Locality of Distributed Graph Problems

by

Yi-Jun Chang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2019

Doctoral Committee:

Professor Seth Pettie, Chair
Assistant Professor Bernhard Haeupler, Carnegie Mellon University
Assistant Professor Viswanath Nagarajan
Assistant Professor Grant Schoenebeck
Assistant Professor Jukka Suomela, Aalto University

Yi-Jun Chang
cyijun@umich.edu
ORCID iD: 0000-0002-0109-2432

Acknowledgments

First and foremost, I would like to thank my advisor Seth Pettie for his guidance and support. I would also like to thank Bernhard Haeupler, Viswanath Nagarajan, Grant Schoenebeck, and Jukka Suomela for serving on my dissertation committee.

During my graduate studies, I was very fortunate to have had the opportunity to collaborate and discuss research with many talented people, from whom I learned a lot. I thank all my past and current collaborators: Alkida Balliu, Sebastian Brandt, Varsha Dani, Manuela Fischer, Mohsen Ghaffari, Thomas P. Hayes, Qizheng He, Kuan-Yi Ho, Wenyu Jin, Tsvi Kopelowitz, Wenzheng Li, Dennis Olivetti, Seth Pettie, Mikaël Rabie, Jukka Suomela, Jara Uitto, Ruosong Wang, Hsu-Chun Yen, Wei Zhan, Hengjie Zhang, Yufan Zheng.

Last but not least, I thank my parents for their unconditional support.

Table of Contents

List of Figures

List of Tables

Abstract

Locality is one of the central themes in distributed computing. Suppose in a network each node only has direct communication with its *local* neighbors, how efficiently can a *global* task be solved? We aim to investigate the locality of fundamental distributed graph problems. Toward this goal, we consider the following three basic abstract models of distributed computing.

- LOCAL: each device has direct communication links with its neighbors, there is no message size constraint.
- CONGEST: each device has direct communication links with its neighbors, the size of each message is at most $O(\log n)$ bits.
- CONGESTED-CLIQUE: each device has direct communication links with all other devices, the size of each message is at most $O(\log n)$ bits.

A brief summary of our results is as follows.

Complexity Theory for the LOCAL Model: We study the spectrum of natural problem complexities that can exist in the LOCAL model. We provide answers to the following fundamental questions regarding the nature of the LOCAL model: (i) How to classify the distributed problems according to their complexities? (ii) How much does randomness help? (iii) Can we solve more problems given more time?

Complexity of Distributed Coloring: The coloring problem is a classical and well-studied problem in distributed computing. We devise distributed algorithms for the edge-coloring problem and the vertex-coloring problem in the LOCAL model that improve upon the previous state of the art.

Bandwidth Constraint: We develop a new framework for algorithm design based on expander decompositions that allows us to apply CONGESTED-CLIQUE techniques to the CONGEST model. Using this approach, we provide improved algorithms for the triangle detection and enumeration problem in CONGEST.

Chapter 1

Introduction

Large and complex networks naturally arise in various disciplines. The internet, social interactions of humans, and some biological systems can be represented and analyzed as networks. A network is often modeled as a graph $G = (V, E)$, where each vertex $v \in V$ represents an node in the network, and each edge $e = \{u, v\} \in E$ represents a communication link between the two nodes u and v.

Our focus is the *locality* of distributed computing. Under the constraint that each node only has direct communication with its *local* neighbors, how efficiently can a *global* task be solved? This type of question has been studied extensively in the field of distributed computing.

Computational Models. In this thesis, we consider the following three basic abstract models of distributed computing.

- LOCAL: each node has direct communication links with its neighbors, there is no message size constraint.
- CONGEST: each node has direct communication links with its neighbors, the size of each message is at most $O(\log n)$ bits.
- CONGESTED-CLIQUE: each node has direct communication links with all other nodes, the size of each message is at most $O(\log n)$ bits.

In all these models, the computation proceeds in synchronized rounds. The *time complexity* of a given distributed task $\mathcal{P}$ is a minimum number of rounds of an algorithm that solves $\mathcal{P}$ on a network of n nodes, where the minimum is taken over all algorithms.

Complexity Theory for the LOCAL Model. Most existing works in distributed computing only focused on *specific* problems, such as leader election, consensus, broadcasting, and finding a maximal independent set. There is a lack of a unified theory that can deal with a large class of distributed problems.[1] This is in contrast to the sequential computation models such as the Turing machine or the RAM model of computation, where a very rich computational complexity theory has been developed [8].

We aim to develop a computational complexity theory for the LOCAL model that captures a wide range of "natural" distributed graph problems. Toward this goal, we will focus our attention to the *locally checkable labelings* (LCL) problems [116], i.e., the distributed problems whose solution is locally verifiable. The class of LCL problems is sufficiently general that it includes many problems that are well-studied in the LOCAL model, such as the maximal independent set problem and the graph coloring problem. By restricting ourselves to LCLs, we can avoid dealing with some uninteresting artificial problems like "gathering the IDs of all vertices within radius $\sqrt{n}$" and other inherently non-local problems such as leader election.

We will study the class of LCL problems in the LOCAL model from the perspective of *computational complexity theory*. Can we solve more problems given more time? Specifically, are there any LCL problems with time complexity $\Theta((\log^* n)^2)$, $\Theta(\log \log n)$, or $\Theta(\sqrt{\log n})$? What is the value of random bits? If each vertex in the network is capable of generating independent random bits, then how much improvement in time complexity do we obtain? Is there a generic way to derandomize any randomized algorithm to a deterministic one? If so, what is the overhead in the transformation? We will address these fundamental questions in this thesis.

