Classification of distributed binary labeling problems

Alkida Balliu · alkida.balliu@aalto.fi · Aalto University

Sebastian Brandt · brandts@ethz.ch · ETH Zurich

Yuval Efron · szxrtde@cs.technion.ac.il · Technion

Juho Hirvonen · juho.hirvonen@aalto.fi · Aalto University

Yannic Maus · yannic.maus@campus.technion.ac.il · Technion

Dennis Olivetti · dennis.olivetti@aalto.fi · Aalto University

Jukka Suomela · jukka.suomela@aalto.fi · Aalto University

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.

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 [26]. 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 [5, 8, 27]; we will give the precise definition in Section 3.

Contributions.   In this work, we focus on *deterministic* distributed algorithms in the LOCAL model of distributed computing, 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 [9, 12, 18].
- 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 [14, 23], and it can be encoded in the edge labeling formalism using an alphabet of size three [5]—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 2, 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* [8, 27]. Previously, it was known that sinkless orientation is a nontrivial *fixed point* for round elimination [9]—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.

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 a brief discussion of the general landscape of distributed computational complexity in Section 2, and then give formal definitions of binary labeling problems in Section 3. We present a summary of our results in Section 4—our main contribution is the characterization of all binary labeling problems in Table 2. We explain the details of the model of computing in Section 5. All algorithms and lower bound proofs related to deterministic complexity are presented in Sections 6–10:

- Section 6: unsolvable problems,
- Section 7: complexity class $O(1)$,
- Section 8: complexity class $\Theta(n)$,
- Section 9: $O(\log n)$ upper bounds,
- Section 10: $\Omega(\log n)$ lower bounds.

We conclude with a discussion of randomized complexity in Section 11.

2 Background and related work

Locality and the LOCAL model. A fundamental concept in distributed computing is *locality*. In brief, the locality of a graph problem is T if all nodes of a graph can produce their own part of the solution by only looking at their radius-T neighborhoods.

The idea of locality is formalized as the time complexity in the *LOCAL model* of distributed computing [23, 28]. In brief, a graph problem Π can be solved in graph family $\mathcal{F}$ in time $T(n)$ in the LOCAL model if there is a mapping $\mathcal{A}$ from radius-$T(n)$ neighborhoods to local outputs such that for any graph $G \in \mathcal{F}$ with n nodes and for any assignment of polynomially-sized unique identifiers in G we can apply mapping $\mathcal{A}$ to each node of G, and the local outputs form a feasible solution for Π. Here $\mathcal{A}$ is a deterministic distributed algorithm in the LOCAL model.

Note that if we interpret graph G as a computer network in which nodes can send messages to their neighbors, then time becomes equal 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*.

Our main focus is on deterministic distributed algorithms, but we will also discuss randomized distributed algorithms; the only difference there is that in a randomized algorithm the nodes are labeled not only with unique identifiers but also with an unbounded stream of random bits. We will define the model of computing in more detail in Section 5.

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. 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* [2–4, 6, 9–12, 15, 19–21, 29, 30].

The key example is the family of *locally checkable labeling* problems (LCLs), introduced by Naor and Stockmeyer [26]. 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 $o(\log n)$ [12].

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 [10, 26]. 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 [4, 10, 26], and some questions related to the complexity of LCLs in trees are also decidable [11]—unfortunately, we run into PSPACE-hardness already in the case of paths and cycles [4].

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.

3 Binary labeling problems

We will now give the formal definition of the family of binary labeling problems, together with some natural examples of graph problems that are contained in this family. LCL problems have been traditionally specified by listing a *collection of permitted local neighborhoods* [26]. However, we will use a more recent *edge labeling* formalism [5, 8, 27], 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 [8, 27]:

- 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.

Now if we set $|\Sigma| = 2$, we arrive at the definition of binary labeling problems.

3.1 General form

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

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

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. We use the term *relevant nodes* to refer to white nodes of degree d and black nodes of degree δ.

Example 3.1 (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.

3.2 Equivalence

If a is a natural number and A is a set of natural numbers, we will define

$$a - A = \{a - x : x \in A\}.$$

Observation 3.2. *The following binary labeling problems have the same distributed complexity up to ± 1 rounds:*

$$\begin{aligned}
\Pi_{00} &= (d, \delta, W, B), \\
\Pi_{01} &= (\delta, d, B, W), \\
\Pi_{10} &= (d, \delta, d - W, \delta - B), \\
\Pi_{11} &= (\delta, d, \delta - B, d - W).
\end{aligned}$$

Proof. Given an algorithm for Π_{x0}, we get an algorithm for Π_{x1} by exchanging the roles of black and white nodes. Given an algorithm for Π_{0x}, we get an algorithm for Π_{1x} by replacing solution X with its complement $E \setminus X$.

When we exchange the roles of black and white nodes, we may need to spend one additional round so that black nodes can inform white nodes of the final output or vice versa; see Section 5 and Observation 5.2 for more details. $\square$

Observation 3.2 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*.

3.3 Restrictions and relaxations

Definition 3.3. Given two problem $\Pi = (d, \delta, W, B)$ and $\Pi' = (d, \delta, W', B')$, we say that Π' is a *restriction* of Π and Π is a *relaxation* of Π' if

$$W' \subseteq W, \quad B' \subseteq B.$$

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

Observation 3.4. *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.*
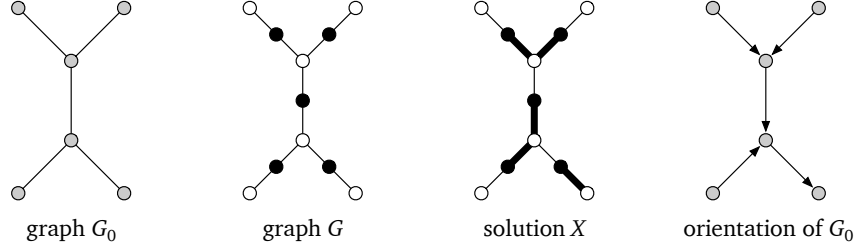
Figure 1: Binary labeling problems with $B = 010$ are orientation problems.

graph G_0 graph G solution X orientation of G_0

3.4 Vector notation

It is convenient to interpret set W as a bit vector $w_0 w_1 \ldots w_d$ 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 3.1b. Using this notation, the problem of Example 3.1 can be represented as follows:

$$W = 01110, \quad B = 01110,$$

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 $W \in \{010, 0110, 01110, \ldots\}$.

3.5 Important special case: $\delta = 2$

At first the bipartite setting 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 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 $V \setminus V_0$.

Now 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 indegree$(v) \in W$.

Example 3.5 (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 3.5b. Let $W = 0111^+$ and $B = 010$. By Observation 3.2, this is equivalent to Example 3.5. In essence, we have merely reversed the roles of heads and tails—now the task is to find a sourceless orientation.
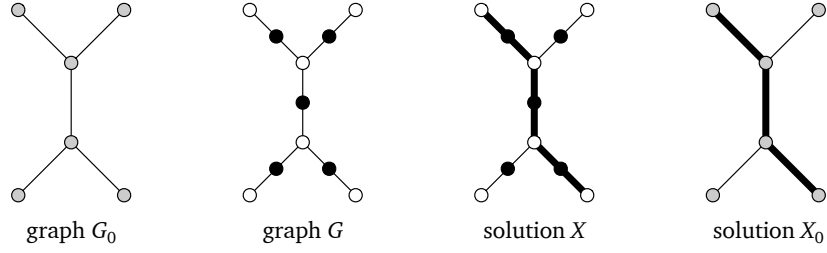
5

graph G_0 graph G solution X solution X_0

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

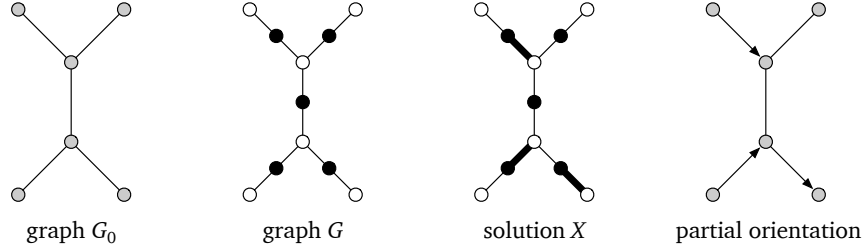graph G_0 graph G solution X partial orientation

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

Example 3.6 (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 3.5; recall Definition 3.3.

Example 3.7 (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 3.8 (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.

Example 3.9 (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 3.10 (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 3.2.

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 3.11 (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 3.12 (trivial). Let $W = 0010^+$ and $B = 111$. Here all white nodes can arbitrarily choose two incident edges and add them to X.

Remark. 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.

3.6   Our focus: trees

Throughout this work, we will study binary labeling problems in trees. Binary labeling problems as such make sense also in general graphs, but as our aim is at understanding LCL problems on trees, we will assume from now on that our input graph is a tree (a simple, connected, acyclic graph).

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 3.9. 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).

3.7   Why exactly this family?

As we have discussed above, our goal here is to initiate a systematic investigation of the distributed complexity of LCL problems on trees, and in this work we will focus on a specific family of LCL problems, binary labeling problems. We will now briefly discuss reasons that led to the choice of this specific family of LCL problems for our work.

Sinkless orientation is a binary labeling problem.   Sinkless orientation (recall Example 3.5) is one of the cornerstones of the modern theory of distributed computational complexity. This problem was introduced in our context in 2016 [9], 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; we will 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 [9, 12, 18]. 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

Deterministic complexity	Randomized complexity	General LCLs	Homogeneous LCLs	**Binary labeling**	Examples
$O(1)$	$O(1)$	YES	YES	**YES**	trivial problems
$\omega(1)$, $o(\log^* n)$	$\omega(1)$, $o(\log^* n)$	?	NO [7]	**NO**	?
$\Theta(\log^* n)$	$\Theta(\log^* n)$	YES	YES	**NO**	$(\Delta + 1)$-coloring [14, 23]
$\Theta(\log n)$	$\Theta(\log \log n)$	YES	YES	**YES**	sinkless orientation [9, 12, 18]
$\Theta(\log n)$	$\Theta(\log n)$	YES	YES	**YES**	even orientation (Example 3.7)
$\Theta(n^{1/k})$	$\Theta(n^{1/k})$	YES	NO [7]	**NO**	$2\frac{1}{2}$-coloring [11]
$\Theta(n)$	$\Theta(n)$	YES	NO [7]	**YES**	2-coloring (Example 3.8)

Table 1: An overview of possible distributed time complexities in trees for different classes of LCLs.

orientation has the same complexity as sinkless orientation; the upper bound is nontrivial [18], but the lower bound trivially comes from the observation that a sinkless and sourceless orientation gives a sinkless orientation. We are 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 do not know if all problems in this complexity class are merely extensions and variants of the same problem!

One of the goals of the present work is to systematically explore a large family of LCL problems that contains the sinkless orientation problem as well as many closely related problems. Our definition of binary labeling problems is guided by this insight: it is large enough to contain many such problems, yet restrictive enough so that there is hope of understanding the distributed complexity of all problems in this family.

As we will see in this work, the study of binary labeling problems sheds new light on this issue. We will encounter 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.

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 is on the latter case.

In the definition of binary labeling problems, the idea of focusing on regular parts is simply 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* [7]. The family of problems that we study in the present work is similar in spirit to homogeneous LCLs, but there is a key difference: all homogeneous LCLs can be solved in $O(\log n)$ rounds in trees, but there are binary labeling problems with distributed time complexity $\Theta(n)$; see Table 1.

Automatic round elimination and fixed points. As we mentioned in Section 3, the edge labeling formalism makes it easy to apply the automatic round elimination technique [8, 27], which is an effective technique for proving lower bounds [5, 7–9, 23, 25]. However, there are still many

open research questions related to round elimination itself. One of the key questions is the existence of fixed points; here the sinkless orientation problem is a good example.

Let Π be the sinkless orientation problem from Example 3.5. 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 [8, 27] 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 3.6. 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.

The family of binary labeling problems is chosen so that it contains both fixed points as well as closely related problems that are not fixed points. As we will see in this work, through a systematic study of all binary labeling problems we are able to discover new nontrivial fixed points.

Restriction to two labels and $\Theta(\log^* n)$ complexity. In the 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.

The case of $|\Sigma| = 2$ that we study here is the smallest nontrivial case, but there is another reason that makes the case of $|\Sigma| = 2$ interesting: we will show 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 [5]; 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.

In this work we will show that none of these problems can be encoded with two labels. We show 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.

Type	Problem family	White constraint	Black constraint	**Deterministic complexity**	Upper bound	Lower bound
I	I.a	100^+	$0**^+$	**unsolvable**		Section 6
	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^+	$O(1)$	Section 7	
	III.b	111^+	non-empty			
IV	IV.a	$1**^+$	$1**^+$			
	IV.b	$**^+1$	$**^+1$			
V	V.a	10^+1	010	$\Theta(n)$	Section 8	Section 8
	V.b	010	10^+1			
VI	VI.a	0^+1*	$*10^+$			
	VI.b	$*10^+$	0^+1*			
VII	VII.a	all other cases		$\Theta(\log n)$	Section 9	Section 10

Table 2: The deterministic distributed complexity of binary labeling problems in trees; 1^+ signifies one or more 1s, and $*$ signifies either 0 or 1. 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.

4 Our contributions

4.1 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 2. Given a binary labeling problem Π, one can simply do pattern matching in this table to first find its problem family and this way also find its deterministic complexity.

For easier reference, we listed all problem families explicitly, but if we take into account the fact that e.g. all problems of family I.a are equivalent to a problem in family I.b and vice versa (recall Observation 3.2), we can partition problems in seven *types*, labeled with **I, II, . . . , VII** in the table.

All proofs and algorithms are presented in Sections 6–10: Section 6 covers the complexity of problems of types **I** and **II**, Section 7 covers types **III** and **IV**, and Section 8 covers types **V** an **VI**. Now by definition all problems that are not of type **I**–**VI** are of type **VII**; in Section 9 we prove that all of them are solvable in $O(\log n)$ rounds and in Section 10 we prove a matching lower bound of $\Omega(\log n)$ for all of these problems. This will then complete the proof that the classification of Table 2 is correct.

The first two parts of Table 2 are trivial:

- 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,

Deterministic complexity	Type	Examples of problems			
		W	B	reference	description
unsolvable	**I**	011^+	100	Example 3.11	contradiction
$O(1)$	**III**	0010^+	111	Example 3.12	trivial
$\Theta(n)$	**V**	10^+1	010	Example 3.8	two-coloring
$\Theta(\log n)$	**VII**	01110	01110	Example 3.1	bipartite splitting
		1010	010	Example 3.7	even orientation
		111^+0	010	Example 3.5	sinkless orientation
		011^+0	010	Example 3.6	sinkless and sourceless orientation
		0100^+	101	Example 3.9	regular matching
		011^+0	101	Example 3.10	splitting
		0100^+	0100^+	Definition 9.3	bipartite matching
		0100^+	10^+1	Definition 9.4	hypergraph matching
		0100^+	110	Definition 9.5	edge grabbing
$\Omega(\log n), O(n)$	**V** or **VII**	1^+01^+	01^+0	Definition 10.2	forbidden degree
	VI or **VII**	11^+0	011^+	Definition 10.3	bipartite sinkless orientation

Table 3: Examples of binary labeling problems and problem families and their deterministic complexities.

> 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 2, 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 4.1. *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 3 shows examples of problems in each complexity class.

4.2 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 Sections 6–10 have also direct implications on randomized complexity. We will summarize all results related to randomized complexity in Section 11, and have a deeper look at questions that are specific to randomized complexity there.

By prior work, it is known that classes $O(1)$ and $\Theta(n)$ remain the same also for randomized complexity—this follows from [20] and (unpublished) extensions of the result in [2]. 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 3.5).

- Deterministic complexity $\Theta(\log n)$ and randomized complexity $\Theta(\log n)$ (e.g. even orientation, Example 3.7).

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.

5   Model of computing

There are two equivalent perspectives that we can use to reason about locality in distributed algorithms:

1. Each node sees its radius-T neighborhood in the input graph and uses this information to choose its own part of solution.

2. The input graph is a computer network and initially each node is only aware of its own input. Nodes can exchange messages with each other for T communication rounds and then they will stop and announce their own part of the solution.

In this work we will use the second approach to formalize the model of computing. Our formalism is equivalent to the usual LOCAL model of distributed computing [23, 28], up to some additive constants in the number of communication rounds.

   We will first define deterministic distributed algorithms in the port-numbering model, which is a rather restricted model of computing. Then we will add unique identifiers to arrive at the deterministic LOCAL model, and finally randomness to arrive at the randomized LOCAL model.

