

The Landscape of Distributed Complexities on Trees and Beyond

CHRISTOPH GRUNAU*, ETH Zurich, Switzerland

VÁCLAV ROZHOŇ, ETH Zurich, Switzerland

SEBASTIAN BRANDT, CISPA Helmholtz Center for Information Security, Germany

We study the local complexity landscape of locally checkable labeling (LCL) problems on constant-degree graphs with a focus on complexities below $\log^* n$. Our contribution is threefold:

- (1) Our main contribution is that we complete the classification of the complexity landscape of LCL problems on trees in the LOCAL model, by proving that every LCL problem with local complexity $o(\log^* n)$ has actually complexity $O(1)$. This result improves upon the previous speedup result from $o(\log \log^* n)$ to $O(1)$ by [Chang, Pettie, FOCS 2017].
- (2) In the related LCA and VOLUME models [Alon, Rubinfeld, Vardi, Xie, SODA 2012, Rubinfeld, Tamir, Vardi, Xie, 2011, Rosenbaum, Suomela, PODC 2020], we prove the same speedup from $o(\log^* n)$ to $O(1)$ for *all* bounded degree graphs.
- (3) Similarly, we complete the classification of the LOCAL complexity landscape of oriented d -dimensional grids by proving that any LCL problem with local complexity $o(\log^* n)$ has actually complexity $O(1)$. This improves upon the previous speed-up from $o(\sqrt[d]{\log^* n})$ by Suomela in [Chang, Pettie, FOCS 2017].



Additional Key Words and Phrases: Graph problems, LCL problems, local complexity, volume model

1 INTRODUCTION

One fundamental aspect in the theory of distributed computing is *locality*: for a given graph problem, how far does each individual node have to see in order to find its own part of a globally correct solution? The notion of locality is captured by the local complexity of the LOCAL model of distributed computing [36]. In recent years, our understanding of locality has improved dramatically by studying it from a complexity theoretical perspective [7, 17, 19, 21, 41], where the goal is to characterize the possible local complexities of all problems from a certain problem class. Most of the past research in this direction has focused on the study of so-called Locally Checkable Labeling problems (LCLs) [41]. Informally speaking, LCLs are graph problems that can be verified locally. That is, if a given output looks correct in the local neighborhood of each node, then the output is guaranteed to be a globally correct solution. Prominent examples of LCLs are Maximal Independent Set, Maximal Matching and all kinds of coloring problems. LCLs are only defined for constant degree graphs.

Interestingly, the study of local complexity has also been shown to be quite relevant for other subfields of computer science such as local computation algorithms or massively parallel computation [28, 44], and is finding its applications in subareas of discrete mathematics as well [12, 15, 34].

Below $\log^ n$.* A classical result of Linial [36] shows that the deterministic LOCAL complexity of $O(\Delta^2)$ -coloring a graph with maximum degree Δ is $\Theta(\log^* n)$.¹ In particular, this result directly implies that many basic symmetry breaking problems like $(\Delta + 1)$ -coloring have a deterministic LOCAL complexity of $\Theta(\log^* n)$ on constant degree graphs. In contrast, there are many basic problems with complexity $O(1)$. An example is to “find the maximum degree of a node in your 2-hop neighborhood”. The common theme for all our contributions is to understand what happens in

*The author ordering was randomized using <https://www.aeaweb.org/journals/policies/random-author-order/generator>. It is requested that citations of this work list the authors separated by \textcircled{r} instead of commas: Grunau  Rozhoň  Brandt.

¹The function $\log^*(n)$ is defined as the number of times one needs to repeatedly apply the log function to n until the result becomes at most 1.

between the two worlds. By a result of Chang and Pettie [21], whose proof was inspired by ideas introduced by Naor and Stockmeyer [41], we know that all LCLs with local complexity $o(\log \log^* n)$ have, in fact, complexity $O(1)$.

Complementing this result, in [10], the authors provided examples of LCLs with complexities between $\Theta(\log \log^* n)$ and $\Theta(\log^* n)$. Roughly speaking, examples of those problems are the following: one is asked to solve a basic $\Theta(\log^* n)$ -complexity problem on a path. The input graph, however, is not just a path. It is a path P plus a shortcutting structure on top of it that ensures that the t -hop neighborhood of each node $u \in P$ in the full input graph G actually contains the $f(t)$ -hop neighborhood of u in the subgraph P for some function f . This results in a problem with complexity equal to $\Theta(f^{-1}(\log^* n))$. The growth rate of the function f can range from linear (if no shortcutting is involved) up to exponential, since the neighborhood at distance t in the full graph G can contain up to roughly Δ^t nodes. Hence we obtain complexities from the range $\Theta(\log \log^* n) - \Theta(\log^* n)$. It should be noted that such constructed problems have a special structure. In fact, one can observe the following two points:

- (1) To construct such a problem, the input graph needs to contain shortcuts and, hence, cycles. In Theorem 1.1 we prove that if the input graph is a tree, there cannot be any such problems. This completes the classification of LCLs on trees.
- (2) Although the necessary radius that a node u needs to check to solve the constructed problem can be only $O(\log \log^* n)$, the number of nodes that u needs to look at to solve the problem is still $O(\log^* n)$. In Theorem 1.3 we prove that in the VOLUME complexity model, where the main measure is the number of nodes u needs to probe, not the radius, there are in fact no complexities between $\omega(1)$ and $o(\log^* n)$.

1.1 Main Contribution: Finishing the Classification of LCLs on Trees

In this section we present a prominent achievement of the theory of LCLs: by a long line of work [8, 10, 13, 19, 21, 23, 25–27, 29, 36, 40, 41, 43], we now know that there are only four types of LCL problems, as described in Figure 1, bottom left:

- (A) Problems that can be solved in $O(1)$ rounds.
- (B) Problems with complexity (both randomized and deterministic) in the range $\Theta(\log \log^* n) - \Theta(\log^* n)$; these include basic symmetry breaking problems like $(\Delta + 1)$ -coloring and maximal independent set.
- (C) Problems with randomized complexity $\text{poly} \log \log n$ and deterministic complexity $\text{poly} \log n$; these problems can be solved by reformulating them as an instance of the Lovász local lemma (LLL).
- (D) Global problems with complexity $\Omega(\log n)$; their randomized and deterministic complexity is the same up to a polylogarithmic factor.

One particular focus in the long line of work on complexities of LCLs has been on understanding the complexity landscape on trees—not only because trees are a natural subclass of general graphs, but also because they turned out to be important for the development of understanding the complexities on general graphs; in particular many lower bounds for LCLs are proven on trees, e.g., via the round elimination technique [14]. As a result the complexity landscape on trees is quite well understood: previous to our work, each LCL on trees was known to have one of the following complexities (unless stated otherwise, the deterministic and randomized complexities are the same):

- (1) $O(1)$,
- (2) in the range $\Theta(\log \log^* n) - \Theta(\log^* n)$,
- (3) deterministic complexity $\Theta(\log n)$ and randomized complexity $\Theta(\log \log n)$,
- (4) complexity $\Theta(\log n)$,

(5) complexity $\Theta(n^{1/k})$ for some positive integer k .

This follows from a long line of work [8, 13, 18–21, 41]. The only missing piece was to determine for which complexities in the range $\Theta(\log \log^* n) - \Theta(\log^* n)$ there actually exists an LCL with that complexity. We complete the complexity landscape of LCLs on trees by showing that any LCL with a complexity in this range has actually complexity $\Theta(\log^* n)$. That is, we prove that any LCL on trees with complexity $o(\log^* n)$ has complexity $O(1)$.

THEOREM 1.1 (INFORMAL VERSION OF THEOREM 3.11). *Let Δ be any fixed positive integer. Any LCL on trees (with maximum degree at most Δ) with LOCAL complexity $o(\log^* n)$ has, in fact, LOCAL complexity $O(1)$.*

Put together with the aforementioned work on the classification of LCLs on trees, we get the following corollary (see Figure 1, top left, for an illustration).

COROLLARY 1.2 (Classification of LCLs on trees, see Figure 1). *Let Π be an LCL on trees. Then the deterministic/randomized LOCAL complexity of Π is one of the following:*

- (1) $O(1)$,
- (2) $\Theta(\log^* n)$,
- (3) $\Theta(\log n)$ deterministic and $\Theta(\log \log n)$ randomized,
- (4) $\Theta(\log n)$,
- (5) $\Theta(n^{1/k})$ for $k \in \mathbb{N}$.

Moreover, all mentioned complexity classes are non-empty, i.e., for each class (and each k), there exists some LCL with the indicated complexity.

An intriguing implication of our work is that the complexity landscape of LCLs on trees is *discrete*, while the landscape of LCLs on general (bounded-degree) graphs contains “dense” regions (in classes (B) and (D) [8, 10]). It would be interesting to see to what extent this is a more general phenomenon beyond LCLs on bounded-degree graphs.

Our method, in a nutshell. Our approach is based on the round elimination technique [14], a highly successful technique for proving local lower bounds. In essence, round elimination is an explicit process that takes an LCL Π on trees as input and returns an LCL Π' with complexity exactly one round less. More precisely:

- (1) If there is a T -round randomized algorithm \mathcal{A} for Π , then there is also a $(T - 1)$ -round randomized algorithm \mathcal{A}' for Π' such that the (local) failure probability of \mathcal{A}' is bounded by a reasonable function of the (local) failure probability of \mathcal{A} .
- (2) If we have a $(T - 1)$ -round algorithm \mathcal{A}' for Π' , we can use it to construct a T -round algorithm \mathcal{A} for the original problem Π ; if \mathcal{A}' is deterministic, then \mathcal{A} is deterministic as well.

So far, in the literature, the standard use case for applying round elimination has been to prove lower bounds for some *concrete, fixed* problem such as maximal matching or Lovász local lemma [4, 6, 13, 14]. We provide a novel application of round elimination by showing that, perhaps surprisingly, it can also be used to prove gap results, which are results that reason about *all* LCLs on a given graph class. More precisely, we show that with the tool of round elimination at hand, there is an elegant way to prove Theorem 1.1, which roughly proceeds as follows.

We start with any problem Π for which there exists a randomized algorithm \mathcal{A} that solves Π in $T(n) = o(\log^* n)$ rounds, with probability $1 - 1/\text{poly}(n)$. We fix some sufficiently large number n_0 of nodes, and apply bullet point (1) $T = T(n_0)$ times to get a 0-round algorithm $\mathcal{A}^{(T)}$ for a certain problem $\Pi^{(T)}$. By analyzing the development of the

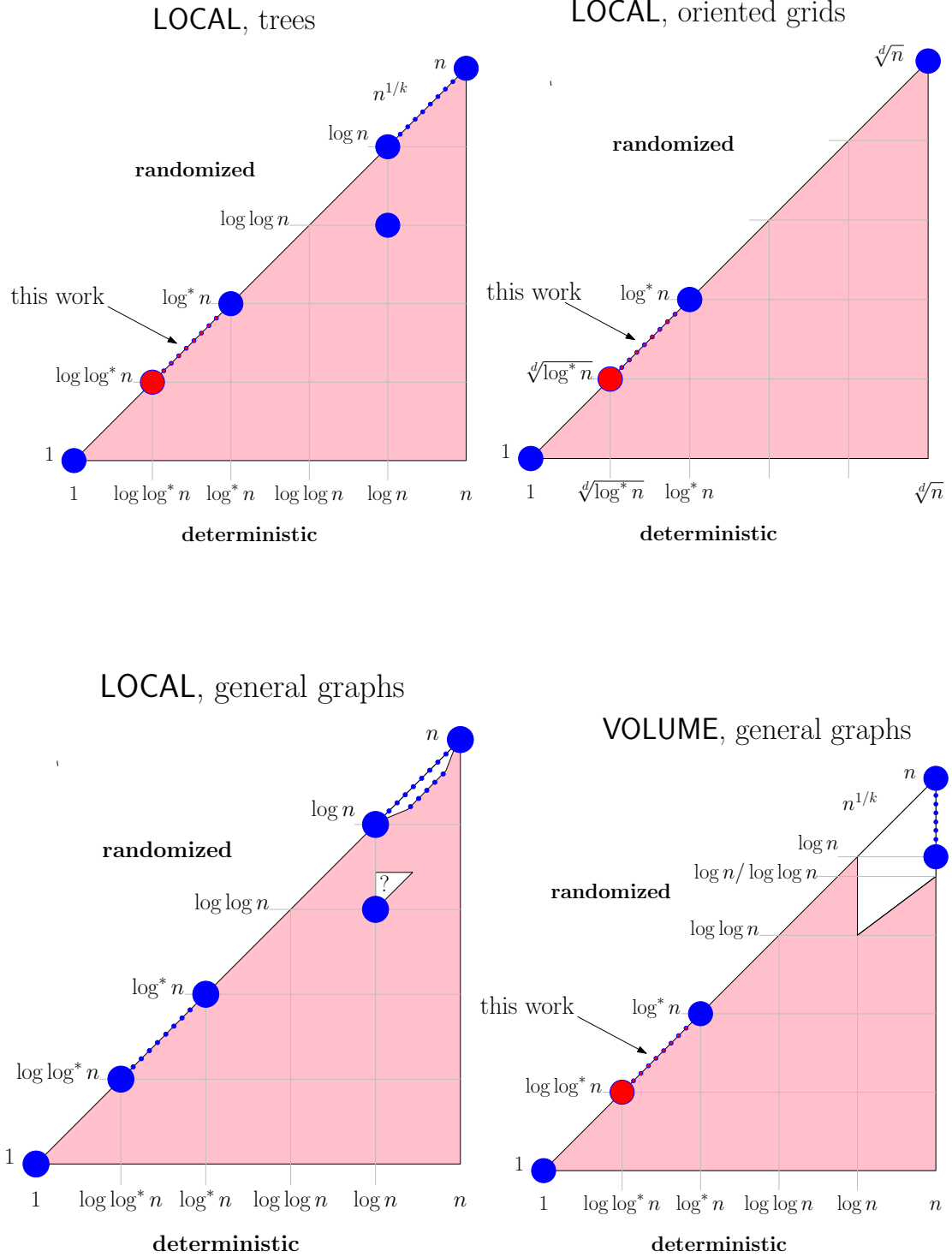


Fig. 1. The LCL landscape of local complexities on trees (top left), oriented grids (top right), general bounded degree graphs (bottom left) and the landscape of the volume model (bottom right). Blue circles correspond to possible deterministic and randomized local complexities, while the red color means that no problems with the given complexity are possible in the area. The arrows point to our contribution: completion of the classification of LCLs on trees and oriented grids by showing that there are no problems with complexities between $\Theta(\log^* n)$ and $O(1)$. We now also understand the landscape of the VOLUME/ LCA model in the regime of low complexities. See [46] for a friendly introduction to the topic of LCL landscapes.

(local) failure probabilities of the algorithms appearing during the T applications of (1), we can show that algorithm $\mathcal{A}^{(T)}$ still has a large probability of success, and the fact that $\mathcal{A}^{(T)}$ is a 0-round algorithm enables us to infer that $\Pi^{(T)}$ is in fact so easy that it can be solved with a *deterministic* 0-round algorithm. Finally, we apply bullet point (2) T times to obtain a deterministic T -round algorithm for the original problem Π . Due to fixing the number of nodes to n_0 , the obtained T -round algorithm is only guaranteed to produce a correct output on n_0 -node trees; however, due to the nature of 0-round algorithms and the precise definition of the round elimination process (which are both independent of the number of nodes of the input graph), the obtained algorithm can be shown to also work on trees with an arbitrary number of nodes, with precisely the same, constant runtime $T(n_0) = O(1)$.

Unfortunately, the known approach for analyzing the change of (local) failure probability in bullet point (1) considers only the restricted setting of regular graphs and LCLs without inputs² (which usually suffices when proving a lower bound for a concrete LCL). One of our technical contributions is to develop an extension that also works in the general setting of irregular graphs and LCLs with inputs, which might be of independent interest.

Further related work. Previous to our work, the gap result of Theorem 1.1 was known for a subclass of LCLs, called homogeneous LCLs [11]. Roughly speaking, problems in this class require the output of a node u to be correct only if the part of the tree around u is a perfect Δ -regular tree without any inputs. In contrast, our argument works in full generality, i.e., the degrees of nodes can differ and there can be inputs in the graph. Also, we believe that our argument is substantially simpler conceptually.

A recent work [7] gave a complete classification of possible complexities of LCLs on *rooted regular* trees, showing that each LCL on such trees has a complexity of $O(1)$, $\Theta(\log^* n)$, $\Theta(\log n)$, or $\Theta(n^{1/k})$ for some positive integer k (and all of these complexity classes are nonempty). Moreover, the complexity of each LCL is independent of whether randomization is allowed and whether the LOCAL or the CONGEST model³ is considered. We note that their approach relies heavily on the provided orientation and, to the best of our knowledge, does not generalize to unrooted trees. We will discuss the decidability results from this work in Section 1.4.

An even more recent work [9] showed that the asymptotic complexity of any LCL problem on trees is the same in the LOCAL and the CONGEST model. This implies that the complexity landscape of LCLs on trees in CONGEST is precisely the same as in LOCAL, and in particular extends our gap between $\omega(1)$ and $o(\log^* n)$ to the CONGEST model. The authors also show that their result does not extend to general graphs by explicitly providing an LCL with different complexities in LOCAL and CONGEST.