One surprising finding of our research is the existence of several *gaps* in the complexity landscape. For example, we show that the deterministic complexity of any LCL problem on bounded degree general graphs is either $\Omega(\log n)$ or $O(\log^* n)$; for paths and cycles, we show that there are only three possible asymptotic complexities: $\Theta(1)$, $\Theta(\log^* n)$, and $\Theta(n)$, and randomness do not help in improving the asymptotic complexity.

These complexity gaps are useful in several ways. For example, if we prove an $\Omega(\log \log n)$ randomized or deterministic lower bound for an LCL problem $\mathcal{P}$ on bounded-degree graphs, then the lower bound is automatically improved to $\Omega(\log n)$ due to the aforementioned $\omega(\log^* n)$—$o(\log n)$ gap. We will show that some gaps are *decidable* in the sense that we

[1] See [66] for a discussion on this issue.

can write a compute program that, given a description of an LCL problem $\mathcal{P}$, decides in a finite amount of time which side of the gap $\mathcal{P}$ is on. In particular, we will show that all gaps for paths and cycles are decidable. As a consequence, the process of designing asymptotically optimal distributed algorithms on paths and cycles can be automated.

Complexity of Distributed Coloring. The goal of the k-vertex coloring (resp., k-edge coloring) problem is to assign each vertex (resp., edge) a color from $\{1, 2, \ldots, k\}$ such that no two adjacent vertices (resp., edges) are assigned the same color. There has been a long line of research aiming at optimizing the time complexity and the number of colors. We continue this line of research and give improved results over the previous works.

We devise a randomized algorithm for $(1+\epsilon)\Delta$-edge coloring, for any $\epsilon = \tilde{\Omega}(1/\sqrt{\Delta})$. As long as $\epsilon \geq (\log^3 \Delta)/\sqrt{\Delta}$, our algorithm takes only $O(\log \Delta \log n)$ time. The palette size of our algorithm improves over the previous work of Elkin, Pettie, and Su [55], and it approaches a natural limit for randomized coloring strategies.[2]

We devise a randomized algorithm for $(\Delta+1)$-vertex coloring running in $2^{O(\sqrt{\log \log n})}$ time. This improves upon the previous state of the art by Harris, Schneider, and Su [86] which requires $O(\sqrt{\log \Delta}) + 2^{O(\sqrt{\log \log n})}$ time, and it achieves the best asymptotic complexity attainable via the graph shattering framework.

Bandwidth Constraint. The assumption of the LOCAL model that there is no message size constraint is somewhat unrealistic. In this thesis, we also investigate the locality of distributed graph problems under a bandwidth constraint. The CONGEST model is a variant of LOCAL that requires the size of each message to be at most $O(\log n)$ bits. The CONGESTED-CLIQUE model is a variant of CONGEST that allows all-to-all communication for all pairs of vertices, even those not joined by an edge.

Intuitively, the CONGEST model captures two constraints in distributed computing: *locality* and *bandwidth*, whereas the CONGESTED-CLIQUE model only focuses on the *bandwidth* constraint. This difference makes the two models behave very differently. For instance, the *minimum spanning tree* (MST) problem can be solved in $O(1)$ rounds in the CONGESTED-CLIQUE [93], but its round complexity is $\tilde{\Theta}(D + \sqrt{n})$ in CONGEST [128, 133], where D is the diameter of the network.

[2] We need $\epsilon = \Omega(1/\sqrt{\Delta})$ to guarantee a constant probability of being able to color an edge e, given a random feasible coloring of its neighboring edges.

Does non-local communication always help?[3] Indeed, non-local communication is very helpful for designing algorithms for many distributed problems. There are already several successful cases where algorithms in CONGESTED-CLIQUE are designed by accelerating known LOCAL or CONGEST algorithms using non-local communication [37, 72, 106, 124, 125]. In this thesis, we try to do the reverse of this. We develop a framework of algorithm design based on *expander decompositions* that allow us to simulate CONGESTED-CLIQUE algorithms efficiently in the CONGEST model. Using this tool, we obtain improved upper bounds for the triangle detection and enumeration problems in CONGEST. This also suggests that these problems might be instances where non-local communication does not help much.

Organization. In Section 1.1, we present the computation models, problems, and tools that we use. In Section 1.2, we present a more detailed overview of our results. The first part of our work (complexity theory of the LOCAL model) is presented in Chapters 2, 3, and 4. The second part of our work (complexity of distributed coloring) is presented in Chapters 5 and 6. The third part of our work (bandwidth constraint) is presented in Chapter 7. We conclude in Chapter 8 with comments on future research directions.

1.1 Computational Models, Problems, and Tools

In this section we review the major computation models, problems, and tools used in this thesis. The main computational model considered is the LOCAL model [107, 127] of distributed message-passing computation. In the LOCAL model, all local computation is free and the size of messages is unbounded. Henceforth "time" refers to the number of rounds. Each vertex v is initially aware of its degree $\deg(v)$, a port numbering mapping its incident edges to $\{1, \ldots, \deg(v)\}$, certain global parameters such as $n \stackrel{\text{def}}{=} |V|$, $\Delta \stackrel{\text{def}}{=} \max_{v \in V} \deg(v)$, and possibly other information. The assumption that global parameters are common knowledge can sometimes be removed; see Korman, Sereni, and Viennot [99]. The most important measure of efficiency is the number of rounds. The differences between the randomized and the deterministic models are as follows.