Port-numbering model.   Given a graph $G = (V, E)$, define the set of *ports* by

$$P = \big\{(v, i) : v \in V, i \in \{1, 2, \ldots, \deg(v)\}\big\}.$$

We say that $p \colon P \to P$ is a *port numbering* of G if for each edge $\{u, v\} \in E$ there are natural numbers $1 \le i \le \deg(u)$ and $1 \le j \le \deg(v)$ such that $p(u, i) = (v, j)$ and $p(v, j) = (u, i)$. We use e.g. the following terminology:

- Port i of node u is connected to port j of node v.
- Port i of node u is connected to node v.
- Node v is the ith neighbor of node u.

A *port-numbered network* is a graph G together with a port numbering p. We can interpret a port-numbered network as a computer network in which a node v is a computer with $\deg(v)$ communication ports and e.g. $p(u, i) = (v, j)$ indicates that if node u sends a message to communication port i, it is received by node v from its communication port j, and vice versa.

   Computation in a port-numbered network (G, p) proceeds as follows:

- All nodes *initialize* their local states. The initial state can only depend on the *local input* of the node and on its degree.

- Then we repeat *synchronous communication rounds* until all nodes have stopped. In each round:

    1. All nodes that have not stopped yet choose what messages to send to each of their communication ports.

    2. Messages are forwarded along the edges to the recipients.

    3. All nodes that have not stopped yet update their local states based on the messages that they received. Optionally, a node may choose to stop and announce its *local output*.

A *distributed algorithm* $\mathcal{A}$ in this model is simply a collection of three procedures: one for initialization, one for constructing outgoing messages, and one for doing state updates after receiving messages.

White and black algorithms. We say that $\mathcal{A}$ is a *white algorithm* that solves an edge labeling problem Π in time $T(n)$ if the following holds:

- We can take any graph $G = (V, E)$ and any two-coloring $f \colon V \to \{\text{black}, \text{white}\}$ of G.

- We can take any port numbering p of G.

- If we run $\mathcal{A}$ in port-numbered network (G, p) with local inputs f, all nodes will stop after at most $T(|V|)$ communication rounds and announce their local outputs.

- For each white node u the local output specifies the output label for each incident edge (as a vector of $\deg(u)$ output labels from some alphabet Σ, ordered by port numbers).

- For each black node v the local output is empty.

- The local outputs of the white nodes constitute a feasible solution for Π, i.e., they satisfy all white constraints and all black constraints.

In particular, if Π is a binary labeling problem $\Pi = (d, \delta, W, B)$, we require that the outputs of the white nodes can be interpreted as a subset of edges $X \subseteq E$ such that X satisfies both white and black constraints.

We say that Π has *white complexity* T in the port-numbering model if T is the pointwise minimum of all functions T' such that there exists a white algorithm $\mathcal{A}$ that solves Π in time T'.

We define a *black algorithm* in analogous manner: black nodes produce output labels for their incident edges and white nodes produce empty outputs. *Black complexity* is the minimum of the running times of all black algorithms.

Observation 5.1. *If we have $\Pi = (d, \delta, W, B)$ and $\Pi' = (\delta, d, B, W)$, then a white algorithm for Π can be interpreted as a black algorithm for Π' and vice versa.*

Observation 5.2. *Black complexity of problem Π and white complexity of the same problem Π differ by at most 1 round.*

Proof. Given a white algorithm $\mathcal{A}$ that solves Π in time T, we can construct a black algorithm $\mathcal{A}'$ that solves Π in $T + 1$ rounds: run $\mathcal{A}$ for T rounds, and then use 1 round to have all white nodes inform their black neighbors about the final output. Conversely, given a white algorithm $\mathcal{A}$ that solves Π in time T, we can construct a white algorithm $\mathcal{A}'$ that solves Π in $T + 1$ rounds. □

Deterministic LOCAL model. In the LOCAL model we augment the port-numbering model with unique identifiers. We assume that if there are n nodes in the graph, each node is labeled with a unique integer from $\{1, 2, \ldots, \text{poly}(n)\}$. This is part of the local input, and a distributed algorithm can use it when it initializes the local state of a node. Other than that, computation proceeds exactly in the same way as in the port-numbering model.

We say that $\mathcal{A}$ is a deterministic white algorithm in the LOCAL model for problem Π if it solves Π correctly for any assignment of unique identifiers. We define black algorithms, deterministic white complexity in the LOCAL model, and deterministic black complexity in the LOCAL model in the analogous way.

For brevity, when we refer to a (deterministic) algorithm or (deterministic) complexity, we mean deterministic white algorithms in the LOCAL model.

Randomized algorithms. Finally, we can let state transitions be probabilistic, and we arrive at randomized algorithms. We can define e.g. randomized white algorithms in the LOCAL model in the natural manner. In this work, randomized algorithms are Monte Carlo algorithms that are correct w.h.p. in the size of the graph.

For brevity, when we refer to a randomized algorithm or randomized complexity, we mean randomized white algorithms in the LOCAL model.

Simulations. As discussed in Section 3.5, it is often convenient to e.g. interpret a graph G_0 as a bipartite graph G by subdividing edges. Now if we have an algorithm $\mathcal{A}$ that we can apply in G, we can easily simulate $\mathcal{A}$ in G_0.

There are some technical details that we need to take into account. A port numbering of G induces a port numbering of G_0. However, a port numbering of G_0 does not directly give port numbers for the black nodes of G. However, we are usually interested in algorithms for the LOCAL model of computing, and we can use the unique identifiers to choose the port numbers for all nodes of G (e.g., port 1 points towards the neighbor with the smallest identifier).

Each node of the (virtual) network G has to be simulated by some node of the (physical) network G_0. In particular, each black node has to be simulated by one of its white neighbors. We can use the port numbers or unique identifiers to choose who simulates what.

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.

6   Unsolvable problems

In this section we prove that any binary labeling problem contained in the following families is unsolvable:

- I.a: $W = 100^+$, $B = 0**^+$,
- I.b: $W = 00^+1$, $B = **^+0$,
- I.c: $W = 0**^+$, $B = 100^+$,
- I.d: $W = **^+0$, $B = 00^+1$,

- II.a: $W = 000^+$, $B = ***^+$,
- II.b: $W = ***^+$, $B = 000^+$.

This completes the classification in the first section of Table 2, i.e., the classification of type I and II. We refer to these problems as *unsolvable* problems but we emphasize that this section does

not contain a proof that there are no other unsolvable problems. We will only learn this later as Sections 7–9 will show that all problems of type **III**–**VII** are solvable.

Theorem 6.1. *All binary LCL problems that are contained in any of the above families cannot be solved on any tree meeting the degree requirements of the respective problem.*

Proof. The proof will consider two cases.

- **Families I.a, I.b, I.c, I.d:** Fix problem $\Pi = (d, \delta, W, B) \in$ I.c with $W = 01^d$, $B = 10^\delta$. Then Π is *easier* than any problem Π' in the family I.c, i.e., $\Pi' \subseteq \Pi$; furthermore any problem in family I.a, I.b or I.d is equivalent (as defined in Subsection 3.2) to some problem in I.c; thus Π is the *easiest* problem in the considered families and if Π is unsolvable any problem in all the families is unsolvable.

  We now show that Π is unsolvable. Given some tree $G = (V, E)$ that meets the degree requirements of Π, assume there is a valid solution to Π in G and consider some white node $v \in V$. Since white nodes are not allowed to have their incident edges labeled all 0, there must be an incident edge labeled 1, that we denote by $e = \{v, u\}$. However, this means that the black node u has the incident edge e labeled with 1. This contradicts our assumption of having a valid solution to Π, since black nodes are only allowed to label all of their incident edges with 0. Thus Π is unsolvable and the proof is complete.

- **Families II.a, II.b:** Fix problem $\Pi = (d, \delta, W, B) \in$ II.b with $W = 1^{d+1}$, $B = 0^{\delta+1}$. Similar to the reasoning in the previous case it is sufficient to show that Π is unsolvable to deduce that any problem in the two families is unsolvable. Given some tree $G = (V, E)$, assume that Π can be solved on G and consider some black node $v \in V$. Denote by i the number of edges incident to v that are labeled with 1. Clearly we have $0 \leq i \leq d$. However, no matter the value of i, black nodes are not allowed i incident edges labeled with 1 in their solution, thus we arrive at a contradiction. Thus Π is unsolvable and the proof is complete. $\square$

7 Constant-time problems

In this section we will prove that any binary labeling problem in the following families can be solved in constant time:

- III.a: $W \neq 000^+$, $B = 111^+$,
- III.b: $W = 111^+$, $B \neq 000^+$,

- IV.a: $W = 1**^+$, $B = 1**^+$,
- IV.b: $W = **^+1$, $B = **^+1$.

This completes the classification in the second section of Table 2, i.e., the classification of types **III** and **IV**. We refer to these problems as *trivial* problems but we emphasize that this section does not contain a proof that there are no other trivial problems, i.e., that there are not other problems solvable in time $O(1)$. We will only learn this later as Section 6 showed that problems of type **I** and **II** are unsolvable and Section 8 and Section 10 will show that all problems of type **V**–**VII** have $\omega(1)$ lower bounds.

Theorem 7.1. *All binary LCL problems that are contained in any of the above families can be solved in constant time on any tree meeting the degree requirements of the respective problem.*

Proof. The proof considers two cases.

15

- **Families III.a, III.b:** Any problem $\Pi' \in$ III.b has an equivalent problem in III.a. Thus it is sufficient to show that any problem in III.a can be solved in constant time. Let $\Pi \in$ III.a be a problem with white degree d. Let $G = (V, E)$ be some input tree. We know that for Π, it holds that $W \neq 0^d$. Thus there exists $0 \leq i \leq d$ such that $w_i = 1$. Thus Π allows white nodes to label i of their incident edges with 1. Now, each white node arbitrarily picks i incident edges and labels them with 1 and labels all other of its incident edges with 0. Thus the labeling of the incident edges around any white node is according to what W allows. The labeling is also feasible for every black node as $B = 111^+$ says that any labeling is allowed for black nodes. Clearly this takes constant time.

- **Families IV.a, IV.b:** Any problem in IV.a can be solved by labeling all edges with 0 and any problem in IV.b can be solved by labeling all edges with 1. Clearly these are constant time algorithms. $\qquad\square$

8 Global problems

In this section we will prove that any binary labeling problem in the following families can be solved in $O(n)$ rounds and they also require $\Omega(n)$ rounds:

- V.a: $W = 10^+1$, $B = 010$,
- V.b: $W = 010$, $B = 10^+1$,

- VI.a: $W = 0^+1*$, $B = *10^+$,
- VI.b: $W = *10^+$, $B = 0^+1*$.

This completes the classification in the third section of Table 2, i.e., the classification of types **V** and **VI**. We refer to these problems as *global* problems but we emphasize that this section does not contain a proof that there are no other global problems. We only learn this because previous sections showed that problems of type **I**–**IV** are either unsolvable or trivial and Section 9 will show that problems of **VII** can be solved in $O(\log n)$ rounds.

We begin with showing that V.a and V.b have complexity $\Theta(n)$

Theorem 8.1. V.a *and* V.b *have complexity* $\Theta(n)$.

Proof. V.b is equivalent to V.a thus we only need to show that V.a has complexity $\Theta(n)$ to prove the theorem. Recall that $\mathsf{V.a} = (d, 2, W, B)$ with $W = 10^+1$ and $B = 010$. As explained in Section 3.5, since the black degree δ equals 2, we can treat the problem as an orientation problem on a given d-regular tree $G = (V, E)$ in which all nodes are white. All nodes have to satisfy the constraint string $W = 10^+1$, thus, for all nodes v, the resulting orientation of the edges must satisfy either indegree$(v) = 0$ or indegree$(v) = d$, where indegree(v) is the number of edges incident to v which are oriented towards v. Now consider two adjacent nodes of degree d in G, denoted by v, u. A key observation is that in any valid orientation of the edges, it holds that indegree$(v) = 0$ implies indegree$(u) = d$ and vice versa. Now, if we treat indegree$(v) = d$ and indegree$(v) = 0$ as two colors, we can deduce that the subgraph of G induced by nodes of degree d has to be properly colored with 2 colors in any valid solution to V.a. We use this observation to show that the V.a has complexity $\Omega(n)$. Consider the tree obtained by starting from a path P and connecting to each node $d - 2$ new nodes. Also, connect an additional node to each endpoint of the path to make the tree d-regular. In a tree of n nodes, this path P has $\Omega(n)$ nodes. Note that each node in the path has degree d and thus must satisfy the constraint of W. Thus, in any valid solution for V.a, the nodes of P must be properly 2-colored. Thus any algorithm A solving V.a in T rounds would be able to solve

the problem of 2-coloring a path properly in $O(T)$ rounds by simply letting each node in a given path simulate A with some additional virtual nodes connected to it as leaves. It is well known that any algorithm solving 2-coloring on a path has complexity $\Omega(n)$. Thus we can deduce that any algorithm solving V.a has complexity $\Omega(n)$, as required.

To show that V.a can also be solved in $O(n)$ rounds it is sufficient that V.a is solvable on any d-regular tree. Any tree is 2-colorable and orienting all edges from one color family to the other solves the problem. $\qquad\square$

As any problem in VI.b is equivalent to a problem in VI.a we will only show that the easiest problem in VI.b has complexity $\Omega(n)$ to prove the following theorem.

Theorem 8.2. *Any problem in family* VI.a *or* VI.b *has complexity* $\Omega(n)$.

Proof. Any problem in VI.a is equivalent to a problem in VI.b. Thus it is sufficient to show that the *easiest* problem in VI.b has complexity $\Omega(n)$ to show that all problems in VI.b and VI.a have complexity $\Omega(n)$. Let $\Pi = (d, \delta, W, B) \in$ VI.b where $W = 110^+$, $B = 0^+11$. Problem Π is the easiest problem in VI.b as $\Pi' \subseteq \Pi$ for any problem $\Pi' \in$ VI.b. We show that Π has complexity $\Omega(n)$.

Let $X \subseteq E$ be the set of marked edges. The set X defines an orientation of the edges, given by orienting edges in X from black to white nodes, and edges in $E \setminus X$ from white to black nodes. Similarly, an orientation defines an assignment for Π. It is easy to see that Π requires to orient edges such that nodes have either 0 or 1 incoming edges.

We now prove that there can be at most a single vertex in the graph having all edges oriented outgoing, implying that Π is equivalent to orienting a tree from some root, namely, to the *oriented tree* problem. Assume towards a contradiction that there are two nodes u, v having all outgoing edges. Denote by P the unique path between u, v in the tree. Starting from u, denote by w the first vertex on P such that the edges incident to w that are part of P are not both oriented towards v. Such a vertex must exist, because the incident edge of v on P is not directed towards v, as all its edges are oriented outgoing. Since w is the first node satisfying the condition, both the incident edges of w on P are oriented towards w, thus w has two incoming edges, giving a contradiction.

All that is left to do, is to prove that the *oriented tree* problem is indeed global. We reduce another problem, namely the *almost oriented path* problem, to the oriented tree problem, and then we show that the almost oriented path problem is hard. The almost oriented path problem requires to orient the edges of a properly 2-colored path such that at most one node of degree 2 has all outgoing edges, and all other nodes of degree 2 have exactly one outgoing edge (nodes of degree 1 are unconstrained).

Now, let us show that an algorithm for the oriented tree problem can be used to solve the almost oriented path problem. In particular, we show how to locally turn an instance of one problem to the other. Start from a properly 2-colored path, and connect to each white node $d - 2$ new black nodes, and to each black node $\delta - 2$ new white nodes. Then, connect an additional node to each endpoint of the path, preserving a proper 2-coloring. Clearly, a solution for the oriented tree problem for this instance can be directly mapped to a solution for the almost oriented path problem on the original instance.

We now prove that the almost oriented path problem requires $\Omega(n)$ time. Assume by contradiction that there is a $T = o(n)$ algorithm A that solves the problem, and consider an instance P of size n such that $T(n) < n/100$. Let u, v be the endpoints of the path. Depending on the output of A, we can give *types* to nodes. A node is of type L if both its incident edges are oriented towards u. A node is of type R if both its incident edges are oriented towards v. Otherwise (if both edges are outgoing), it is of type C. Consider four nodes v_1, v_2, v_3, v_4 at distance at least $3T$ between each other and from the endpoints of the path, where v_1 is the nearest to u among them, and v_4

17

is the nearest to v. Notice that at least one of the two following statements must hold: both v_1 and v_2 are of type L, or both v_3 and v_4 are of type R, since otherwise there must be a node in the path having two incoming edges. Let assume w.l.o.g. that we are in the first case. Let Q be the subpath of length $2T + 1$ centered at node v_1. We now construct a new path P', by starting from P and replacing Q by its mirror. The algorithm A, while running on v_1 and v_2, can not distinguish the instances P and P', thus it must output the same on both instances, but since Q has been mirrored, this implies that v_1 on P' has type R. Since in P' the node v_1 has type R and v_2 has type L, this implies that there is some node in the subpath between v_1 and v_2 having two incoming edges, contradicting the correctness of A. $\qquad\square$

Theorem 8.3. *Any problem in family* VI.a *or* VI.b *has complexity* $O(n)$.