1.2 Speedup in the VOLUME Model

Recently, Rosenbaum and Suomela[42] initiated the study of the complexity landscape of locally checkable problems in the VOLUME model on general constant degree graphs, as was done before in the LOCAL model. In the VOLUME model, a node v can adaptively probe its local neighborhood in order to determine its local output. In comparison to the LOCAL model, v does not learn its entire local neighborhood. Instead, it can only explore parts of it before computing its output. The VOLUME model is very similar (and, in the complexity regime we consider, identical, due to the work of [30]) to the well-studied LCA model [1, 44]. The basic landscape of LCLs in the VOLUME model following from [16, 42] is summarized in the bottom right part of Figure 1. Note that while their work focuses on giving examples of problems

²We say that an LCL is an *LCL without inputs* if the correctness of a solution does not depend on input labels in the graph (such as lists in a list coloring problem). In the general setting, an LCL allows the correctness to depend on input labels (though we might emphasize this by using the term *LCL with inputs*).

³The CONGEST model differs from the LOCAL model in that it only allows messages of size $O(\log n)$ bits.

of high complexities, such as $\Theta(n^{1/k})$ for $k \in \mathbb{N}$, we settle how the landscape looks for the class of symmetry breaking problems.

THEOREM 1.3 (INFORMAL VERSION OF THEOREM 4.1). *If the deterministic or randomized VOLUME complexity of an LCL is $o(\log^* n)$, it is, in fact, $O(1)$.*

We note that together with a result of [16], Theorem 1.3 implies that the only deterministic VOLUME complexities for LCLs when identifiers can be from an exponential range are $\Theta(1)$, $\Theta(\log^* n)$ and $\Theta(n)$. We find it interesting that the VOLUME complexity landscape is in this regard substantially cleaner than the LOCAL complexity landscape that contains complexities between $\Theta(\log \log^* n)$ and $\Theta(\log^* n)$, while one would, a priori, assume that the usage of a finer measure in the VOLUME model will lead to a subtler and more complicated landscape. The reason is simple and sketched next.

Our method in a nutshell. We adapt the Ramsey-theoretical argument of [41], which was improved by [21], to the VOLUME model. We observe that the heart of their argument is essentially a VOLUME model argument that shows that algorithms with $o(\log^* n)$ VOLUME complexity can be made order-invariant and, hence, they can be sped up to complexity $O(1)$. The VOLUME nature of the argument is exactly the reason why the original argument applied to the LOCAL model gives only an $o(\log \log^* n)$ speedup in general graphs and an $o(\sqrt{\log^* n})$ speedup in two-dimensional grids as this is the local complexity that implies $o(\log^* n)$ volume. The main conceptual difference between our proof and the original approach [21, 41] is that VOLUME model algorithms can perform *adaptive* probes unlike LOCAL algorithms.

1.3 Speedup in Oriented Grids

Our third result completes the LOCAL complexity landscape for LCLs on oriented grids by proving that there are no local complexities between $\omega(1)$ and $o(\log^* n)$.

In an oriented grid, all the edges are oriented in a consistent matter. Moreover, each edge is labeled with a value from $[d]$, indicating the dimension the edge corresponds to. Oriented grids may appear as a quite specific setting to consider from the perspective of distributed computing. However, in related fields where ideas from local algorithms can be applied, such as the study of factors of iid solutions [32–34, 45] or constructions in descriptive combinatorics including the famous circle squaring problem [24, 31, 35, 38, 39], oriented grids are often the graph class under consideration.

THEOREM 1.4. *[Informal version of Theorem 5.1] Let d be a fixed positive constant. Any LCL on a d -dimensional oriented grid with local complexity $o(\log^* n)$ has, in fact, local complexity $O(1)$.*

The above theorem together with the work of [17, 19, 21] implies the following characterization of possible local complexities on oriented grids.

Corollary 1.5. *Let d be a fixed positive constant. Then the deterministic/randomized complexity of an LCL problem on a d -dimensional oriented grid is one of the following:*

- (1) $O(1)$,
- (2) $\Theta(\log^* n)$,
- (3) $\Theta(\sqrt[d]{n})$.

Sometimes one considers the case of unoriented grids, that is, just the underlying graph of an oriented grid, without any additional labeling (in the study of randomized processes and descriptive combinatorics, the grid is usually oriented).

Our result unfortunately does not generalize to this case, as those graphs do not locally induce an implicit order on vertices. We conjecture, however, that the local complexity landscape remains the same.

Conjecture 1.6. *On unoriented grids, all LCL problems with complexity $o(\log^* n)$ have local complexity $O(1)$.*

1.4 Decidability

While understanding the complexity landscapes of LCL problems in different models and on different graph classes is certainly interesting in its own right, it also constitutes the first step towards a more ambitious goal—that of *deciding* for a given LCL problem how fast it can be solved, i.e., into which complexity class it falls. Unfortunately, it is known that this is not possible on general graphs: a result by Naor and Stockmeyer [41] states that it is undecidable whether a given LCL can be solved in constant time. More specifically, the work of Naor and Stockmeyer implies that, on d -dimensional oriented grids, it is undecidable whether a given LCL has complexity $O(1)$ or $\Theta(\sqrt[d]{n})$. For the case of toroidal oriented grids, i.e., grids that wrap around (in the natural sense), it is undecidable whether a given LCL has complexity $O(\log^* n)$ or is a global problem [17].

However, if we restrict attention to trees, no undecidability results are known, i.e., it is entirely possible that one day a procedure is developed that takes as input an arbitrary LCL and returns its complexity on trees. While it turned out that decidability questions for LCLs are quite complex already in very simple settings, considerable effort has gone into making progress towards this exciting goal. On the positive side, it is known that in paths and cycles the only LOCAL complexities are $O(1)$, $\Theta(\log^* n)$, and $\Theta(n)$, and it can be decided in polynomial time into which class a given LCL problem falls, provided that the LCL does not have inputs [17, 21, 22, 41]. When considering general LCLs (i.e., LCLs with inputs) on paths and cycles, it remains decidable which asymptotic complexity a given LCL has, but the question becomes PSPACE-hard [2]. The PSPACE-hardness result extends to the case of LCLs without inputs on bounded-degree trees, which was subsequently improved by Chang [20] who showed that in this setting the question becomes EXPTIME-hard.

Nevertheless, there are also good news regarding decidability on trees beyond cycles and paths. For instance, the asymptotic complexity of problems from a natural subclass of LCLs, called *binary labeling problems* can be decided efficiently [3]. Moreover, in their work providing the complexity classification of LCLs on regular rooted trees [7], the authors showed that it is decidable into which of the four complexity classes $O(1)$, $O(\log^* n)$, $\Theta(\log n)$, and $n^{\Theta(1)}$ a given LCL without inputs falls. The decidability is achieved by defining so-called “certificates” for $O(\log n)$ -round, $O(\log^* n)$ -round, and constant-round solvability—for each T from $\{O(\log n), O(\log^* n), O(1)\}$, an LCL problem is T -round solvable if and only if there exists a certificate for T -round solvability for the given LCL problem, and the existence of such a certificate is decidable. Unfortunately, the definitions of these certificates rely heavily on the fact that the input tree is rooted (in particular on the availability of a canonical direction in the tree, which is given by the parent-child relation); it is entirely unclear how such an approach could be extended to unrooted trees.

On unrooted trees, Chang and Pettie [21] showed that it is decidable whether a given LCL can be solved in logarithmic time or requires polynomial time, and Chang [20] showed that in the latter case (i.e., if the given LCL has complexity $\Theta(n^{1/k})$ for some positive integer k) the exact exponent is decidable. However, decidability on (unrooted) trees below $\Theta(\log n)$ is wide open; in fact, even the following simple question is a major open problem in the complexity theory of LCLs.

Question 1.7. Is it decidable whether a given LCL can be solved in constant time on (bounded-degree) trees?

Note that constant-time-solvability of LCLs on trees is semidecidable as for each constant c and fixed LCL Π , there are only constantly many different candidate c -round LOCAL algorithms for solving Π —the difficult direction is to prove (or disprove) semidecidability for the impossibility of constant-time-solvability. While our proof of Theorem 1.1 does not settle Question 1.7, it may be a first step towards a resolution as it reduces the problem to proving (or disproving) semidecidability of an $\Omega(\log^* n)$ -round lower bound for the given LCL. Note that this avoids consideration of all the “messy” complexities shown to exist between $\Theta(\log \log^* n)$ and $\Theta(\log^* n)$ in general (bounded-degree) graphs (such as $2^{\Theta(\log^\alpha \log^* n)}$ for any positive rational number $\alpha \leq 1$) [10].

1.5 Organization of the Paper

In Section 2 we formally define the settings that we work with, and prove basic technical results. In Section 3 we prove the speedup theorem for trees in the LOCAL model, i.e., Theorem 1.1. In Section 4 we prove the speedup theorem in the VOLUME model, i.e., Theorem 1.3. In Section 5 we prove the speedup theorem for oriented grids in the LOCAL model, i.e., Theorem 1.4.

2 PRELIMINARIES

We use classical graph-theoretical notation, e.g. we write $G = (V, E)$ for an unoriented graph. A *half-edge* is a pair $h = (v, e)$, where $v \in V$, and $e \in E$ is an edge incident to v . We denote the set of half-edges of G by $H = H(G)$, i.e., $H = \{(v, e) \mid v \in e, v \in V, e \in E\}$. Furthermore, for every vertex v' , we denote the set of half-edges $(v, e) \in H$ where $v = v'$ by $H[v']$, and for every edge e' , we denote the set of half-edges $(v, e) \in H$ where $e = e'$ by $H[e']$. Often we assume that G additionally carries a labeling of vertices or half-edges. We use $B_G(u, r)$ to denote the ball of radius r around a node u in G and we call it the *r-hop neighborhood* of u . When talking about half-edges in $B_G(u, r)$, we mean all half-edges (v, e) such that $v \in B_G(u, r)$. For example, $B_G(u, 0)$ contains all half-edges incident to u .

The reader should keep in mind that our setting is graphs of maximum degree bounded by some constant Δ . This is sometimes explicitly stated (or it is implied by the constraints) but most of the time it is tacitly assumed. Of special interest to us will be the class of all trees with maximum degree at most Δ , which we denote by \mathcal{T} . Similarly, we denote the class of all forests with maximum degree at most Δ by \mathcal{F} . Moreover, for any positive integer n , any set N of positive integers, and any $\mathcal{G} \in \{\mathcal{F}, \mathcal{T}\}$, we will use \mathcal{G}_n , resp. \mathcal{G}_N , to denote the class of members of \mathcal{G} with n nodes, resp. with a number of nodes that is contained in N .

2.1 LOCAL Model and LCL Problems

In this section, we define our main model of computation and discuss the problem class considered in this work. Our main model of computation is the LOCAL model [37]. Although it is often convenient to think about LOCAL algorithms as message-passing procedures, it will be simpler to work with the following equivalent definition.

Definition 2.1 (LOCAL model). The input to a problem in the LOCAL model is an n -node graph G , for some positive integer n . Each node of the graph is considered as a computational entity and equipped with a globally unique identifier, i.e., positive integer from a polynomial range (in the case of deterministic algorithms), or with a private random bit string (in the case of randomized algorithms). Additionally, depending on the problem considered, other inputs might be stored at the nodes or half-edges. In a T -round algorithm each node v is aware of the number n of nodes of the input

graph⁴ and of its T -hop neighborhood, i.e., of all nodes in distance at most T from v , all edges that have at least one endpoint in distance at most $T - 1$ from v , and all half-edges whose endpoint is in distance at most T from v (as well as all inputs stored therein). Based on this information, v has to decide on its output whose specification is given by the considered problem. In other words, a T -round algorithm is simply a function (parameterized by n) from the space of all possible (labeled) T -hop neighborhoods of a node to the space of outputs. Which neighborhoods are “possible” is determined by the considered problem and the considered graph class.

For technical reasons, we will also assume that each graph comes with a *port numbering*, i.e., each node v has $\deg(v)$ ports $1, \dots, \deg(v)$, and each edge incident to v is connected to v via a unique one of those ports. In other words, the ports at a node provide a total order on the set of incident edges. It is straightforward to verify that the addition of a port numbering does not change the computational power of the model (asymptotically) as each node can infer a total order on the set of incident edges from the unique identifiers of its neighbors (in the deterministic case) or from the random bits of its neighbors, with arbitrarily large success probability (in the randomized case).

The class of problems considered in this work are *LCL problems* (or *LCLs*, for short), which were introduced by Naor and Stockmeyer [41]. In their seminal paper, Naor and Stockmeyer provided a definition for LCL problems where input and output labels were assigned to nodes, and remarked that a similar definition can be given for edge-labeling problems. A modern definition that captures both kinds of LCL problems (and their combinations) assigns labels to *half-edges* (instead of vertices or edges). Before we can provide this definition, we need to define some required notions.

A *half-edge labeling* of a graph G (with labels from a set Σ) is a function $f: H(G) \rightarrow \Sigma$. A Σ_{in} - Σ_{out} -labeled graph is a triple $(G, f_{\text{in}}, f_{\text{out}})$ consisting of a graph G and two half-edge labelings $f_{\text{in}}: H(G) \rightarrow \Sigma_{\text{in}}$ and $f_{\text{out}}: H(G) \rightarrow \Sigma_{\text{out}}$ of G . We analogously define a Σ_{in} -labeled graph by omitting f_{out} .

We can now define an LCL problem as follows.

Definition 2.2 (LCL problem). An LCL problem Π is a quadruple $(\Sigma_{\text{in}}, \Sigma_{\text{out}}, r, \mathcal{P})$ where Σ_{in} and Σ_{out} are finite sets, r is a positive integer, and \mathcal{P} is a finite collection of Σ_{in} - Σ_{out} -labeled graphs. A correct solution for an LCL problem Π on a Σ_{in} -labeled graph (G, f_{in}) is given by a half-edge labeling $f_{\text{out}}: H(G) \rightarrow \Sigma_{\text{out}}$ such that, for every node $v \in V(G)$, the triple $(B_G(v, r), f'_{\text{in}}, f'_{\text{out}})$ is isomorphic to a member of \mathcal{P} , where f'_{in} and f'_{out} are the restriction of f_{in} and f_{out} , respectively, to $B_G(v, r)$.

Intuitively, the collection \mathcal{P} provides the constraints of the problem by specifying how a correct output looks *locally*, depending on the respective local input. From the definition of a correct solution for an LCL problem it follows that members of \mathcal{P} that have radius $> r$ can be ignored. Also the finiteness of \mathcal{P} automatically implies that we are restricting ourselves to graphs of degree at most Δ for some constant Δ .

The main tool in our proof of Theorem 1.1, the round elimination technique, applies (directly) only to a subclass of LCL problems: roughly speaking, it is required that the local correctness constraints specified by \mathcal{P} can be translated into node and edge constraints, i.e., correctness constraints that can be verified by looking at the label configurations on each edge and around each node. The definition of this subclass of LCL problems is given in the following. Note that, despite the complicated appearance, the definition is actually quite intuitive: essentially, in order to define the LCL problem, we simply specify a set of label configurations that are allowed on an edge, a set of label configurations that

⁴Our results work equally well in the setting where only some upper bound n' on n is given to the nodes; in that case, as usual, the complexities are functions of n' instead of of n . We remark that our setting where the nodes are aware of the exact value of n is in fact the more difficult one (for our approach) as evidenced by the issue discussed in Section 3.1.

are allowed around a node, and an input-output label relation that specifies for each input label which output label is allowed at the same half-edge.

Definition 2.3 (Node-edge-checkable LCL problem). A *node-edge-checkable LCL* Π is a quintuple $(\Sigma_{\text{in}}^\Pi, \Sigma_{\text{out}}^\Pi, \mathcal{N}_\Pi, \mathcal{E}_\Pi, g_\Pi)$, where Σ_{in}^Π and Σ_{out}^Π are finite sets, \mathcal{E}_Π is a collection of cardinality-2 multisets $\{B_1, B_2\}$ with $B_1, B_2 \in \Sigma_{\text{out}}^\Pi$, $\mathcal{N}_\Pi = (\mathcal{N}_\Pi^1, \mathcal{N}_\Pi^2, \dots)$ consists of collections \mathcal{N}_Π^i of cardinality- i multisets $\{A_1, \dots, A_i\}$ with $A_1, \dots, A_i \in \Sigma_{\text{out}}^\Pi$, and $g_\Pi: \Sigma_{\text{in}}^\Pi \rightarrow 2^{\Sigma_{\text{out}}^\Pi}$ is a function that assigns to each label from Σ_{in}^Π a subset of the labels of Σ_{out}^Π . A correct solution for a node-edge-checkable LCL Π on a Σ_{in}^Π -labeled graph (G, f_{in}) is given by a half-edge labeling $f_{\text{out}}: H(G) \rightarrow \Sigma_{\text{out}}^\Pi$ such that

- (1) for every node $v \in V(G)$, the multiset consisting of the labels assigned by f_{out} to the half-edges in $H[v]$ is contained in $\mathcal{N}_\Pi^{\deg(v)}$,
- (2) for every edge $e \in E(G)$, the multiset consisting of the labels assigned by f_{out} to the half-edges in $H[e]$ is contained in \mathcal{E}_Π , and
- (3) for every half-edge $h \in H(G)$, the label $f_{\text{out}}(h)$ is contained in the label set $g_\Pi(f_{\text{in}}(h))$.

We call \mathcal{N}_Π the *node constraint* of Π and \mathcal{E}_Π the *edge constraint* of Π . Moreover, we call the elements $\{A_1, \dots, A_i\}$ of \mathcal{N}_Π^i *node configurations* and the elements $\{B_1, B_2\}$ of \mathcal{E}_Π *edge configurations* (of Π). In a LOCAL algorithm solving a node-edge-checkable problem Π , each node is supposed to output a label for each incident half-edge such that the induced global half-edge labeling is a correct solution for Π .

