in the future*. Before this work, in essence the only known way to prove a nontrivial $\Omega(\log n)$ lower bound for some locally checkable problem Π was to come up with a reduction showing that Π is at least as hard as sinkless orientation. Our systematic exploration of binary labeling problems led to the discovery of an entire family of simple graph problems that are not directly comparable with sinkless orientation, but for which we can prove tight $\Omega(\log n)$ lower bounds directly with round elimination. In the future, whenever we encounter a graph problem Π of an unknown complexity, we can try to relate it not only to sinkless orientation but also to one of the new problems for which we now have new lower bounds.

Open questions. The main open question that we leave for future work is extending the characterization to randomized distributed algorithms: some binary labeling problems can be solved in $\Theta(\log \log n)$ rounds with randomized algorithms, but it is not yet known exactly which binary labeling problems belong to this class. Our work takes the first steps towards developing such a classification.

Structure. We start with the model of computation in Section 1.1 and a brief discussion of the general landscape of distributed computational complexity in Section 1.2, and then give the formal definitions of binary labeling problems in Section 2 that are needed to present our results in a concise manner, which we do in Section 3 – our main contribution is the characterization of all binary labeling problems in Table 1. Section 4 discusses the expressive power of binary labeling problems and collects examples of interesting binary labeling problems. In Section 5, we explain our proof techniques, and, more importantly, the key ideas of the technically most involved results. We conclude with Section 6, in which we discuss additional connections between the present work and related work – in particular, we highlight the role of binary labeling problems in the recent developments in the distributed computational complexity theory – and we will also introduce new directions for future research. All algorithms and lower bound proofs related to deterministic complexity and a discussion of randomized complexity are presented in the full version [2].

1.1 Model of computing

Deterministic LOCAL model. In this work, we use the standard LOCAL model of distributed computing [24, 29]. In this model, the input graph G represents a communication network, where each node is a computer and each edge is a communication link. If there are n nodes in the graph, each node is labeled with a *unique identifier* from $\{1, 2, \ldots, \mathrm{poly}(n)\}$. This is part of the local input, and a distributed algorithm can use it when it initializes the local state of a node. Computation proceeds in synchronous rounds, and in each round each node:

- sends a message to each neighbor,
- receives a message from each neighbor,
- performs local computation and updates its local state.

After each round, a node can choose to stop and produce its *local output*.

Let Π be a graph problem. We say that $\mathcal{A}$ is a deterministic algorithm that solves problem Π in $T(n)$ rounds if, for any input graph G with n nodes and for any assignment of unique identifiers, each node stops after at most $T(n)$ rounds and the local outputs of the nodes form a feasible solution of Π in G; if the task is to label edges we require that both endpoints agree on a label for the edge. We say that Π has complexity T in the LOCAL model if T is the pointwise minimum of all functions T' such that there exists an algorithm $\mathcal{A}$ that solves Π in time T'.

Note that in this model time is equivalent to distance: in T synchronous communication rounds all nodes can gather their radius-T neighborhoods (and nothing more). Hence we can interchangeably refer to *locality*, *distributed time complexity*, and the *number of communication rounds*.

Randomized LOCAL model. Our main focus is on deterministic distributed algorithms, but we will also discuss randomized distributed algorithms. In the randomized LOCAL model, in addition to having unique identifiers, nodes are labeled with an unbounded stream of random bits. Hence, we can let state transitions be probabilistic, and we arrive at randomized algorithms. In this work, randomized algorithms are Monte Carlo algorithms that are correct w.h.p. in the size of the graph.

1.2 Background and related work

Distributed complexity theory and LCL problems. The study of distributed graph algorithms has traditionally focused on specific graph problems – for example, investigating exactly what is the locality of finding a maximal independent set [3, 4]. However, in the recent years we have seen more focus on the development of a distributed complexity theory with which we can reason about entire *families of graph problems* [1, 5–7, 10, 11, 15, 16, 18, 19, 21, 23, 30, 31].

The key example is the family of *locally checkable labeling* problems (LCLs), introduced by [27]. Informally, a problem is locally checkable if the feasibility of a solution can be verified by looking at all constant-radius neighborhoods. For example, maximal independent sets are locally checkable, as we can verify both independence and maximality by looking at radius-1 neighborhoods.

In this line of research, among the most intriguing results are various *gap theorems*: For example, there are LCL problems solvable in $\Theta(\log^* n)$ rounds, while some LCL problems require $\Theta(\log n)$ rounds. However, between these two classes there is a gap: there are no LCL problems whose deterministic complexity in bounded-degree graphs is between $\omega(\log^* n)$ and $o(\log n)$ [15].

Decidability of distributed computational complexity. The existence of such a gap immediately suggests a follow-up question: given the description of an LCL problem, can we *decide* on which side of the gap it lies? And if so, can we automatically construct an asymptotically optimal algorithm for solving the problem?

As soon as we look at a family of graphs that contains e.g. 2-dimensional grids, questions related to the distributed complexity of a given LCL problem become undecidable [11, 27]. However, if we look at the case of paths and cycles, we can at least in principle write a computer program that determines the computational complexity of a given LCL problem [1, 11, 27], and some questions related to the complexity of LCLs in trees are also decidable [16] – unfortunately, we run into PSPACE-hardness already in the case of paths and cycles [1].