Proof. Let $\Pi = (d, \delta, W, B)$ where $W = 010^+$, $B = 0^+10$. Any problem in VI.a is equivalent to a problem in VI.b and $\Pi \in$ VI.b is the *hardest* problem in VI.b, i.e., $\Pi \subseteq \Pi'$ for all $\Pi' \in$ VI.b. Thus it is sufficient to show that Π can be solved in $O(n)$ rounds.

Since any problem that is solvable can be solved in $O(n)$ rounds, it is enough to show that a solution for Π always exists in any tree that meets the degree requirements of Π. It is clear that the problem described in Theorem 8.2 is always solvable, since it is always possible to orient a tree towards a root. The current problem Π is slightly harder, since it does not allow any white node of degree d and any black node of degree δ to have all incoming edges. We can solve this problem anyway, by choosing a leaf of the tree as a root, i.e., a node with degree 1 and orient all edges towards the leaf. $\qquad\square$

9 Logarithmic upper bounds

In this section we will show that all binary labeling problems that are not of the type discussed in Section 6, Section 7, or Section 8 can be solved in $O(\log n)$ rounds with deterministic distributed algorithms. This will give the upper bounds for the final section of Table 2. In this section, we formally prove the following theorem.

Theorem 9.1. *Any problem that is not of type* I, II, III, IV, V *or* VI *has complexity* $O(\log n)$.

To prove Theorem 9.1, we first introduce the concept of *resilience*, and three families of binary labeling problems: *bipartite matching*, *hypergraph matching*, and *edge grabbing*. In Lemma 9.6 we will show that all problems that we have not yet covered in previous sections are resilient or they are contained in one of the three families. Later, in Section 9.6, we will show that resilient problems are solvable in $O(\log n)$ rounds, and finally in Sections 9.7–9.9 we show that all problems in the three families can also be solved in $O(\log n)$ rounds.

9.1 Definitions

Definition 9.2 (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 $0^{\delta+1-s}$.

Remark. Recall the *even orientation* problem from Example 3.7: $W = 1010$ and $B = 010$. This is a $(2, 1)$-resilient problem: W does not contain a substring 00, and B does not contain a substring 00.

Definition 9.3 (Bipartite matching)**.** Problems of the form $\Pi = (0100^+, 0100^+)$ are called *bipartite matching problems.*

Remark. Bipartite matching problems satisfy $d \geq 3$ and $\delta \geq 3$. In a d-regular bipartite graph, any solution to the bipartite matching problem is a perfect matching.

Definition 9.4 (Hypergraph matching)**.** Problems of the form $\Pi = (0100^+, 10^+1)$ are called *hypergraph matching problems.*

Remark. Hypergraph matching problems satisfy $d \geq 3$ and $\delta \geq 2$. By interpreting black nodes as hyperedges the problem equals the hypergraph matching problem with hyperedges of rank δ. In any d-regular hypergraph any such matching is a perfect hypergraph matching.

Definition 9.5 (Edge grabbing)**.** Problems of the form $\Pi = (0100^+, 110)$ are called *edge grabbing problems.*

Remark. Edge grabbing problems satisfy $d \geq 3$ and $\delta = 2$. If we interpret black nodes as edges of a simple graph, in a solution to the edge grabbing problem each white vertex *grabs* exactly one of its incident edges where no edge is grabbed by both of its endpoints.

9.2 Coverage

We now show that the definition of resilience and the three problem families defined above cover all binary labeling problems that are not unsolvable, trivial, or global. Recall Observation 3.4 for the notion of a restriction of a problem.

Lemma 9.6. *Let Π be a binary labeling problem that is not of type **I**, **II**, **III**, **IV**, **V** or **VI**. Then there is a restriction $\Pi' \subseteq \Pi$ such that one of the following holds:*

1. *Π' is $(2, 1)$-resilient or $(1, 2)$-resilient,*
2. *Π' is equivalent to a bipartite matching problem,*
3. *Π' is equivalent to a hypergraph matching problem,*
4. *Π' is equivalent to an edge grabbing problem.*

Proof. Let $\Pi = (d, \delta, W, B)$ be some problem that does not belong to type **I**–**VI**. If Π is $(2, 1)$-resilient or $(1, 2)$-resilient, we are done. For the rest of the proof assume that Π is neither $(2, 1)$-resilient nor $(1, 2)$-resilient. Therefore both of the following hold:

- W contains 0^{d-1} or B contains 0^δ.
- W contains 0^d or B contains $0^{\delta-1}$.

Now, we will go over all possible combinations and analyze them. First assume that W contains 0^d. We have the following cases:

- $W = 000^+$: We have $\Pi \in$ **II**.a.
- $W = 100^+$ and $B = 0**^+$: We have $\Pi \in$ **I**.a.
- $W = 100^+$ and $B = 1**^+$: We have $\Pi \in$ **IV**.a.
- $W = 00^+1$ and $B = **^+0$: We have $\Pi \in$ **I**.b.
- $W = 00^+1$ and $B = **^+1$: We have $\Pi \in$ **IV**.b.

But we assumed $\Pi \notin$ **I**, **II**, **IV**, so none of these are possible.

Second, assume that B contains 0^δ. This is equivalent to W containing 0^d; we would have $\Pi \in$ **I**, $\Pi \in$ **II**, or $\Pi \in$ **IV**.

The final possibility is that W does not contain 0^d, and B does not contain 0^δ. Then W has to contain 0^{d-1}, and B has to contain $0^{\delta-1}$, i.e.,

$$W \in \{10^+1,\, 010^+,\, 0^+10,\, 110^+,\, 0^+11\},$$
$$B \in \{10^+1,\, 010^+,\, 0^+10,\, 110^+,\, 0^+11\}.$$

We go through all combinations of possible values of W and B in these sets:

- $W = 10^+1$ and $B = 10^+1$: We have $\Pi \in$ IV.a.
- $W = 10^+1$ and $B = 010^+$:
  - If $B = 010$, we have $\Pi \in$ V.a.
  - Otherwise $B = 0100^+$, and Π is equivalent to a hypergraph matching problem.
- $W = 10^+1$ and $B = 0^+10$: Equivalent to the case $W = 10^+1$ and $B = 010^+$ above.
- $W = 10^+1$ and $B = 110^+$: We have $\Pi \in$ IV.a.
- $W = 10^+1$ and $B = 0^+11$: We have $\Pi \in$ IV.b.

- $W = 010^+$ and $B = 10^+1$: Equivalent to the case $W = 10^+1$ and $B = 010^+$ above.
- $W = 010^+$ and $B = 010^+$:
  - If $W = 010$, we have $\Pi \in$ VI.a.
  - If $B = 010$, we have $\Pi \in$ VI.b.
  - Otherwise $W = 0100^+$ and $B = 0100^+$, and Π is equivalent to a bipartite matching problem.
- $W = 010^+$ and $B = 0^+10$: We have $\Pi \in$ VI.b.
- $W = 010^+$ and $B = 110^+$:
  - If $W = 010$ and $B = 110^+$, we have $\Pi \in$ VI.a.
  - If $W = 0100^+$ and $B = 110$, then Π is equivalent to an edge grabbing problem.
  - Otherwise $W = 0100^+$ and $B = 1100^+$. Define $\Pi' = (0100^+, 0100^+)$. Now $\Pi' \subseteq \Pi$, and Π' is equivalent to a bipartite matching problem.
- $W = 010^+$ and $B = 0^+11$: We have $\Pi \in$ VI.b.

- $W = 0^+10$: Equivalent to one of the cases with $W = 010^+$ above.

- $W = 110^+$ and $B = 10^+1$: We have $\Pi \in$ IV.a.
- $W = 110^+$ and $B = 010^+$: Equivalent to $W = 010^+$ and $B = 110^+$ above.
- $W = 110^+$ and $B = 0^+10$: We have $\Pi \in$ VI.b.
- $W = 110^+$ and $B = 110^+$: We have $\Pi \in$ IV.a.
- $W = 110^+$ and $B = 0^+11$: We have $\Pi \in$ VI.b.

- $W = 0^+11$: Equivalent to one of the cases with $W = 110^+$ above. $\qquad\square$

9.3 Rake & compress procedures

We now present two procedures, which will be are our main tool in order to show $O(\log n)$ upper bounds. These procedures are based on the rake and compress technique introduced by Miller and Reif [24].

Let $G = (V, E)$ be a tree whose vertices are properly colored with two colors, called black and white. We start with two definitions; see Figure 4 for examples.
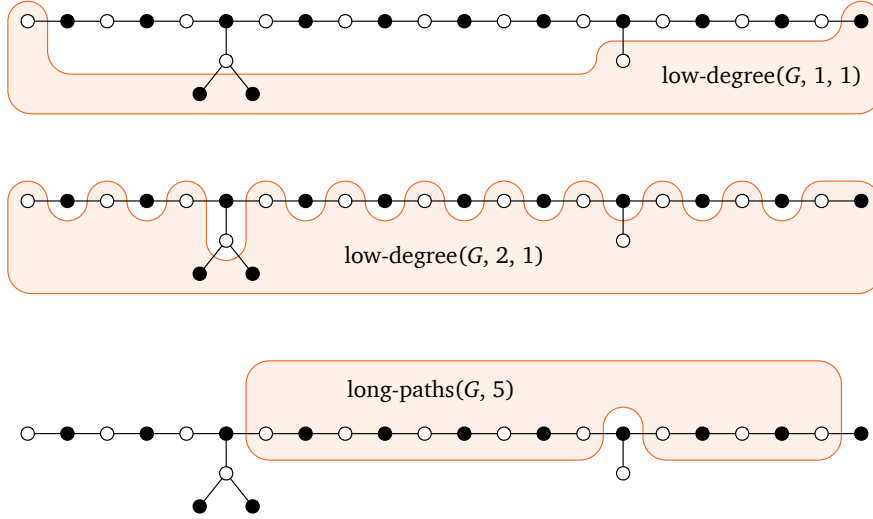
Figure 4: Examples of low-degree(G, w, b) from Definition 9.7 and long-paths(G, p) from Definition 9.8.

Definition 9.7 (low-degree). Let $w, b \in \{1, 2\}$. We define that low-degree(G, w, b) $\subseteq V$ is the set of all white nodes with degree at most w and all black nodes with degree at most b.

Definition 9.8 (long-paths). Let $p \in \{1, 2, \dots\}$. Let $X \subseteq V$ consist of all nodes of degree exactly 2. We define that long-paths(G, p) $\subseteq X$ consists of those nodes that belong to a connected component of size at least p in the subgraph of G induced by X.

Note that both of the definitions are local; low-degree only depends on the degree of a node and to find out whether a given node v is in long-paths(G, p), it is sufficient to explore its radius-p neighborhood:

Observation 9.9. *For any constants w, b, p, there is a constant-round algorithm that finds which nodes belong to the set* low-degree(G, w, b) *and which nodes belong to the set* long-paths(G, p).

Now we are ready to define two specific rake & compress procedures, RC and RCP. For a graph $G = (V, E)$ and a set of nodes X, we write $G - X$ for the subgraph of G induced by $V \setminus X$, i.e., it is the subgraph formed by deleting nodes in X. See Figure 5 for illustrations.

Definition 9.10 (RC). Let $w, b \in \{1, 2\}$. Procedure RC(w, b) partitions the set of nodes V into non-empty sets $V_1, V_2, \dots, V_L$ for some L as follows:

$$
\begin{aligned}
G_0 &= G, \\
V_{i+1} &= \text{low-degree}(G_i, w, b), & i &= 0, 1, \dots \text{ and } G_{i-1} \text{ is non-empty}, \\
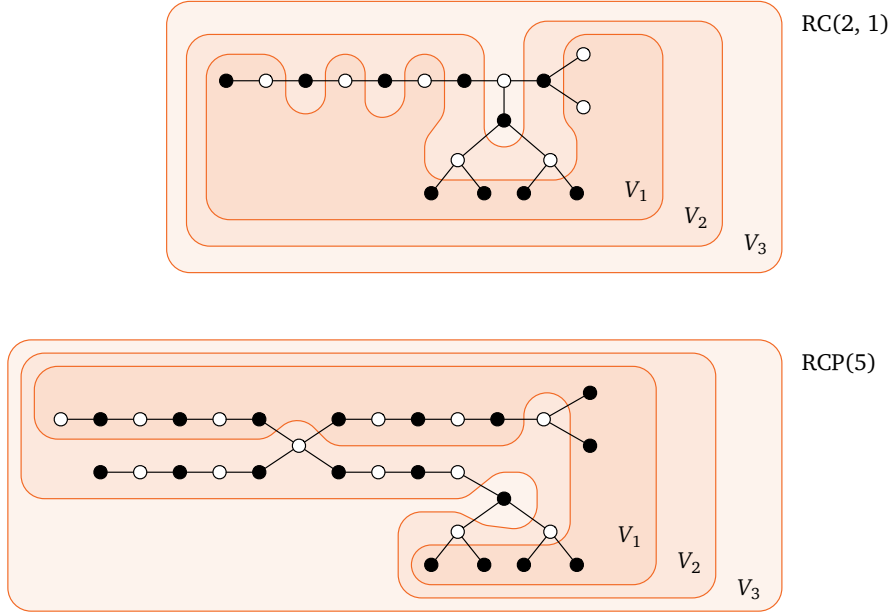G_{i+1} &= G_i - V_{i+1}, & i &= 0, 1, \dots \text{ and } G_{i-1} \text{ is non-empty}.
\end{aligned}
$$

Definition 9.11 (RCP). Let $p \in \{1, 2, \dots\}$. Procedure RCP(p) partitions the set of nodes V into non-empty sets $V_1, V_2, \dots, V_L$ for some L as follows:

$$
\begin{aligned}
G_0 &= G, \\
V_{i+1} &= \text{low-degree}(G_i, 1, 1) \cup \text{long-paths}(G_i, p), & i &= 0, 1, \dots \text{ and } G_{i-1} \text{ is non-empty}, \\
G_{i+1} &= G_i - V_{i+1}, & i &= 0, 1, \dots \text{ and } G_{i-1} \text{ is non-empty}.
\end{aligned}
$$

21

Figure 5: Examples of processes $RC(w, b)$ from Definition 9.10 and $RCP(p)$ from Definition 9.11.

We call the sets V_i *layers*. As we start with a tree G_0, each subgraph G_i is a forest, and as long as it is non-empty there is at least one node of degree at most 1. Therefore V_{i+1} is also non-empty, and hence both RC and RCP eventually terminate after $L = O(n)$ layers. The key observation is that $L = O(\log n)$ layers suffices, as long as we do not use $RC(1, 1)$; this is what we will prove next.

9.4 Rake & compress runs in logarithmic time

Lemma 9.12. *Fix a $p = 1, 2, \ldots$, and let $G = (V, E)$ be a tree with n nodes. At least one of the following holds:*

$$\left|\text{low-degree}(G, 1, 1)\right| \geq \frac{n}{6p} \quad \text{or} \quad \left|\text{long-paths}(G, p)\right| \geq \frac{n}{3}.$$

Proof. If $n = 1$, then $|\text{low-degree}(G, 1, 1)| = 1 = n$, and the claim holds. Otherwise $n > 1$, and the minimum degree of G is 1.

Let n_1 be the number of nodes of degree 1, let n_2 be the number of nodes of degree 2, and let n_3 be the number of nodes of degree at least 3. We have $n = n_1 + n_2 + n_3$. The number of edges in tree G is $m = n - 1$, and by counting half-edges, we get $n_1 + 2n_2 + 3n_3 \leq 2m = 2n_1 + 2n_2 + 2n_3 - 2$. Hence $n_3 < n_1$.

We have $|\text{low-degree}(G, 1, 1)| = n_1$, so if $n_1 \geq n/(6p)$, the claim holds. In what follows, assume that $n_1 < n/(6p)$. This implies that $n_1 + n_3 < 2n_1 < n/(3p)$.