Even though the round elimination technique can be applied directly only to node-edge-checkable LCL problems, the results we obtain apply to all LCL problems. The reason for this is that for each LCL problem Π there exists a node-edge-checkable LCL problem Π' such that the time complexities of Π and Π' differ only by an additive constant, as we show in Lemma 2.6. This fact suffices to lift our results for node-edge-checkable LCL problems to general LCL problems: in particular, the existence of an LCL problem with time complexity in $\omega(1)$ and $o(\log^* n)$ would imply the existence of a node-edge-checkable LCL problem with the same complexity constraints, leading to a contradiction.

Before stating and proving Lemma 2.6, we formally define the local failure probability of an algorithm solving a node-edge-checkable LCL, and the complexity of an LCL problem.

Definition 2.4 (Local failure probability). Let $\Pi = (\Sigma_{\text{in}}^\Pi, \Sigma_{\text{out}}^\Pi, \mathcal{N}_\Pi, \mathcal{E}_\Pi, g_\Pi)$ be some node-edge-checkable LCL problem. We say that a half-edge labeling $f_{\text{out}}: H(G) \rightarrow \Sigma_{\text{out}}^\Pi$ is *incorrect on some edge* $e = \{u, v\}$ of graph (G, f_{in}) if

- (1) $\{f_{\text{out}}((u, e)), f_{\text{out}}((v, e))\} \notin \mathcal{E}_\Pi$, or
- (2) $f_{\text{out}}((u, e)) \notin g_\Pi(f_{\text{in}}((u, e)))$ or $f_{\text{out}}((v, e)) \notin g_\Pi(f_{\text{in}}((v, e)))$.

Similarly, we say that f_{out} is *incorrect at some node* v if

- (1) $\{f_{\text{out}}(h)\}_{h \in H[v]} \notin \mathcal{N}_\Pi^{\deg(v)}$, or
- (2) $f_{\text{out}}(h) \notin g_\Pi(f_{\text{in}}(h))$ for some $h \in H[v]$.

We say that an algorithm \mathcal{A} *fails on some edge* e , *resp. at some node* v , if the output produced by \mathcal{A} is incorrect on e , *resp.* at v . Furthermore, we say that a (randomized) algorithm \mathcal{A} has *local failure probability* p on some graph G if p is the smallest (real) number such that, for each edge e and node v in G , the probability that \mathcal{A} fails on e , *resp.* at v , is upper bounded by p . Moreover, for each n , the local failure probability of \mathcal{A} on some class of n -node graphs is the maximum of the local failure probabilities of \mathcal{A} on the graphs in the class. (In contrast, the definition of *global failure probability* is as commonly used, i.e., we say that \mathcal{A} has (global) failure probability $p = p(n)$ if the (worst-case) probability that \mathcal{A} does not produce a correct solution for Π is upper bounded by p and p is minimal under this constraint.)

The LOCAL complexity of a (node-edge-checkable or common) LCL is simply the minimum complexity of an algorithm \mathcal{A} that solves it on all graphs:

Definition 2.5 (Complexity of an LCL problem). The *deterministic (round) complexity* of an LCL Π is the function $T: \mathbb{N} \rightarrow \mathbb{N} \cup \{0\}$ satisfying that for each $n \in \mathbb{N}$, there exists a deterministic algorithm \mathcal{A}_n solving Π in $T(n)$ rounds on all n -node graphs G with each half-edge labeled with a label from Σ_{in} , but no deterministic algorithm solving Π in $T(n) - 1$ rounds on this class of graphs. The *randomized (round) complexity* of an LCL Π is defined analogously, where deterministic algorithms are replaced by randomized algorithms with a (global) failure probability of at most $1/n$.

When we talk about the complexity of an LCL on trees, we further restrict the above definition to graphs that are trees (and similarly for other graph classes).

Now we are ready to state and prove the following lemma, which ensures that we can restrict attention to node-edge-checkable LCLs.

Lemma 2.6. *For any LCL problem Π , there exists a node-edge-checkable LCL problem Π' such that (in both the randomized and deterministic LOCAL model) the complexities of Π and Π' on trees (and on forests) are asymptotically the same.*

PROOF. Suppose $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, r, \mathcal{P})$ is an LCL. We create a node-edge-checkable LCL $\Pi' = (\Sigma_{\text{in}}^{\Pi'}, \Sigma_{\text{out}}^{\Pi'}, \mathcal{N}_{\Pi'}, \mathcal{E}_{\Pi'}, g_{\Pi'})$ as follows:

- $\Sigma_{\text{in}}^{\Pi'} = \Sigma_{\text{in}}$.
- $\Sigma_{\text{out}}^{\Pi'}$ contains all possible labelings of r -hop neighborhoods of a node, each neighborhood has marked a special half-edge, each vertex and each edge has an order on incident half-edges and each half-edge is labeled with a label from Σ_{out} , moreover, the labeling by Σ_{out} has to be accepted by \mathcal{P} .
- $\mathcal{N}_{\Pi'}$ contains such sets $S = \{\sigma_1, \dots, \sigma_d\}$ with $\sigma_i \in \Sigma_{\text{out}}^{\Pi'}$ such that there exists an r -hop neighborhood N of a node u of degree d together with each node and each edge having an order on incident half edges and such that half-edges have labels from Σ_{out} such that we can assign labels from S to half-edges around u in such a way that each σ_i assigned to (u, e_i) describes N with the special half-edge of σ_i being e_i .
- $\mathcal{E}_{\Pi'}$ is defined analogously to $\mathcal{N}_{\Pi'}$, but we require an existence of a neighborhood of an edge e that is consistent from the perspective of labels from $\Sigma_{\text{out}}^{\Pi'}$ assigned to (u, e) and (v, e) .
- $g_{\Pi'}$ maps each label $\tau \in \Sigma_{\text{in}}$ to the set of labels $\sigma \in \Sigma_{\text{out}}^{\Pi'}$ such that the special half-edge of σ is labeled by τ .

Suppose we have a valid solution of Π . Then, in r rounds each half-edge can decide on its Π' -label by encoding its r -hop neighborhood, including port numbers of each vertex and each edge that give ordering on its half-edges, into a label from $\Sigma_{\text{out}}^{\Pi'}$. The constraints $\mathcal{N}_{\Pi'}$, $\mathcal{E}_{\Pi'}$, $g_{\Pi'}$ will be satisfied.

On the other hand, suppose we have a valid solution for Π' . In 0-rounds, each half-edge (v, e) can label itself with the label on the special half-edge in its Π' -label $f'_{\text{out}}((v, e))$. We claim that the Π -labeling we get this way around a half-edge (u, e) is isomorphic to the Σ_{in} -labeling described by the label $f_{\text{out}}((v, e))$, hence the new labeling is a solution to Π' . To see this, consider running a BFS from (v, e) . The node constraints $\mathcal{N}_{\Pi'}$ and the edge constraints $\mathcal{E}_{\Pi'}$ are ensuring that the labels from $\Sigma_{\text{out}}^{\Pi'}$ of visited half-edges are describing compatible neighborhoods, while the function g ensures that the description of Σ_{in} labels in the r -hop neighborhood of u by the labels from $\Sigma_{\text{out}}^{\Pi'}$ agrees with the actual Σ_{in} labeling of the r -hop neighborhood of u . As the label $f'_{\text{out}}((v, e))$ needs to be accepted by \mathcal{P} , we get that \mathcal{P} accepts the r -hop neighborhood of u , as needed. \square

It is crucial that the way in which we define the node-edge-checkable LCL problem Π' in Lemma 2.6 guarantees that the considered (input-labeled) graph class remains the same as for Π (and does not turn into a graph class with a

promise on the distribution of the input labels, which would be the result of the straightforward approach of defining Π' by encoding the input labels contained in a constant-sized ball in Π in a single input label in Π' , and doing the same for output labels). If this property was not guaranteed, it would be completely unclear (and perhaps impossible) how to extend the round elimination framework of [14] to our setting with input labels.

2.2 Order-Invariant Algorithms

In this section, we formally define the notion of an order-invariant algorithm and introduce further computational models of interest. We also show that oftentimes order-invariant algorithms can be sped up to improve the round/probe complexity, both in the LOCAL and the VOLUME model.

Definition 2.7 (Order-invariant LOCAL algorithm [41]). A deterministic $T(n)$ -round LOCAL algorithm \mathcal{A} is called order-invariant if the following holds: Consider two assignments of distinct identifiers to nodes in $B_G(v, T(n))$ denoted by μ and μ' . Assume that for all $u, w \in B_G(v, T(n))$ it holds that $\mu(u) > \mu(w)$ if and only if $\mu'(u) > \mu'(w)$. Then, the output of \mathcal{A} on the set of half-edges $H[v]$ will be the same in both cases.

Next, we define the notions necessary for Section 4. We start by defining the VOLUME model [42]. We define the VOLUME model in a more mathematically rigorous way compared to [42], as this will help us later with the proofs. Before defining the VOLUME model, we start with one more definition.

Definition 2.8. For an arbitrary $S \subseteq \mathbb{N}$, we define

$$Tuples_S = \{(id, deg, in) : id \in S, deg \in [\Delta], in : [deg] \mapsto \Sigma_{in}\}$$

and for $i \in \mathbb{N}$, we define

$$Tuples_{i,S} = \{(t_1, t_2, \dots, t_i) : \text{for every } j \in [i], t_j \in Tuples_S\}.$$

For a given $i \in \mathbb{N}$ and $\ell \in [2]$, let $t^{(\ell)} = ((id_1^\ell, deg_1, in_1), (id_2^\ell, deg_2, in_2), \dots, (id_i^\ell, deg_i, in_i)) \in Tuples_{i,\mathbb{N}}$ be arbitrary. We say that the tuples $t^{(1)}$ and $t^{(2)}$ are almost identical if for every $j_1, j_2 \in [i]$, $id_{j_1}^1 < id_{j_2}^2$ implies $id_{j_1}^2 < id_{j_2}^2$, $id_{j_1}^1 > id_{j_2}^1$ implies $id_{j_1}^2 > id_{j_2}^2$ and $id_{j_1}^1 = id_{j_2}^1$ implies $id_{j_1}^2 = id_{j_2}^2$.

A tuple in $Tuples_S$ can encode the local information of a node v , including its ID, its degree and the input assigned to each of its incident half-edges, i.e., $in(k)$ is the input assigned to the k -th half edge. We denote with t_v the tuple that encodes the local information of v . A tuple in $Tuples_{i,S}$ can be used to encode all the information (modulo the number of nodes of the input graph) that a node knows about the input graph after having performed $i - 1$ probes. The notion of almost identical tuples will be helpful for defining the notion of order-invariance for the VOLUME model.

Definition 2.9 (VOLUME model). Let $\Pi = (\Sigma_{in}, \Sigma_{out}, r, \mathcal{P})$ be an LCL problem. A deterministic VOLUME model algorithm \mathcal{A} for Π with a probe complexity of $T(n)$ can be seen as a set of computable functions $f_{n,i}$ for every $n \in \mathbb{N}$ and $i \in [T(n) + 1]$ with $f_{n,i} : Tuples_{i,\mathbb{N}} \mapsto [i] \times [\Delta]$ for $i \in [T(n)]$ encoding the i -th adaptive probe and $f_{n,T(n)+1} : Tuples_{T(n)+1,\mathbb{N}} \mapsto \Sigma_{out}^{[\Delta]}$, where $\Sigma_{out}^{[\Delta]}$ refers to the set of functions mapping each value in $[\Delta]$ to a value in Σ_{out} , encoding the output that \mathcal{A} assigns to the incident half edges of the queried node. Next, we have to define what it means for \mathcal{A} to be a valid algorithm. To that end, let (G, f_{in}) be an arbitrary Σ_{in} -labeled graph on n nodes with each node in G being equipped with an identifier and a port assignment (To simplify the technical definition, we assume that G does not contain any isolated node).

The algorithm \mathcal{A} defines a half-edge labeling $f_{out, \mathcal{A}, (G, f_{in})} : H(G) \mapsto \Sigma_{out}$ as follows: Let $(v, e) \in H(G)$ be an arbitrary half-edge of G . We define $t^{(0)} = (t_0)$ and for $i \in \{1, 2, \dots, T(n)\}$, we obtain $t^{(i)}$ from $t^{(i-1)} = (t_{v_0}, t_{v_1}, \dots, t_{v_i})$ as follows. Let $(j, p) = f_{n,i}(t^{(i-1)})$ and v_{i+1} the node in G such that $\{v_j, v_{i+1}\}$ is the p -th edge incident to v_j (we assume that the degree of v_j is at least p). Then, $t^{(i)} = (t_{v_0}, t_{v_1}, \dots, t_{v_{i+1}})$. Finally, $f_{n, T(n)+1}(t_{v_0}, t_{v_1}, \dots, t_{v_{T(n)}})$ defines a function g mapping each value in $[\Delta]$ to a value in Σ_{out} . Let e be the p -th edge incident to v . We define $f_{out, \mathcal{A}, (G, f_{in})}((v, e)) = g(p)$.

We say that \mathcal{A} solves Π if $f_{out, \mathcal{A}, (G, f_{in})}$ is a valid output labeling for every Σ_{in} -labeled graph (G, f_{in}) with each node in G having a unique ID from a polynomial range, is equipped with a port labeling, and G does not contain an isolated node.

The notion of an order-invariant algorithm naturally extends to algorithms in the VOLUME model.

Definition 2.10. (Order-invariant VOLUME algorithm) We say that \mathcal{A} is order invariant if for every $n \in \mathbb{N}$ and $i \in [T(n) + 1]$, $f_{n,i}(t) = f_{n,i}(t')$ for every $t, t' \in \text{Tuples}_{i,S}$ with t and t' being almost identical.

We will use the following basic speed-up result for order-invariant algorithms in both the LOCAL and the VOLUME model.

THEOREM 2.11 (SPEED-UP OF ORDER-INVARIANT ALGORITHMS (cf. [19])). *Let \mathcal{A} be an order-invariant algorithm solving a problem Π in $f(n) = o(\log n)$ rounds of the LOCAL model or with $f(n) = o(n)$ probes of the VOLUME model. Then, there is an order-invariant algorithm solving Π in $O(1)$ rounds of the LOCAL model or, equivalently, $O(1)$ probes in the VOLUME model.*

PROOF. The result for the LOCAL model is proven in [19]. The proof for the VOLUME model follows in the exact same manner. We provide it here for completeness. Let n_0 be a fixed constant such that $\Delta^{r+1} \cdot (T(n_0) + 1) \leq n_0/\Delta$. We now define a VOLUME algorithm \mathcal{A}' with probe complexity $T'(n) = \min(n, T(n_0)) = O(1)$ as follows. For $n \in \mathbb{N}$ and $i \in [T'(n) + 1]$, we define $f_{n,i}^{\mathcal{A}'} = f_{\min(n, n_0), i}^{\mathcal{A}}$. It remains to show that \mathcal{A}' indeed solves Π . For the sake of contradiction, assume that this is not the case. This implies the existence of a Σ_{in} -labeled graph (G, f_{in}) on $n \geq n_0$ nodes (with IDs from a polynomial range, port assignments, and no isolated nodes) such that \mathcal{A}' "fails" on (G, f_{in}) . Put differently, there exists a node v such that \mathcal{A}' produces a mistake in the r -hop neighborhood of v . The r -hop neighborhood of v consists of at most Δ^{r+1} vertices. To answer a given query, \mathcal{A}' "sees" at most $T(n_0) + 1$ nodes. Hence, to compute the output of all the nodes in the r -hop neighborhood of v , \mathcal{A}' "sees" at most $\Delta^{r+1}(T(n_0) + 1) \leq \frac{n_0}{\Delta}$ many nodes. We denote the set consisting of those nodes as $V^{visible}$. Now, let (G', f'_{in}) be a Σ_{in} -labeled graph on n' nodes (with IDs from a polynomial range, port assignments, and no isolated nodes) such that every $u \in V^{visible}$ is also contained in G' , with its degree being the same in both graphs, as well as the input assigned to each of its incident half edges. The assigned ID can be different, however, the relative orders of the IDs assigned to nodes in $V^{visible}$ in G and G' are the same. As $\Delta^{r+1}(T(n_0) + 1) \leq \frac{n_0}{\Delta}$, such a (G', f'_{in}) exists. As \mathcal{A} is order invariant, so is \mathcal{A}' . Moreover, $f_{n,i}^{\mathcal{A}'} = f_{n',i}^{\mathcal{A}'}$ for any i . Hence, it follows that \mathcal{A}' assigns the same output to all the half-edges in the r -hop neighborhood of v in G and G' . Therefore, \mathcal{A}' also fails on the graph G' . From the way we defined \mathcal{A}' , this directly implies that \mathcal{A} also fails on G , a contradiction with the assumption that \mathcal{A} is a correct algorithm. This finishes the proof. \square

LCA model. We now briefly discuss the related LCA model. A deterministic local computation algorithm (LCA) is similar to a deterministic VOLUME algorithm, with two small differences. First, an LCA can perform so-called far probes and second, it can assume that each node in the n -node input graph has a unique ID in the set $\{1, 2, \dots, n\}$. However, far probes are not of any help in the complexity regime we consider.