We conjecture that *all* questions about the distributed complexity of LCL problems in trees are decidable. Proving (or disproving) the conjecture is a major research program, but in this work we take one step towards proving the conjecture and we bring plenty of good news: we introduce a family of LCL problems, so-called *binary labeling problems*, and we show that we can completely characterize the deterministic distributed complexity of every binary labeling problem in trees. In particular, all questions about the deterministic distributed complexity of these problems are decidable not only in principle but also in practice – using our results, a human being or a computer can easily find an optimal algorithm for solving any given binary labeling problem.

2 Binary labeling problems

We will now give the formal definition of the family of binary labeling problems. LCL problems have been traditionally specified by listing a *collection of permitted local neighborhoods* [27]. However, we will use the more recent *edge labeling* formalism [3, 9, 28], which is equally expressive in the case of trees, and it has the additional benefit that it makes it very convenient to apply the automatic round elimination technique [9, 28], which is very helpful to prove lower bounds:

- we have a bipartite graph and the nodes are colored with two colors, white and black,
- the task is to *label edges* with symbols from some alphabet Σ,
- there are *both white and black constraints* that define the graph problem.

We emphasize that despite its bipartite appearance, the edge labeling formalism can easily be used to describe problems on general graphs by interpreting edges as "black nodes". In fact, as explained in detail in Section 4, problems on general graphs are a special case in the edge labeling formalism – in general the formalism describes problems on hypergraphs.

2.1 General form

If we set $|\Sigma| = 2$ in the edge labeling formalism, we arrive at the definition of binary labeling problems. Formally, a *binary labeling problem* is a tuple $\Pi = (d, \delta, W, B)$, where

- $d \in \{2, 3, \dots\}$ is the *white degree*,
- $\delta \in \{2, 3, \dots\}$ is the *black degree*,
- $W \subseteq \{0, 1, \dots, d\}$ is the *white constraint*, and
- $B \subseteq \{0, 1, \dots, \delta\}$ is the *black constraint*.

An *instance* of problem Π is a pair (G, f), where

- $G = (V, E)$ is a simple graph,
- $f \colon V \to \{\text{black}, \text{white}\}$ is a proper 2-coloring of G, i.e., $f(u) \neq f(v)$ for all $\{u, v\} \in E$.

In the distributed setting, we will assume that the color $f(v)$ is part of the local input of node $v \in V$.

Let $X \subseteq E$ be a subset of edges. For each node $v \in V$, its X-degree $\deg_X(v)$ is the number of edges in X that are incident to v. We say that $X \subseteq E$ is a *solution* to binary labeling problem Π if it satisfies the following constraints for all $v \in V$:

- If $f(v) = \text{white}$ and $\deg(v) = d$, then $\deg_X(v) \in W$.
- If $f(v) = \text{black}$ and $\deg(v) = \delta$, then $\deg_X(v) \in B$.

We use the term *relevant nodes* to refer to white nodes of degree d and black nodes of degree δ. The interpretation is that problem Π is interesting in regular neighborhoods in which all white nodes have degree d and all black nodes have degree δ; any irregularities make the problem easier to solve, as all other nodes are unconstrained.

▶ **Example 2.1** (bipartite splitting)**.** Let $d = \delta = 4$ and $W = B = \{1, 2, 3\}$. We can interpret a solution $X \subseteq E$ as a coloring: edges in X are colored red and all other edges are colored blue. Now Π is equivalent to the following graph problem on bipartite graphs: color all edges red or blue such that all degree-4 nodes are incident to at least one blue edge and at least one red edge.

In order to be able to state our results in a concise manner, we will introduce some necessary notation and terminology in the following.

2.2 Equivalence, restrictions, and relaxations

If a is a natural number and A is a set of natural numbers, we will define $a - A = \{a - x : x \in A\}$.

▶ **Definition 2.2** (Equivalence)**.** *For any d, δ, W, B we call the following four problems* equivalent

$$\Pi_{00} = (d, \delta, W, B), \qquad \Pi_{01} = (\delta, d, B, W),$$
$$\Pi_{10} = (d, \delta, d - W, \delta - B), \qquad \Pi_{11} = (\delta, d, \delta - B, d - W).$$

Definition 2.2 partitions binary labeling problems in equivalence classes, each of them with at most four distinct problems. We use notation $\Pi \sim \Pi'$ for this equivalence relation, and say that Π and Π' are *equivalent*.

▶ **Observation 2.3.** *If* $\Pi \sim \Pi'$ *then* Π *and* Π' *have the same distributed complexity up to* ± 1 *round.*

Proof. Given an algorithm for Π, we get an algorithm for Π' by either exchanging the roles of black and white nodes or by replacing solution X with its complement $E \setminus X$. As incident nodes need to agree on the output on edges the complexity can differ by 1 round. ◀

The following definition is helpful to show that our classification is complete.

▶ **Definition 2.4.** *Given two problems* $\Pi = (d, \delta, W, B)$ *and* $\Pi' = (d, \delta, W', B')$*, we say that* Π' *is a* restriction *of* Π *and* Π *is a* relaxation *of* Π' *if* $W' \subseteq W$*, and* $B' \subseteq B$*.*

We use notation $\Pi' \subseteq \Pi$ to denote that Π' is a restriction of Π.

▶ **Observation 2.5.** *If* $\Pi' \subseteq \Pi$*, then any feasible solution for* Π' *is also a feasible solution for* Π*. In particular, if* Π' *can be solved in* T *rounds, then* Π *can also be solved in* T *rounds.*

2.3 Vector notation

It is convenient to interpret set W as a bit vector $w_0 w_1 \dots w_d$ with $d+1$ bits, so that bit $w_i = 1$ if $i \in W$ and $w_i = 0$ if $i \notin W$. Similarly, B can be interpreted as a bit vector with $\delta + 1$ bits.