DetLOCAL: In the deterministic model, all vertices are assumed to hold unique $\Theta(\log n)$-

[3] An analogous question has been considered in the *centralized local computation model* [82], and their conclusion is that *non-local* probes are not helpful for a wide range of graph problems.

bit IDs. This assumption is needed in order to avoid trivial impossibilities. Except for the information about $\deg(v)$, $\mathrm{ID}(v)$, and the port numbering, the initial state of v is identical to every other vertex. The algorithm executed at each vertex is deterministic.

RandLOCAL: In the randomized model each vertex may locally generate an unbounded number of independent truly random bits, but there are no globally shared random bits. Except for the information about $\deg(v)$ and its port numbering, the initial state of v is identical to every other vertex. Algorithms in this model operate for a specified number of rounds and have some probability of *failure*, the definition of which is problem specific. Unless otherwise stated, the maximum tolerable global probability of failure is $1/\mathrm{poly}(n)$. Throughout the thesis, the term "w.h.p." refers to a success probability of $1 - 1/\mathrm{poly}(n)$.

Clearly RandLOCAL algorithms are capable of generate distinct IDs (w.h.p.) if desired. Observe that the role of the parameter "n" is different in the two LOCAL models: in DetLOCAL it affects the ID length whereas in RandLOCAL it affects the failure probability.

1.1.1 Locally Checkable Labeling

Naor and Stockmeyer [116] introduced *locally checkable labelings* (LCL) to formalize a large class of natural graph problems in the LOCAL model. Fix a class $\mathcal{G}$ of possible input graphs and let Δ be the maximum degree in any such graph. Formally, an LCL problem $\mathcal{P}$ for $\mathcal{G}$ has a radius $r = O(1)$, input and output alphabets $\Sigma_{\mathrm{in}}, \Sigma_{\mathrm{out}}$ (which can depend on Δ, but not n), and a set $\mathcal{C}$ of acceptable configurations.

Each acceptable configuration $C \in \mathcal{C}$ is a graph centered at a specific vertex, in which each vertex has a degree, a port numbering, and two labels from Σ_{in} and Σ_{out}. Given the input graph $G(V, E, \phi_{\mathrm{in}})$ where $\phi_{\mathrm{in}} : V(G) \to \Sigma_{\mathrm{in}}$, an acceptable output is any function $\phi_{\mathrm{out}} : V(G) \to \Sigma_{\mathrm{out}}$ such that for each $v \in V(G)$, the "labeled subgraph" induced by $N^r(v)$ is isomorphic to a member of $\mathcal{C}$. Here the notion $N^r(v)$ not only considers the topology of the r-neighborhood of v, but also considers the following information stored within vertices in $N^r(v)$: (i) vertex degrees, (ii) port numberings, (iii) input labels, and (iv) output labels.

For bounded degree graphs, an LCL can be described explicitly by enumerating a finite number of acceptable configurations. For graph classes with unbounded degrees, LCLs can be defined by describing the set of acceptable configurations through some logical

expression. Many natural symmetry breaking problems can be expressed as LCLs, such as MIS, maximal matching, (α, β)-ruling sets, $(\Delta+1)$-vertex coloring, and sinkless orientation.

Note that the class of LCL problems is analogous to the class NP in the sequential computation model in the sense that a distributed problem $\mathcal{P}$ is an LCL if its solution can be verified in $r = O(1)$ rounds in the LOCAL model.

The following problems are examples of LCLs for $r = 1$ without input labels.

Maximal Independent Set (MIS). Given a graph $G = (V, E)$, find a set $I \subseteq V$ such that for any vertex $v \in V$, we have $N(v) \cap I = \emptyset$ iff $v \in I$.

k-Coloring. Given a graph $G = (V, E)$, find a labeling $V \to \{1, 2, \ldots, k\}$ such that for each edge $\{u, v\} \in E$, u and v are labeled with different numbers (also called colors).

For MIS it suffices to label vertices with $\Sigma_{\text{out}} = \{0, 1\}$ indicating whether they are in the MIS. For k-Coloring we use $\Sigma_{\text{out}} = \{1, \ldots, k\}$.

In this thesis we also consider the *list coloring* problem. In a k-list coloring problem, each vertex v is given a palette $\Psi(v)$ of k colors from an arbitrary universe. Each vertex v is required to color itself by a color from its own list $\Psi(v)$, and no two adjacent colors are colored the same.

1.1.2 Distributed Lovász Local Lemma

The Lovász local lemma (LLL) is a tool in probabilistic method defined as follows. Consider a set of independent random variables $\mathcal{V}$ and a set of *bad* events $\mathcal{E}$, where each $A \in \mathcal{E}$ depends on a subset $\text{vbl}(A) \subset \mathcal{V}$. Define the dependency graph as

$$G_{\mathcal{E}} = (\mathcal{E}, \{(A, B) \mid \text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset)\}).$$

Symmetric versions of the Lovász local lemma are stated in terms of d, the maximum degree in $G_{\mathcal{E}}$, and $p = \max_{A \in \mathcal{E}} \Pr[A]$. A standard version of the LLL says that if $ep(d + 1) < 1$ then $\Pr[\cap_{A \in \mathcal{E}} \overline{A}] > 0$, i.e., it is possible to avoid all bad events.