THEOREM 2.12 (CF. [30], THEOREM 1). *Any LCL problem that can be solved by an LCA with probe complexity $t(n)$ can also be solved by an LCA with probe complexity $O(t(n^{\log(n)}))$ that does not perform any far probes, provided $t(n) = o(\sqrt{\log(n)})$.*

In particular, an LCA with probe complexity $o(\log^* n)$ implies an LCA for the same problem that does not perform any far probes and has a probe complexity of $o(\log^* n)$. Hence, we can focus on showing that any LCA that does not perform any far probes with a probe complexity of $o(\log^* n)$ implies an LCA for the same problem with a probe complexity of $O(1)$. However, an LCA that does not perform any far probes is the same as a VOLUME algorithm, with the only difference being that the LCA algorithm only has to produce a valid output if each node in the n -node input graph has a unique ID from the set $\{1, 2, \dots, n\}$. However, one can show with a very simple argument that a VOLUME algorithm (LCA without far probes) with probe complexity $T(n) = o(\log^* n)$ that assumes unique IDs from the set $\{1, 2, \dots, n\}$ implies a volume algorithm with probe complexity $T'(n) = T(n^k) = o(\log^* n)$ that only assumes IDs from the set $\{1, 2, \dots, n^k\}$ for an arbitrary constant k . Hence, from the discussion above, a speed-up result from $o(\log^* n)$ to $O(1)$ in the VOLUME model directly implies the same speed-up in the LCA model.

3 THE LOCAL MODEL GAP ON TREES

In this section we prove the $\omega(1) - o(\log^* n)$ gap for LCLs on trees in the LOCAL model. We do so by proving Theorem 3.11 (which is the slightly more formal version of Theorem 1.1) by explicitly designing, for any given (node-edge-checkable) LCL problem Π with complexity $o(\log^* n)$, a constant-round algorithm. As explained in Section 1.1, a very rough outline of our approach is to generate from Π a sequence of node-edge-checkable LCL problems of decreasing randomized local complexities (where we allow the local failure probability to grow along the problems in the sequence), find a problem in the sequence that can be solved in 0 rounds with a reasonably low local failure probability, show that there exists a 0-round deterministic algorithm for that problem, and turn this algorithm into a constant-round algorithm for Π by going back up the sequence of problems and arguing that the deterministic complexities increase slowly along the sequence in this direction. While the round elimination framework [4, 14] provides a blueprint how to generate a suitable sequence, it unfortunately only does so for LCLs on regular trees without inputs. We provide an extension of the framework that also works for LCLs on irregular trees (or forests) with inputs.

We will start in Section 3.1 by extending the definition of the round elimination problem sequence to the setting with inputs (and taking care of a technical issue). In Section 3.2, we will carefully bound the evolution of failure probabilities along a sequence of algorithms with decreasing runtimes that solve the problems in the defined problem sequence. Section 3.3 takes care of the reverse step, i.e., showing that the deterministic complexities of the problems in the problem sequence do not increase fast when traversed towards Π . Finally, in Section 3.4, we will put everything together and prove Theorem 3.11.

3.1 The Problem Sequence

Similarly to the approach in [14], we define, for any node-edge-checkable LCL problem Π , two node-edge-checkable problems $\mathcal{R}(\Pi)$ and $\bar{\mathcal{R}}(\Pi)$. The problems in the aforementioned sequence are then obtained by iteratively applying $\bar{\mathcal{R}}(\mathcal{R}(\cdot))$, starting with Π .

Definition 3.1 ($\mathcal{R}(\Pi)$). Let $\Pi = (\Sigma_{\text{in}}^\Pi, \Sigma_{\text{out}}^\Pi, \mathcal{N}_\Pi, \mathcal{E}_\Pi, g_\Pi)$ be a node-edge-checkable LCL problem. We define a new node-edge-checkable LCL problem $\mathcal{R}(\Pi) = (\Sigma_{\text{in}}^{\mathcal{R}(\Pi)}, \Sigma_{\text{out}}^{\mathcal{R}(\Pi)}, \mathcal{N}_{\mathcal{R}(\Pi)}, \mathcal{E}_{\mathcal{R}(\Pi)}, g_{\mathcal{R}(\Pi)})$ by specifying the five components. We start by setting $\Sigma_{\text{in}}^{\mathcal{R}(\Pi)} := \Sigma_{\text{in}}^\Pi$ and $\Sigma_{\text{out}}^{\mathcal{R}(\Pi)} := 2^{\Sigma_{\text{out}}^\Pi}$, i.e., the input label set of $\mathcal{R}(\Pi)$ is simply the input label set of Π , and the output label set of $\mathcal{R}(\Pi)$ is the power set of the output label set of Π . Next, we define $g_{\mathcal{R}(\Pi)}$ by setting

$g_{\mathcal{R}(\Pi)}(\ell) := 2^{g_{\Pi}(\ell)}$ for any label $\ell \in \Sigma_{\text{in}}^{\Pi}$, i.e., intuitively speaking, in problem $\mathcal{R}(\Pi)$ an input label ℓ on some half-edge requires that the output label on the same half-edge is a subset of the set of output labels that were allowed in Π on a half-edge with input label ℓ .

We define the edge constraint $\mathcal{E}_{\mathcal{R}(\Pi)}$ of $\mathcal{R}(\Pi)$ as the set of all cardinality-2 multisets $\{B_1, B_2\}$ such that $B_1, B_2 \in \Sigma_{\text{out}}^{\mathcal{R}(\Pi)}$ and, for all $b_1 \in B_1, b_2 \in B_2$, we have $\{b_1, b_2\} \in \mathcal{E}_{\Pi}$. Finally, we define the node constraint $\mathcal{N}_{\mathcal{R}(\Pi)}$ of $\mathcal{R}(\Pi)$ as follows. For each integer $i \geq 1$, define $\mathcal{N}_{\mathcal{R}(\Pi)}^i$ as the set of all cardinality- i multisets $\{A_1, \dots, A_i\}$ such that $A_1, \dots, A_i \in \Sigma_{\text{out}}^{\mathcal{R}(\Pi)}$ and there exists some selection (a_1, \dots, a_i) of labels from $A_1 \times \dots \times A_i$ such that $\{a_1, \dots, a_i\} \in \mathcal{N}_{\Pi}^i$.

Note that our definition of $\mathcal{R}(\Pi)$ differs slightly from the usual definition of $\mathcal{R}(\Pi)$ as given in, e.g., [6] (beyond the obvious differences due to the fact that we consider LCL problems with input labels): our definition does not remove so-called “non-maximal” configurations. Removing such configurations can be beneficial when trying to determine the complexity of specific problems, but is not required (or helpful) in our setting, where we want to argue about the complexity of many problems at once.

Definition 3.2 ($\overline{\mathcal{R}}(\Pi)$). The problem $\overline{\mathcal{R}}(\Pi)$ differs from $\mathcal{R}(\Pi)$ only in the node and edge constraints; for the remaining three parameters we set $\Sigma_{\text{in}}^{\overline{\mathcal{R}}(\Pi)} := \Sigma_{\text{in}}^{\mathcal{R}(\Pi)}$, $\Sigma_{\text{out}}^{\overline{\mathcal{R}}(\Pi)} := \Sigma_{\text{out}}^{\mathcal{R}(\Pi)}$, and $g_{\overline{\mathcal{R}}(\Pi)} := g_{\mathcal{R}(\Pi)}$.

We define the node constraint $\mathcal{N}_{\overline{\mathcal{R}}(\Pi)}$ of $\overline{\mathcal{R}}(\Pi)$ as follows. For each integer $i \geq 1$, define $\mathcal{N}_{\overline{\mathcal{R}}(\Pi)}^i$ as the set of all cardinality- i multisets $\{A_1, \dots, A_i\}$ such that $A_1, \dots, A_i \in \Sigma_{\text{out}}^{\overline{\mathcal{R}}(\Pi)}$ and, for all $(a_1, \dots, a_i) \in A_1 \times \dots \times A_i$, we have $\{a_1, \dots, a_i\} \in \mathcal{N}_{\Pi}^i$. Moreover, we define the edge constraint $\mathcal{E}_{\overline{\mathcal{R}}(\Pi)}$ of $\overline{\mathcal{R}}(\Pi)$ as the set of all cardinality-2 multisets $\{B_1, B_2\}$ such that $B_1, B_2 \in \Sigma_{\text{out}}^{\overline{\mathcal{R}}(\Pi)}$ and there exists some selection (b_1, b_2) of labels from $B_1 \times B_2$ such that $\{b_1, b_2\} \in \mathcal{E}_{\Pi}$.

Note that, although the function $\overline{\mathcal{R}}(\cdot)$ can take any arbitrary node-edge-checkable LCL problem as argument, we will use as arguments only problems that are of the form $\mathcal{R}(\Pi)$ for some node-edge-checkable LCL problem Π .

Recall that \mathcal{T} , resp. \mathcal{F} , denotes the class of all trees, resp. forests, of maximum degree at most Δ . Before turning to the analysis of the evolution of the aforementioned local failure probabilities, there is a technical issue we have to discuss. A crucial argument in said analysis is, roughly speaking, that if you consider some (sufficiently small) neighborhood of a node (or edge), and a set of extensions of this neighborhood via different edges leaving the neighborhood⁵, then there must also be a graph in the considered graph class that (simultaneously) contains all extensions of the set (together with the neighborhood). Here the term “considered graph class” describes the input graph class restricted to the members that are consistent with the knowledge of the nodes about the number n of nodes, i.e., in particular if all nodes are aware of the exact value of n (as they are in our definition of the LOCAL model), then the considered graph class contains only n -node graphs. Now, if the input graph class is \mathcal{T} , it follows that the aforementioned crucial argument does not hold in general: if all extensions in the considered set “conclude” the tree (i.e., do not have leaving edges except those connecting them to the considered initial neighborhood) and the combined number of nodes in the initial neighborhood and the considered extensions does not happen to be precisely n , then there is no n -node tree that contains the neighborhood together with all considered extensions.

We solve this issue by proving our main theorem first for the class \mathcal{F} of forests (which do not have the aforementioned issue as the number of nodes⁶ in some maximal connected component is not known to the nodes) in Theorem 3.10 and

⁵An extension via some leaving edge is simply a possibility of how the graph could continue for the next hop beyond the respectively chosen leaving edge that is consistent with (some graph in) the considered graph class.

⁶We remark that the issue does not occur in the variant of the LOCAL model in which nodes are only aware of some upper bound on the number of nodes, but we believe that it is important to ensure that the correctness of (our) results does not depend on such minor details in the model specification.

then lifting it to \mathcal{T} in Theorem 3.11 by showing that, for node-edge-checkable LCL problems, a complexity of $o(\log^* n)$ on trees implies a complexity of $o(\log^* n)$ on forests. Lemma 3.3 provides this relation between trees and forests; in Sections 3.2 and 3.3 we will then exclusively work with forests. Note that the complexity of any LCL problem Π on \mathcal{F} is trivially at least as large as its complexity on \mathcal{T} as \mathcal{T} is a subclass of \mathcal{F} .

Lemma 3.3. *Let Π be some node-edge-checkable LCL problem that has deterministic, resp. randomized, complexity $o(\log^* n)$ on \mathcal{T} . Then the deterministic, resp. randomized, complexity of Π on \mathcal{F} is in $o(\log^* n)$.*

PROOF. Let \mathcal{A} be a (deterministic or randomized) algorithm solving Π on \mathcal{T} in $T(n) \in o(\log^* n)$ rounds (and observe that the existence of \mathcal{A} (even if it is randomized) implies that a correct global solution exists). We design a new algorithm \mathcal{A}' solving Π on \mathcal{F} in $o(\log^* n)$ rounds. Algorithm \mathcal{A}' proceeds as follows on any n -node input forest $G' \in \mathcal{F}$, where, for any node u , we denote the (maximal) connected component containing u by C_u and the number of nodes in C_u by $|C_u|$.

First, each node collects its $(2T(n^2) + 2)$ -hop neighborhood in G' . Then, based on the collected information, each node u determines whether there exists a node v in C_u such that the $(T(n^2) + 1)$ -hop neighborhood of v contains all of C_u .

If such a node v exists, then each node in C_u is aware of the whole component C_u and can simply choose the same solution for Π on C_u (by mapping component C_u (including unique identifiers or random bits) in some arbitrary, but fixed, deterministic fashion to some correct solution), and then output the part of the solution it is responsible for. Note that this requires all nodes in C_u to be distinguishable from each other (so that each node knows which part of the solution on C_u it is responsible for); for deterministic algorithms this is guaranteed by the unique identifiers, for randomized algorithms it is guaranteed with probability at least $1 - 1/n^2$ by having each node interpret its first $\lceil 4 \log n \rceil$ random bits as an identifier (which guarantees uniqueness of the created identifiers with probability at least $1 - 1/n^2$).

If no such node v exists, then u simply executes \mathcal{A} with input parameter⁷ n^2 (and each node in C_u will do likewise). In this case, due to the fact that no $(T(n^2) + 1)$ -hop node neighborhood fully contains C_u , it holds for each node v in C_u that the $(T(n^2) + 1)$ -hop neighborhood of v in G' is isomorphic to the $(T(n^2) + 1)$ -hop neighborhood of some node w in some n^2 -node tree $G \in \mathcal{T}$. Hence, if \mathcal{A}' fails on some node v in C_u or on some edge incident to v , then \mathcal{A} fails on some node w in some n^2 -node tree, or on some edge incident to w . Since the failure probability of \mathcal{A} on n^2 -node trees is at most $1/n^2$, it holds for any node w in any n^2 -node tree that the probability that \mathcal{A} fails on w or an edge incident to w is at most $1/n^2$. It follows for each node v in C_u that the probability that \mathcal{A}' fails on v or an edge incident to v is at most $1/n^2$.

Now, a union bound over all components C_u of the first kind (“such a node v exists”) and all nodes in components C_u of the second kind (“no such node v exists”) yields that \mathcal{A}' fails with probability at most $1/n$. Note that if \mathcal{A} is deterministic, then all of the above failure probabilities are 0, and \mathcal{A}' is deterministic as well.

For the runtime of \mathcal{A}' , observe that in either of the two considered cases, the initial collection of u ’s $(2T(n^2) + 2)$ -hop neighborhood suffices to compute u ’s output in \mathcal{A}' . Since $T(n) \in o(\log^* n)$ implies $2T(n^2) + 2 \in o(\log^* n)$, it follows that the runtime of \mathcal{A}' , and therefore also the complexity of Π on \mathcal{F} , is in $o(\log^* n)$. \square

⁷Recall that each node receives as input a parameter representing the number of nodes. Note that nothing prevents us from executing an algorithm using an input parameter that does not represent the correct number of nodes.

3.2 From Harder to Easier Problems

Recall that, for any set N of positive integers, \mathcal{F}_N denotes the class of forests with a number of nodes that is contained in N . The goal of this section is to prove the following theorem.

THEOREM 3.4. *Let Π be a node-edge-checkable LCL problem and \mathcal{A} a randomized algorithm solving Π on \mathcal{F} with runtime $T(n)$ and local failure probability at most $p \leq 1$.⁸ Let N be the set of all positive integers n satisfying $T(n) + 2 \leq \log_\Delta n$. Then, there exists a randomized algorithm \mathcal{A}' solving $\overline{\mathcal{R}}(\mathcal{R}(\Pi))$ on \mathcal{F}_N with runtime $\max\{0, T(n) - 1\}$ and local failure probability at most $Sp^{1/(3\Delta+3)}$, where*

$$S = (10\Delta(|\Sigma_{\text{in}}^\Pi| + \max\{|\Sigma_{\text{out}}^\Pi|, |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|\}))^{4\Delta^{T(n)+1}}.$$

In other words, we want to show, roughly speaking, that we can solve $\overline{\mathcal{R}}(\mathcal{R}(\Pi))$ at least one round faster than Π if we allow the indicated increase in the local failure probability of the randomized algorithm solving the problem.

For the proof of Theorem 3.4, we will make use of an approach that is an extension of the approach known for the setting of LCLs on regular trees without inputs (see, e.g., [4–6]). More specifically, we will explicitly define an algorithm \mathcal{A}' (depending on \mathcal{A}) that satisfies the properties stated in Theorem 3.4. While in the regular setting without inputs, the possibilities how the input graph could continue beyond the view of a node v differ only in the random bits of the nodes beyond v 's view, the irregularity and inputs in our setting require us to first perform a simulation step in the definition of \mathcal{A}' that simulates all possible topologies and inputs that a node could encounter (not too far) beyond its view before considering the randomness contained in each such extension defined by the combination of topology and inputs. As we show, the resulting more complex definition of \mathcal{A}' still allows us to give an upper bound on the increase of the local failure probability from \mathcal{A} to \mathcal{A}' that suffices for our purposes.

It should be noted that, due to the fact that we consider a large class of LCL problems at once (and not a single fixed LCL problem, in which case certain simplification techniques might be applicable), the increase in the number of output labels from Π to $\overline{\mathcal{R}}(\mathcal{R}(\Pi))$ is doubly exponential. Since the bound on the local failure probability of \mathcal{A}' depends (superlinearly) on the number of output labels of Π , and, ultimately, we want to apply Theorem 3.4 iteratively (starting with Π), we cannot apply Theorem 3.4 more than $\Theta(\log^* n)$ times before the local failure probability grows too large. This provides a (different) explanation why we cannot extend the $\omega(1) - o(\log^* n)$ gap further with our approach (which also follows from the fact that there are problems with complexity $\Theta(\log^* n)$).

Note that we will define \mathcal{A}' for all forests from \mathcal{F} , but will prove the guarantee on the local failure probability of \mathcal{A}' only for the forests in \mathcal{F}_N . Also recall that, for any node u in a graph G , we denote the r -hop neighborhood of u in G by $B_G(u, r)$. By abuse of notation, we will use $B_G(u, r)$ both for all the information contained in the r -hop neighborhood of u (i.e., the topology, inputs, and random bits) and for the respective subgraph of G (including input information, but no random bits).