▶ **Example 2.6.** Using this notation, the problem of Example 2.1 can be represented with $W = 01110$ and $B = 01110$, or, in brief, $\Pi = (01110, 01110)$.

Note that when we use vector notation, vectors W and B fully determine problem Π; therefore we do not need to specify d and δ separately and we can simply write $\Pi = (W, B)$.

We will use shorthand notation such as 1^x for a vector of x 1s and 1^+ for a vector of one or more 1s, and we will use $*$ to refer to a bit of any value. For example, $W = 0{*}{*}0$ is a shorthand for $W \in \{0000, 0010, 0100, 0110\}$ and $W = 01^+0$ is a shorthand for $W \in \{010, 0110, 01110, \dots\}$.

3 Our contributions

3.1 Main results: deterministic complexity

Our main contribution is a complete classification of the deterministic distributed complexity of binary labeling problems in trees – this is presented in Table 1, and our results hold in the standard LOCAL model of distributed computing [24, 29].

Next, we discuss the classification in more detail: One can partition all binary labeling problems in 15 *families of problems* based on the structure of their white and black constraints such that, given a binary labeling problem Π in the vector notation, one can simply do pattern matching in Table 1 to first find its problem family and this way also find its deterministic complexity. For easier reference, we have listed all problem families explicitly, but if we combine families whose problems are equivalent (recall Observation 2.3) we can partition problems in seven *types*, labeled with $\mathsf{I}, \mathsf{II}, \dots, \mathsf{VII}$ in the table. Types also reflect the structure of our proofs. The mapping from types (and families) to distributed complexity is also shown in Table 1. It then directly follows that the distributed complexity of any given binary labeling problem is decidable, and efficiently computable.

Table 1 The deterministic distributed complexity of binary labeling problems in trees; 1^+ signifies one or more 1s, 0^+ signifies one or more 0s, and $*$ signifies either 0 or 1. Non-empty means that the constraint is not 0^+. Note that e.g. all problem families of type I are equivalent to each other in the following sense: for any problem Π of type I, there is exactly one problem equivalent to Π in each of the four families I.a, I.b, I.c, and I.d.

Type	Problem family	White constraint	Black constraint	**Deterministic complexity**
I	I.a	100^+	$0{*}{*}^+$	**unsolvable**
	I.b	00^+1	${*}{*}^+0$	
	I.c	$0{*}{*}^+$	100^+	
	I.d	${*}{*}^+0$	00^+1	
II	II.a	000^+	${*}{*}{*}^+$	
	II.b	${*}{*}{*}^+$	000^+	
III	III.a	non-empty	111^+	$\boldsymbol{O(1)}$
	III.b	111^+	non-empty	
IV	IV.a	$1{*}{*}^+$	$1{*}{*}^+$	
	IV.b	${*}{*}^+1$	${*}{*}^+1$	
V	V.a	10^+1	010	$\boldsymbol{\Theta(n)}$
	V.b	010	10^+1	
VI	VI.a	$0^+1{*}$	${*}10^+$	
	VI.b	${*}10^+$	$0^+1{*}$	
VII	VII.a	all other cases		$\boldsymbol{\Theta(\log n)}$

Table 2 Examples of binary labeling problems and problem families and their deterministic complexities.

Deterministic complexity	Type	Examples of problems			
		W	B	reference	description
unsolvable	I	011^+	100	Example 4.8	contradiction
$O(1)$	III	0010^+	111	Example 4.9	trivial
$\Theta(n)$	V	10^+1	010	Example 4.5	two-coloring
$\Theta(\log n)$	VII	01110	01110	Example 2.1	bipartite splitting
		1010	010	Example 4.4	even orientation
		111^+0	010	Example 4.1	sinkless orientation
		011^+0	010	Example 4.3	sinkless & sourceless orientation
		0100^+	101	Example 4.6	regular matching
		011^+0	101	Example 4.7	splitting
		0100^+	0100^+	Section 5	bipartite matching
		0100^+	10^+1	Section 5	hypergraph matching
		0100^+	110	Section 5	edge grabbing
$\Omega(\log n)$, $O(n)$	V or VII	1^+01^+	01^+0	Section 5	forbidden degree
	VI or VII	11^+0	011^+	Section 5	bipartite sinkless orientation

We next detail on the different types. Problems of types **I** and **II** are unsolvable, problems of type **III** and **IV** have complexity $O(1)$, and problems of types **V** an **VI** have complexity $\Theta(n)$. Now by definition all problems that are not of type **I**–**VI** are of type **VII**; we prove that all of them are solvable in $O(\log n)$ rounds and we also prove a matching lower bound of $\Omega(\log n)$. This completes the proof that the classification of Table 1 is correct.

Finding problems that are unsolvable or need $O(1)$ rounds is not difficult:

- There are two types of problems that are unsolvable for trivial reasons. For example, if all white nodes must have X-degree 0 and all black nodes must have a non-zero X-degree, no solution exists as long as the tree is large enough.
- There are two types of problems that are solvable in $O(1)$ time for trivial reasons. For example, if black nodes are happy with any X-degree, then white nodes can simply pick some number $x \in W$ and choose arbitrarily x adjacent edges.

However, what is not obvious is that this list is exhaustive: no other problems are unsolvable, and no other problem is solvable in $O(1)$ time.