Consider the subgraph G_2 induced by degree-2 nodes of G. If we contract each connected component of G_2 into an edge, we obtain a tree G' in which we have $n' = n_1 + n_3$ nodes and $m' = n_1 + n_3 - 1$ edges, and each edge represents at most one connected component of G_2. Hence there are fewer than $n/(3p)$ components in G_2.

Components of size less than p can contain fewer than $pn/(3p) = n/3$ nodes in total, but the total number of nodes of degree 2 is $n_2 = n - (n_1 + n_3) > n - n/(3p) \geq 2n/3$. Therefore there have to be at least $n/3$ nodes in components of size at least p, and hence $|\text{long-paths}(G, p)| \geq n/3$. $\square$

Lemma 9.13. *Let $G = (V, E)$ be a tree with n nodes. Then*

$$\left|\text{low-degree}(G, 1, 2)\right| \geq \frac{n}{12} \quad \text{and} \quad \left|\text{low-degree}(G, 2, 1)\right| \geq \frac{n}{12}.$$

Proof. Apply Lemma 9.12 with $p = 2$. If $|\text{low-degree}(G, 1, 1)| \geq n/(6p) = n/12$, we are done. Otherwise $|\text{long-paths}(G, 2)| \geq n/3$. Each connected component in the subgraph induced by long-paths$(G, 2)$ is a two-colored path of length at least 2, and hence at least a fraction $1/3$ of the nodes in each such component is black. Therefore there are at least $n/9$ black nodes of degree 2, and $|\text{low-degree}(G, 1, 2)| \geq n/9$. The same holds for white nodes and $|\text{low-degree}(G, 2, 1)|$. $\qquad\square$

Lemma 9.14. *Procedures* RC$(1, 2)$, RC$(2, 1)$, *and* RCP(p) *terminate after* $L = O(\log n)$ *layers.*

Proof. We show that we eliminate at least a constant fraction of nodes per step. We can assume that G_i is connected; otherwise we can analyze each connected component separately. For RC(w, b), we can apply Lemma 9.13 to see that we eliminate at least a fraction $1/12$ of the nodes in each step. For RCP(p), we can apply Lemma 9.12 to see that we eliminate at least a fraction $1/(6p)$ of the nodes in each step. $\qquad\square$

Corollary 9.15. *Procedures* RC$(1, 2)$, RC$(2, 1)$, *and* RCP(p) *can be implemented in* $O(\log n)$ *rounds in the LOCAL model of computation.*

Proof. Follows from Observation 9.9 and Lemma 9.14: it is sufficient to repeat a constant-time step for $O(\log n)$ times. $\qquad\square$

9.5 Properties of layer decompositions

Given a decomposition of nodes in layers $V_1, V_2, \ldots, V_L$ and a node $v \in V$, we define

$$N(v, i) = \big\{ u \in V_i : \{u, v\} \in E \big\}, \qquad\qquad d(v, i) = |N(v, i)|,$$
$$N(v, i \ldots j) = N(v, i) \cup N(v, i + 1) \cup \cdots \cup N(v, j), \qquad d(v, i \ldots j) = |N(v, i \ldots j)|.$$

That is, $N(v, i \ldots j)$ is the set of neighbors of v that are on layers $i, i + 1, \ldots, j$.

Claim 9.16. *In a decomposition $V_1, V_2, \ldots, V_L$ produced by* RC(w, b), *we will have* $d(v, i \ldots L) \leq w$ *for all white nodes $v \in V_i$ and $d(v, i \ldots L) \leq b$ for all black nodes $v \in V_i$.*

Proof. Note that $d(v, i \ldots L)$ is equal to the number of neighbors of v in graph G_{i-1}, and by assumption we had $v \in V_i = \text{low-degree}(G_{i-1}, w, b)$, so if v is white, there are at most w neighbors in G_{i-1}, and if v is black, there are at most b neighbors in G_{i-1}. $\qquad\square$

Claim 9.17. *In a decomposition $V_1, V_2, \ldots, V_L$ produced by* RCP(p), $p > 1$, *we will have*

$$d(v, i \ldots L) \leq 2 \quad \text{and} \quad d(v, i + 1 \ldots L) \leq 1$$

for all nodes $v \in V_i$.

Proof. We have $v \in V_i$, and hence one of the following holds:

- $v \in \text{low-degree}(G_{i-1}, 1, 1)$. Then v has at most 1 neighbor in G_{i-1}, and hence

$$d(v, i + 1 \ldots L) \leq d(v, i \ldots L) \leq 1.$$

- $v \in \text{long-paths}(G_{i-1}, p)$. Then v has at most 2 neighbors in G_{i-1}. Furthermore, v is a part of a connected component of size at least p, and the entire component belongs to V_i; hence there is a neighbor u of v in V_i, and in particular $u \in N(v, i \ldots L)$ but $u \notin N(v, i + 1 \ldots L)$. We have

$$d(v, i \ldots L) = 2,$$
$$d(v, i + 1 \ldots L) \leq d(v, i \ldots L) - 1 = 1. \qquad\square$$

23

9.6 Algorithm for resilient problems

Now we are ready to use RC and RCP to develop $O(\log n)$-time algorithms for all four problem classes that we defined in Section 9.1. We will start with resilient problems in this section; we will discuss bipartite matching, hypergraph matching, and edge grabbing later.

In what follows, we will often deal with *partial* labelings of graphs:

Definition 9.18 (Partial labeling). A *partial labeling* of a tree $G = (V, E)$ is a labeling $g\colon E \to \{0, 1, \bot\}$, where $g(e) = \bot$ indicates that edge e is unlabeled. Given two partial labelings g and g', we say that g' is a *completion* of g if $g'(e) = \bot$ implies $g(e) = \bot$ and $g(e) \neq \bot$ implies $g'(e) = g(e)$; that is, g' is constructed from g by changing some empty labels to non-empty labels. We say that the labeling around v is complete in g if all edges incident to v have non-empty labels.

Given a problem $\Pi = (d, \delta, W, B)$ and a partial labeling g, we say that the labeling around v is *valid* in g if it is complete around v and v satisfies the constraints of Π, i.e., the number of incident edges labeled with 1 is in W if v is a white node and it is in B if v is a black node. Finally, we say that a labeling g *can be completed* around v if there exists a completion g' of g such that the labeling around v is valid in g'. We extend these definitions to sets of nodes in a natural manner.

Note that if v is not a relevant node (recall that only white nodes of degree d and black nodes of degree δ are relevant), then any partial labeling can be trivially completed around v.

Now we are ready to connect the definition of resilience with the definition of completability:

Lemma 9.19. *Let $\Pi = (d, \delta, W, B)$ be a (t, s)-resilient problem, let $G = (V, E)$ be a tree, and let g be a partial labeling of G. Then g is completable around any $v \in V$ that satisfies one of the following:*

- *v is a white node incident to at most t edges with non-empty labels,*
- *v is a black node incident to at most s edges with non-empty labels.*

Proof. We prove the claim for white nodes; the case of black nodes is analogous. Let v be a white node incident to at most t edges with non-empty labels. If the degree of v is not d, the claim trivially holds, so assume that v is a node of degree d.

Let $i \leq t$ be the number of incident edges that are labeled with 1. By the definition of resilience, there has to be an index $j \in W$ such that

$$j \in \{i, i+1, \ldots, i+d-t\}.$$

If this was not the case, we would have $d + 1 - t$ consecutive indexes j with $j \notin W$, i.e., a substring of the form 0^{d+1-t} in the bit string representation of set W. Note that $j - i \leq d - t$ and therefore there are at least $j - i$ unlabeled edges around v.

This gives now a simple way to find a valid labeling for the unlabeled edges around v: take the first $j - i$ unlabeled edges incident to v, label them with 1, and label all other edges incident to v with 0. $\qquad\square$

Remark. The converse of Lemma 9.19 also holds—it turns out that resilience is *equivalent* to completability. However, one direction is sufficient for us.

Theorem 9.20. *If $\Pi = (d, \delta, W, B)$ is $(2, 1)$-resilient or $(1, 2)$-resilient, then Π can be solved in $O(\log n)$ rounds.*

Proof. Without loss of generality, let us consider a $(2, 1)$-resilient problem. We use the rake and compress procedure RC$(2, 1)$ to construct a layer decomposition $V_1, V_2, \ldots, V_L$; we have $L = O(\log n)$ due to Lemma 9.14 and this takes $O(\log n)$ rounds due to Corollary 9.15.
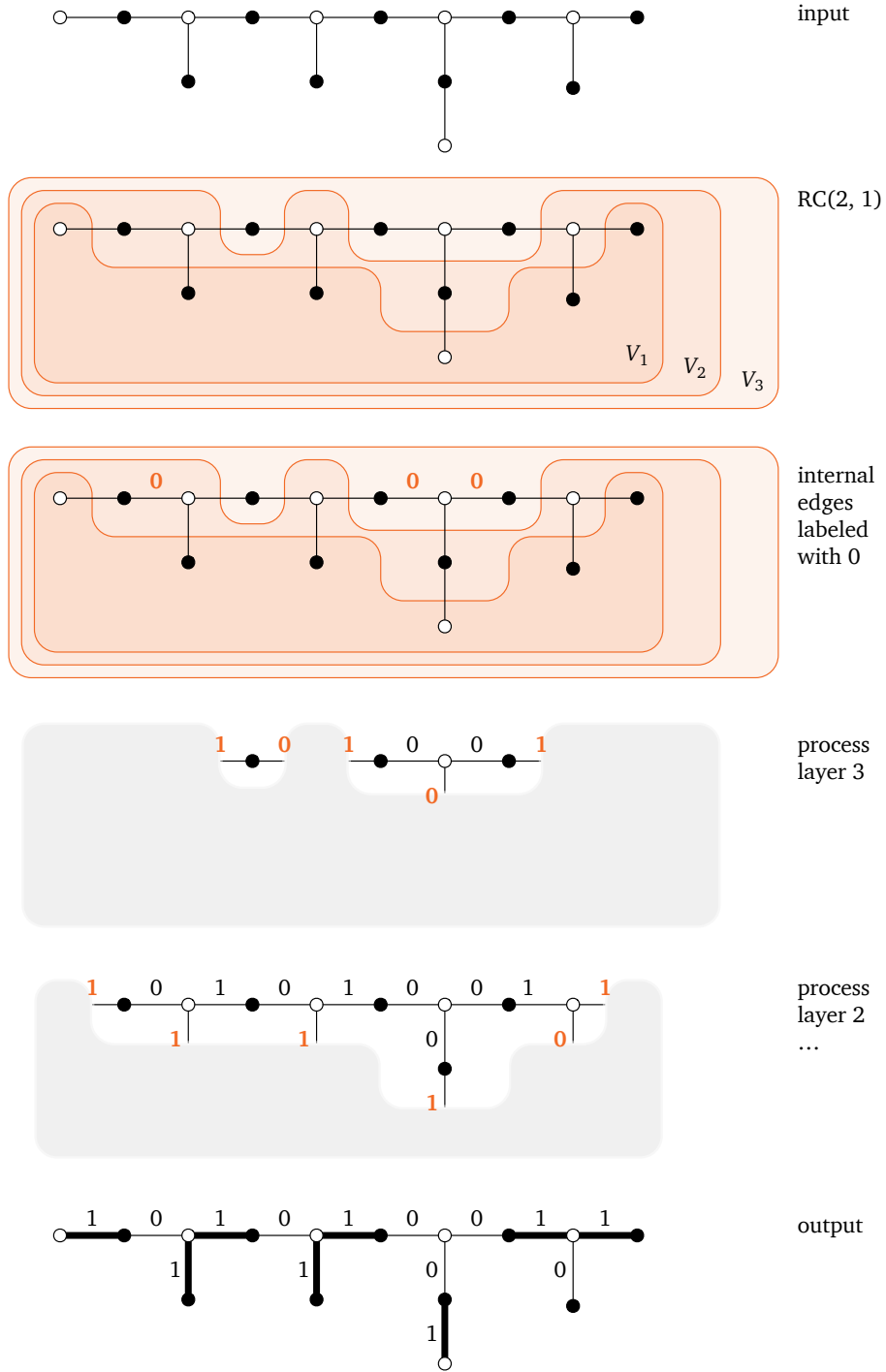
Figure 6: Illustration of the proof of Theorem 9.20: solving $(2, 1)$-resilient problems in $O(\log n)$ rounds, with the help of $\mathrm{RC}(2, 1)$ procedure. Here we use the *even orientation* problem (Example 3.7) as an example—the constraints are $W = 1010$ and $B = 010$, and the constraint need to be satisfied for white nodes of degree $d = 3$ and black nodes of degree $\delta = 2$.

We then use the layer decomposition to construct a feasible solution X to Π; see Figure 6 for an example. First, edges between nodes of the same layer get labeled with 0. Then, nodes proceed sequentially by layer number, from layer L to layer 1. For each layer i, assume that all nodes at higher layers have already labeled their incident edges in a valid manner. We prove that we can complete the labeling of the edges incident to nodes of layer i in a valid manner.

Consider a white node v at layer i. By Claim 9.16, $d(v, i \ldots L) \leq 2$, and hence at most two edges incident to v have already fixed their labels. Since the problem is $(2, 1)$-resilient, by Lemma 9.19 white nodes can complete any such partial labeling in their neighborhood.

Similarly, black nodes have at most one edges incident to them that are already labeled, and as the problem is $(2, 1)$-resilient, black nodes can also complete any such partial labeling in their neighborhood. Note that the choices of two adjacent nodes that are on the same layer do not interfere with each other, as all layer-internal edges are fixed to label 0. Further, in any layer, particularly in layer 1, that contains all the leaves, there might white nodes of degree $< d$, and black nodes of degree $< \delta$. These nodes are always satisfied and can always complete their labeling in an arbitrary manner.

We need $O(1)$ rounds per layer and hence $O(\log n)$ rounds in total to construct a feasible solution to Π. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ $\square$

9.7 Algorithm for bipartite matching problems

Theorem 9.21 (Bipartite matching). *Problems of the form $\Pi = (0100^+, 0100^+)$ can be solved in $O(\log n)$ rounds.*

Proof. We apply the procedure RCP(3) to the input tree G. Denote by $V_1, \ldots, V_L$ the resulting decomposition. We process the layers one by one, from layer L to 1.

Consider the nodes on layer i, and assume that all nodes of layers higher than i have labeled their incident edges in a valid manner. We show that nodes of layer i can label their incident edges as well. Recall that only white nodes of degree d and black nodes of degree δ are *relevant* and all other nodes are unconstrained. Note that in bipartite matching problems, all relevant nodes have degree at least 3.

By Claim 9.17 each node v at layer i satisfies $d(v, i \ldots L) \leq 2$ and $d(v, i+1 \ldots L) \leq 1$. Hence relevant nodes have at most one incident edge fixed by higher-level nodes, and they have at least one edge pointing towards lower-level nodes. In particular, relevant nodes are currently incident to at most one edge with label 1, and each relevant node v at level i can pick one incident edge $e = \{v, u\}$ such that u is on a lower level. Node v then labels all other unlabeled incident edges with 0, and finally it assigns 0 or 1 to edge e in order to satisfy its constraints. Nodes that are not relevant can simply assign 0 to all incident unlabeled edges.