Deriving \mathcal{A}' . Let \mathcal{A} be a randomized algorithm for some node-edge-checkable LCL problem Π with runtime $T = T(n)$. If $T = 0$, we can simply let \mathcal{A}' simulate \mathcal{A} , and then, for each half-edge h , transform the intermediate output $\ell \in \Sigma_{\text{out}}^\Pi$ returned by \mathcal{A} on h into the final output $\{\{\ell\}\} \in \Sigma_{\text{out}}^{\overline{\mathcal{R}}(\mathcal{R}(\Pi))}$ on h . By construction, \mathcal{A}' fails on some edge, resp. node, if and only if \mathcal{A} fails on the same edge, resp. node; since $p \leq Sp^{1/(3\Delta+3)}$ for the S specified in Theorem 3.4, it follows that \mathcal{A}' satisfies the properties required in Theorem 3.4. Hence, we will assume in the following that $T \geq 1$.

⁸Note that (bounds on) local failure probabilities such as p also (possibly) depend on the number n of nodes; however, for better readability we will omit this dependency.

In order to derive \mathcal{A}' from \mathcal{A} , we first derive an “intermediate” algorithm $\mathcal{A}_{1/2}$ for $\mathcal{R}(\Pi)$ from \mathcal{A} , and then we derive \mathcal{A}' from $\mathcal{A}_{1/2}$. As we will show later, $\mathcal{A}_{1/2}$ solves $\mathcal{R}(\Pi)$ (on \mathcal{F}_N) with a moderately increased local failure probability (and, intuitively speaking, very slightly reduced runtime) compared to \mathcal{A} ; a similar moderate increase in local failure probability (and slight decrease in runtime) is incurred when going from $\mathcal{A}_{1/2}$ to \mathcal{A}' .

Deviating from the usual convention that nodes are the entities performing the computation, we will assume for $\mathcal{A}_{1/2}$ that the edges of the input forest perform the computation. This is not in contradiction with the definition of the LOCAL model as $\mathcal{A}_{1/2}$ is only a construct defined for the design of \mathcal{A}' ; the actual computation in \mathcal{A}' is performed by the nodes. Algorithm $\mathcal{A}_{1/2}$ proceeds as follows.

Each edge $e = \{u, v\}$ in the input forest $G \in \mathcal{F}$ first collects all information (i.e., the topology, inputs, and random bits) contained in the union $B_G(e, T - 1/2) := B_G(u, T - 1) \cup B_G(v, T - 1)$ of the $(T - 1)$ -hop neighborhoods of u and v . Then, e determines the output label ℓ' it outputs on half-edge (u, e) as follows, depending on some parameter $0 < K \leq 1$ that we will choose later. Label ℓ' is simply the set of all labels ℓ such that there exists an input forest $G' \in \mathcal{F}$ (including input labels) and an edge e' in G' such that $B_{G'}(e', T - 1/2) \cong B_G(e, T - 1/2)$ and the probability that the node u' corresponding to u in the isomorphism outputs ℓ on (u', e') according to \mathcal{A} is at least K , conditioned on the assumption that the random bits in $B_{G'}(e', T - 1/2)$ are the same as in $B_G(e, T - 1/2)$. Here, the isomorphism is w.r.t. the topology and the input labels. In other words, in $\mathcal{A}_{1/2}$, edge e outputs on (u, e) the set of all labels ℓ for which the probability that, in \mathcal{A} , node u outputs ℓ on (u, e) , conditioned on the random bits that e has collected, is at least K for at least one possible extension of (the topology and input labels of) the graph beyond the $(T - 1/2)$ -hop view of e . Edge e computes the output label on half-edge (v, e) analogously. This concludes the description of $\mathcal{A}_{1/2}$; in the following we derive \mathcal{A}' from $\mathcal{A}_{1/2}$ in a fashion dual to how we derived $\mathcal{A}_{1/2}$ from \mathcal{A} .

In \mathcal{A}' , each node u first collects all information contained in $B_G(u, T - 1)$. Then, for each incident edge e , node u determines the output label ℓ'' it outputs on half-edge (u, e) as follows, depending on some parameter $0 < L \leq 1$ that we will choose later. Label ℓ'' is simply the set of all labels ℓ' such that there exists an input forest $G'' \in \mathcal{F}$ and a node u'' in G'' such that $B_{G''}(u'', T - 1) \cong B_G(u, T - 1)$ and the probability that the edge e'' corresponding to e in the isomorphism outputs ℓ' on (u'', e'') according to $\mathcal{A}_{1/2}$ is at least L , conditioned on the assumption that the random bits in $B_{G''}(u'', T - 1)$ are the same as in $B_G(u, T - 1)$. In other words, in \mathcal{A}' , node u outputs on (u, e) the set of all labels ℓ' for which the probability that, in $\mathcal{A}_{1/2}$, edge e outputs ℓ' on (u, e) , conditioned on the random bits that u has collected, is at least L for at least one possible extension of (the topology and input labels of) the graph beyond the $(T - 1)$ -hop view of u . This concludes the description of \mathcal{A}' .

In the following, for all forests in \mathcal{F}_N , we bound the local failure probability of $\mathcal{A}_{1/2}$, depending on (the bound on) the local failure probability of \mathcal{A} . We start by proving two helper lemmas. For any edge e , resp. node u , let p_e^* , resp. p_u^* , denote the probability that $\mathcal{A}_{1/2}$ fails at edge e , resp. node u . Moreover, for graphs G, G' , let f_{in} , resp. f'_{in} , be the functions that, for each half-edge in G , resp. G' , return the input label of the half-edge. Finally, we will use $\mathcal{A}(h)$ and $\mathcal{A}_{1/2}(h)$ to denote the labels that \mathcal{A} and $\mathcal{A}_{1/2}$, respectively, output on some half-edge h .

Lemma 3.5. *Let $e = \{u, v\}$ be an arbitrary edge in an arbitrary forest $G \in \mathcal{F}_N$. It holds that $p_e^* \leq ps/(K^2)$, where $s = (3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}}$.*

PROOF. Consider an arbitrary assignment of random bits in $B_G(e, T - 1/2)$ for which $\mathcal{A}_{1/2}$ fails on e , i.e., for which

- (1) the cardinality-2 multiset of labels that $\mathcal{A}_{1/2}$ outputs on (u, e) and (v, e) is not contained in $\mathcal{E}_{\mathcal{R}(\Pi)}$,
i.e., $\{\mathcal{A}_{1/2}((u, e)), \mathcal{A}_{1/2}((v, e))\} \notin \mathcal{E}_{\mathcal{R}(\Pi)}$, or
- (2) we have $\mathcal{A}_{1/2}((u, e)) \notin g_{\mathcal{R}(\Pi)}(f'_{\text{in}}((u, e)))$ or $\mathcal{A}_{1/2}((v, e)) \notin g_{\mathcal{R}(\Pi)}(f'_{\text{in}}((v, e)))$.

First, consider the case that Condition 1 is satisfied. Then, by the definition of $\mathcal{R}(\Pi)$, there are two labels $\ell_u \in \mathcal{A}_{1/2}((u, e))$, $\ell_v \in \mathcal{A}_{1/2}((v, e))$ such that $\{\ell_u, \ell_v\} \notin \mathcal{E}_\Pi$. Moreover, by the definition of $\mathcal{A}_{1/2}$, there is a forest $G' \in \mathcal{F}_N$ containing⁹ $B_G(e, T - 1/2)$ such that $|V(G')| = |V(G)|$ and, conditioned on the already fixed random bits in $B_{G'}(e, T - 1/2) = B_G(e, T - 1/2)$, the probability that \mathcal{A} , when executed on G' , returns ℓ_u on (u, e) and ℓ_v on (v, e) is at least K^2 . Note that we use here that the input graph is a forest: in order to be able to simply multiply the two probabilities of ℓ_u being returned on (u, e) and ℓ_v being returned on (v, e) (which are both lower bounded by K), we require that those two probabilities are independent (which is guaranteed if the input graph is a forest, as then $B_{G'}(u, T) \setminus B_{G'}(e, T - 1/2)$ and $B_{G'}(v, T) \setminus B_{G'}(e, T - 1/2)$ are disjoint). Note further that we use that $G \in \mathcal{F}_N$ (which implies that the number of nodes in $B_G(e, T - 1/2)$ is sufficiently small compared to $|V(G)|$) to guarantee the property $|V(G')| = |V(G)|$, and that this property is needed because the lemma statement relating the probabilities p_e^* and p (which technically speaking are functions of the number n of nodes) is supposed to hold for any fixed n . Finally, note that this is a place where it is crucial that we consider forests, not trees, as on trees it might be the case that no graph G' as described exists: if the two extensions beyond $B_G(e, T - 1/2)$ that are responsible for the containment of ℓ_u in $\mathcal{A}_{1/2}((u, e))$ and of ℓ_v in $\mathcal{A}_{1/2}((v, e))$ both have no edges leaving the respective extension except those connecting them to $B_G(e, T - 1/2)$, and the total number of nodes in the union of $B_G(e, T - 1/2)$ and those two extensions does not happen to be precisely n , then those two extensions cannot appear simultaneously if we assume the input graph to be an n -node tree.

Now consider the case that Condition 2 is satisfied. Then, by the definition of $\mathcal{R}(\Pi)$, there is some label $\ell_u \in \mathcal{A}_{1/2}((u, e))$ satisfying $\ell_u \notin g_\Pi(f'_{\text{in}}((u, e)))$ or some label $\ell_v \in \mathcal{A}_{1/2}((v, e))$ satisfying $\ell_v \notin g_\Pi(f'_{\text{in}}((v, e)))$. With an analogous argumentation to the one used in the previous case, we obtain that there is a forest $G' \in \mathcal{F}_N$ containing $B_G(e, T - 1/2)$ such that $|V(G')| = |V(G)|$ and the probability that \mathcal{A} , when executed on G' , fails on e is at least K .

Since $0 < K \leq 1$, we can conclude that in either case the probability (conditioned on the already fixed random bits in $B_{G'}(e, T - 1/2)$) that, in G' , \mathcal{A} fails on e is at least K^2 . Observe that the output of \mathcal{A} on the two half-edges belonging to e depends only on $B_{G'}(e, T + 1/2) = B_{G'}(u, T) \cup B_{G'}(v, T)$. Given the fixed topology and input in $B_{G'}(e, T - 1/2)$, there are at most $s := (3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}}$ different possibilities for the topology and input in $B_{G'}(e, T + 1/2)$: there are at most $2\Delta^T$ nodes in $B_{G'}(e, T + 1/2) \setminus B_{G'}(e, T - 1/2)$, and for each of the Δ possible ports of such a node, there are at most 3 possibilities regarding topology (being connected to a node in $B_{G'}(e, T - 1/2)$, being connected to a node outside of $B_{G'}(e, T + 1/2)$, or non-existing due to the degree of the node being too small) and at most $|\Sigma_{\text{in}}^\Pi|$ possibilities regarding the input label on the half-edge corresponding to that port. Let \mathcal{B} denote the set of different balls of the form $B_{G'}(e, T + 1/2)$ (for some G') that contain $B_G(e, T - 1/2)$. As shown above, $|\mathcal{B}| \leq s$.

Now, forget the fixing of the random bits in $B_G(e, T - 1/2)$. By the above discussion, it follows that there is some ball $B \in \mathcal{B}$ (and therefore also some forest G') containing $B_G(e, T - 1/2)$ such that the probability that \mathcal{A} , when executed on B (or G'), fails on e is at least $(p_e^*/s) \cdot K^2$. Since this probability is upper bounded by p , we obtain $p_e^* \leq ps/(K^2)$, as desired. \square

Lemma 3.6. *Let u be an arbitrary node in an arbitrary forest $G \in \mathcal{F}_N$. It holds that $p_u^* \leq p + |\Sigma_{\text{out}}^\Pi| \Delta K + \frac{ps\Delta}{K}$, where $s = (3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}}$.*

PROOF. Let $e_1, \dots, e_{\deg(u)}$ denote the edges incident to u . Moreover, let $p_u^{(1)}$ denote the probability that $\{\mathcal{A}_{1/2}((u, e_i))\}_{1 \leq i \leq \deg(u)} \notin \mathcal{N}_{\mathcal{R}(\Pi)}^{\deg(u)}$ and $p_u^{(2)}$ the probability that there exists some $1 \leq i \leq \deg(u)$ satisfying

⁹For better readability, we refrain from using the mathematically precise term “isomorphism” in the following, and instead identify isomorphic objects with each other, e.g., we consider $B_G(e, T - 1/2)$ to be a subgraph of G' if it is isomorphic to some subgraph of G' .

$\mathcal{A}_{1/2}((u, e_i)) \notin g_{\mathcal{R}(\Pi)}(f_{\text{in}}((u, e_i)))$. Since those two conditions together cover all cases in which $\mathcal{A}_{1/2}$ fails at u , we have $p_u^* \leq p_u^{(1)} + p_u^{(2)}$. We start by bounding $p_u^{(1)}$.

Observe that correctness at u , for both \mathcal{A} and $\mathcal{A}_{1/2}$, depends only on $B_G(u, T)$. Given the topology and input in $B_G(u, T)$ (which is already fixed since we are considering some fixed graph G), we call, for each label $\ell \in \Sigma_{\text{out}}^\Pi$ and each $1 \leq i \leq \deg(u)$, an assignment of random bits in $B_G(u, T)$ *bad for the pair* (ℓ, i) if $\mathcal{A}((u, e_i)) = \ell$ and $\ell \notin \mathcal{A}_{1/2}((u, e_i))$ (under this assignment). Observe that the definition of $\mathcal{A}_{1/2}$ ensures, for each fixed assignment of random bits in $B_G(e_i, T - 1/2)$, that if $\ell \notin \mathcal{A}_{1/2}((u, e_i))$ under this assignment¹⁰, then the probability that $\mathcal{A}((u, e_i)) = \ell$ (conditioned on the fixed assignment in $B_G(e_i, T - 1/2)$) is smaller than K . Hence, (prior to fixing any random bits) for each pair $(\ell, i) \in \Sigma_{\text{out}}^\Pi \times \{1, \dots, \deg(u)\}$, the probability that an assignment of random bits in $B_G(u, T)$ is bad for (ℓ, i) is smaller than K . It follows by a union bound that the probability that an assignment of random bits is bad for *some* pair $(\ell, i) \in \Sigma_{\text{out}}^\Pi \times \{1, \dots, \deg(u)\}$ is upper bounded by $|\Sigma_{\text{out}}^\Pi| \cdot \Delta \cdot K$.

Now, consider an arbitrary assignment of random bits in $B_G(u, T)$ such that $\{\mathcal{A}_{1/2}((u, e_i))\}_{1 \leq i \leq \deg(u)} \notin \mathcal{N}_{\mathcal{R}(\Pi)}^{\deg(u)}$ under this assignment. We argue that then \mathcal{A} fails at u or there is some pair $(\ell, i) \in \Sigma_{\text{out}}^\Pi \times \{1, \dots, \deg(u)\}$ such that the assignment is bad for (ℓ, i) : indeed, if there is no such pair and \mathcal{A} does not fail at u , then, for each $1 \leq i \leq \deg(u)$, we have $\mathcal{A}((u, e_i)) \in \mathcal{A}_{1/2}((u, e_i))$, which (combined again with the correctness of \mathcal{A} at u) would imply, by the definition of $\mathcal{N}_{\mathcal{R}(\Pi)}^{\deg(u)}$, that $\{\mathcal{A}_{1/2}((u, e_i))\}_{1 \leq i \leq \deg(u)} \in \mathcal{N}_{\mathcal{R}(\Pi)}^{\deg(u)}$, contradicting the stated property of the considered assignment. We conclude that $p_u^{(1)} \leq p + |\Sigma_{\text{out}}^\Pi| \cdot \Delta \cdot K$.

Next, we bound $p_u^{(2)}$. By a union bound, it follows from the definition of $p_u^{(2)}$ that there is some $1 \leq i \leq \deg(u)$ such that the probability that $\mathcal{A}_{1/2}((u, e_i)) \notin g_{\mathcal{R}(\Pi)}(f_{\text{in}}((u, e_i)))$ is at least $p_u^{(2)} / \deg(u) \geq p_u^{(2)} / \Delta$.

Consider an arbitrary assignment of random bits in $B_G(e_i, T - 1/2)$ such that $\mathcal{A}_{1/2}((u, e_i)) \notin g_{\mathcal{R}(\Pi)}(f_{\text{in}}((u, e_i)))$. Then, by the definition of $\mathcal{R}(\Pi)$, there is some label $\ell \in \mathcal{A}_{1/2}((u, e_i))$ satisfying $\ell \notin g_{\Pi}(f_{\text{in}}((u, e_i)))$. Moreover, by the definition of $\mathcal{A}_{1/2}$, there is some ball $B_{G'}(u, T)$ (in some forest $G' \in \mathcal{F}_N$ satisfying $|V(G')| = |V(G)|$) containing $B_G(e_i, T - 1/2)$ such that, conditioned on the already fixed random bits in $B_{G'}(e_i, T - 1/2) = B_G(e_i, T - 1/2)$, the probability that \mathcal{A} , when executed on $B_{G'}(u, T)$ (or G'), returns ℓ on (u, e_i) is at least K . Note that since $\ell \notin g_{\Pi}(f_{\text{in}}((u, e_i)))$, returning ℓ on (u, e_i) implies that \mathcal{A} fails at u .