Furthermore, as we can see in Table 1, there are only very few binary labeling problems that are solvable but inherently global, i.e., they require $\Theta(n)$ rounds to solve. What is perhaps the biggest surprise is that all other problems can be solved in $O(\log n)$ rounds, and they also require $\Omega(\log n)$ rounds. In particular:

▶ **Theorem 3.1.** *There are no binary labeling problems with deterministic distributed complexity in the following ranges:*

- *between* $\omega(1)$ *and* $o(\log n)$,
- *between* $\omega(\log n)$ *and* $o(n)$.

Table 2 shows examples of problems in each complexity class.

3.2 Additional results: randomized complexity

While our focus in this work is on deterministic complexity, we will also explore the randomized complexity of binary labeling problems. Many of our theorems from the deterministic parts have direct implications on randomized complexity. By prior work, it is known that classes $O(1)$ and $\Theta(n)$ remain the same also for randomized complexity – this follows from [19] and (unpublished) extensions of the result in [5]. However, class $\Theta(\log n)$ is more interesting, as there are binary labeling problems of the following types:

- Deterministic complexity $\Theta(\log n)$ and randomized complexity $\Theta(\log \log n)$, e.g. sinkless orientation, Example 4.1.
- Deterministic complexity $\Theta(\log n)$ and randomized complexity $\Theta(\log n)$, e.g. even orientation, Example 4.4.

That is, randomness helps with some binary labeling problems but not all. While we present a partial classification of the randomized complexity of binary labeling problems, the main open question for the future work is coming up with a complete characterization of exactly which binary labeling problems can be solved in $O(\log \log n)$ rounds with randomized algorithms.

4 Expressive power of binary labeling problems

At first the bipartite setting in the definition of binary labeling problems in Section 2.1 may seem restrictive – indeed, why would we care about graphs that are properly 2-colored. However, we can take any graph and interpret edges as "black nodes" and this way many graph problems of interest can be represented in the binary labeling formalism. More precisely, let $G_0 = (V_0, E_0)$ be a graph. Subdivide (i.e., add a vertex in the middle of each edge) all edges of G_0 to construct a new graph $G = (V, E)$, and assign the color white to all original nodes in V_0 and color black to the new nodes in $V \setminus V_0$.

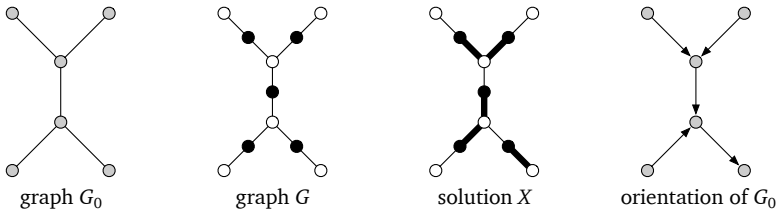

Figure 1 Binary labeling problems with $B = 010$ are orientation problems.

To simulate an algorithm $\mathcal{A}$ for the (virtual) network G in the communication network G_0 each vertex of G has to be simulated by a vertex of G_0 and we use node identifiers (or randomness) to choose which endpoint simulates the black node of an edge. Finally, for the purposes of lower bounds we can also go in the other direction and take an algorithm for G_0 and simulate it in graph G. This increases the round complexity by a factor of two, which we can ignore as we are only interested in asymptotics.

Consider a binary labeling problem Π with $B = 010$. Assume that we have a solution X to Π in G. We can now interpret X as an *orientation* of the original graph G_0: If an edge $e = \{u, v\} \in E_0$ was subdivided in two edges, $e_1 = \{u, x\}$, and $e_2 = \{x, v\}$, we will have exactly one of these edges in set X. If we have $e_1 \in X$, we can interpret it so that edge $\{u, v\}$ is oriented from v to u, and otherwise it is oriented from u to v; see Figure 1. In essence, Π is now equivalent to the following problem: Find an orientation of G_0 such that all nodes with $\deg(v) = d$ have $\text{indegree}(v) \in W$.

▶ **Example 4.1** (sinkless orientation)**.** Let $d > 2$, $\delta = 2$, $W = 111^+0$, and $B = 010$. Now all edges must be properly oriented: there is exactly one head. Furthermore, for all nodes of degree d, they must have indegree in $\{0, 1, \ldots, d-1\}$. Put otherwise, degree-d nodes must have outdegree at least one. Hence a feasible solution represents an orientation in which none of degree-d nodes are sinks.

▶ **Example 4.2.** Let $W = 0111^+$ and $B = 010$. By Observation 2.3, this is equivalent to Example 4.1. In essence, we have merely reversed the roles of heads and tails – now the task is to find a sourceless orientation.

▶ **Example 4.3** (sinkless and sourceless orientation)**.** Let $W = 011^+0$ and $B = 010$. Now in d-regular graphs the task is to find an orientation such that all nodes have at least one incoming and at least one outgoing edge.

Note that this problem is a restriction of Example 4.1; recall Definition 2.4.

▶ **Example 4.4** (even orientation)**.** Let $W = 1010$ and $B = 010$. In 3-regular graphs the task is to find an orientation such that all nodes have an even number of outgoing edges.

▶ **Example 4.5** (two-coloring)**.** Let $W = 10^+1$ and $B = 010$. In d-regular graphs this is a 2-coloring problem: each node v is either "red" (indegree 0) or "blue" (indegree d), and all edges between such nodes are properly colored (they are always oriented from red to blue).