The constructive LLL problem is to assign values to all variables in $\mathcal{V}$ such that no event in $\mathcal{E}$ happens. The distributed LLL problem [44] is to assign values to all variables in $\mathcal{V}$ such that no event in $\mathcal{E}$ happens in the LOCAL model, where the communication network is identical to the dependency graph $G_{\mathcal{E}}$ of the LLL system. Specifically, each vertex v in $G_{\mathcal{E}}$ corresponds to an event A in $\mathcal{E}$, and v is initially aware of $\text{vbl}(A)$ and

$N(v) = \{A' \in \mathcal{E} \setminus \{A\} \mid \text{vbl}(A) \cap \text{vbl}(A') \neq \emptyset\}$, but no other information about the global structure of $G_{\mathcal{E}}$. It is straightforward to see that the distributed LLL is an LCL problem.

Moser and Tardos's [112] parallel resampling algorithm for the constructive LLL implies an $O(\log^2 n)$ time RandLOCAL algorithm under the LLL criterion $ep(d+1) < 1$. Distributed LLL was introduced by Chung, Pettie, and Su [44], and they gave an $O(\log_{1/epd^2} n)$ time algorithm under the LLL criterion $epd^2 < 1$, an $O(T_{\text{weak-MIS}} \cdot \log_{1/ep(d+1)} n)$ time algorithm under the LLL $ep(d+1) < 1$, and an $O(\log n / \log\log n)$ time algorithm under criterion $p \cdot \text{poly}(d)2^d < 1$. They observed that under *any* criterion of the form $p \cdot f(d) < 1$, $\Omega(\log^* n)$ time is necessary. Ghaffari's [68] weak MIS algorithm, together with [44], implies an $O(\log d \cdot \log_{1/ep(d+1)} n)$ algorithm under LLL criterion $ep(d+1) < 1$. Brandt et al. [31] proved that $\Omega(\log_{\log(1/p)} \log n)$ time in RandLOCAL is necessary, even under the permissive LLL criterion $p2^d \leq 1$.

Roughly speaking, distributed LLL can be seen as a black box tool that is able to amplify the (local) failure probability from $1/\text{poly}(d)$ to $1/\text{poly}(n)$. We illustrate a simple example of applying distributed Lovász Local Lemma in algorithm design in RandLOCAL. Consider the k-vertex coloring problem on a graph $G = (V, E)$, and consider a randomized procedure where each vertex chooses a color uniformly at random. For each edge $e = \{u, v\}$, we define $\mathcal{E}_e$ as the event where u and v are assigned the same color. The set of bad events is defined as $\mathcal{E} = \{\mathcal{E}_e \mid e \in E\}$. Each bad event in $\mathcal{E}$ occurs with probability $p = 1/k$ and depends on at most $d = 2(\Delta - 1)$ other events. It is clear that a distributed algorithm on $G_{\mathcal{E}}$ can be simulated by a distributed algorithm on G with a constant factor overhead. Therefore, as long as $ep(d+1) = e(2\Delta - 1)/k < 1$, a k-vertex coloring can be computed in time asymptotic to the time complexity of distributed LLL under the criterion $ep(d+1) < 1$. If one applies the $O(\log d \cdot \log_{1/ep(d+1)} n)$-time algorithm of [44], then we obtain the runtime of $O(\log d \cdot \log_{1/ep(d+1)} n) = O(\log n \log \Delta)$. Of course, there are better algorithms (that do not resort to distributed LLL) for solving the k-coloring problem when $k > e(2\Delta - 1)$. Nevertheless, there are several instances where the usage of distributed LLL appears to be very essential [38, 44, 55, 61, 129]. Our edge coloring algorithm in Chapter 5 will also use distributed LLL.

1.1.3 Graph Shattering

Graph shattering is a framework for designing RandLOCAL algorithms. This framework was introduced in [23] (using the ideas in [25]) and was further used in [27, 44, 55, 68,

71, 86, 100, 129]. The idea is to apply a randomized procedure that fixes some fragment of the output (e.g., part of the MIS is fixed, part of the coloring is fixed, etc.), thereby effectively removing a large fraction of the vertices from further consideration. If it can be shown that the connected components in the subgraph still under consideration have size $\text{poly}(\log n)$, one can revert to the best available *deterministic* algorithm and solve the problem on each component of the "shattered" graph in parallel. The randomized part is called the *pre-shattering* phase; the deterministic part is called the *post-shattering* phase.

Next, we present a sufficient condition for the graph to be shattered into small connected components. For a graph $G = (V, E)$ we say that a subset $S \subseteq V$ is a *distance-k* set if the following two conditions are met:

1. The distance between any two distinct vertices $u, v \in S$ is at least k.
2. Define $G^k = (V, E^k)$, where there is an edge $\{u, v\} \in E^k$ if and only if $\text{dist}_G(u, v) = k$. Then it is required that S is connected in G^k.