As already established in the proof of Lemma 3.5, there are at most $s := (3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}}$ different possibilities for the topology and input of a ball $B_{G'}(e_i, T + 1/2)$ containing $B_G(e_i, T - 1/2)$, which implies the same bound on the number of different balls $B_{G'}(u, T)$ containing $B_G(e_i, T - 1/2)$. Analogously to before (and forgetting the fixing of the random bits in $B_G(e_i, T - 1/2)$), we obtain that there is some ball $B_{G'}(u, T)$ (in some forest G' satisfying $|V(G')| = |V(G)|$) containing $B_G(e_i, T - 1/2)$ such that the probability that \mathcal{A} , when executed on $B_{G'}(u, T)$ (or G'), fails at u is at least $((p_u^{(2)} / \Delta) / s) \cdot K$. Since this probability is upper bounded by p , we obtain $p_u^{(2)} \leq ps\Delta / K$, which implies

$$p_u^* \leq p + |\Sigma_{\text{out}}^\Pi| \Delta K + \frac{ps\Delta}{K}, \quad (3.1)$$

as desired. \square

By using Lemmas 3.5 and 3.6 and choosing K suitably, we now give an upper bound on the local failure probability of $\mathcal{A}_{1/2}$ on \mathcal{F}_N .

Lemma 3.7. *Algorithm $\mathcal{A}_{1/2}$ has local failure probability at most $2\Delta(s + |\Sigma_{\text{out}}^\Pi|)p^{1/3}$ on \mathcal{F}_N , where $s = (3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}}$.*

¹⁰Note that $\mathcal{A}_{1/2}((u, e_i))$ is uniquely determined after fixing the random bits in $B_G(e_i, T - 1/2)$.

PROOF. Choose the parameter in the definition of $\mathcal{A}_{1/2}$ as $K := p^{1/3}$. Then, by Lemma 3.6, for each node u in the input forest G , we obtain

$$p_u^* \leq p + |\Sigma_{\text{out}}^\Pi| \Delta p^{1/3} + s \Delta p^{2/3} \leq 2\Delta(s + |\Sigma_{\text{out}}^\Pi|) p^{1/3} ,$$

and, by Lemma 3.5, for each edge e in G , we obtain

$$p_e^* \leq s p^{1/3} \leq 2\Delta(s + |\Sigma_{\text{out}}^\Pi|) p^{1/3} .$$

The lemma statement follows by the definition of local failure probability. \square

Next, we prove an analogous statement to Lemma 3.7 relating the local failure probabilities of $\mathcal{A}_{1/2}$ and \mathcal{A}' . For any edge e , resp. node u , let p'_e , resp. p'_u , denote the probability that \mathcal{A}' fails at edge e , resp. node u .

Lemma 3.8. *If $\mathcal{A}_{1/2}$ has local failure probability at most $p^* \leq 1$ on \mathcal{F}_N , then \mathcal{A}' has local failure probability at most $3(s + |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|)(p^*)^{1/(\Delta+1)}$ on \mathcal{F}_N , where $s = (3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}}$.*

PROOF. We start by obtaining analogous statements to Lemmas 3.5 and 3.6. By simply exchanging the roles of nodes and edges¹¹ and reducing the radii of the considered balls by $1/2$ (as well as using parameter L instead of K , and using $\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}$ instead of Σ_{out}^Π), we directly obtain analogous proofs resulting in the following two statements, (assuming the stated upper bound p^* on the local failure probability of $\mathcal{A}_{1/2}$):

- (1) For any node u in the input forest G , we have $p'_u \leq p^* s / (L^\Delta)$.
- (2) For any edge e in G , we have $p'_e \leq p^* + |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}| \cdot 2L + p^* \cdot 2s/L$.

Note that, compared to Lemmas 3.5 and 3.6, each 2 has been replaced by Δ and vice versa, simply because the roles of edges (that end in 2 nodes) and nodes (that “end” in at most Δ edges) are reversed in the analogous proofs. Also observe that, technically, the expression for s that we obtain in the new proofs is $(3|\Sigma_{\text{in}}^\Pi|)^{\Delta \cdot \Delta^{(T-1)+1}}$; however, since this is upper bounded by the original expression for s , the obtained statements also hold for the original $s = (3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}}$ (which we will continue to use).

Now, analogously to the choice of $K = p^{1/(2+1)}$ in the proof of Lemma 3.7, set $L := (p^*)^{1/(\Delta+1)}$. We obtain, for each edge e in G ,

$$p'_e \leq p^* + |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}| \cdot 2(p^*)^{1/(\Delta+1)} + 2s(p^*)^{\Delta/(\Delta+1)} \leq 3(s + |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|)(p^*)^{1/(\Delta+1)} ,$$

and, for each node u in G ,

$$p'_u \leq s(p^*)^{1/(\Delta+1)} \leq 3(s + |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|)(p^*)^{1/(\Delta+1)} .$$

Again, the lemma statement follows by the definition of local failure probability. \square

Combining Lemmas 3.7 and 3.8, we are finally ready to prove Theorem 3.4.

PROOF OF THEOREM 3.4. Let S be as specified in the theorem, and let \mathcal{A}' be as derived before. As already argued during the definition of \mathcal{A}' , if the runtime $T = T(n)$ of \mathcal{A} is 0, then \mathcal{A}' satisfies the stated properties regarding runtime and local failure probability. Hence, assume in the following that $T \geq 1$.

From the definition of \mathcal{A}' , it follows directly that the runtime of \mathcal{A}' is $T - 1$. It remains to show that \mathcal{A}' has local failure probability at most $S p^{1/(3\Delta+3)}$ on \mathcal{F}_N . We start by showing that \mathcal{A}' has local failure probability at most

¹¹Note that, as usual for round elimination, all proofs also directly extend to hypergraphs. Hence, exchanging the roles of nodes and edges is very natural: a node v of degree $\deg(v)$ simply becomes a hyperedge containing $\deg(v)$ endpoints, while an edge becomes a node of degree 2.

$S'p^{1/(3\Delta+3)}$ on \mathcal{F}_N , where

$$S' := 3 \cdot ((3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}} + |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|) \cdot (2\Delta((3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}} + |\Sigma_{\text{out}}^\Pi|))^{1/(\Delta+1)}.$$

Indeed, if p satisfies $2\Delta((3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}} + |\Sigma_{\text{out}}^\Pi|)p^{1/3} \leq 1$, then this is a direct consequence of Lemmas 3.7 and 3.8. If p does not satisfy the given condition, then the straightforward combination of Lemmas 3.7 and 3.8 does not work, as the condition $p^* \leq 1$ in Lemma 3.8 is not satisfied. However, in this case, i.e., if $2\Delta((3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}} + |\Sigma_{\text{out}}^\Pi|)p^{1/3} > 1$, we obtain $S'p^{1/(3\Delta+3)} > 1$, which trivially implies that \mathcal{A}' has local failure probability at most $S'p^{1/(3\Delta+3)}$.

Now, the theorem statement follows from the fact that $S' \leq S$, which in turn follows from

$$\begin{aligned} S' &= 3 \cdot ((3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}} + |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|) \cdot (2\Delta((3|\Sigma_{\text{in}}^\Pi|)^{2\Delta^{T+1}} + |\Sigma_{\text{out}}^\Pi|))^{1/(\Delta+1)} \\ &\leq (9(|\Sigma_{\text{in}}^\Pi| + |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|))^{2\Delta^{T+1}} \cdot (6\Delta(|\Sigma_{\text{in}}^\Pi| + |\Sigma_{\text{out}}^\Pi|))^{2\Delta^{T+1}} \\ &\leq (10\Delta(|\Sigma_{\text{in}}^\Pi| + \max\{|\Sigma_{\text{out}}^\Pi|, |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|\}))^{2\Delta^{T+1}} \cdot (10\Delta(|\Sigma_{\text{in}}^\Pi| + \max\{|\Sigma_{\text{out}}^\Pi|, |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|\}))^{2\Delta^{T+1}} \\ &= (10\Delta(|\Sigma_{\text{in}}^\Pi| + \max\{|\Sigma_{\text{out}}^\Pi|, |\Sigma_{\text{out}}^{\mathcal{R}(\Pi)}|\}))^{4\Delta^{T+1}} \\ &= S. \end{aligned}$$

□

3.3 From Easier to Harder Problems

In this section, we prove the following lemma, which, in a sense, provides a counterpart to Theorem 3.4: it states that the time needed to solve some problem Π is not much larger than the time needed to solve $\overline{\mathcal{R}}(\mathcal{R}(\Pi))$. It is a simple extension of one direction of [14, Theorem 4.1] to the case of graphs with inputs. We note that the lemma also holds for randomized algorithms (with essentially the same proof), but we will only need the deterministic version.

Lemma 3.9. *Let Π be a node-edge-checkable LCL problem and let \mathcal{A} be a deterministic algorithm solving $\overline{\mathcal{R}}(\mathcal{R}(\Pi))$ in $T(n)$ rounds, for some function T (on some arbitrary class of graphs). Then there exists a deterministic algorithm \mathcal{A}' solving Π in $T(n) + 1$ rounds.*

PROOF. We will define \mathcal{A}' as follows, depending on \mathcal{A} .¹²

For each half-edge h , let $\mathcal{A}(h)$ denote the output label that \mathcal{A} outputs at h . In \mathcal{A}' , each node v starts by computing $\mathcal{A}(h)$ for each half-edge $h = (w, e)$ such that w is a neighbor of or identical to v and e is an edge incident to v . As, in \mathcal{A} , each neighbor of v computes its output in $T(n)$ rounds, v can compute all mentioned $\mathcal{A}(h)$ in $T(n) + 1$ rounds, by simulating \mathcal{A} . Node v will decide on its output solely based on the information collected so far, which implies that the runtime of \mathcal{A}' is $T(n) + 1$.

For choosing its output, v proceeds in two steps (computed without gathering any further information). In the first step, for each incident edge $e = \{v, w\}$, node v chooses, in some deterministic fashion, a label $L_{(v,e)} \in \mathcal{A}((v, e))$ and a label $L_{(w,e)} \in \mathcal{A}((w, e))$ such that $\{L_{(v,e)}, L_{(w,e)}\} \in \mathcal{E}_{\mathcal{R}(\Pi)}$. Such a pair of labels exists, by the definition of $\mathcal{E}_{\overline{\mathcal{R}}(\mathcal{R}(\Pi))}$ (and the fact that \mathcal{A} correctly solves $\overline{\mathcal{R}}(\mathcal{R}(\Pi))$). Moreover, the definition of $\mathcal{N}_{\mathcal{R}(\Pi)}^{\deg(v)}$ implies that the multiset $\{L_{(v,e')}\}_{e' \ni v}$ is contained in $\mathcal{N}_{\mathcal{R}(\Pi)}^{\deg(v)}$, and the definition of $g_{\overline{\mathcal{R}}(\mathcal{R}(\Pi))}$ implies that $L_{(v,e)} \in g_{\mathcal{R}(\Pi)}(i_{(v,e)})$ and $L_{(w,e)} \in g_{\mathcal{R}(\Pi)}(i_{(w,e)})$ where $i_{(v,e)}, i_{(w,e)} \in \Sigma_{\text{in}}^\Pi$ denote the input labels assigned to (v, e) and (w, e) , respectively. Note also that since the labels $L_{(v,e)}$ and $L_{(w,e)}$ are chosen in a deterministic fashion, depending only on $\mathcal{A}((v, e))$ and

¹²Note that the \mathcal{A} and \mathcal{A}' considered here are not the same as the ones considered in Section 3.2.

$\mathcal{A}((w, e))$, node v and node w will compute the same label $L_{(v,e)}$ and the same label $L_{(w,e)}$. We conclude that labeling each half-edge h with label L_h yields a correct solution for $\mathcal{R}(\Pi)$.

In the second step, v computes the final output for each half-edge incident to v , in a fashion analogous to the first step. More precisely, for each incident edge e , node v chooses a final output label $\ell_{(v,e)} \in L_{(v,e)}$ such that the multiset $\{\ell_{(v,e')}\}_{e' \ni v}$ is contained in $\mathcal{N}_{\Pi}^{\deg(v)}$. Such labels $\ell_{(v,e)}$ exist, by the definition of $\mathcal{N}_{\mathcal{R}(\Pi)}^{\deg(v)}$ (and the fact that the labeling assigning L_h to each half-edge h correctly solves $\mathcal{R}(\Pi)$). Moreover, the definition of $\mathcal{E}_{\mathcal{R}(\Pi)}$ implies that, for any edge $e = \{v, w\}$, the multiset $\{\ell_{(v,e)}, \ell_{(w,e)}\}$ is contained in \mathcal{E}_{Π} , and the definition of $\mathcal{G}_{\mathcal{R}(\Pi)}$ implies that, for any half-edge (v, e) , we have $\ell_{(v,e)} \in \mathcal{G}_{\Pi}(i_{(v,e)})$.

We conclude that labeling each half-edge h with label ℓ_h yields a correct solution for Π . It follows that \mathcal{A}' solves Π in $T(n) + 1$ rounds, as desired. \square

3.4 Proving the Gap

In this section, we will finally prove our main result that on forests or trees, any LCL problem with complexity $o(\log^* n)$ can be solved in constant time. We will first take care of the case of forests.

THEOREM 3.10. *Let Π be an arbitrary LCL problem that has (deterministic or randomized) complexity $o(\log^* n)$ on \mathcal{F} . Then Π can be solved in constant time on \mathcal{F} (both deterministically and randomized).*

PROOF. Observe that, by Lemma 2.6, it suffices to prove the theorem for node-edge-checkable LCL problems. Hence, assume in the following that Π is node-edge-checkable (and has deterministic or randomized complexity $o(\log^* n)$ on \mathcal{F}).

Observe further that if each node independently chooses an identifier (of length $O(\log n)$ bits) uniformly at random from a set of size n^3 , then the probability that there are two nodes that choose the same identifier is at most $1/n$, by a union bound over all pairs of nodes. Hence, a deterministic algorithm solving Π in $o(\log^* n)$ rounds can be transformed into a randomized algorithm with the same runtime and a failure probability of at most $1/n$ by having each node create its own identifier from its random bits before executing the deterministic algorithm. Thus, we can assume that the $o(\log^* n)$ -round algorithm for Π guaranteed in the theorem is randomized and fails with probability at most $1/n$. Let \mathcal{A} denote this algorithm, and observe that the bound of $1/n$ on the failure probability of \mathcal{A} implies that also the local failure probability of \mathcal{A} is bounded by $1/n$.

Let $T(n)$ denote the runtime of \mathcal{A} on the class of n -node forests from \mathcal{F} . Let n_0 be a sufficiently large positive integer; in particular, we require that

$$T(n_0) + 2 \leq \log_{\Delta} n_0, \quad (3.2)$$

$$2T(n_0) + 5 \leq \log^* n_0, \quad (3.3)$$

and

$$\left((S^*)^2 \cdot (\log n_0)^{2\Delta} \right)^{(3\Delta+3)T(n_0)} < n_0 \quad (3.4)$$

where $S^* := (10\Delta(|\Sigma_{\text{in}}^{\Pi}| + \log n_0))^{4\Delta T(n_0)+1}$. Since $T(n) \in o(\log^* n)$ (and Δ and Σ_{in}^{Π} are fixed constants), such an n_0 exists. Moreover, for simplicity, define $f(\cdot) := \overline{\mathcal{R}}(\mathcal{R}(\cdot))$, and recall that \mathcal{F}_{n_0} denotes the class of all forests from \mathcal{F} with n_0 nodes. Repeatedly applying $f(\cdot)$, starting with Π , yields a sequence of problems $\Pi, f(\Pi), f^2(\Pi), \dots$. Our first goal is to show that there is a 0-round algorithm solving $f^{T(n_0)}(\Pi)$ on \mathcal{F}_{n_0} with small local failure probability, by applying Theorem 3.4 repeatedly.