Note that the choices of the nodes on the same level do not conflict with each other, as all edges between nodes of level i are labeled with 0. Thus we can safely label each layer in $O(1)$ rounds. $\quad\square$

9.8 Algorithm for hypergraph matching problems

Theorem 9.22 (Hypergraph matching). *Problems of the form $\Pi = (0100^+, 10^+1)$ can be solved in $O(\log n)$ rounds.*

Proof. We apply the procedure RCP(5) to the input tree G to construct a decomposition $V_1, \ldots, V_L$; see Figure 7 for an example. This choice of the parameter implies that the connected components of $G[V_i]$ are isolated nodes and paths of length at least 5. Recall that Claim 9.17 shows that each
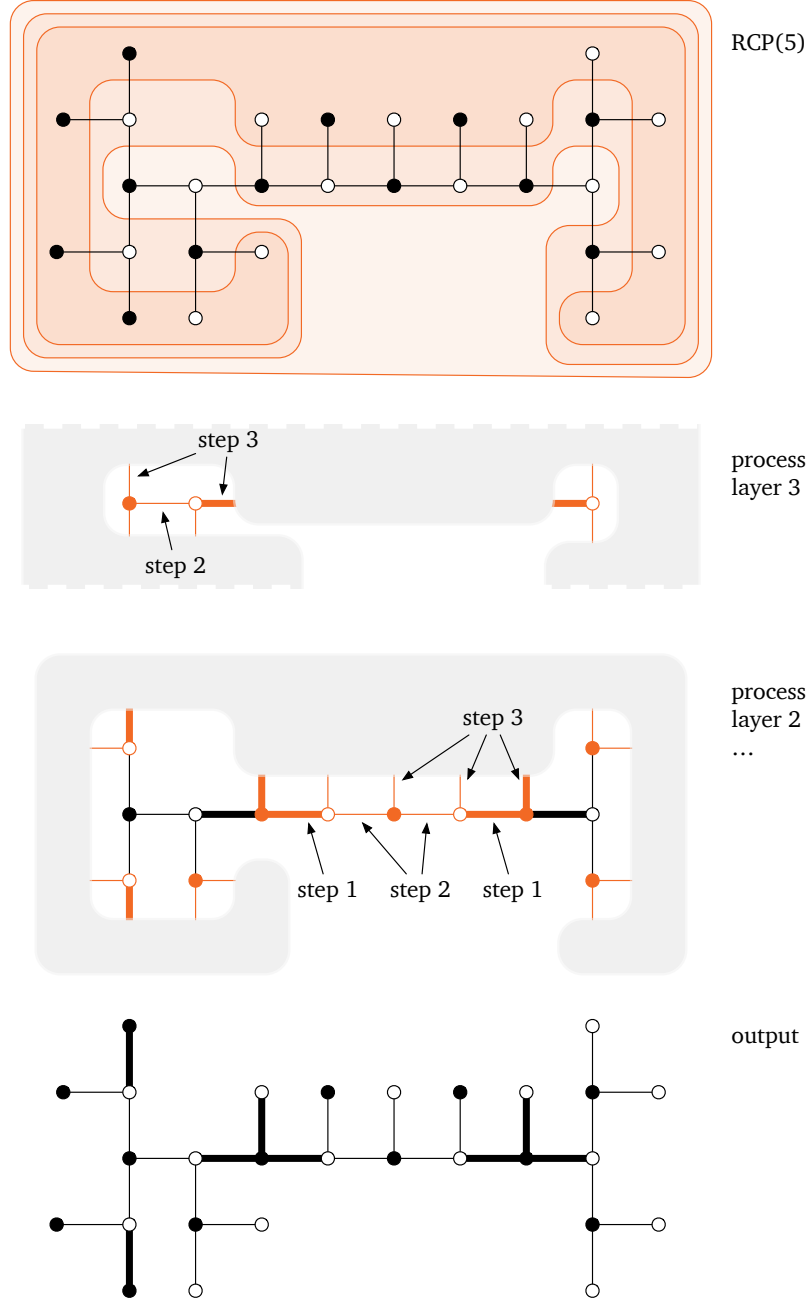
Figure 7: Illustration of the proof of Theorem 9.22: solving hypergraph matching in $O(\log n)$ rounds, with the help of RCP(5) procedure. Here we use the case of $d = \delta = 3$ as an example, i.e., the problem to solve is $W = 0100$, $B = 1001$.

node $v \in V_i$ has at most two neighbors in layers $i, i+1, \ldots, L$ and at most one neighbor in layers $i+1, \ldots, L$.

We iterate through the layers in a backwards order and in iteration $i = L, \ldots, 1$ the vertices in V_i use a three step procedure to label all their incident edges that have not been labeled in any of the previous iterations:

- **Step 1**: Assume that v is a black node, it is adjacent to a higher-layer node $u \in N(v, i+1 \ldots L)$, and the edge $\{v, u\}$ is already labeled with 1. Now if v is adjacent to another node $s \in V_i$ on the same layer, we label the edge $\{v, s\}$ with 1.

- **Step 2:** Label all edges between vertices in V_i that are unlabeled after step 1 with 0.

- **Step 3:** Each vertex completes the labeling of its incident edges going to lower layers in an arbitrary valid way.

Vertices only need to know their $O(1)$ neighborhood to execute the three steps. We show that each vertex can complete its labeling in step 3 in a valid way:

- *Black nodes:* A black node v either has all of its already labeled edges labeled with a 0 or it has two edges incident edges labeled with a 1. Thus any black node can complete the labeling of its incident edges towards lower levels with all 0s in the former and with all 1s in the latter case.

- *White nodes with higher-level neighbors:* Consider a white node $v \in V_i$ and assume that v has a higher-layer neighbor u. Then v cannot have a black neighbor v_1 with a higher-layer neighbor u_1, as otherwise v and v_1 would only form a connected component that is a path of length 2, a contradiction with the fact that $G[V_i]$ consist of isolated nodes and paths of length at least 5. Thus v has at most one incident edge labeled with a 1 after step 2, i.e., the edge $\{v, u\}$, and v can complete the labeling.

- *White nodes without higher-level neighbors:* Consider a white node $v \in V_i$ and assume that v has no neighbors on higher layers. Then v can have at most one black neighbor that participated in step 1 because if it had two such neighbors v_1 and v_2, then v_1, v and v_2 would again induce a connected component that is a path of length 3 in $G[V_i]$, a contradiction. Thus v has at most one black neighbor v_1 that potentially labeled the edge $\{v, v_1\}$ with a 1, all other already labeled incident edges of v must be labeled with 0, that is, v can complete the labeling. $\square$

9.9 Algorithm for edge grabbing problems

We next show that any problem in the edge grabbing family can be reduced to sinkless orientation which implies that it can be solved in $O(\log n)$ rounds.

Theorem 9.23 (Edge grabbing). *Problems of the form $\Pi = (0100^+, 110)$ can be solved in $O(\log n)$ rounds.*

Proof. We will show this upper bound by reducing the problem to sinkless orientation, that is, we reduce Π to a problem of the form $\Pi' = (0111^+, 010)$, which can be solved in $O(\log n)$ rounds by Theorem 9.20.

Assume we are given a solution to Π'. Each white node v of degree d is incident to at least one edge that is labeled with 1. Node v chooses one such edge (say, the one with the smallest port number) and marks it *red*. All other edges are *black*.

Let X be the set of red edges. These edges form a feasible solution to Π: each white node of degree d is incident to exactly one red edge, and each black node of degree 2 is incident to at most one red edge. $\qquad\square$

9.10 Proof of Theorem 9.1

Lemma 9.6 shows that any problem that is not of type **I**, **II**, **III**, **IV**, **V** or **VI** falls 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.

Theorems 9.20, 9.21, 9.22, and 9.23 show that problems in each of the four classes can be solved in $O(\log n)$ rounds. This completes the proof.

10 Logarithmic lower bounds

In this section we will show that all binary labeling problems that are not of the type discussed in Section 6, Section 7, or Section 8 require $\Omega(\log n)$ rounds with deterministic distributed algorithms. This will give the lower bounds for the final section of Table 2, and hence it completes the classification of all binary labeling problems. Formally we prove the following theorem.

Theorem 10.1. *Any problem that is not of type* **I**, **II**, **III**, **IV**, **V** *or* **VI** *has complexity* $\Omega(\log n)$.

To prove Theorem 10.1, we first introduce two families of problems: *forbidden degree problems* and *sinkless orientation problems*. In Lemma 10.4 we will prove that any problem not covered in Sections 6–8 is at least as hard as a forbidden degree problem or a sinkless orientation problem. Then we will prove a lower bound of $\Omega(\log n)$ for both problem families.

10.1 Definitions

Definition 10.2 (Forbidden degree). Problems of the form $\Pi = (1^+01^+, 01^+0)$ are called *forbidden degree problems*.

Remark. Note that here we can have any degrees $d, \delta \geq 2$. In forbidden degree problems, there is one value $i \notin W$, $0 < i < d$. This value is known as the forbidden degree; the intuition here is that white nodes can have any X-degree except i.

Definition 10.3 (Bipartite sinkless orientation). Problems of the form $\Pi = (11^+0, 011^+)$ are called *bipartite sinkless orientation problems*.

Remark. Note that here we can have any degrees $d, \delta \geq 2$. For bipartite sinkless orientation, it is known [9, 12, 13] that the same lower bounds of $\Omega(\log \log n)$ and $\Omega(\log n)$ as above hold for the case $d, \delta \geq 3$, and it is not hard to extend these bounds to the case where $d = 2$ or $\delta = 2$. However, to avoid case distinctions and make sure that the bounds also hold in our setting (where, e.g., we have a tree instead of a general graph), we will prove these lower bounds for bipartite sinkless orientation in an analogous fashion to the lower bound proofs for the forbidden degree problems.

10.2 Coverage

We will first 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:

Lemma 10.4. *Let $\Pi = (d, \delta, W, B)$ be a problem that is not of type* **I***,* **II***,* **III** *or* **IV***. Then there is a relaxation $\Pi' \supseteq \Pi$ such that Π' is equivalent to a forbidden degree problem or a bipartite sinkless orientation problem.*

Proof. As before, we denote the bits in W and B by $w_0, w_1, \ldots, w_d$ and $b_0, b_1, \ldots, b_\delta$, respectively. Consider all possible 4-bit strings $w_0 \, b_0 \, w_d \, b_\delta$:

- $11**$: We have $\Pi = (1**^+, 1**^+) \in$ **IV**.a.

- $**11$: We have $\Pi = (**^+1, **^+1)$, which is equivalent to $\Pi = (1**^+, 1**^+)$ above.

- $0**0$: We have $\Pi = (0**^+, **^+0) \subseteq \Pi' = (011^+, 11^+0)$.
  Now Π' is a bipartite sinkless orientation problem.

- $*00*$: We have $\Pi = (**^+0, 0**^+)$, which is equivalent to $\Pi = (0**^+, **^+0)$ above.

- 1010: We have $\Pi = (1*^+1, 0*^+0)$.
  If $\Pi = (111^+, 0*^+0)$, then $\Pi \in$ **III**.b.
  Otherwise $\Pi \subseteq \Pi' = (1^+01^+, 01^+0)$, and Π' is a forbidden degree problem.

- 0101: We have $\Pi = (0*^+0, 1*^+1)$, which is equivalent to $\Pi = (1*^+1, 0*^+0)$ above. $\square$

10.3 Automatic round elimination

We will show in this section that all forbidden degree problems require $\Omega(\log \log n)$ randomized time in the LOCAL model, and the same holds also for bipartite sinkless orientation. To prove the claim, we will use *round elimination*. We begin by defining the necessary concepts.

The various round elimination proofs [5, 7–9, 13, 22, 25] have all used slightly different notation. In this section we adopt the framework of Brandt [8] and much of the notation of Balliu et al. [5].

Model of computing. We use the randomized model of computing from Section 5, with two important distinctions.

1. We always assume that the algorithm is running in the middle of a 2-colored tree with white degree d and black degree δ. In particular, we assume that the nodes never see any irregularities, such as leaves. This implies that the topology of the input network is fixed. We show that the problem is hard even with this promise.

2. We assume that the nodes have random port numbers assigned independently from their random bits. This assumption simplifies the analysis of Lemma 10.5. This model can simulate the randomized LOCAL model, since there an algorithm must work with high probability for *any* fixed port numbering.

Bipartite labeling problems. In this section we consider a generalized version of binary labeling problems with $|\Sigma| > 2$. A *bipartite labeling problem* is given by a tuple $(\Sigma, d, \delta, W, B)$, where Σ is the alphabet, d is the white degree, δ is the black degree, W is the set of *white configurations*, and B is the set of *black configurations*. Here, each white configuration is a multiset containing exactly d (not necessarily distinct) labels from Σ, and each black configuration is a multiset containing exactly δ (not necessarily distinct) labels from Σ. The problem is solved correctly if each edge is labeled with some $\sigma \in \Sigma$, and for each white, resp. black, node, the multiset $\{x_1, x_2, \ldots, x_k\}$, where $k = d$, resp. $k = \delta$, of incident edge labels is a configuration contained in W, resp. B.

We represent (sets of) configurations as regular expressions [5]. We use the shorthands $[x_1 x_2 \cdots x_k] = x_1 | x_2 | \cdots | x_k$ and x^k for x repeated k times, e.g., the regular expression $XY^2[XZ]$ stands for the set containing the two multisets $\{X, X, Y, Y\}$ and $\{X, Y, Y, Z\}$.

Output problems. For each bipartite labeling problem $\Pi = (\Sigma, d, \delta, W, B)$ there exists a problem $\Pi' = (2^\Sigma, d, \delta, W', B')$ such that if Π can be solved in T rounds by a white algorithm, then Π' can be solved in $T - 1$ rounds by a black algorithm (with slightly worse success probability). We will provide a definition for such a problem below and call the problem Π' specified by the definition the *black output problem of* Π. Similarly, if Π can be solved in T rounds by a black algorithm, we define a *white output problem of* Π that can be solved by a $(T-1)$-round white algorithm. Note that the definitions of the black and white output problems of Π depend only on Π and not on T.

In the following we define the black output problem Π' of Π. The white output problem is obtained by applying the same operations with the roles of white and black reversed.

Given Π, we form the black constraint B' of Π' as follows. A multiset $X_1 X_2 \ldots X_\delta$, where $X_i \subseteq 2^\Sigma$, is in B' if and only if for all $(x_1, x_2, \ldots, x_\delta) \in X_1 \times X_2 \times \cdots \times X_\delta$ we have that $x_1 x_2 \ldots x_\delta$ is in B. The white constraint is formed as follows. A multiset $Y_1 Y_2 \ldots Y_d$, where $Y_i \subseteq 2^\Sigma$, is in W' if and only if there exist $y_1 \in Y_1, y_2 \in Y_2, \ldots, y_d \in Y_d$ such that $y_1 y_2 \ldots y_d$ is in W.

Given a solution to the black (resp. white) output problem of Π, the white (resp. black) nodes can solve the initial problem Π in one additional round by construction [8]. We show essentially the converse in Lemma 10.5.

Simplification via maximality. We simplify any black output problem $\Pi' = (2^{|\Sigma|}, d, \delta, W', B')$ using an observation of Brandt [8, Section 4.2]. We consider only *maximal* black configurations: if there are black configurations $Y_1 Y_2 \cdots Y_\delta$ and $Z_1 Z_2 \cdots Z_\delta$ such that $Y_i \subseteq Z_i$ for all i, then we can safely remove the constraint $Y_1 Y_2 \cdots Y_\delta$ from B' without changing the (black) complexity of the problem. In particular, any black algorithm A solving Π' can always safely output $Z_1 Z_2 \cdots Z_\delta$ instead of $Y_1 Y_2 \cdots Y_\delta$ since this can only increase the chances that the white constraint is satisfied at a white node, according to our definition of a black output problem. Moreover, we can remove all white configurations that contain labels (from 2^Σ) that are not contained in any of the black configurations (after applying the above maximality argument), as those labels cannot occur in a correct output. We will use these simplifications (and the analogous versions for white output problems) throughout the proofs of Lemmas 10.8 and 10.9.

Local failure probability. Way say that an algorithm A has local failure probability at most p if for all nodes, the probability that the labeling incident to that node does not satisfy the constraint of that node is at most p. Since randomness is the only input, all white nodes have degree d, all black nodes have degree δ, and the graph is a tree, all white nodes and all black nodes in the middle of the tree, respectively, have the same local failure probability.

10.4 Local success probability decay

In this section we show how to bound the failure probability of the round elimination. This proof is essentially present in previous round elimination arguments, but has always been presented in a problem-specific way. Lemma 10.5 presents it here in full generality. In particular, our argument does not depend on the particular structure of the LCL, only the number of different output labels and the degrees of the nodes.

Lemma 10.5. *Let A be a white T-round anonymous randomized algorithm for a bipartite labeling problem Π with local failure probability at most p in the randomized LOCAL model. Then there exists a black $(T-1)$-round algorithm A' for Π', the black output problem of Π, with local failure probability at most $Kp^{1/(\delta+1)}$, for some constant $K = K(\delta, d, |\Sigma|)$. Respectively, for a black T-round algorithm for Π with local failure probability p there exists a white $(T-1)$-round algorithm for the white output problem of Π with local failure probability at most $Kp^{1/(d+1)}$ for some constant $K = K(d, \delta, |\Sigma|)$.*

Proof. We prove the case of initial white algorithm. The case of a black algorithm is symmetric. We connect the failure probability of algorithm A' to the failure probability of the original algorithm A. We denote the T-hop ball around node v, considered as a graph, by $B_T(v)$. We denote the labeled graph obtained by labeling $B_T(v)$ with the inputs of each node by $I_T(v)$.