Lemma 1.1 ([23])**.** *The number of distinct distance-k sets of size t is less than $4^t \cdot n \cdot \Delta^{k(t-1)}$.*

Proof. A distance-k set is spanned by a tree in G^k. There are less than 4^t distinct unlabeled trees of t vertices, and there are less than $n\Delta^{k(t-1)}$ ways to embed a t-vertex tree in G^k. The lemma follows since there is an injective mapping from the family of distance-k sets of size t to subtrees of t vertices in G^k. □

Lemma 1.2 (The Shattering Lemma)**.** *Consider a randomized procedure that generates a subset of vertices $B \subseteq V$. Suppose that for each $v \in V$, we have $\Pr[v \in B] \leq \Delta^{-3c}$, and this holds even if the random bits not in $N^c(v)$ are determined adversarially. With probability at least $1 - n^{-\Omega(c')}$, each connected component in the graph induced by B has size at most $(c'/c)\Delta^{2c} \log_\Delta n$.*

Proof. This lemma is well-known; see e.g., [23, 39, 61]. For the sake of completeness, we provide a proof here. We set $k = 2c + 1$ and $t = (c'/c) \log_\Delta n$. By Lemma 1.1, the number of distinct distance-k sets of size t is less than

$$\begin{aligned} 4^t \cdot n \cdot \Delta^{k(t-1)} &= \Delta^{\frac{2t}{\ln \Delta} + k(t-1) + \log_\Delta n} \\ &= \Delta^{2c'(1+o(1)) \log_\Delta n}. \end{aligned}$$

Given a size-t distance-k set T, the probability that $T \subseteq B$ is at most $\Delta^{-3ct} = \Delta^{-3c' \log_\Delta n}$. By a union bound over all possible T, with probability $1 - n^{-\Omega(c')}$, there is no size-t distance-k set T such that $T \subseteq B$.

Let S be a connected subset of B of size at least $(c'/c)\Delta^{2c} \log_\Delta n$. Then there is a distance-$(2c+1)$ set $T \subseteq S$ of size t, which can be constructed greedily. By the above discussion, with probability $1 - n^{-\Omega(c')}$, there is no such set S. □

The shattering lemma allows us to argue that the bad vertices form connected components of size $O(\mathrm{poly}(\Delta) \cdot \log n)$ with probability $1 - 1/\mathrm{poly}(n)$, where n is the number of vertices in the underlying network.

We remark that there is one subtle issue [23, Remark 3.6] about the post-shattering phase that is often not discussed in the literature. To execute a τ-time DetLOCAL algorithm on an n'-vertex graph, it is required that all vertices are equipped with IDs of length $\ell = O(\log n')$ that are distinct within distance $O(\tau)$. If $n' = O(\mathrm{poly} \log n)$, then we have $\ell = O(\log \log n)$. Therefore, we cannot use the initial $O(\log n)$-bit IDs to run the DetLOCAL algorithm in the post-shattering phase.

There exist DetLOCAL algorithms whose runtime depends on the ID length. For example, the ruling set computation used in the network decomposition algorithm of [120] takes time linear in the ID-length.

Fortunately, such $O(\log \log n)$-bit IDs can be generated in $O(\tau)$ time, and so the asymptotic time complexity is not affected. The algorithm for generating such IDs is based on the following theorem of Linial.

Theorem 1.1 (Linial's Coloring [107])**.** *Let G be a graph which has been k-colored.*

- *There is a* DetLOCAL *algorithm that computes a $5\Delta^2 \log k$-coloring in one round.*
- *There is a* DetLOCAL *algorithm that computes a $\beta \cdot \Delta^2$-coloring in $O(\log^* k - \log^* \Delta + 1)$ time, where $\beta > 0$ is a universal constant.*

The initial $O(\log n)$-bit IDs in DetLOCAL can be viewed as a $\mathrm{poly}(n)$-coloring. The desired $O(\log \log n)$-bit IDs can be generated by running Linial's coloring on the graph $G^{O(\tau)}$ for $O(1)$ iterations. Note that one round of G^s can be simulated using $O(s)$ rounds on the underlying network G.

1.1.4 Other Computational Models

Recall that the LOCAL model does not have a message size constraint, and so all problems can be solved in $O(D)$ time by learning the entire graph topology, where D is the diameter of the graph. This assumption is somewhat unrealistic, and so this motivates us to consider computational models that impose a message size constraint. The CONGEST model is the variant of the LOCAL model that imposes an $O(\log n)$ message size constraint. The CONGESTED-CLIQUE model is a variant of CONGEST that allows all-to-all communication. In CONGESTED-CLIQUE, each vertex initially knows its adjacent edges and the set of vertex IDs, which we can assume w.l.o.g. is $\{1, \ldots, |V|\}$. In each round, each vertex transmits $n - 1$ $O(\log n)$-bit messages, one addressed to each vertex in the graph.

One of the main reasons that some problems can be solved efficiently in CONGESTED-CLIQUE is due to the routing algorithm of Lenzen [105]. As long as each vertex v is the source and the destination of at most $O(n)$ messages, we can deliver all messages in $O(1)$ rounds.

Lemma 1.3 (Lenzen's Routing)**.** *Consider a graph $G = (V, E)$ and a set of point-to-point routing requests, each given by the* IDs *of the corresponding source-destination pair. As long as each vertex v is the source and the destination of at most $O(n)$ messages, namely $O(n \log n)$ bits of information, we can deliver all messages in $O(1)$ rounds in the* CONGESTED-CLIQUE *model.*

Using this routing algorithm [105] as a communication primitive, many parallel algorithms can be transformed to efficient CONGESTED-CLIQUE algorithms [34]. For example, consider the distributed matrix multiplication problem, where the input matrices are distributed to the vertices such that the ith vertex initially knows the ith row. The problem can be solved in the CONGESTED-CLIQUE model in $\tilde{O}(n^{1/3})$ rounds over semirings, or $n^{1-(2/\omega)+o(1)} = o(n^{0.158})$ rounds over rings [34].