Recall that for any integer $i \geq 0$, we have $\Sigma_{\text{out}}^{\mathcal{R}(f^i(\Pi))} = 2^{\Sigma_{\text{out}}^{f^i(\Pi)}}$ and $\Sigma_{\text{out}}^{\overline{\mathcal{R}}(\mathcal{R}(f^i(\Pi)))} = 2^{\Sigma_{\text{out}}^{\mathcal{R}(f^i(\Pi))}}$, by definition. This implies that for any $0 \leq i \leq T(n_0)$, we have

$$\max\{|\Sigma_{\text{out}}^{f^i(\Pi)}|, |\Sigma_{\text{out}}^{\mathcal{R}(f^i(\Pi))}|\} \leq 2^{2^{\dots 2^{\Sigma_{\text{out}}^{\Pi}}}}$$

where the power tower is of height $2T(n_0) + 3$. By (3.3), we obtain that

$$\max\{|\Sigma_{\text{out}}^{f^i(\Pi)}|, |\Sigma_{\text{out}}^{\mathcal{R}(f^i(\Pi))}|\} \leq \log n_0 \text{ for all } 0 \leq i \leq T(n_0). \quad (3.5)$$

Recall that $S^* = (10\Delta(|\Sigma_{\text{in}}^{\Pi}| + \log n_0))^{4\Delta T(n_0)+1}$ and note that $\Sigma_{\text{in}}^{f^{T(n_0)}(\Pi)} = \Sigma_{\text{in}}^{f^{T(n_0)-1}(\Pi)} = \dots = \Sigma_{\text{in}}^{\Pi}$. By the above discussion, we conclude that when applying Theorem 3.4 to problem $f^i(\Pi)$ for some $0 \leq i \leq T(n_0)$, then the parameter S in the obtained upper bound $S p^{1/(3\Delta+3)}$ (in Theorem 3.4) is upper bounded by S^* . Hence, by applying Theorem 3.4 $T(n_0)$ times, each time using the upper bound $S^* p^{1/(3\Delta+3)}$ (instead of $S p^{1/(3\Delta+3)}$ with the respective S from Theorem 3.4), we obtain that there exists a randomized algorithm \mathcal{A}^* solving $f^{T(n_0)}(\Pi)$ on \mathcal{F}_{n_0} in 0 rounds with local failure probability p^* at most

$$\begin{aligned} & S^* \cdot (S^*)^{1/(3\Delta+3)} \cdot (S^*)^{1/(3\Delta+3)^2} \cdot \dots \cdot (S^*)^{1/(3\Delta+3)^{T(n_0)-1}} \cdot 1/(n_0)^{1/(3\Delta+3)^{T(n_0)}} \\ &= (S^*)^{\sum_{i=0}^{T(n_0)-1} (1/(3\Delta+3)^i)} \cdot 1/(n_0)^{1/(3\Delta+3)^{T(n_0)}} \\ &\leq (S^*)^2 \cdot 1/(n_0)^{1/(3\Delta+3)^{T(n_0)}} \\ &< 1/(\log n_0)^{2\Delta} \\ &\leq 1/\left(\Sigma_{\text{out}}^{f^{T(n_0)}(\Pi)}\right)^{2\Delta}, \end{aligned}$$

where the second-to-last inequality follows from (3.4), and the last inequality from (3.5). Note that (3.2) guarantees the applicability of Theorem 3.4 to the graph class \mathcal{F}_{n_0} for our purposes. Moreover, we can assume that \mathcal{A}^* outputs only labels from $\Sigma_{\text{out}}^{f^{T(n_0)}(\Pi)}$ (even if it fails), as otherwise we can turn \mathcal{A}^* into such an algorithm (with the same runtime and a smaller or equally small local failure probability) by simply replacing each label that \mathcal{A}^* outputs at some node and is not contained in $\Sigma_{\text{out}}^{f^{T(n_0)}(\Pi)}$ by some arbitrary label from $\Sigma_{\text{out}}^{f^{T(n_0)}(\Pi)}$.

Our next step is to show that the obtained bound on the local failure probability p^* of \mathcal{A}^* implies that there exists a *deterministic* algorithm solving $f^{T(n_0)}(\Pi)$ on \mathcal{F}_{n_0} in 0 rounds. To this end, let \mathcal{I} be the set of all tuples $I = (i_1, \dots, i_k)$ consisting of $k \in \{1, \dots, \Delta\}$ labels from Σ_{in}^{Π} , and \mathcal{O} the set of all tuples $O = (o_1, \dots, o_k)$ consisting of $k \in \{1, \dots, \Delta\}$ labels from $\Sigma_{\text{out}}^{f^{T(n_0)}(\Pi)}$. We say that $I = (i_1, \dots, i_k)$, resp. $O = (o_1, \dots, o_k)$, is the *input tuple*, resp. *output tuple*, of some node v if $\deg(v) = k$ and, for each $1 \leq j \leq k$, we have that i_j is the input label assigned to, resp. the output label returned by \mathcal{A}^* at, the half-edge incident to v corresponding to port j at v .

Since the runtime of \mathcal{A}^* is 0 rounds, any node v chooses its output tuple solely based on its random bit string and its input tuple. We now define a function $\mathcal{A}_{\text{det}}: \mathcal{I} \rightarrow \mathcal{O}$ (that will constitute the desired deterministic algorithm) as follows. Consider an arbitrary tuple $I = (i_1, \dots, i_k) \in \mathcal{I}$ and let v be a node with input label I . Since \mathcal{A}^* outputs only labels from $\Sigma_{\text{out}}^{f^{T(n_0)}(\Pi)}$ and $k \leq \Delta$, there are at most $\left(\Sigma_{\text{out}}^{f^{T(n_0)}(\Pi)}\right)^{\Delta}$ different possibilities for the output tuple of v . Hence, there exists a tuple $O \in \mathcal{O}$ that v outputs with probability at least $1/\left(\Sigma_{\text{out}}^{f^{T(n_0)}(\Pi)}\right)^{\Delta}$ (when executing \mathcal{A}^*). Fix such an O arbitrarily, and set $\mathcal{A}_{\text{det}}(I) = O$. This concludes the description of \mathcal{A}_{det} . Note that the definition of \mathcal{A}_{det} is independent

of the choice of v (under the given restrictions) as the only relevant parameters are the input tuple at v (which is fixed) and the random bit string at v (which comes from the same distribution for each node).

We claim that for any two (not necessarily distinct) configurations $I, I' \in \mathcal{I}$, and any two (not necessarily distinct) labels $o \in \mathcal{A}_{\text{det}}(I)$ and $o' \in \mathcal{A}_{\text{det}}(I')$, we have $\{o, o'\} \in \mathcal{E}_{f^{T(n_0)}(\Pi)}$. For a contradiction, assume that there are two configurations $I = (i_1, \dots, i_k), I' = (i'_1, \dots, i'_{k'})$ from \mathcal{I} and two labels $o \in \mathcal{A}_{\text{det}}(I)$ and $o' \in \mathcal{A}_{\text{det}}(I')$ satisfying $\{o, o'\} \notin \mathcal{E}_{f^{T(n_0)}(\Pi)}$. Let $O = (o_1, \dots, o_k)$ and $O' = (o'_1, \dots, o'_{k'})$ be the tuples that I and I' , respectively, are mapped to by \mathcal{A}_{det} , i.e., $\mathcal{A}_{\text{det}}(I) = O$ and $\mathcal{A}_{\text{det}}(I') = O'$. Let $j \in \{1, \dots, k\}$ and $j' \in \{1, \dots, k'\}$ be two ports/indices such that $o_j = o$ and $o'_{j'} = o'$. Consider a forest (with n_0 nodes) containing two adjacent nodes v, v' such that I is the input tuple of v , I' is the input tuple of v' , and the edge $e := \{v, v'\}$ corresponds to port j at v and to port j' at v' . By the definition of \mathcal{A}_{det} , we know that, when executing \mathcal{A}^* , the probability that v outputs o on half-edge (v, e) and the probability that v' outputs o' on half-edge (v', e) are each at least $1/\left(\sum_{\text{out}}^{f^{T(n_0)}(\Pi)}\right)^\Delta$. Since these two events are independent (as one depends on the random bit string of v and the other on the random bit string of v'), it follows that the probability that the output on edge e is $\{o, o'\}$ is at least $1/\left(\sum_{\text{out}}^{f^{T(n_0)}(\Pi)}\right)^{2\Delta}$. Now, $\{o, o'\} \notin \mathcal{E}_{f^{T(n_0)}(\Pi)}$ yields a contradiction to the fact the local failure probability of \mathcal{A}^* is strictly smaller than $1/\left(\sum_{\text{out}}^{f^{T(n_0)}(\Pi)}\right)^{2\Delta}$, proving the claim.

Observe that for any configuration $I = (i_1, \dots, i_k)$ from \mathcal{I} , we also have that

- (1) $\mathcal{A}_{\text{det}}(I)$ (or, more precisely, the unordered underlying multiset) is contained in $\mathcal{N}_{f^{T(n_0)}(\Pi)}^k$, and
- (2) for each port $1 \leq j \leq k$, the entry from $\mathcal{A}_{\text{det}}(I)$ corresponding to port j is contained in $g_{f^{T(n_0)}(\Pi)}(i_j)$,

as otherwise for each node v with input tuple I , algorithm \mathcal{A}^* would fail at v with probability at least $1/\left(\sum_{\text{out}}^{f^{T(n_0)}(\Pi)}\right)^\Delta$, yielding again a contradiction to the aforementioned upper bound on the local failure probability of \mathcal{A}^* . By the above discussion, we conclude that the output returned by \mathcal{A}_{det} is correct at each node and each edge, and therefore \mathcal{A}_{det} constitutes a deterministic 0-round algorithm solving $f^{T(n_0)}(\Pi)$ on \mathcal{F}_{n_0} .

By definition, algorithm \mathcal{A}_{det} is simply a function from \mathcal{I} to \mathcal{O} . Hence, while technically \mathcal{A}_{det} has been defined only for forests from \mathcal{F}_{n_0} , we can execute \mathcal{A}_{det} also on forests with an arbitrary number of nodes and it will still yield a correct output. We conclude that \mathcal{A}_{det} constitutes a deterministic 0-round algorithm solving $f^{T(n_0)}(\Pi)$ on \mathcal{F} .

Now, we can simply apply Lemma 3.9 $T(n_0)$ times, starting with $f^{T(n_0)}(\Pi)$, and obtain that there exists a deterministic algorithm solving Π on \mathcal{F} in $T(n_0)$ rounds (which implies the existence of a randomized algorithm solving Π on \mathcal{F} in $T(n_0)$ rounds, using the same argument as in the beginning of the proof). As n_0 is a fixed positive integer (depending only on Δ and Π), it follows that Π can be solved in constant time on \mathcal{F} (both deterministically and randomized). \square

Now, by combining Theorem 3.10 with Lemma 3.3 and Lemma 2.6 (which guarantees that Lemma 3.3, which is stated for node-edge-checkable LCL problems, can be applied), we obtain our main result as a corollary.

THEOREM 3.11 (FORMAL VERSION OF THEOREM 1.1). *Let Π be an arbitrary LCL problem that has (deterministic or randomized) complexity $o(\log^* n)$ on \mathcal{T} . Then Π can be solved in constant time on \mathcal{T} (both deterministically and randomized).*

4 THE VOLUME MODEL GAP

In this section, we show that a deterministic or randomized VOLUME algorithm (LCA) with a probe complexity of $o(\log^* n)$ implies a deterministic VOLUME algorithm (LCA) with a probe complexity of $O(1)$. We first show the speed-up only for deterministic VOLUME algorithms and later discuss how to extend the result to the full generality.

THEOREM 4.1. *There does not exist an LCL with a deterministic VOLUME complexity between $\omega(1)$ and $o(\log^* n)$.*

PROOF. Consider some LCL $\Pi = (\Sigma_{\text{in}}, \Sigma_{\text{out}}, r, \mathcal{P})$. Let \mathcal{A} be a VOLUME model algorithm with a probe complexity of $T(n) = o(\log^* n)$ that solves Π . We show that this implies the existence of a VOLUME model algorithm that solves Π and has a probe complexity of $O(1)$. To do so, we first show that we can turn \mathcal{A} into an order-invariant algorithm with the same probe complexity. Once we have shown this, Theorem 2.11, which shows a speed-up result for order-invariant algorithms, implies that there also exists a VOLUME model algorithm for Π with a probe complexity of $O(1)$, as needed.

We start by proving the lemma below, which, informally speaking, states that there exists a sufficiently large set $S_n \subseteq [n]$ such that \mathcal{A} is order-invariant as long as the IDs it "encounters" are from the set S_n . The proof adapts a Ramsey-theoretic argument first introduced in [41] and further refined in [19] in the context of the LOCAL model to the VOLUME model.

Lemma 4.2. *There exists a $n_0 \in \mathbb{N}$ such that the following holds for every $n \geq n_0$. There exists a set $S_n \subseteq [n]$ of size $(T(n) + 1) \cdot \Delta^{r+1}$ such that for every $i \in [T(n) + 1]$ and $t, t' \in \text{Tuples}_{i, S_n}$ it holds that $f_{n,i}(t) = f_{n,i}(t')$ if t and t' are almost identical.*

PROOF. Consider an $n \in \mathbb{N}$. We denote with H a complete $(T(n) + 1)$ -uniform hypergraph on n nodes. Each node in H corresponds to a (unique) ID in the set $\{1, 2, \dots, n\}$. For each hyperedge $X \subseteq \{1, 2, \dots, n\}$ we define a function f_X .

The input to f_X is a tuple $t = (t_1, \dots, t_{T(n)+1}) \in \text{Tuples}_{T(n)+1, [T(n)+1]}$. For $j \in [T(n) + 1]$, let $t_j = (id_j, deg_j, in_j)$. We define a new tuple $t_j^X = (id_j^X, deg_j, in_j)$, where id_j^X is the id_j -th smallest element in X . We now define

$$f_X(t) = (f_{n,1}(t_1^X), f_{n,2}(t_1^X, t_2^X), \dots, f_{n,T(n)+1}(t_1^X, t_2^X, \dots, t_{T(n)+1}^X)).$$

We now prove two things. First, we show that for n being sufficiently large, there exists a set $S_n^{\text{big}} \subseteq [n]$ of size $(T(n) + 1) \cdot \Delta^{r+1} + T(n) + 1$ such that for any two hyperedges $X, Y \subseteq S_n^{\text{big}}$, $f_X = f_Y$. This will follow by Ramsey Theory and an upper bound on the number of possible different functions f_X . Second, let S_n be the set one obtains from S_n^{big} by discarding the $T(n) + 1$ largest elements of S_n^{big} . We show that S_n satisfies the conditions of Lemma 4.2.

For the Ramsey-theoretic argument, we start by upper bounding the total number of possible functions. Note that $|\text{Tuples}_{[T(n)+1]}| \leq (T(n) + 1) \cdot \Delta \cdot |\Sigma_{\text{in}}|^\Delta$ and therefore the possible number of inputs to the function f^X is $|\text{Tuples}_{T(n)+1, [T(n)+1]}| \leq \left((T(n) + 1) \cdot \Delta \cdot |\Sigma_{\text{in}}|^\Delta \right)^{T(n)+1} = T(n)^{O(T(n))}$. Note that the output of f_X is contained in the set $\left(\bigtimes_{i=1}^{T(n)} ([i] \times [\Delta]) \right) \times (\Sigma_{\text{out}})^{|\Delta|}$ and therefore there are at most $(T(n) \cdot \Delta)^{T(n)} \cdot |\Sigma_{\text{out}}|^\Delta = T(n)^{O(T(n))}$ possible outputs. Hence, there exist at most $\left(T(n)^{O(T(n))} \right)^{T(n)^{O(T(n))}}$ different possible functions. Let $R(p, m, c)$ denote the smallest number such that any p -uniform complete hypergraph on $R(p, m, c)$ nodes with each hyperedge being assigned one of c colors contains a monochromatic clique of size m . It holds that $\log^*(R(p, m, c)) = p + \log^* m + \log^* c + O(1)$ [21].

Setting $p = T(n) + 1$, $m = (T(n) + 1) \cdot \Delta^{r+1} + T(n) + 1$ and $c = \left(T(n)^{O(T(n))} \right)^{T(n)^{O(T(n))}}$, we can conclude that H contains a set $S_n^{\text{big}} \subseteq [n]$ of size m such that for any two hyperedges $X, Y \subseteq S_n^{\text{big}}$, $f_X = f_Y$ as long as $\log^*(n) \geq T(n) + 1 + \log^* m + \log^* c + O(1)$, which is the case for sufficiently large n as $T(n) = o(\log^* n)$, $\log^* m = o(T(n))$ and $\log^* c = o(T(n))$.

Now, let S_n be the set one obtains from S_n^{big} by discarding the $T(n) + 1$ largest elements from S_n^{big} . Let $i \in [T(n) + 1]$ and $t^{(\ell)} = ((id_1^\ell, deg_1, in_1), (id_2^\ell, deg_2, in_2), \dots, (id_i^\ell, deg_i, in_i)) \in \text{Tuples}_{i, S_n}$ for $\ell \in [2]$ such that $t^{(1)}$ and $t^{(2)}$ are almost

identical. It remains to show that $f_{n,i}(t^{(1)}) = f_{n,i}(t^{(2)})$. For $\ell \in [2]$, let $X^\ell \subseteq S_n^{big}$ such that $\{id_1^\ell, id_2^\ell, \dots, id_i^\ell\}$ contains the $|\{id_1^\ell, id_2^\ell, \dots, id_i^\ell\}|$ -th lowest elements of X^ℓ . Now, let $t = (t_1, t_2, \dots, t_{T(n)+1}) \in S_{T(n)+1, [T(n)+1]}$ such that for every $j \in [i]$, $t_j^{X^\ell} = (id_j^\ell, deg_j, in_j)$. Note that it follows from the way we defined X^1 and X^2 and the fact that $t^{(1)}$ and $t^{(2)}$ are almost identical that such a tuple t exists. As $X^\ell \subseteq S_n^{big}$, we have $f_{X^1}(t) = f_{X^2}(t)$. In particular, this implies that

$$f_{n,i}(t^{(1)}) = f_{n,i}(t_1^{X^1}, t_2^{X^1}, \dots, t_i^{X^1}) = f_{n,i}(t_1^{X^2}, t_2^{X^2}, \dots, t_i^{X^2}) = f_{n,i}(t^{(2)}),$$

which finishes the proof. \square

We now construct an order-invariant algorithm \mathcal{A}' with probe complexity $T'(n) = \max(O(1), T(n)) = o(\log^* n)$. Note that it is easy to make \mathcal{A}' order-invariant for every input graph having fewer than n_0 nodes. For $n \geq n_0$, we have $T'(n) = T(n)$ and for every $i \in \{1, 2, \dots, T'(n)\}$ and tuple $t = ((id_1, deg_1, in_1), (id_2, deg_2, in_2), \dots, (id_i, deg_i, in_i)) \in \text{Tuples}_{i, \mathbb{N}}$, we define

$$f_{n,i}^{\mathcal{A}'}(t) = f_{n,i}^{\mathcal{A}}(t' = (id'_1, deg_1, in_1), (id'_2, deg_2, in_2), \dots, (id'_i, deg_i, in_i))$$

where id'_1, \dots, id'_i is chosen in such a way that $\{id'_1, \dots, id'_i\}$ contains the $|\{id'_1, \dots, id'_i\}|$ -th smallest elements of S_n and t and t' are almost identical. It is easy to verify that \mathcal{A}' is indeed order-invariant.