Another interesting special case is $B = 101$. Now for each original edge of G_0 we will select either both of the half-edges or none of the half-edges, and hence a solution $X \subseteq E$ can be interpreted in a natural way as a *subset of edges* $X_0 \subseteq E_0$ in the original graph; see Figure 2. This is a splitting problem: partition E_0 in two classes, X_0 and $E_0 \setminus X_0$, and W determines how many incident edges in each class we can have.

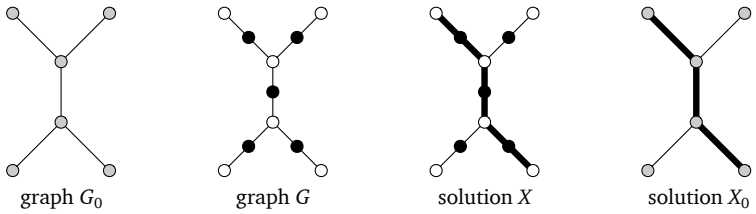

Figure 2 Binary labeling problems with $B = 101$ are splitting problems.

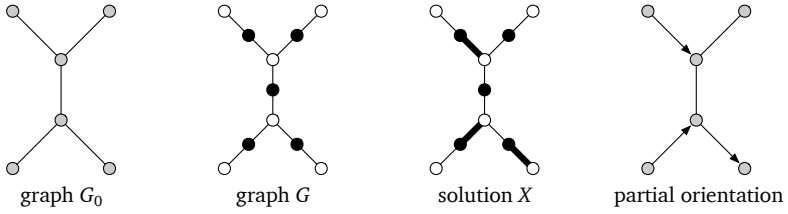

Figure 3 Binary labeling problems with $B = 110$ are partial orientation problems.

▶ **Example 4.6** (regular matching)**.** Let $W = 0100^+$ and $B = 101$. The task is to find a set $X_0 \subseteq E_0$ such that all nodes of degree d are incident to exactly one edge in X_0. In particular, if we have a d-regular graph, X_0 is a perfect matching.

▶ **Example 4.7** (splitting)**.** Let $W = 011^+0$ and $B = 101$. In this problem we will need to color edges red and blue such that all degree-d nodes are incident to at least one red edge and at least one blue edge.

We can also consider e.g. $B = 110$ and interpret X as a *partial orientation*: some edges of G_0 are oriented and some may be left unoriented and again W indicates which indegrees are permitted; see Figure 3. The case of $B = 011$ is equivalent to $B = 110$ – recall Observation 2.3.

As we will see, all other cases $B = 000$, $B = 100$, $B = 001$, and $B = 111$ are either trivial or unsolvable; we will give two examples:

▶ **Example 4.8** (contradiction)**.** Let $W = 011^+$ and $B = 100$. Here black nodes must have X-degree 0, so we must have $X = \emptyset$. However, all white nodes must be adjacent to at least one edge in X. Hence there is no solution to the problem.

▶ **Example 4.9** (trivial)**.** Let $W = 0010^+$ and $B = 111$. Here all white nodes can arbitrarily choose two incident edges and add them to X.

▶ Remark 4.10. Similar ideas can be generalized to hypergraphs. In essence, we can interpret the bipartite graph G as a hypergraph H, where white nodes of G correspond to nodes of H and black nodes of G correspond to hyperedges of H. Now Π is in essence a hypergraph problem. If we set $B = 010^+$, a solution X can be interpreted as an orientation of hyperedges, and if we set $B = 10^+1$, a solution X can be interpreted as a subset of hyperedges. Sinkless orientations in hypergraphs have been studied, e.g., in [12].

4.1 Binary labeling problems in trees

All of the above examples are well-defined also in trees; however, some care is needed when we interpret them. For example, let us revisit the "regular matching" problem from Example 4.6. Let $W = 0100^+$ and $B = 101$. Now in the case of a tree, the task is to find a subset of edges

X such that internal nodes of degree d are incident to exactly one such edge. However, leaf nodes are unconstrained. Informally, if we are in the middle of a d-regular tree, we will need to find a solution that locally looks like a perfect matching, but near leaf nodes and other irregularities the output is more relaxed. One consequence is that for this problem a solution always exists (while there are of course trees in which a perfect matching does not exist).

5 Overview of the key technical ideas

Due to space constraints, the full proofs are deferred to the full version [2]. Here, we give an overview of the key techniques and ideas where we focus on the technically most involved parts. In these parts we show that all problems that cannot be (easily) put into the categories *unsolvable*, $O(1)$ round complexity or $\Theta(n)$ round complexity have complexity $\Theta(\log n)$.

Problems of types I and II are unsolvable. We show that problems of types I and II are problems in which a contradiction occurs, e.g., black nodes can only label their incident edges with 0, but white nodes must have at least one incident edge labeled 1. Such problems clearly cannot be solved.

Problems of types III and IV can be solved in $O(1)$ rounds. We show that problems of types III and IV can be solved without any communication. These problem classes consist of problems in which all nodes can output the same label on all their incident edges, and problems in which black (white) nodes are happy with any labeling.

Problems of types V and VI require $\Theta(n)$ rounds. We show that problems of types V and VI consist of variants of the 2-coloring problem and variants of the problem of consistently orienting a given tree. We prove that all such problems require $\Omega(n)$ rounds, using indistinguishability arguments. In essence, we prove that some nodes need to coordinate their outputs over a linear distance.

> **Key idea: all remaining cases have complexity $\Theta(\log n)$.** So far we have covered relatively simple cases. Surprisingly, we can show that for all problems not of types I–VI there are deterministic $O(\log n)$ upper bounds and deterministic $\Omega(\log n)$ lower bounds.