1.2 Overview of Our Results

In this section, we present an overview of our results.

1.2.1 Complexity Theory for the LOCAL Model

In this thesis we investigate the complexity of the *locally checkable labeling* (LCL) problems in the LOCAL model. We are able to demonstrate that there exist several "gaps" in the possible complexity landscape, and these gaps will reveal some interesting properties of the LOCAL model. We will mostly focus on the case $\Delta = O(1)$, but we note that all of our results can be generalized to the unbounded degree setting by adding a suitable dependence on Δ.

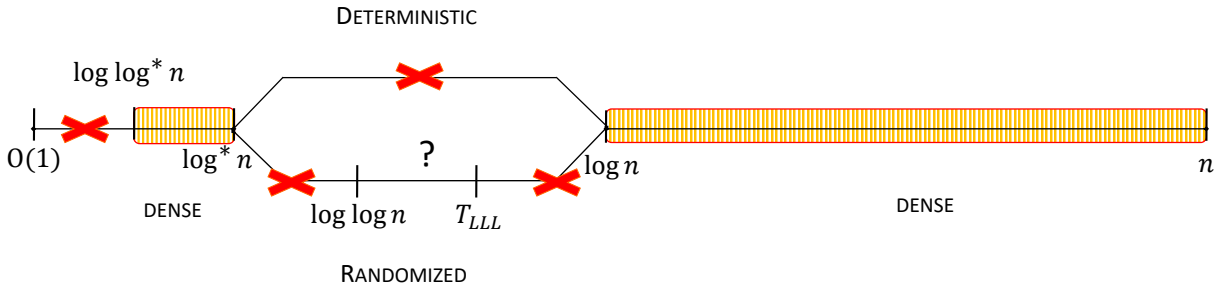

Figure 1.1: Complexity landscape for bounded-degree general graphs.

Bounded Degree General Graphs. See Figure 1.1 for an overview of the complexity landscape of the LOCAL model on bounded degree general graphs. The four gaps indicated by X are proved in Chapter 2. For instance, we show that the deterministic complexity of any LCL problem on bounded degree general graphs is either $\Omega(\log n)$ or $O(\log^* n)$.

Surprisingly, in a recent work of Balliu et al. [15], they showed that the complexity landscape of the two regions $[\Theta(\log\log^* n), \Theta(\log^* n)]$ and $[\Theta(\log n), \Theta(n)]$ are very *dense* in the sense that there are many time complexities in these regions that can be realized by LCL problems.[4]

Therefore, the only major puzzle (the question mark in Figure 1.1) that remains unsolved is whether $T_{\mathrm{LLL}} = \Theta(\log\log n)$ on bounded degree general graphs, which we conjecture to be true.

[4]Chronologically, in the conference version of [42] we proved the first time hierarchy-type theorem for the LOCAL model by demonstrating an infinite sequence of LCL problems with complexities $\Theta(n^{1/k})$, for $k \in \mathbb{Z}^+$, and this applies to both trees and general graphs. In the conference version of [42] we conjectured that there is an $\omega(n^{1/(k+1)})$—$o(n^{1/k})$ gap for each $k \in \mathbb{Z}^+$ for bounded degree general graphs. Later, Balliu et al. [15] disproved our conjecture by showing that the complexity landscape for bounded degree general graph is in fact very dense in some regions.

Bounded Degree Trees. The complexity landscape for trees is very different from the case of general graphs. In particular, the aforementioned result of Balliu et al. [15] does not apply to trees. In Chapter 3, we investigate the complexity landscape for trees, and we have the following results.

We define an infinite class of LCL problems called *Hierarchical* $2\frac{1}{2}$*-Coloring*. We show that the complexity of the k-level Hierarchical $2\frac{1}{2}$-Coloring problem is $\Theta(n^{1/k})$, for $k \in \mathbb{Z}^+$. The upper bound holds in DetLOCAL on general graphs, and the lower bound holds even on degree-3 trees in RandLOCAL. Therefore, not only general graphs, but also trees support an *infinite* number of natural problem complexities.

We prove that on the class of bounded degree trees, no LCL has complexity within the range $[\omega(\log n), n^{o(1)}]$. Moreover, we show that this gap is *decidable* in the sense that there is a sequential algorithm that is able to do the following. Given any description of an LCL problem $\mathcal{P}$, within a finite amount of time the algorithm decides which side of the gap $\mathcal{P}$ belongs to.

Paths and Cycles. For the case of paths and cycles, using the techniques developed in Chapter 2, we can show that there are only three possible complexities that can be realized by an LCL problem: $\Theta(1)$, $\Theta(\log^* n)$, and $\Theta(n)$, and random bits do not help in improving the asymptotic time complexity. However, these proofs are non-constructive by nature. In Chapter 4 we re-prove these gaps using a different approach that is able to give us a decidability result.

In contrast, some complexity gaps are known to be *undecidable*. It is a well-known result of Naor and Stockmeyer [116] that it is undecidable whether a given LCL problem can be solved in $O(1)$ time, even on grids and tori [32, 116].

Implications of the Complexity Gaps. These complexity gaps are useful in several ways. In what follows, we discuss some implications of these complexity gaps.