Given a T-round white algorithm A, black nodes can execute a $(T-1)$-round *simulation* algorithm A' as follows. Each black node v looks at the input, including the random inputs and the random port numbers, inside its $(T-1)$-neighborhood. It simulates A at each of its neighbors, treating them as the active nodes, over all possible assignments of random inputs outside $B_{T-1}(v)$. We observe that given $I_{T-1}(v)$, the outputs of the neighbors of v, as random variables, are independent from each other: for all $u, w \in N(v)$ we have that $(B_T(u) \cap B_T(w)) \setminus B_{T-1}(v)$ is empty. This follows from the fact that we assumed the input graph to be a tree and implies independence since we assumed that the nodes do not have any correlated inputs, in particular that the nodes do not have unique identifiers. Each black node v can independently compute the conditional probabilities $\Pr[A(u, v) = \sigma \mid I_{T-1}(v)]$ for all labels σ and neighbors u, that is, the probability of u outputting σ on edge $\{u, v\}$ as an active node executing algorithm A after fixing $I_{T-1}(v)$.

We say that an output label σ is *frequent* in A at node u for v if

$$\Pr\big[A(u, v) = \sigma \mid I_{T-1}(v)\big] \geq f.$$

We will set the value of f later. The output of the simulation algorithm A' for edge $\{v, u\}$ at v, denoted by $A'(v, u)$, is defined as the set of frequent output labels, that is,

$$A'(v, u) = S_v(u) = \{\sigma \in \Sigma \mid \sigma \text{ is frequent in } A \text{ at } u \text{ for } v\}.$$

Note that v can determine $A'(v, u)$ for each neighbor u based on $I_{T-1}(v)$.

We want to consider only the random assignments $I_{T-1}(v)$ that make A likely to succeed, and discard the tail of assignments that cause A to fail with a large probability. We say that the $(T-1)$-neighborhood $I_{T-1}(v)$ of a black node v is *good* if A is likely to succeed around it. Let $E(v)$ denote the event that for a black node v the outputs $A(u, v)$ for all $u \in N(v)$ form a black configuration in Π. For a parameter r to be fixed later, we say that $I_{T-1}(v)$ is good if and only if $\Pr[E(v) \mid I_{T-1}(v)] \geq 1 - r$. Since the outputs of the neighbors are independent, this probability can again be determined by v based on $I_{T-1}(v)$.

Most random assignments $I_{T-1}(v)$ must be good, as otherwise the original algorithm A would not work with probability $1 - p$: we have that $r \cdot \Pr[I_{T-1}(v)$ is not good$] \leq p$. We get that

$$\Pr[I_{T-1}(v) \text{ is good}] = 1 - \Pr[I_{T-1}(v) \text{ is not good}]$$
$$\geq 1 - \frac{p}{r}.$$

If the neighborhood is not good, the conditional failure probability of A is at least r. As the total failure probability is at most p, we get the above. We can lower bound the success probability of black nodes in A' by observing that

$$\Pr[A' \text{ succeeds at } v] \geq \Pr[I_{T-1}(v) \text{ is good}](1 - \Pr[A' \text{ fails at } v \mid I_{T-1}(v) \text{ is good}]). \qquad (1)$$

Assume that the $(T-1)$-neighborhood of a black node v is good. Let $u_1, \ldots, u_\delta$ be the neighbors of v. Assume that A' fails at node v, i.e., assume that there exist $\sigma_1 \in S_v(u_1), \sigma_2 \in S_v(u_2), \ldots, \sigma_\delta \in S_v(u_\delta)$ such that the multiset $\{\sigma_1, \ldots, \sigma_\delta\}$ is not contained in the black constraint B of Π. This implies that there is a significant probability for A to fail in this neighborhood as well. Since the labels $\sigma_1, \sigma_2, \ldots, \sigma_\delta$ appear independently and are all frequent, the probability having $A(u_i, v) = \sigma_i$ for all i given $I_{T-1}(v)$ is at least f^δ. We have that

$$\Pr[I_{T-1}(v) \text{ is good}] \cdot \Pr[A' \text{ fails at } v \mid I_{T-1}(v) \text{ is good}] \cdot f^\delta \leq \Pr[A \text{ fails at } v].$$

Since the local failure probability of A is at most p, we have that

$$\Pr[A' \text{ fails at } v \mid I_{T-1}(v) \text{ is good}] \leq \frac{\Pr[A \text{ fails at } v]}{\Pr[I_{T-1}(v) \text{ is good}] \cdot f^\delta}$$
$$\leq \frac{p}{(1 - p/r) \cdot f^\delta}.$$

By (1) we get that black nodes succeed in A' with probability at least

$$\left(1 - \frac{p}{r}\right)\left(1 - \frac{p}{(1 - p/r) \cdot f^\delta}\right) = 1 - \frac{p}{r} - \frac{p}{f^\delta}.$$

By choosing $r = f^\delta$, we obtain that

$$\Pr[A' \text{ succeeds at a black node}] \geq 1 - \frac{2p}{f^\delta}.$$

Next we consider the white nodes in the simulation algorithm A'. According to the definition of Π', each white node u succeeds if and only if the sets algorithm A' outputs on u's incident edges allow for selecting one label (from Σ) from each set such that the resulting selected multiset is a configuration in W. This is guaranteed to happen when the true outputs $A(u, v)$ of u are included in the corresponding sets $A'(v, u)$, that is, they are all frequent.

We begin by upper bounding the probability that a true output is not frequent. Consider a white node u, and its black neighbor v. From the definition of a frequent label it follows that, for (any) fixed $I_{T-1}(v)$ and any label $\sigma \in \Sigma$, the conditional probability that $\sigma = A(u, v)$ but σ is not frequent is at most f. We obtain that

$$\Pr[A(u, v) \notin A'(v, u)] \leq |\Sigma| f$$

by a union bound. A white node can fail in A' if A would have failed in the neighborhood or if at least one edge does not have the true output as a frequent output. Taking another union bound over all the edges, we have that a white node succeeds with probability at least $1 - p - d|\Sigma|f$.

It remains to set f to minimize the local failure probability of the simulation. First, note that we must have $f \leq 1/|\Sigma|$ to guarantee that there is a frequent output. We choose f to upper bound the maximum of $p + d|\Sigma|f$ and $2p/f^\delta$. By choosing for example $f = Cp^{1/(\delta+1)}$ for $C = (2/(d|\Sigma|))^{1/(\delta+1)}$, we get for black nodes that

$$\Pr[A' \text{ fails at a black node}] \leq \frac{2p}{f^\delta} = 2pC^{-\delta}p^{-\delta/(\delta+1)} = 2C^{-\delta}p^{1/(\delta+1)},$$

and for white nodes that

$$\Pr[A' \text{ fails at a white node}] \leq d|\Sigma|f + p = d|\Sigma|Cp^{1/(\delta+1)} + p.$$

By the choice of C we have that both terms are bounded by

$$2^{1/(\delta+1)}(d|\Sigma|)^{\delta/(\delta+1)}p^{1/(\delta+1)} + p \leq Kp^{1/(\delta+1)}$$

for some constant $K = K(\delta, d, |\Sigma|)$. $\qquad\square$

10.5   FDSO problem

Our proof employs an intermediate (non-binary) LCL problem, which we refer to as *forbidden degree or sinkless orientation* (FDSO). As we show in Lemma 10.7, this problem is at least as easy as both the forbidden degree problem and bipartite sinkless orientation, i.e., as easy as the two problems that play important roles in Lemma 10.4. It is defined as follows in our white-black formalism.

Definition 10.6. Forbidden degree s or sinkless orientation ($\mathsf{FDSO}(s)$), for $0 < s < d$ is a bipartite labeling problem $(\Sigma, d, \delta, W, B)$ over four labels: $\Sigma = \{X, H, T, A\}$. The white constraint W is given by the configurations

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

and the black constraint B is given by the configurations

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

Lemma 10.7. $\mathsf{FDSO}(s)$ *is at least as easy as the bipartite sinkless orientation problem (for the same d, δ), and at least as easy as the forbidden degree problem (for the same d, δ, s).*

Proof. Fix $d, \delta \geq 2$ and $0 < s < d$ arbitrarily. We first show that $\mathsf{FDSO}(s)$ is at least as easy as bipartite sinkless orientation, i.e., as the problem with $W = 11^+0$ and $B = 011^+$. Given a solution to bipartite sinkless orientation, a white node labels one 0-edge with A, and the remaining edges with X. This is clearly a valid solution for $\mathsf{FDSO}(s)$: each white node is labeled AX^{d-1} and each black node has a 1-edge that is labeled X.

Now, we show that $\mathsf{FDSO}(s)$ is at least as easy as the forbidden degree problem (with parameters d, δ, s). Assume we are given a solution to $\Pi = (d, \delta, W', B')$ for $W' = [d] \setminus \{s\}$, and $B' = \{1, 2, \ldots, \delta - 1\}$. In the solution to Π, white nodes have either at most $s - 1$ or at least $s + 1$ incident edges labeled 1. In the first case, they will select $d - s + 1$ edges among the 0-labeled edges and output T on those, and X on the rest. In the second case, they will select $s + 1$ edges among the

1-labeled edges and output H on these, and X on the rest. A white node is always happy according to $\mathsf{FDSO}(s)$. Assume that a black node does not have any incident label X, and therefore does not satisfy the first set of configurations $X[AHTX]^{\delta-1}$. Then, it must have incident edges labeled 0 and 1 in Π, which must be labeled T and H in $\mathsf{FDSO}(s)$, respectively, and it follows that it satisfies the second set of configurations $HT[AHTX]^{\delta-2}$. $\qquad\square$

10.6 FDSO is a fixed point

In this section we show that, for any s, $\mathsf{FDSO}(s)$ is a fixed point. In particular, after two round elimination steps starting from $\mathsf{FDSO}(s)$, the output problem is again $\mathsf{FDSO}(s)$.

Lemma 10.8. *Let $\Pi_0 = \mathsf{FDSO}(s)$ for some $0 < s < d$. Then $\Pi_1 = (2^\Sigma, d, \delta, W_1, B_1)$, the black output problem of Π_0, is the following problem. Given the renaming of labels as*

$$\begin{aligned}
\mathbf{A} &= \{X\} \\
\mathbf{H} &= \{H, X\} \\
\mathbf{T} &= \{T, X\} \\
\mathbf{X} &= \{A, H, T, X\}\,,
\end{aligned}$$

the white constraint W_1 is given by the configurations

$$\begin{aligned}
&\mathbf{X}[\mathbf{AHTX}]^{d-1} \\
&\mathbf{H}^{s+1}[\mathbf{AHTX}]^{d-s-1} \\
&\mathbf{T}^{d-s+1}[\mathbf{AHTX}]^{s-1},
\end{aligned}$$

and the black constraint B_1 by the configurations

$$\begin{aligned}
&\mathbf{AX}^{\delta-1} \\
&\mathbf{HTX}^{\delta-2}.
\end{aligned}$$

Proof. According to the definition of a black output problem, we have to show that

(1) the specified black constraint B_1 contains exactly those maximal configurations $\mathbf{Y}_1 \ldots \mathbf{Y}_\delta$ for which for every choice $Y_1 \in \mathbf{Y}_1, \ldots, Y_\delta \in \mathbf{Y}_\delta$, we have that $Y_1 \ldots Y_\delta$ is a black configuration for Π_0, and

(2) the specified white constraint W_1 contains exactly those configurations $\mathbf{Z}_1 \ldots \mathbf{Z}_\delta$ for which there exists a choice $Z_1 \in \mathbf{Z}_1, \ldots, Z_\delta \in \mathbf{Z}_\delta$ such that $Z_1 \ldots Z_\delta$ is a white configuration for Π_0.

We start by proving (1). Recall that the black constraint of Π_0 is given by $X[AHTX]^{\delta-1}$ and $HT[AHTX]^{\delta-2}$. Consider the first configuration $\mathbf{AX}^{\delta-1}$ in the black constraint B_1 of Π_1. Clearly, choosing one original label (i.e., a label from Σ) from $\mathbf{A}$, and one from each of the $\delta - 1$ copies of $\mathbf{X}$ will result in a configuration that is contained in $X[AHTX]^{\delta-1}$ since the only choice to pick from $\mathbf{A}$ is X. Similarly, for the second configuration $\mathbf{HTX}^{\delta-2}$ in B_1, any choice of labels (from Σ) from $\mathbf{H}$, $\mathbf{T}$, and $\delta - 2$ copies of $\mathbf{X}$ yields a configuration contained in $X[AHTX]^{\delta-1}$ (if at least one X is chosen), or a configuration contained in $HT[AHTX]^{\delta-2}$ (if no X is chosen, which implies that the two labels chosen from $\mathbf{H}$ and $\mathbf{T}$ are H and T).

What is left to be shown is that $\mathbf{AX}^{\delta-1}$ and $\mathbf{HTX}^{\delta-2}$ are the only maximal configurations. (It is straightforward to check that these configurations are indeed maximal by checking that any new

element added to any of the sets in one of these configurations will break the property given in the definition of the black constraint of the black output problem.) Now assume that there is another maximal configuration $\mathbf{Y}_1 \ldots \mathbf{Y}_\delta$. Then, due to its maximality, each of the $\mathbf{Y}_i$ has to contain at least one label Y_i such that $Y_i \neq X$ (as otherwise $\mathbf{Y}_1 \ldots \mathbf{Y}_\delta$ would be "dominated" by $\mathbf{A}\mathbf{X}^{\delta-1}$), and, using this fact, we can also infer that there is a $\mathbf{Z} \in \{\mathbf{H}, \mathbf{T}\}$ such that, for all i, we have $\mathbf{Y}_i \setminus \mathbf{Z} \neq \emptyset$ (as otherwise $\mathbf{Y}_1 \ldots \mathbf{Y}_\delta$ would be dominated by $\mathbf{HTX}^{\delta-2}$). From this, it follows that we can pick $Y_1 \in \mathbf{Y}_1, \ldots, Y_\delta \in \mathbf{Y}_\delta$ such that $X \neq Y_i \neq H$ for all i (if $\mathbf{Z} = \mathbf{H}$), or $X \neq Y_i \neq T$ for all i (if $\mathbf{Z} = \mathbf{T}$). But this implies that $Y_1 \ldots Y_\delta$ is contained neither in $X[AHTX]^{\delta-1}$, nor in $HT[AHTX]^{\delta-2}$, yielding a contradiction. Hence, $\mathbf{A}\mathbf{X}^{\delta-1}$ and $\mathbf{HTX}^{\delta-2}$ are the only maximal configurations. As $\mathbf{A}$, $\mathbf{H}$, $\mathbf{T}$, and $\mathbf{X}$ are the only labels that occur in the black configurations of Π_1, we can restrict attention to these labels also when we discuss the white constraints next.

To show (2), consider the white configurations in Definition 10.6. In the following, we list all white configurations of the black output problem of Π_0, by simply going through the three white configurations of Π_0 one by one, each time listing all "superconfigurations" consisting of δ labels from $\{A, H, T, X\}$ (i.e., those configurations from which the considered white configuration of Π_0 can be picked). For the first configuration of Π_0, this means anything containing an A and anything containing X (i.e. any label) in the remaining positions. We obtain the set of configurations $\{A, H, T, X\}[\{X\}\{H, X\}\{T, X\}\{A, H, T, X\}]^{d-1}$ which is $\mathbf{X}[\mathbf{AHTX}]^{d-1}$ after renaming. For the second configuration, we require $s + 1$ new labels containing H and the remaining labels containing X. We get the set of configurations

$$[\{H, X\}\{A, H, T, X\}]^{s+1}[\{X\}\{H, X\}\{T, X\}\{A, H, T, X\}]^{d-s-1}$$

which is $[\mathbf{H}, \mathbf{X}]^{s+1}[\mathbf{AHTX}]^{d-s-1}$ after renaming. We note that any contained configuration containing the label $\mathbf{X}$ is already contained in the first obtained set of configurations, and therefore we can simplify the second set to the claimed $\mathbf{H}^{s+1}[\mathbf{AHTX}]$. The last white configuration of Π_0 is similar to the second. We require $d - s + 1$ new labels containing the original label T and the remaining to contain the label X. We get the set of configurations

$$[\{T, X\}\{A, H, T, X\}]^{d-s+1}[\{X\}\{H, X\}\{T, X\}\{A, H, T, X\}]^{s-1}$$

which is $[\mathbf{T}, \mathbf{X}]^{d-s+1}[\mathbf{AHTX}]^{s-1}$ after renaming. Again, any configuration containing the label $\mathbf{X}$ is already contained in the first obtained set of configurations, so we can simplify the third set to $\mathbf{T}^{d-s+1}[\mathbf{AHTX}]^{s-1}$. As we can see, the obtained white configurations are exactly those listed in the lemma, which completes the proof. $\qquad\square$

We proceed to prove that the white output problem of Π_1 is $\mathsf{FDSO}(s)$.