All other problems can be solved in $O(\log n)$ rounds. To show the upper bound, we define the following notion of *resilience*. Let $\Pi = (d, \delta, W, B)$ be a binary labeling problem. For $0 \le t \le d+1$ and $0 \le s \le \delta+1$ we say that Π is (t,s)-*resilient* if both of the following hold:

- bit string W does not contain a substring of the form 0^{d+1-t},
- bit string B does not contain a substring of the form $0^{\delta+1-s}$.

We also introduce three special problem families:

- Problems of the form $\Pi = (0100^+, 0100^+)$ are called *bipartite matching problems*.
- Problems of the form $\Pi = (0100^+, 10^+1)$ are called *hypergraph matching problems*.
- Problems of the form $\Pi = (0100^+, 110)$ are called *edge grabbing problems*.

We show that any problem that is *not* of types I–VI has to fall in one of the following four classes:

(1) $(2,1)$-resilient problems and $(1,2)$-resilient problems.
(2) Bipartite matching and equivalent problems.

(3) Hypergraph matching and equivalent problems.
(4) Edge grabbing and equivalent problems.

We show how to use the rake & compress technique [25] to design $O(\log n)$-round algorithms for all of these problems. In brief, we decompose the vertex set of the tree into $L = O(\log n)$ layers, and sequentially construct a valid solution in $O(\log n)$ iterations. We first fix the solutions of all nodes in layer L in parallel, then all nodes in layer $L-1$ in parallel, and so on. Here resilience is crucial, as it guarantees lower layers can still satisfy their requirements even if the higher layers have made arbitrary choices.

All other problems require $\Omega(\log n)$ rounds. To show the matching lower bound, we define two more problem families:

- Problems of the form $\Pi = (1^+01^+, 01^+0)$ are called *forbidden degree problems*.
- Problems of the form $\Pi = (11^+0, 011^+)$ are called *bipartite sinkless orientation problems*.

We then show that all problem that are not unsolvable (types **I** and **II**) or trivial (types **III** and **IV**) are at least as hard as forbidden degree problems or bipartite sinkless orientation problems. Finally, we show that both of these problems require $\Omega(\log n)$ rounds.

The lower bounds for bipartite sinkless orientation problems essentially follow from [10]; however, for forbidden degree problems no lower bounds were known previously.

We show the lower bound using the round elimination technique [9]. We define a new parameterized problem family that we call $\mathsf{FDSO}(s)$ – we emphasize that this is *not* a binary labeling problem, but it turns out that an efficient algorithm for the forbidden degree problem implies an efficient algorithm for $\mathsf{FDSO}(s)$. In the language of [3, 28], $\mathsf{FDSO}(s)$ is defined by the white constraint

$$AX^{d-1}, \quad H^{s+1}X^{d-s-1}, \quad T^{d-s+1}X^{s-1},$$

and black constraint

$$X[AHTX]^{\delta-1}, \quad HT[AHTX]^{\delta-2}.$$

We show that $\mathsf{FDSO}(s)$ is a *fixed point* for the round elimination technique, and a lower bound of $\Omega(\log n)$ then follows.

Randomized complexity. To conclude this work, we take a closer look at the problems of type **VII**, i.e., problems with deterministic complexity $\Theta(\log n)$. Together with prior work [14, 16], our results imply that all such problems have randomized complexity either $\Theta(\log \log n)$ or $\Theta(\log n)$ rounds. Moreover, there are already many known examples of problems of type **VII** that fall in the class of $\Theta(\log \log n)$; examples include sinkless orientation [15], as well as many orientation and splitting problems that are known to be as easy as sinkless orientation [20]. We focus on new hardness results: we give a list of problem families within type **VII** that fall in the class of $\Theta(\log n)$-round randomized complexity.

Our list of problems of type **VII** that require $\Theta(\log n)$ rounds with randomized algorithms is *not* complete. The main open question for future work is completing the list of all such binary labeling problems.

6 Discussion and additional related work

In this work we initiated a systematic investigation of the distributed complexity of LCL problems on trees, and we presented a complete characterization of the deterministic distributed complexity of a specific family of LCL problems, binary labeling problems. To conclude this

work, we will discuss additional connections between the present work and related work – in particular, highlighting the role of binary labeling problems in the recent developments in the distributed computational complexity theory – and we will also introduce new directions for future research.

Sinkless orientation is a binary labeling problem. Sinkless orientation (Example 4.1) is one of the cornerstones of the modern theory of distributed computational complexity. This problem was introduced in our context in 2016 [10], and in Google Scholar there are already more than 30 papers written since 2016 that mention the term "sinkless orientation", all of them related to the theory of distributed computing. There are many variants of the definition, but for our purposes all of them are in essence equivalent to each other, so let us use the following version: there is a fixed parameter $d > 2$, and the task is to orient edges so that nodes of degree d are not sinks (note that low-degree nodes can be sinks and the problem is therefore trivial in cycles).

The distributed complexity of solving sinkless orientation is now completely understood: $\Theta(\log n)$ rounds with deterministic algorithms and $\Theta(\log \log n)$ rounds with randomized algorithms [10, 15, 22]. Yet there are many fundamental questions related to the role of sinkless orientation that we do not understand at all.