Separation between the Randomized and Deterministic Complexities. Since there exists a problem (precisely, Δ-coloring trees) whose randomized complexity is $O(\log_\Delta \log n)$ [129], our deterministic $\omega(\log^* n)$—$o(\log_\Delta n)$ gap provides the first instance of an *exponential separation* between the randomized and deterministic complexities in LOCAL.

Classification of Distributed Problems. "How to classify distributed problems into complexity classes?" has been identified as the most intriguing question for the LOCAL model [102, Section 6]. The existence of gaps implies that the distributed problems can be naturally divided into classes. For instance, on paths/cycles topology, there are only three possible complexities for LCL problems: $\Theta(1)$, $\Theta(\log^* n)$, and $\Theta(n)$. Thus, distributed problems on paths/cycles naturally form three complexity classes.

Automatic Speedup. Underlying each complexity gap we proved is an *automatic speedup theorem*, which is a tool that is sometimes applicable in algorithm design or lower bound proofs. For instance, there is an a $2^{O(\sqrt{\log \log n})}$-time randomized algorithm for Lovász local lemma on bounded-degree graphs [61], which can be derandomized into a $2^{O(\sqrt{\log n})}$-time deterministic algorithm; our $\omega(\log n)$—$n^{o(1)}$ gap for bounded-degree trees immediately implies that this algorithm can be accelerated to run in $O(\log n)$ time deterministically on bounded-degree trees. Similarly, the complexity gaps can be used to improve lower bounds. In view of the $\omega(\log^* n)$—$o(\log_\Delta \log n)$ gap in the randomized model, any randomized lower bound that is in between $\omega(\log^* n)$ and $o(\log_\Delta \log n)$ is automatically improved to $\Omega(\log_\Delta \log n)$. In Chapter 5, we apply this trick in our lower bound for edge-coloring.

Decidability. Some of the complexity gaps we proved are *decidable*. This suggests that we might be able to automate the design of distributed algorithm for some cases. In fact, in a recent work of Brandt et al. [32], they wrote a program to automate the design of some distributed algorithms. Recall that in this thesis we prove that all complexity gaps on paths and cycles are decidable. As a consequence, one can write a computer program to automatically design an asymptotically optimal distributed algorithm for any given LCL problem on paths or cycles.

Complete Problems. In this thesis we show the $\omega(T_{LLL})$—$o(\log n)$ gap in randomized LOCAL for bounded-degree graphs, where $T_{LLL} = o(\log n)$ [61] is the randomized complexity of distributed Lovász local lemma (under the criterion $pd^c < 1$ for some constant c). This implies that the distributed Lovász local lemma is a *complete problem* for the class $\mathcal{C}$ of randomized sublogarithmic-time problems in the following sense. If the distributed Lovász local lemma can be solved in randomized $T(n)$ time on bounded-degree graphs, then all problems in $\mathcal{C}$ can be solved in randomized $T(n)$ time on bounded-degree graphs.

Value of Random Bits. In Chapter 2, we prove that the randomized complexity for any LCL problem on instances of size n is at least its deterministic complexity on instances of size $\sqrt{\log n}$. There are some implications of this result.

Optimally of Graph Shattering. It is impossible to improve the $2^{O(\sqrt{\log\log n})}$ terms in the randomized $O(\log\Delta + 2^{O(\sqrt{\log\log n})})$-time MIS algorithm of [68] without also improving the $2^{O(\sqrt{\log n})}$-time deterministic algorithm of Panconesi and Srinivasan [120]. More generally, any randomized algorithm based on the *graph shattering* technique [23] is conditionally optimal, as we observe that the graph shattering often results in complexities of the form $f(\Delta) + T_{\mathrm{Det}}(\mathrm{poly}\log n)$, where $T_{\mathrm{Det}}(n')$ is the deterministic complexity of the same problem on an n'-vertex graph.

Derandomization. The $2^{O(\sqrt{\log\log n})}$-time randomized algorithm for edge-coloring in Chapter 5 and the $2^{O(\sqrt{\log\log n})}$-time distributed Lovász local lemma algorithm in [61] can be derandomized to run in deterministically $2^{O(\sqrt{\log n})}$ time. Also, any randomized algorithm for an LCL problem taking $t(n) = 2^{O(\log^* n)}$ time can be derandomized without asymptotic penalty. This improves upon a classical result of Naor and Stockmeyer [116]—the class of $O(1)$-time LCL problems in randomized and deterministic models is identical.

We note that our randomized $\omega(\log^* n)$—$o(\log\log n)$ gap is in fact proved by combining this derandomization tool and the deterministic $\omega(\log^* n)$—$o(\log n)$ gap.

1.2.2 Complexity of Distributed Coloring

Much of the research effort on the LOCAL model have been devoted to understanding the complexity of the four canonical symmetry breaking problems and their variants: maximal independent set (MIS), $(\Delta+1)$-vertex coloring, maximal matching, and $(2\Delta-1)$-edge coloring. In this thesis we focus on the coloring problems.

The numbers "$2\Delta-1$" and "$\Delta+1$" arise because they are the *smallest* palette size with the property that any partial coloring can be extended to a complete coloring, by the trivial greedy algorithm. Elkin, Pettie, and Su [55] gave randomized algorithms for $(1+\epsilon)\Delta$-edge coloring, for any constant $\epsilon > 0$, and Δ sufficiently lartge depending on ϵ. Vizing's theorem guarantees the existence of a $(\Delta+1)$-edge coloring; but it is unknown whether such a coloring can be efficiently computed in LOCAL.