Lemma 10.9. *The white output problem Π_2 of Π_1, from Lemma 10.8, is $\mathsf{FDSO}(s)$. In particular, Π_2 is the following problem. Given a renaming of the labels as*

$$
\begin{aligned}
A &= \{\mathbf{X}\} \\
H &= \{\mathbf{H}, \mathbf{X}\} \\
T &= \{\mathbf{T}, \mathbf{X}\} \\
X &= \{\mathbf{A}, \mathbf{H}, \mathbf{T}, \mathbf{X}\},
\end{aligned}
$$

the white constraint W_2 of Π_2 is given by the configurations

$$
\begin{aligned}
&AX^{d-1} \\
&H^{s+1}X^{d-s-1} \\
&T^{d-s+1}X^{s-1},
\end{aligned}
$$

36

and the black constraint B_2 of Π_2 is given by the configurations

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

Proof. The proof proceeds similarly to the proof of Lemma 10.8. According to the definition of a white output problem, we have to show that

(1) the specified white constraint W_2 contains exactly those maximal configurations $Y_1 \ldots Y_d$ for which for every choice $\mathbf{Y}_1 \in Y_1, \ldots, \mathbf{Y}_d \in Y_d$, we have that $\mathbf{Y}_1 \ldots \mathbf{Y}_d$ is a black configuration for Π_1, and

(2) the specified black constraint B_2 contains exactly those configurations $Z_1 \ldots Z_d$ for which there exists a choice $\mathbf{Z}_1 \in Z_1, \ldots, \mathbf{Z}_d \in Z_d$ such that $\mathbf{Z}_1 \ldots \mathbf{Z}_d$ is a white configuration for Π_1.

We start by proving (1). Recall that the white constraint of Π_1 is given by $\mathbf{X}[\mathbf{AHTX}]^{d-1}$, $\mathbf{H}^{s+1}[\mathbf{AHTX}]^{d-s-1}$, and $\mathbf{T}^{d-s+1}[\mathbf{AHTX}]^{s-1}$. Consider the first configuration AX^{d-1} in the white constraint W_2 of Π_2. Clearly, choosing one label from A, and one from each of the $d-1$ copies of X will result in a configuration that is contained in $\mathbf{X}[\mathbf{AHTX}]^{d-1}$ since the only choice to pick from A is $\mathbf{X}$. Similarly, for the second configuration $H^{s+1}X^{d-s-1}$ in W_2, any choice of $s+1$ labels from H and $d-s-1$ labels from X yields a configuration contained in $\mathbf{X}[\mathbf{AHTX}]^{d-1}$ (if at least one $\mathbf{X}$ is chosen), or a configuration contained in $\mathbf{H}^{s+1}[\mathbf{AHTX}]^{d-s-1}$ (if no $\mathbf{X}$ is chosen, which implies that all $s+1$ labels chosen from H are $\mathbf{H}$). Finally, for the third configuration $T^{d-s+1}X^{s-1}$ in W_2, any choice of $d-s+1$ labels from T and $s-1$ labels from X yields a configuration contained in $\mathbf{X}[\mathbf{AHTX}]^{d-1}$ (if at least one $\mathbf{X}$ is chosen), or a configuration contained in $\mathbf{T}^{d-s+1}[\mathbf{AHTX}]^{s-1}$ (if no $\mathbf{X}$ is chosen, which implies that all $d-s+1$ labels chosen from T are $\mathbf{T}$).

What is left to be shown is that AX^{d-1}, $H^{s+1}X^{d-s-1}$, and $T^{d-s+1}X^{s-1}$ are the only maximal configurations. (Again, it is straightforward to check that these configurations are indeed maximal.) Now assume that there is another maximal configuration $Y_1 \ldots Y_d$. Then, due to its maximality, each of the Y_i has to contain at least one label $\mathbf{Y}_i$ such that $\mathbf{Y}_i \neq \mathbf{X}$ (as otherwise $Y_1 \ldots Y_d$ would be "dominated" by AX^{d-1}), at least $d-s$ many of the Y_i have to contain a label that is not contained in H (as otherwise $Y_1 \ldots Y_d$ would be dominated by $H^{s+1}X^{d-s-1}$), and at least s many of the Y_i have to contain a label that is not contained in T (as otherwise $Y_1 \ldots Y_d$ would be dominated by $T^{d-s+1}X^{s-1}$). Now we claim that we can pick $\mathbf{Y}_1 \in Y_1, \ldots, \mathbf{Y}_d \in Y_d$ such that $\mathbf{Y}_i \neq \mathbf{X}$ for all i, $\mathbf{Y}_i \neq \mathbf{H}$ for $d-s$ many i, and $\mathbf{Y}_i \neq \mathbf{T}$ for s many i as follows.

Let $J_H \in [d]$ be a set of indexes such that $Y_i \setminus H \neq \emptyset$ for all $i \in J_H$ and $|J_H| = d-s$, and let $J_T \in [d]$ be a set of indexes such that $Y_i \setminus T \neq \emptyset$ for all $i \in J_T$ and $|J_T| = s$. These index sets exists due to our observations above. Now for all $i \in J_H \setminus J_T$, pick a $\mathbf{Y}_i$ satisfying $\mathbf{X} \neq \mathbf{Y}_i \neq \mathbf{H}$; for all $i \in J_T \setminus J_H$, pick a $\mathbf{Y}_i$ satisfying $\mathbf{X} \neq \mathbf{Y}_i \neq \mathbf{T}$; and for all $i \notin J_H \cup J_T$, pick a $\mathbf{Y}_i$ satisfying $\mathbf{X} \neq \mathbf{Y}_i$. Since $H \cap T = \{\mathbf{X}\}$, we can see that for each $i \notin J_H \cup J_T$, we have that $\mathbf{Y}_i$ is not contained in H or not contained in T. Furthermore, at least $(d-s) - |J_H \cap J_T|$ of the $d - |J_H \cap J_T|$ many $\mathbf{Y}_i$ we fixed so far are not contained in H, and at least $s - |J_H \cap J_T|$ are not contained in T. Since $d = (d-s) + s$, it follows that we can fix the remaining $|J_H \cap J_T|$ choices $\mathbf{Y}_i$ (where $i \in J_H \cap J_T$) so that at least $(d-s)$ of all $\mathbf{Y}_i$ are not contained in H, and at least s of all $\mathbf{Y}_i$ are not contained in T. This concludes the proof of the claim.

The choice of our $\mathbf{Y}_i$ immediately implies that $\mathbf{Y}_1 \ldots \mathbf{Y}_d$ is not a configuration contained in the white constraint W_1 of Π_1, yielding a contradiction to our assumption. Hence, AX^{d-1}, $H^{s+1}X^{d-s-1}$, and $T^{d-s+1}X^{s-1}$ are the only maximal configurations.

To show (2), consider the black configurations in Π_1. In the following, we list all black configurations of the white output problem of Π_1, by going through the two black configurations of Π_1 one by one, analogously to the approach in the proof of Lemma 10.8.

The first black configuration requires one element containing the label **A** and $\delta - 1$ elements containing the label **X**. This gives the set of configurations $X[AHTX]^{\delta-1}$ after renaming. The second black configuration requires an element containing **H**, an element containing **T**, and the remaining elements to contain **X**. This gives the set of configurations $[HX][TX][AHTX]^{\delta-2}$ after renaming. We observe that anything containing the element X is already covered by the first set of configurations, so we can simplify the second set of configurations to $HT[AHTX]^{\delta-2}$. As we can see, the obtained black configurations are exactly those listed in the lemma, which completes the proof. $\qquad\square$

10.7 FDSO is nontrivial

It remains to show that the FDSO problem cannot be solved in zero rounds.

Lemma 10.10. *Any 0-round randomized algorithm for* FDSO(s) *has local failure probability* $p \geq C$ *for some constant* C.

Proof. An FDSO(s) algorithm fails at a black node if and only if it is labeled with A^δ, H^δ, or A^δ. A white 0-round algorithm assigns probabilities p_1, p_2, and p_3 to outputting the three white configurations AX^{d-1}, $H^{s+1}X^{d-s-1}$, and $T^{d-s+1}X^{s-1}$. If $1 - p_1 - p_2 - p_3 = \Omega(1)$, then it outputs an illegal output with a constant probability, and we are done. If not, we have that at least one of p_i is $\Omega(1)$. We only consider the case $p_1 = \Omega(1)$, as the other cases are similar.

Consider an arbitrary black node u. Each incident white node v chooses the configuration AX^{d-1} independently with probability p_1. Since the white nodes can see only their random bits, they will assign the label A to a uniformly random incident edge. We get that the incident edges of u are labeled A^δ with probability at least $(p_1/d)^\delta$, which is $\Omega(1)$, as required. $\qquad\square$

10.8 Logarithmic lower bound for FDSO

We are now ready to put the pieces together and prove that any forbidden degree problem and bipartite sinkless orientation require $\Omega(\log \log n)$ randomized time.

Theorem 10.11. *All forbidden degree problems require* $\Omega(\log \log n)$ *randomized time in the LOCAL model. The same holds for bipartite sinkless orientation.*

Proof. By Lemma 10.7, it suffices to prove the desired lower bound for FDSO(s). For the sake of contradiction, assume that there exists an algorithm A with running time $2T = o(\log \log n)$ for solving FDSO(s) with probability $1 - 1/n$. This implies that A solves FDSO(s) with local failure probability at most $1/n$.

Using Lemmas 10.8 and 10.9, we have that after two round elimination steps, starting with FDSO(s), the obtained problem is again FDSO(s). Assuming that FDSO(s) can be solved with local failure probability p in time T' and given Lemma 10.5, we get that FDSO(s) can be solved in time $T' - 2$ with local failure probability at most

$$p' \leq K_2\Big(K_1 p^{1/(d+1)}\Big)^{1/(\delta+1)},$$

where $K_1 = K(d, \delta, 4)$ and $K_2 = K(\delta, d, 4)$. Let $\Delta = \max\{d, \delta\}$. We have that

$$p' \leq K_2 K_1^{1/(\delta+1)}\Big(p^{1/(\Delta+1)}\Big)^{1/(\Delta+1)} \leq K_2 K_1^{1/(\delta+1)} p^{1/(\Delta+1)^2}.$$

Starting with the assumed $2T$-time algorithm and iterating the above step T times we arrive at a 0-round algorithm for $\mathsf{FDSO}(s)$ with local failure probability at most some r. Since for some large enough constant C we have that $K_2 C^{1/(\delta+1)} \leq C$ and $K_1 C^{1/(d+1)} \leq C$, we get that the multiplicative term in front of p converges to some constant K. We get the bound

$$r \leq K \left(\frac{1}{n}\right)^{1/(\Delta+1)^T}.$$

If $T = o(\log\log n)$, then $r = o(1)$. By Lemma 10.10 we know that for any 0-round algorithm $r = \Omega(1)$, a contradiction with the assumption that $T = o(\log\log n)$. $\qquad\square$

Corollary 10.12. *All forbidden degree problems require $\Omega(\log n)$ deterministic time in the LOCAL model. The same holds for bipartite sinkless orientation.*

Proof. By Theorem 10.11, the deterministic complexity of any forbidden degree problem is $\Omega(\log\log n)$. Since there are no LCLs with deterministic complexity between $\omega(\log^* n)$ and $o(\log n)$ [12], the complexity must be $\Omega(\log n)$. $\qquad\square$

10.9 Proof of Theorem 10.1

Lemma 10.4 shows that any problem that is not of type **I**, **II**, **III**, **IV**, **V** or **VI** can be relaxed to a forbidden degree problem, the bipartite sinkless orientation problem, or a problem equivalent to one of those. Corollary 10.12 shows that all forbidden degree problems and bipartite sinkless orientation require $\Omega(\log n)$ rounds. Observation 3.2 ensures that this lower bound also holds for all problems that are equivalent to a forbidden degree problem or bipartite sinkless orientation. Hence, Theorem 10.1 follows.

11 Randomized complexity

We have now completed the classification of the deterministic distributed complexity of binary labeling problems in trees; all solvable problems fall in one of the complexity classes of $O(1)$, $\Theta(\log n)$, and $\Theta(n)$. Trivially, $O(1)$ deterministic complexity also implies $O(1)$ randomized complexity, and it is straightforward to verify that all problems discussed in Section 8 require $\Theta(n)$ rounds not only for deterministic algorithms but also for randomized algorithms. However, the class of $\Theta(\log n)$ is much more interesting, as it contains both problems in which randomness helps and also problems in which randomness does not help. In this section we will explore this in more detail and show how to classify *some* binary labeling problems according to their randomized complexity. Throughout the section, it is assumed that $d \geq 3$, or $\delta \geq 3$.

11.1 Propagation lower bound

In this section, we prove that if $\Pi = (d, \delta, W, B)$ is a binary LCL with deterministic complexity $\Theta(\log n)$ such that both white and black constraints do not have a so called *escape*, then also the randomized complexity of Π is $\Theta(\log n)$. We begin with the definition of an escape using the vector notation (cf. Section 3).

Definition 11.1 (Escape)**.** Given a binary LCL $\Pi = (d, \delta, W, B)$ we say that Π has a *white* (resp. *black*) *escape* if W (resp. B) contains two consecutive 1s.

The definition of an escape immediately implies the following observation.

Observation 11.2 (Propagation)**.** *Let Π be a binary LCL with no white and no black escape. If we fix for a given node $v \in V$ the labeling of all but one port in any way, then there is at most a single way to complete the labeling of the last port to produce a valid output for node v.*

Theorem 11.3 (Randomized propagation lower bound)**.** *Let $\Pi = (d, \delta, W, B)$ be a binary LCL such that there is neither a white nor a black escape and assume that Π has deterministic complexity $\Theta(\log n)$ on trees. Then Π has randomized complexity $\Theta(\log n)$ on trees.*

Proof. Let Π be a problem satisfying the conditions of the theorem and let d denote the degree of white nodes and let δ be the degree of black nodes (note that the considered trees are regular except for the degrees at the leaves). We emphasize that the degrees d and δ are fixed and constant throughout this proof. As there is a deterministic $O(\log n)$-round algorithm for Π there is also a randomized algorithm that solves Π in $O(\log n)$ rounds.

Next, we show that there is no randomized algorithm for Π with complexity $o(\log n)$. Assume towards a contradiction that there is an algorithm $\mathcal{A}$ solving Π in $o(\log n)$ rounds and fix a constant $0 < c < (100 \log \max\{d, \delta\})^{-1}$. By the assumption there exists some n_0 such that $\mathcal{A}$ solves Π on any tree with n_0 nodes (that satisfies the degree restrictions of Π) in less than $k(n_0) = c \cdot \log n_0$ rounds. Fix any n larger than n_0 for which a tree meeting the degree requirements of Π exists. We show that there is a tree with n nodes and a port assignment on which algorithm $\mathcal{A}$ has to fail. To this end let $T = (V, E)$ be a tree (satisfying the degree restrictions of Π) with n nodes such that it has an internal white node u that is more than $10k$ hops away from any leaf of T. Due to the choice of c such a tree exists. However, constants in this proof are not optimized to ease presentation. Denote by $v_i, 1 \le i \le d$ the neighbors of u, by $e_i = (u, v_i)$ the incident edges of u and by $T_i = (V_i, E_i)$ the tree of depth $2k - 1$ rooted at v_i such that $u \notin V_i$. For each tree T_i, denote by $S_i \subseteq V_i$ the subset of nodes that are at distance $2k$ from u, that is, S_i is the set of leaves of T_i. Let F_i be the set of edges of T_i that are incident to a node in S_i. Equip the tree with any port labeling such that *corresponding nodes* in S_i and S_j for $1 \le i, j \le d$ have the exact same k-hop view. Since both white and black constraints have no escape, we have full propagation in the following sense:

Propagation Claim: For any $1 \le i \le d$ and under the assumption that $\mathcal{A}$ is a correct algorithm for Π the output of $\mathcal{A}$ on the edges of F_i uniquely determines the output labels of all other edges of T_i and also the output label of the edge e_i.

Let us prove the propagation claim. If we fix for a given node v the labeling of all but one ports in any way, there is at most a single way to complete the labeling of the last port to produce a valid output for v (cf. Observation 11.2). The claim follows by an induction on the levels of the tree T_i starting with the 'leaf edges' incident to S_i. Hence the claim follows.