One of the mysteries is the following observation: whenever we encounter a problem Π that turns out to be as hard as sinkless orientation, usually the reason for Π being hard is that Π is directly related to sinkless orientation through reductions. For example, sinkless and sourceless orientation (Example 4.3) has the same complexity as sinkless orientation; the upper bound is nontrivial [22], but the lower bound trivially comes from the observation that a sinkless and sourceless orientation gives a sinkless orientation. Before this work, we were not aware of any LCL problem Π that has the same complexity as sinkless orientation – $\Theta(\log n)$ rounds deterministic and $\Theta(\log \log n)$ rounds randomized – with a nontrivial lower bound that is not merely an observation that an algorithm for solving Π directly gives an algorithm for solving sinkless orientation.

Indeed, we did not know if all problems in this complexity class are merely extensions and variants of the same problem!

Our systematic study of binary labeling problems succeeded in shedding new light on this issue: we identified problems that are as hard as sinkless orientation, yet require a *new lower bound proof* that is not based on the previous result of the hardness of sinkless orientation.

Automatic round elimination and fixed points. The new problems that we discovered are also directly related to another ongoing research topic: understanding the *automatic round elimination technique* [9, 28]. As we mentioned in Section 2, the edge labeling formalism makes it easy to apply the round elimination technique, which is an effective technique for proving lower bounds [3, 4, 8–10, 13, 24, 26]. However, there are still many open research questions related to round elimination itself. One of the key questions is about fixed points; here the sinkless orientation problem is a good example.

Let Π be the sinkless orientation problem from Example 4.1. If we start with the assumption that the complexity of Π is $T = o(\log n)$ rounds in regular trees, and apply the round elimination technique [9, 28] in a mechanical manner, we will immediately get the result that the complexity of the same problem Π is $T - 1$ rounds, which is absurd, and hence we immediately get a lower bound (at least for some weak models of distributed computing). A bit more formally, Π is a *nontrivial fixed point* for round elimination, and such fixed points immediately imply nontrivial lower bounds. This is particularly interesting as fixed points can be detected automatically with a computer.

Now let Π' be the problem of finding a sinkless and sourceless orientation from Example 4.3. For a human being, it is now trivial that Π' is at least as hard as Π. However, Π' is not a fixed point for round elimination, and we do not seem to have any automatic way for proving lower bounds for problems similar to Π'.

What is not understood at all is what is the fundamental difference between Π and Π' and why one of them is a fixed point and the other one is not. Indeed, so far there have not been many examples of nontrivial fixed points that are not merely trivial variants of sinkless orientations. Our work now presented the first new examples of nontrivial fixed points, and these new examples will hopefully inspire follow-up work towards a better understanding of fixed points in round elimination in general.

Binary labeling problems vs. homogeneity. LCLs come in many flavors. Some problems are trivial in a regular unlabeled graph – fractional problems are a good example here. However, for many distributed problems the interesting part is specifically what to do in the middle of a regular unlabeled tree – symmetry-breaking problems and orientation and splitting problems fall in this category. Our focus was on the latter case.

In the definition of binary labeling problems, the idea of *focusing on regular parts* is captured in the constraint that only white nodes of degree d and only black nodes of degree δ are relevant. In essence, as soon as we see some irregularities (e.g., leaf nodes of the tree), the problem becomes potentially easier to solve.

The idea of investigating what happens in the middle of a regular tree was previously formalized using a somewhat different idea of *homogeneous LCLs* [8]. Both homogeneous LCLs and binary labeling problems are defined so that the problem is interesting in a d-regular part of a tree and any irregularities make the problem easier to solve. However, this is achieved through a different mechanism:

- Homogeneous LCLs [8]: Instead of solving the original problem Π in some neighborhood, you can create a pointer. The pointers have to form a chain, and a chain of pointers can only terminate at a node with degree different from d.
- Binary labeling problems (this work): The labels are constrained only for nodes of degree d. The labels incident to other nodes are unconstrained.

In practice, this means that homogeneous LCLs are always solvable in trees, and they can be solved in $O(\log n)$ rounds (assuming $d > 2$): we can always find a node v of degree 1 or 2 within distance $O(\log n)$, and hence instead of solving the original problem, we can construct a pointer chain towards v. Therefore homogeneous LCLs are well-suited for the original purpose of investigating distributed computational complexity in the $o(\log n)$ region, but they can neither provide insights into the $\Omega(\log n)$ regime nor classify unsolvable problems. Our work on binary LCLs gives complete answers for both.

On the other hand, homogeneous problems are not restricted to two labels, and hence it is possible that there are complexity classes below $o(\log n)$ that do not exist for binary labeling problems. As we see in Table 3, this is indeed the case: problems with complexity $\Theta(\log^* n)$ exist among homogeneous problems, but as we have seen in this work, they do not exist among binary labeling problems.

Restriction to two labels and $\Theta(\log^* n)$ complexity. In a binary labeling problem we label edges (or half-edges) with two labels: whether it is part of solution X or not. If we looked at the more general case of $|\Sigma| = O(1)$ possible edge labels, we would have a family of problems that is, in essence, more expressive than the family of all homogeneous LCLs.

Table 3 An overview of possible distributed time complexities in trees for different classes of LCLs.