It remains to show that \mathcal{A}' indeed solves Π . For the sake of contradiction, assume that this is not the case. This implies the existence of a Σ_{in} -labeled graph (G, f_{in}) on n nodes (with each node having a unique ID from a polynomial range, a port assignment, and G does not have any isolated nodes) such that \mathcal{A}' "fails" on (G, f_{in}) . Put differently, there exists a node v such that \mathcal{A}' produces a mistake in the r -hop neighborhood of v . The r -hop neighborhood of v consists of at most Δ^{r+1} vertices. To answer a given query, \mathcal{A}' "sees" at most $T(n) + 1$ nodes. Hence, to compute the output of all the half-edges in the r -hop neighborhood of v , \mathcal{A}' "sees" at most $\Delta^{r+1}(T(n) + 1) \leq |S_n|$ many nodes. We denote this set of nodes by $V^{visible}$. Even if the IDs of nodes outside of $V^{visible}$ are changed, \mathcal{A}' still fails around v . Moreover, as \mathcal{A}' is order-invariant, changing the IDs of the nodes in $V^{visible}$ in a manner that preserves the relative order does not change the fact that \mathcal{A}' fails around v . Hence, we can find a new assignment of IDs such that each node in $V^{visible}$ is assigned an ID from the set S_n such that \mathcal{A}' still fails around v . However, from the way we defined \mathcal{A}' and the property that S_n satisfies, it follows that \mathcal{A}' and \mathcal{A} produce the same output in the r -hop neighborhood around v . This contradicts the fact that \mathcal{A} is a correct algorithm. \square

We already argued in the preliminaries that Theorem 4.1 also implies that there does not exist an LCL with a deterministic LCA complexity between $\omega(1)$ and $o(\log^* n)$.

As noted in [42], the derandomization result by [19] can be used to show that randomness does not help (up to an additive constant in the round/probe complexity) in the LOCAL and VOLUME model for complexities in $O(\log^* n)$, and the same is true for the LCA model.

Hence, we obtain the following more general theorem.

THEOREM 4.3. *There does not exist an LCL with a randomized/deterministic LCA/VOLUME complexity between $\omega(1)$ and $o(\log^* n)$.*

5 SPEEDUP IN GRIDS

In this section we prove that on d -dimensional oriented grids, any $o(\log^* n)$ -round LOCAL algorithm for an LCL problem Π (with input labels) can be turned into a LOCAL algorithm for Π with a round complexity of $O(1)$. An oriented d -dimensional grid is a grid where each edge is labeled with a label from $[d]$ that says which dimension this edge corresponds to. Moreover, all edges corresponding to the same dimension are consistently oriented. The grid is assumed to be toroidal, that is, without boundaries. We believe, but do not prove, that the same result can be shown also for non-toroidal grids, that is, grids with boundaries. From now on fix a dimension d .

THEOREM 5.1. *There is no LCL problem Π with a randomized/deterministic LOCAL complexity between $\omega(1)$ and $o(\log^* n)$ in d -dimensional oriented grids for $d = O(1)$.*

The proof of the speedup is a relatively simple extension of an argument of Suomela in the paper of Chang and Pettie [19]. There, the argument is proven for LCLs without inputs on unoriented grids. It shows that every $o(\log^* n)$ -round algorithm can be even sped up to a 0-round algorithm. This is not the case anymore with inputs, which makes the proof slightly more involved.

The high-level idea of the proof is the following. First, we observe that a LOCAL algorithm on oriented grids implies a PROD-LOCAL algorithm with the same asymptotic complexity. A PROD-LOCAL algorithm is defined as follows.

Definition 5.2 (PROD-LOCAL model). An algorithm in the PROD-LOCAL model is defined as an algorithm in the LOCAL model, but it expects that each node $u \in V(G)$ for G an oriented grid gets an ordered sequence of d identifiers $id_1(u), \dots, id_d(u) \in [n^{O(1)}]$, one for each dimension of the input grid G . We require that the i -th identifier $id_i(u), id_i(v)$ of two nodes u, v is the same if and only if the two nodes have the same i -th coordinate in G .

An order-invariant PROD-LOCAL model is defined as follows. We say that two identifier assignments are order-indistinguishable in some neighborhood around a node u if for any two nodes v, w in that neighborhood of u and any $i, j \in [d]$, the i -th identifier of one node is either smaller than the j -th identifier of the other node in both ID assignments or larger in both ID assignments. Then, an order-invariant $t(n)$ -round PROD-LOCAL algorithm outputs the same value at the half edges of a node in case two ID assignments are order-indistinguishable in its $t(n)$ -hop neighborhood.

We have the following fact:

Proposition 5.3. *If an LCL Π allows has local deterministic/randomized complexity $t(n)$ in the LOCAL model, it has complexity $t(n)$ in the PROD-LOCAL model.*

PROOF. By result of [19], a $t(n)$ round randomized algorithm in the local model implies a $t(2^{O(n^2)})$ round deterministic algorithm. This implies that we can turn an $O(\log^* n)$ -rounds randomized algorithm into $O(\log^* n)$ -rounds deterministic one.

Each of the d identifiers of a node are bounded by n^c for some fixed constant c in the PROD-LOCAL model. Assigning each node the identifier $I = \sum_{i=1}^d I_i \cdot n^{c(i-1)}$, where (I_i) is its i -th identifier results in globally unique identifiers from a polynomial range. This allows simulation of the deterministic LOCAL algorithm in the PROD-LOCAL model as needed. \square

Next, we argue that an $o(\log^* n)$ -round PROD-LOCAL algorithm can be turned into an order-invariant PROD-LOCAL algorithm. To show the existence of such an order-invariant algorithm, we use a Ramsey theory based argument very similar to the one in Section 4 and [19].

Proposition 5.4. *If there is a $t(n) = o(\log^* n)$ round PROD-LOCAL algorithm for an LCL Π , then there is also an order-invariant PROD-LOCAL algorithm with the same round complexity.*

PROOF. The algorithm \mathcal{A} of local complexity $t(n)$ can be seen as a function that maps $(2t(n) + 1) \times (2t(n) + 1)$ sized grids with edges consistently oriented and labeled with numbers from $[d]$. Moreover, each vertex has a d -tuple of its identifiers and each half-edge a label from Σ_{in} . The function \mathcal{A} maps this input to an output label for each half-edge incident to a given node. Let $R(p, m, c)$ be defined as in the previous section and let H be a hypergraph on $n^{O(1)}$ nodes, each representing an identifier.

- (1) Assume we have a $t(n)$ -hop neighborhood already labeled without assigned identifiers and we want to assign the identifiers. There are at most $p = d \cdot (2t(n) + 1)$ numbers to assign.
- (2) Assume we have a $t(n) + r$ -hop neighborhood already labeled without assigned identifiers and we want to assign the identifiers. There are at most $m = d \cdot (2(t(n) + r) + 1)$ numbers to assign. Here, r is the local checkability radius of the problem.
- (3) The number z is defined to be the number of possible input neighborhoods not labeled with identifiers. We have $z \leq |\Sigma_{in}|^{2d(2t(n)+1)^d}$. Note that the extra $2d$ -factor in the exponent comes from the fact that we label half-edges.
- (4) Finally, c is the number of colors such that each color encodes a distinct function which takes as input one of the z possible neighborhoods labeled with Σ_{in} inputs. Moreover, for each such neighborhood we fix a given permutation on p elements π . The function outputs a tuple of d output labels from Σ_{out} , one entry corresponding to each half-edge incident to a vertex. Hence, for the number of such colors c we have $c = |\Sigma_{out}|^{dp!z}$.

We color each p -edge T of H by the following color out of c colors: we consider all of the z possible ways of labeling the $t(n)$ -hop neighborhood with labels from Σ_{in} and all $p!$ ways of how one can assign a set of p different identifiers of T to that neighborhood (we think of T as a sorted list on which we apply π and then in this order we fill in identifiers to the $t(n)$ -hop neighborhood). For each such input (out of $p! \cdot z$ possible) we now apply \mathcal{A} to that neighborhood and record the output from Σ_{out}^d . This gives one of c possible colors. We now use the bound $\log^* R(p, m, c) = p + \log^* m + \log^* c + O(1)$ as in the previous section. Hence, as we assume $t(n) = o(\log^* n)$, this implies $|V(H)| \geq R(p, m, c)$, and therefore there exists a set $S \subseteq V(H)$ of m distinct IDs such that all p -sized subsets $T \subseteq S$ have the same color.

We now define the new algorithm \mathcal{A}' as follows. Let N be a neighborhood from the order-invariant PROD-LOCAL model; take an arbitrary p -sized subset $T \subseteq S$ and label N with identifiers from T by the permutation π given by the order on N that we got. Then, apply \mathcal{A} on this input.

First, this algorithm is well defined as it does not matter which T we pick, the algorithm always answers the same because every $T \subseteq S$ is colored by the same color. Second, the algorithm is correct, since for any $(t(n) + r)$ -hop neighborhood that has at most m vertices we can choose some way of assigning identifiers from S ($S = m$) to that neighbourhood and \mathcal{A}' answers the same as what \mathcal{A} would answer with this assignment of identifiers for all nodes within the local checkability radius. \square

Once we have the order-invariant algorithm with a round complexity of $o(\log^* n)$, an easy adaptation of the proof for Theorem 2.11 implies that we can turn it into an order-invariant PROD-LOCAL algorithm with a round complexity of $O(1)$. However, because of the oriented-grid, we get a local order on the vertices for free, so we can turn the order-invariant PROD-LOCAL algorithm into an (order-invariant) LOCAL algorithm, thus finishing the proof.

Proposition 5.5. *If an LCL Π has complexity $o(n^{1/d})$ in the order-invariant PROD-LOCAL model, it has deterministic/randomized/deterministic order-invariant local complexity $O(1)$ in the LOCAL model.*

PROOF. This is an argument very similar to [21]. Let \mathcal{A} be the algorithm in the PROD-LOCAL model. We choose n_0 large enough and run \mathcal{A} “fooled” into thinking that the size of the grid is n_0 . As the order on the identifiers, choose the order where two identifiers $id_i(u), id_j(v)$ have $id_i(u) < id_j(v)$ if and only if either $i < j$ or $i = j$ and v is further than u in the i -th coordinate (the notion of “further than” is given by the orientation of the grid). A standard argument as in [21] shows that if n_0 is chosen large enough, the “fooled” algorithm needs to be correct, as otherwise \mathcal{A} would be incorrect when being run on grids of size n_0 . \square

Theorem 5.1 follows by application of Propositions 5.3 to 5.5.

ACKNOWLEDGEMENTS

We would like to thank Yi-Jun Chang, Jan Grebík, and Dennis Olivetti for helpful comments and discussions.

REFERENCES

- [1] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1132–1139. SIAM, 2012.
- [2] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. The distributed complexity of locally checkable problems on paths is decidable. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 262–271. ACM, 2019. doi:10.1145/3293611.3331606.
- [3] Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, pages 17:1–17:17, 2020. arXiv:1911.13294, doi:10.4230/LIPIcs.DISC.2020.17.
- [4] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *Proc. Foundations of Computer Science (FOCS)*, 2019.
- [5] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed Δ -coloring plays hide-and-seek. CoRR, abs/2110.00643, 2021. URL: <https://arxiv.org/abs/2110.00643>, arXiv:2110.00643.
- [6] Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 365–376, 2020. arXiv:2004.08282, doi:10.1109/FOCS46700.2020.00042.
- [7] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. Locally checkable problems in rooted trees. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 263–272. ACM, 2021. doi:10.1145/3465084.3467934.
- [8] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the local model. *Distributed Computing*, pages 1–23, 2020.
- [9] Alkida Balliu, Keren Censor-Hillel, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Locally checkable labelings with small messages. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.DISC.2021.8.
- [10] Alkida Balliu, Juho Hirvonen, Janne H Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1307–1318, 2018.
- [11] Alkida Balliu, Juho Hirvonen, Dennis Olivetti, and Jukka Suomela. Hardness of minimal symmetry breaking in distributed computing. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 369–378, 2019.
- [12] Anton Bernshteyn. Distributed algorithms, the Lovász local lemma, and descriptive combinatorics. *arXiv preprint arXiv:2004.04905*, 2020.
- [13] S. Brandt, O. Fischer, J. Hirvonen, B. Keller, T. Lempiäinen, J. Rybicki, J. Suomela, and J. Uitto. A lower bound for the distributed Lovász local lemma. In *Proc. 48th ACM Symp. on Theory of Computing (STOC)*, pages 479–488, 2016.
- [14] Sebastian Brandt. An automatic speedup theorem for distributed problems. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 379–388, 2019. doi:10.1145/3293611.3331611.
- [15] Sebastian Brandt, Yi-Jun Chang, Jan Grebík, Christoph Grunau, Václav Rozhoň, and Zoltán Vidnyánszky. Local problems on trees from the perspectives of distributed algorithms, finitary factors, and descriptive combinatorics. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 29:1–29:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.29.
- [16] Sebastian Brandt, Christoph Grunau, and Václav Rozhoň. The randomized local computation complexity of the Lovász local lemma. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 307–317. ACM, 2021. doi:10.1145/3465084.3467931.

- [17] Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznanski. LCL problems on grids. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 101–110. ACM, 2017. doi : 10.1145/3087801.3087833.
- [18] Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto. The complexity of distributed edge coloring with small palettes. In *Proc. 29th ACM-SIAM Symp. on Discrete Algorithms*, pages 2633–2652, 2018.
- [19] Y.-J. Chang, T. Kopelowitz, and S. Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. In *Proc. 57th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2016.
- [20] Yi-Jun Chang. The Complexity Landscape of Distributed Locally Checkable Problems on Trees. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing (DISC 2020)*, volume 179 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/opus/volltexte/2020/13096>, doi : 10.4230/LIPIcs.DISC.2020.18.
- [21] Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. In *Proc. 58th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 156–167, 2017.
- [22] Yi-Jun Chang, Jan Studený, and Jukka Suomela. Distributed graph problems through an automata-theoretic lens. In Tomasz Jurdzinski and Stefan Schmid, editors, *Structural Information and Communication Complexity - 28th International Colloquium, SIROCCO 2021, Wrocław, Poland, June 28 - July 1, 2021, Proceedings*, volume 12810 of *Lecture Notes in Computer Science*, pages 31–49. Springer, 2021. doi : 10.1007/978-3-030-79527-6_3.
- [23] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.
- [24] Clinton T. Conley, Jan Grebik, and Oleg Pikhurko. Divisibility of spheres with measurable pieces, 2020. arXiv:2012.07567.
- [25] Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *Proc. 58th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2017.
- [26] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. Improved deterministic network decomposition. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2904–2923. SIAM, 2021. doi : 10.1137/1.9781611976465.173.
- [27] Mohsen Ghaffari, David Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *Proc. Foundations of Computer Science (FOCS)*, pages 662–673, 2018.
- [28] Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. Conditional hardness results for massively parallel computation from distributed lower bounds. In *Proc. Foundations of Computer Science (FOCS)*, 2019.
- [29] Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2505–2523. Society for Industrial and Applied Mathematics, 2017.
- [30] Mika Göös, Juho Hirvonen, Reut Levi, Moti Medina, and Jukka Suomela. Non-local probes do not help with many graph problems. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing*, pages 201–214, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
- [31] Lukasz Grabowski, András Máthé, and Oleg Pikhurko. Measurable circle squaring. *Ann. of Math.*, 185(2):671–710, 2017.
- [32] Jan Grebik and Václav Rozhoň. Of toasts and tails, 2021. arXiv:2103.08394.
- [33] Alexander Holroyd, Thomas Liggett, et al. Symmetric 1-dependent colorings of the integers. *Electronic Communications in Probability*, 20, 2015.
- [34] Alexander E. Holroyd, Oded Schramm, and David B. Wilson. Finitary coloring. *Ann. Probab.*, 45(5):2867–2898, 09 2017. doi : 10.1214/16-AOP1127.
- [35] Miklós Laczkovich. Equidecomposability and discrepancy; a solution of Tarski’s circle-squaring problem. *J. Reine Angew. Math.*, 404:77–117, 1990.
- [36] Nathan Linial. Distributive graph algorithms – global solutions from local data. In *Proc. 28th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 331–335, 1987.
- [37] Nati Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.
- [38] Andrew S. Marks and Spencer T. Unger. Borel circle squaring. *Ann. of Math.*, 186(2):581–605, 2017.
- [39] András Máthé, Jonathan A. Noel, and Oleg Pikhurko. Circle squaring with Jordan measurable pieces. *work in progress*.
- [40] Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991.
- [41] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.
- [42] Will Rosenbaum and Jukka Suomela. Seeing far vs. seeing wide: Volume complexity of local graph problems. In *Proceedings of the 39th Symposium on Principles of Distributed Computing, PODC ’20*, page 89–98, New York, NY, USA, 2020. Association for Computing Machinery. doi : 10.1145/3382734.3405721.
- [43] Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proc. Symposium on Theory of Computation (STOC)*, 2020.
- [44] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. *arXiv preprint arXiv:1104.1377*, 2011.
- [45] Yinon Spinka. Finitely dependent processes are finitary. *Ann. Probab.*, 48(4):2088–2117, 2020.
- [46] Jukka Suomela. Landscape of locality. <https://jukkasuomela.fi/landscape-of-locality/>, 2020.