As $\mathcal{A}$ runs in less than k rounds for all $1 \le i \le d$ and the nodes in S_i have distance $2k$ from u, the nodes in S_i decide on their output independently from the information in the graph T_j for all $j \ne i$ and also independent from e_j. Even stronger this independence is not just pairwise but the output on the sets of edges $F_1, \ldots, F_d$ are fully independent. Furthermore, as the graph and the port numbering around all these sets is identical the outputs produced by the randomized algorithm $\mathcal{A}$ on $F_1, \ldots, F_d$ follow the same probability distribution. Using the propagation claim we obtain that also all edges $e_1, \ldots, e_d$ follow the same output distribution and the output on these edges is fully independent. Thus all edges $e_1, \ldots, e_d$ are in one of three following (non exclusive) cases: For all $1 \le i \le d$

- **Case 1:** the output of S_i forces e_i to be labeled 0 with probability at least $1/4$ and it forces e_i to be labeled with 1 with probability at least $1/4$, or

- **Case 2:** with probability at least $1/4$, the output of the nodes in S_i forces e_i to be labeled with 1, or

- **Case 3:** with probability at least $1/4$, the output of the nodes in S_i forces e_i to be labeled with 0.

We deduce a contradiction for each case using that all edges are in the same case and the problem Π is non trivial as it has deterministic complexity $\Theta(\log n)$ by assumption.

Case 1: As Π is not a trivial problem there exists some $m \leq d$ such that Π does not allow for exactly $m \leq d$ ports to be labeled 1 incident around white nodes. However, as each edge $e_1, \ldots, e_d$ is independently labeled with 1 or 0 each with probability at least $1/4$ the probability that the first m edges $e_1, \ldots, e_m$ incident to u are labeled with 1 and the remaining incident edges $e_{m+1}, \ldots e_d$ are labeled with 0 is constant, i.e., it is at least $(1/4)^m \cdot (1/4)^{d-m} = (1/4)^d = \Omega(1)$. As this labeling is not allowed for u the algorithm fails at least with constant probability at u, a contradiction.[1]

Cases 2 and 3: Both cases are symmetric with interchanging the role of the labels. Assume we are in case 2, and first assume Π does not allow white nodes to label all incident edges with 1. Then with constant probability, i.e., with probability at least $(1/4)^d$, algorithm $\mathcal{A}$ labels all edges incident to u with label 1, i.e., $\mathcal{A}$ fails to produce a valid output for u with constant probability, a contradiction.

If we are in case 2 and Π does allow white nodes to label all ports with 1 we construct a new tree and a port numbering on which algorithm $\mathcal{A}$ fails for a black node with constant probability: Consider a black node t and denote by $u_1, \ldots, u_\delta$ its white neighbors, now similarly fix the trees and the port numbers rooted at the children of $u_i, 1 \leq i \leq \delta$ to be such that the edges incident to $u_i, 1 \leq i \leq \delta$ are in case 2, i.e. labeled 1 with probability at least $1/4$, except for the edge connected to t.[2] Now since we have full propagation and no escape, and we know that white nodes are allowed to label their ports with all 1, we deduce that also the edges $(t, u_i), 1 \leq i \leq \delta$ must be labeled with 1 with probability at least $1/4$, since by the no escape property, an output of a single port labeled 0 is not allowed. Thus we have that with probability at least $(1/4)^\delta = \Omega(1)$, t labels all its edges with 1. As this cannot be a feasible output, otherwise the problem would be trivial as then white and black nodes would accept all edges to be labeled with 1, the algorithm fails with constant probability, a contradiction.

In either case, we reach a contradiction, thus there is no algorithm that solves Π w.h.p. in less than k rounds which proves the claim. □

11.2 Lower bounds using independent sets

In this section, we prove the following theorem.

Theorem 11.4. *Consider the following binary LCL problem* $\Pi = (d, 2, W, B)$, $W = 110^{d-2}1$, $B = 010$. *Then for all* $d \geq 20$, *this problem has randomized complexity* $\Omega(\log n)$.

To prove this theorem, we employ the following lemma, whose proof can be found in [1, Lemma 21].

---

[1] The actual probability of failure at a white node in this case is much larger but the presented probability is sufficient for a contradiction.

[2] The constant c is chosen small enough such that we do not run out of nodes in this construction if we cap trees at a large enough radius that does not influence the output at the considered vertices.

Lemma 11.5. *Let $d \geq 3, \epsilon > 0$, then there exists a d-regular graph G on n vertices such that G has girth $\Omega(\log n)$, and furthermore $\alpha(G) \leq \left(\frac{2 \ln d}{d} + \epsilon\right) n$. Here, $\alpha(G)$ denotes the size of the largest independent set in G.*

Now we are ready to prove Theorem 11.4.

Proof. Given some graph $G = (V, E)$ in the black/white formalism, denote by V_W, V_B the sets of black and white nodes respectively. Note that the constraints of the black nodes allow us to view the problem as an orientation problem on white nodes. Let $c : E \to \{0, 1\}$ be some labeling to the edges that induces a valid solution to Π. Define

$$E' = \left\{ (v_1, v_2) \mid v_1, v_2 \in V_W, \exists b \in V_B, (v_1, b) \in E, (b, v_2) \in E \right\}.$$

We can abuse notation and view c as function from $E' \to \{\leftarrow, \rightarrow\}$. Basically, as an orientation function. This is since given some black node $b \in B$ and his white neighbors v, u, assume w.l.o.g. that $c(u, b) = 1, c(v, b) = 0$, then we define that $c(u, v) = \rightarrow$, and vice versa. Here $\rightarrow$ means that (u, v) is oriented towards v, $\leftarrow$ is defined similarly. Note that each white node still has total degree d, thus we are now working with a d-regular graph. From here on in, we treat E' as our set of edges, and W as our set of nodes (we refer to them using E, V, respectively). Now, outdegree(v) of a node $v \in V_W$ is the number of edges incident to it oriented away from v. Similarly, we define indegree(v) of a node $v \in V_W$. Denote by x_0, x_1, x_d the number of nodes with outdegree $0, 1, d$ respectively.

Now, since each edge in the graph is oriented, we have that

$$\sum_{v \in V} \text{indegree}(v) = \sum_{v \in V} \text{outdegree}(v).$$

Or, in other words, since the total degree of each node is d:

$$dx_0 + (d - 1)x_1 = x_1 + dx_d.$$

Thus $x_d = x_0 + \frac{d-2}{d}x_1$. We also know that $x_0 + x_1 + x_d = n$. From these two equations we continue with two possible cases:

- $x_d \geq \frac{n}{3}$.
- $x_1 \geq \frac{n}{3}$, in which case we have that $x_d \geq \frac{d-2}{3d}n$.

Now since $d \geq 11$, we can simply deduce that $x_d \geq \frac{d-2}{3d}n$ in all cases.

Denote by V_d all the nodes with outdegree d, note that V_d is an independent set and that $|V_d| = x_d$. Thus we can conclude that any d-regular graph in which there is a valid solution to Π, there is an independent set of size at least $\frac{d-2}{3d}n$.

Now assume towards a contradiction that Π is solvable on d-regular trees in $o(\log n)$ rounds. In particular, following from an indistinguishability argument, which holds also for randomized algorithms, the same algorithm should solve Π on d-regular graphs with girth $\Omega(\log n)$. However, according to Lemma 11.5, there is a d-regular graph with girth $\Omega(\log n)$ and $\alpha(G) \leq \left(\frac{2 \ln d}{d} + \epsilon\right) n$, for any $\epsilon > 0$, denote such a graph by G_0. Furthermore, each valid solution to Π in a graph G induces an independent set of size at least $\frac{d-2}{3d}n$ in G, notice that for all $d \geq 20$, it holds that $\frac{d-2}{3d}n > \left(\frac{2 \ln d}{d} + \epsilon\right) n$. According to our assumptions Π will be solved correctly on G_0, since it has $\Omega(\log n)$ girth. This is a contradiction to the fact that any valid solution to Π induces an independent set of size at least $\frac{d-2}{3d}n$. Thus, our assumption was wrong and no such algorithm exists. In particular, Π has randomized complexity $\Omega(\log n)$ as required. $\qquad \square$

11.3   Lower bounds using cuts

In this section, our goal is to prove the following theorem,

Theorem 11.6. *Given some integers $r, d, r < \frac{d}{2}$, consider the following binary LCL $\Pi = (d, 2, W, B)$, $W = 1^{r+1}0^{d-2r-1}1^{r+1}$, $B = 010$. Then Π has randomized complexity $\Omega(\log n)$ whenever $r < \frac{d}{4} - \frac{\sqrt{d-1}}{2}$.*

To prove the theorem, we will employ the following lemma.

Lemma 11.7. *Given an integer $d \geq 3$ and any $\epsilon > 0$, there exists a d-regular graph G on n vertices of girth $\Omega(\log n)$, and for all $S \subseteq V$, it holds that $\frac{E(S)}{|E|} \leq \frac{1}{2} + \frac{\sqrt{d-1}+\epsilon}{d}$. Here, $E(S)$ is the number of edges in G crossing the cut S.*

The lemma follows from previous work on the second eigenvalue of random d-regular graphs.

Proof sketch. Friedman [16, 17] showed that for the smallest eigenvalue λ_n of the adjacency matrix of a random d-regular graph (in the standard configuration model, for every positive integer d) we have that $|\lambda_n| \leq 2\sqrt{d-1} + \varepsilon$, for any $\varepsilon > 0$, with probability $1 - O(n^{-c(d)})$ for some $c(d) = \Omega(\sqrt{d})$.

It is known that the size of maximum cut of a d-regular graph is bounded by $(1/2 + |\lambda_n|/(2d))|E|$ (see for example Trevisan [31]). This implies that most random d-regular graphs do not have large cuts. It remains to construct one with large girth.

We use the standard trick of cutting cycles (see e.g. [1, Section 2]). It is known that with probability at least $1/2$ there are at most $2\sqrt{n}$ cycles of length at most $0.5 \log n / \log d$ in a random d-regular graph. Therefore there exists such a graph with no large cuts. It is possible to cut each cycle by removing and adding exactly two edges without introducing any new short cycles. In the worst case all of the new edges are cut by some maximum cut of the original graph. We have that the size of the maximum cut in the new graph is bounded by $(1/2 + (2\sqrt{d-1} + \varepsilon)/(2d))dn/2 + 2\sqrt{n}$, which is at most $(1/2 + (\sqrt{d-1} + 2\varepsilon)/d$ for large enough n. □

Now, we are ready to prove Theorem 11.6.

Proof. As in the proof of Theorem 11.4, given some graph $G = (V, E)$ in the black/white formalism, denote by V_W, V_B the sets of black and white nodes respectively. Note that the constraints of the black nodes allow us to view the problem as an orientation problem on white nodes. Let $c : E \rightarrow \{0, 1\}$ be some labeling to the edges that induces a valid solution to Π. Define

$$E' = \big\{ (v_1, v_2) \mid v_1, v_2 \in V_W, \exists b \in V_B, (v_1, b) \in E, (b, v_2) \in E \big\}.$$

We can abuse notation and view c as function from $E' \rightarrow \{\leftarrow, \rightarrow\}$. Basically, as an orientation function. This is since given some black node $b \in B$ and his white neighbors v, u, assume w.l.o.g. that $c(u, b) = 1, c(v, b) = 0$, then we define that $c(u, v) = \rightarrow$, and vice versa. Here $\rightarrow$ means that (u, v) is oriented towards v, $\leftarrow$ is defined similarly. Note that each white node still has total degree d, thus we are now working with a d-regular graph. From here on in, we treat E' as our set of edges, and W as our set of nodes (we refer to them using E, V, respectively). Now, outdegree(v) of a node $v \in V_W$ is the number of edges incident to it oriented away from v. We define similarly indegree(v) of a node $v \in V_W$.

Now note that Π is the problem in which every white node has to have outdegree at most r, or indegree at most r. Given some solution to Π denote by X_0, X_1 the sets of nodes of outdegree at most r and indegree at most r, respectively. Note that $V \backslash X_0 = X_1$, thus these two sets define a cut in the graph. Now we want to show that the number of edges leaving X_0 into X_1 is bounded

from below by $(d - 2r)|X_0|$. Assume by towards a contradiction that this is not the case. Denote by E_ℓ the set of edges incident to a node in X_0 that cross the cut (X_0, X_1). Denote by E_s the set of edges in which both endpoints are in X_0. Our assumption is that $|E_\ell| < (d - 2r)|X_0|$. Note that we have that $d|X_0| = 2|E_s| + |E_\ell|$ simply since $d|X_0|$ is the sum of degrees of nodes in X_0. From our assumption we can then deduce that $|E_s| > r|X_0|$. Now, we know that for all $v \in X_0$, outdegree$(v) \le r$. Thus $\sum_{v \in X_0}$ outdegree$(v) \le r|X_0|$. However, this is a contradiction, since by definition of E_s and the fact that each edge is oriented and the fact that $|E_s| > r|X_0|$ we deduce that $\sum_{v \in X_0}$ outdegree$(v) > r|X_0|$.

Thus, we know that $|E(X_0, X_1)| \ge (d - 2r)|X_0|$. Similarly, we can prove that $|E(X_0, X_1)| \ge (d - 2r)|X_1|$. Assume w.l.o.g. that $|X_0| \ge |X_1|$. Thus we get that

$$\frac{|E(X_0, X_1)|}{|E|} \ge \frac{|X_0|(d - 2r)}{0.5d(|X_0| + |X_1|)} \ge \frac{|X_0|(d - 2r)}{d|X_0|} = 1 - \frac{2r}{d}.$$

Thus, we proved, that in any d-regular graph $G = (V, E)$, a solution to Π induces a cut $S \subseteq V$ s.t. $\frac{E(S)}{|E|} \ge 1 - \frac{2r}{d}$.

Now, assume towards a contradiction that there exists an algorithm A, that runs in $o(\log n)$ rounds, that solves Π w.h.p. on any d-regular tree G. Let G_0 be the graph promised from Lemma 11.7 for a sufficiently small $\epsilon > 0$.

Now, since G_0 has girth $\Omega(\log n)$, from an indistinguishability argument, that also holds for randomized algorithms, it must hold that A also solves Π correctly on G_0 w.h.p., since for every node $v \in V$, the view of v when executing A is that of a d-regular tree. Thus we can deduce that G_0 has a cut S induced by X_0 and X_1 as explained before, of relative size $1 - \frac{2r}{d}$. But, by our assumption, it holds that

$$r < \frac{d}{4} - \frac{\sqrt{d-1}}{2}.$$

Thus, we have that

$$1 - \frac{2r}{d} > \frac{1}{2} + \frac{\sqrt{d-1}}{d}.$$

Since we treat d, r as constants, this is a contradiction with Lemma 11.7 for a sufficiently small $\epsilon > 0$.

Thus, we conclude that no such algorithm A exists. Thus any algorithm that solves Π correctly w.h.p. on d-regular trees must have rounds complexity $\Omega(\log n)$. Thus Π has randomized complexity $\Omega(\log n)$ whenever $r < \frac{d}{4} - \frac{\sqrt{d-1}}{2}$ as required. $\qquad\square$

Acknowledgments

References

[1] Noga Alon. On Constant Time Approximation of Parameters of Bounded Degree Graphs. In Oded Goldreich, editor, *Property Testing: Current Research and Surveys*, volume 6390 of *LNCS*, pages 234–239. Springer, 2010. doi:10.1007/978-3-642-16367-8_14.

[2] 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). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.DISC.2018.9.

[3] 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.

[4] 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.

[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)*, 2019.

[6] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. How much does randomness help with locally checkable problems?, 2019. URL http://arxiv.org/abs/1902.06803.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] Joel Friedman. A proof of Alon's second eigenvalue conjecture. In *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 720–724. ACM Press, 2003. doi:10.1145/780542.780646.

[17] Joel Friedman. A proof of Alon's second eigenvalue conjecture and related problems. *Memoirs of the American Mathematical Society*, 195(910), 2008. doi:10.1090/memo/0910.

[18] 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.

[19] 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.

[20] 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)*, 2018. doi:10.1109/FOCS.2018.00069. URL http://arxiv.org/abs/1711.02194.

[21] 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.

[22] Juhana Laurinharju and Jukka Suomela. Brief announcement: Linial's lower bound made easy. In *Proc. 33rd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2014)*, pages 377–378. ACM Press, 2014. doi:10.1145/2611462.2611505.

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

[24] 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.

[25] 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.

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

[27] Dennis Olivetti. Round Eliminator: a tool for automatic speedup simulation, 2019. URL https://github.com/olidennis/round-eliminator.

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

[29] Will Rosenbaum and Jukka Suomela. Seeing Far vs. Seeing Wide: Volume Complexity of Local Graph Problems, 2019. URL http://arxiv.org/abs/1907.08160.

[30] Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-Time Deterministic Network Decomposition and Distributed Derandomization. 2019. URL http://arxiv.org/abs/1907.10937.

[31] Luca Trevisan. Max Cut and the Smallest Eigenvalue. *SIAM Journal on Computing*, 41(6): 1769–1786, 2012. doi:10.1137/090773714.