Deterministic complexity	Randomized complexity	General LCLs	Homog. LCLs	**Binary labeling**	Examples
$O(1)$	$O(1)$	YES	YES	**YES**	trivial problems
$\omega(1)$, $o(\log^* n)$	$\omega(1)$, $o(\log^* n)$	?	NO [8]	**NO**	?
$\Theta(\log^* n)$	$\Theta(\log^* n)$	YES	YES	**NO**	$(\Delta+1)$-coloring [17, 24]
$\Theta(\log n)$	$\Theta(\log \log n)$	YES	YES	**YES**	sinkless orientation [10, 15, 22]
$\Theta(\log n)$	$\Theta(\log n)$	YES	YES	**YES**	even orientation (Ex. 4.4)
$\Theta(n^{1/k})$	$\Theta(n^{1/k})$	YES	NO [8]	**NO**	$2\frac{1}{2}$-coloring [16]
$\Theta(n)$	$\Theta(n)$	YES	NO [8]	**YES**	2-coloring (Ex. 4.5)

The case of $|\Sigma| = 2$ that we studied here is the smallest nontrivial case, but there is another reason that makes the case of $|\Sigma| = 2$ interesting: as mentioned above, we have seen that the *complexity class* $\Theta(\log^* n)$ *disappears* when we go from $|\Sigma| = 3$ down to $|\Sigma| = 2$.

There are numerous symmetry-breaking problems that can be solved in $\Theta(\log^* n)$ rounds in graphs of maximum degree $\Delta = O(1)$. Examples include maximal matching, maximal independent set, minimal dominating set, $(\Delta+1)$-vertex coloring, $(2\Delta-1)$-edge coloring, weak 2-coloring, and many variants of these problems.

It is known that with 3 edge labels we can encode e.g. the problem of finding a maximal matching [3]; hence as soon as $|\Sigma| \geq 3$, there are edge labeling problems solvable in $\Theta(\log^* n)$ rounds. However, previously it was not known if the use of $|\Sigma| \geq 3$ labels was necessary in order to encode any such problem. After all, the use of $|\Sigma| \geq 3$ labels seems unnatural when we consider problems such as maximal matching, maximal independent set, and weak 2-coloring, all of which in essence ask one to find a subset of edges or a subset of nodes subject to local constraints.

The results of this work imply that none of these problems can be encoded with two labels. We have showed that for binary labeling problems, there is a gap in the deterministic complexity between $\omega(1)$ and $o(\log n)$. Hence binary labeling problems give rise to a different landscape of computational complexity in comparison with any other number of labels. We conjecture that there is no such qualitative difference between e.g. alphabets of size 3 or 4.

References

1 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.

2 Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems, 2019. arXiv:1911.13294.

3 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.

4 Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed Lower Bounds for Ruling Sets. In *Proc. 61st IEEE Symposium on Foundations of Computer Science (FOCS 2020)*. IEEE, 2020. arXiv:2004.08282.

5 Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the LOCAL model. In *Proc. 32nd International Symposium on Distributed Computing (DISC 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 9:1–9:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.DISC.2018.9.

6 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)*. ACM Press, 2020. doi:10.1145/3382734.3405715.

7 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.

8 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.

9 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 Press, 2019. doi:10.1145/3293611.3331611.

10 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.

11 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.

12 Sebastian Brandt, Yannic Maus, and Jara Uitto. A sharp threshold phenomenon for the distributed complexity of the lovász local lemma. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 389–398. ACM Press, 2019. doi:10.1145/3293611.3331636.

13 Sebastian Brandt and Dennis Olivetti. Truly Tight-in-Δ Bounds for Bipartite Maximal Matching and Variants. In *Proc. 39th ACM Symposium on Principles of Distributed Computing (PODC 2020)*. ACM Press, 2020. doi:10.1145/3382734.3405745.

14 Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. The Complexity of Distributed Edge Coloring with Small Palettes. In *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 2633–2652. Society for Industrial and Applied Mathematics, 2018. doi:10.1137/1.9781611975031.168.

15 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. In *Proc. 57th IEEE Symposium on Foundations of Computer Science (FOCS 2016)*, pages 615–624. IEEE, 2016. doi:10.1109/FOCS.2016.72.

16 Yi-Jun Chang and Seth Pettie. A Time Hierarchy Theorem for the LOCAL Model. *SIAM Journal on Computing*, 48(1):33–69, 2019. doi:10.1137/17M1157957.

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

18 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)*, pages 18:1–18:16, 2017. doi:10.4230/LIPIcs.DISC.2017.18.

19 Mohsen Ghaffari, David G Harris, and Fabian Kuhn. On Derandomizing Local Distributed Algorithms. In *Proc. 59th IEEE Symposium on Foundations of Computer Science (FOCS 2018)*, pages 662–673, 2018. doi:10.1109/FOCS.2018.00069.

20 Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, Yannic Maus, Jukka Suomela, and Jara Uitto. Improved distributed degree splitting and edge coloring. In *Proc. 31st International Symposium on Distributed Computing (DISC 2017)*, volume 91 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.DISC.2017.19.

21 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proc. 49th ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 784–797. ACM Press, 2017. doi:10.1145/3055399.3055471.

22 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.

23 Janne H Korhonen and Jukka Suomela. Towards a complexity theory for the congested clique. In *Proc. 30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2018)*, pages 163–172. ACM Press, 2018. doi:10.1145/3210377.3210391.

24 Nathan Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

25 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.

26 Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991. doi:10.1137/0404036.

27 Moni Naor and Larry Stockmeyer. What Can be Computed Locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

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

29 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. doi:10.1137/1.9780898719772.

30 Will Rosenbaum and Jukka Suomela. Seeing Far vs. Seeing Wide: Volume Complexity of Local Graph Problems. In *Proc. 39th ACM Symposium on Principles of Distributed Computing (PODC 2020)*. ACM Press, 2020. doi:10.1145/3382734.3405721.

31 Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-Time Deterministic Network Decomposition and Distributed Derandomization. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC 2020)*, 2020. doi:10.1145/3357713.3384298.