To what extent the runtime and the number of colors can be lowered? This is a major research topic in the field of distributed graph algorithms. In this thesis we investigate this problem, and our main results are as follows.

Edge Coloring with Small Palettes. In Chapter 5 we devise a randomized algorithm for $(1+\epsilon)\Delta$-edge coloring, for any $\epsilon = \tilde{\Omega}(1/\sqrt{\Delta})$. The palette size of our algorithm approaches a natural limit for randomized coloring strategies. This is the threshold at which we have a constant probability of being able to color an edge e, given a *random* feasible coloring of its neighborhood. The round complexity of our algorithm is of the form

$$O(\log(1/\epsilon)) \cdot T_{\text{LLL}} + (\log\log n)^{3+o(1)},$$

where the parameters of LLL are $d = \text{poly}(\Delta)$ and $p = \exp(-\Omega(\epsilon^2\Delta/\log^{4+o(1)}\Delta))$. In particular, this is always upper bounded by $O(\log\Delta\log n)$ when $\epsilon \geq (\log^3\Delta)/\sqrt{\Delta}$.

A natural approach to computing $(\Delta+1)$-edge colorings (Vizing's theorem) is to extend partial colorings by iteratively re-coloring parts of the graph (via "alternating paths"). We prove that this approach may be viable, but in the worst case requires recoloring subgraphs of diameter $\Omega(\Delta\log n)$. This stands in contrast to distributed algorithms for Brooks' theorem [75, 122], which exploit the existence of $O(\log_\Delta n)$-length augmenting paths.

Faster Vertex Coloring Algorithm. In Chapter 6 we present a new algorithm for $(\Delta+1)$-list coloring in the randomized LOCAL model running in $O(\log^*\Delta + \mathsf{Det}_d(\text{poly}\log n)) = O(\mathsf{Det}_d(\text{poly}\log n))$ time, where $\mathsf{Det}_d(n')$ is the deterministic complexity of $(\deg+1)$-list coloring (v's palette has size $\deg(v)+1$) on n'-vertex graphs. This improves upon a previous randomized algorithm of Harris, Schneider, and Su [86] with complexity $O(\sqrt{\log\Delta} + \log\log n + \mathsf{Det}_d(\text{poly}\log n))$.

Our algorithm *appears* to be optimal, as it reaches the limit within the graph shattering framework. It *sort of* matches the $\Omega(\mathsf{Det}(\text{poly}\log n))$ randomized lower bound (Theorem 2.1), where Det is the deterministic complexity of $(\Delta+1)$-list coloring. The best known upper bounds on $\mathsf{Det}_d(n')$ and $\mathsf{Det}(n')$ are both $2^{O(\sqrt{\log n'})}$ (Panconesi and Srinivasan [120]) and it is quite plausible that the complexities of both problems are the same, asymptotically.

In Chapter 5 we also study the complexity of LLL on tree-structured dependency graphs, i.e., the underlying graph is T^r for some tree T and some constant r, under the condition

$p(ed)^\lambda < 1$. This dependency graph arises naturally when we consider a constant-round randomized procedure on a tree. We show that in this setting LLL can be solved in RandLOCAL in $O(\max\{\log_\lambda \log n, \frac{\log\log n}{\log\log\log n}\})$ time, when λ is a sufficiently large constant depending on r.

The motivation for us to study the complexity of LLL on tree-structured graphs is to establish nearly tight upper and lower bounds for edge coloring trees. Incorporating this new LLL algorithm to our edge coloring algorithm, we infer that the RandLOCAL time complexity of $(1+\epsilon)\Delta$-edge coloring trees is

$$O\left(\log(1/\epsilon) \cdot \max\{\tfrac{\log\log n}{\log\log\log n}, \log_{\log\Delta} \log n\}\right).$$

This matches our RandLOCAL $\Omega(\log_\Delta \log n)$ lower bound for $(2\Delta-2)$-edge coloring when $1/\epsilon = O(1)$ and $\Delta = O(1)$.

1.2.3 Bandwidth Constraint

In Chapter 7 we investigate the locality of distributed graph problems under a bandwidth constraint. In particular, we aim at understanding the difference between CONGEST (where only local communication is allowed) and CONGESTED-CLIQUE (where all-to-all communication is possible).

We know that there are several successful cases where algorithms in CONGESTED-CLIQUE are designed by accelerating known LOCAL or CONGEST algorithms using all-to-all communication [37, 72, 106, 124, 125]. Are there any techniques that allow us to efficiently adapt tools from CONGESTED-CLIQUE to CONGEST? Is there any scenario that allows us to design a CONGEST algorithm with a time complexity near the optimal complexity in CONGESTED-CLIQUE?

Ghaffari, Kuhn, and Su [77] proved that if each vertex v is the source and the destination of at most $O(\deg(v))$ messages, then all messages can be routed to their destinations in $\tau_{\text{mix}}(G) \cdot 2^{O(\sqrt{\log n \log\log n})}$ rounds, where $\tau_{\text{mix}}(G)$ is the *mixing time* of the lazy random walk on the graph G. The $2^{O(\sqrt{\log n \log\log n})}$ factor was later improved [78] to $2^{O(\sqrt{\log n})}$. The implication of this result is that many problems that can be solved efficiently in the CONGESTED-CLIQUE can also be solved efficiently in CONGEST, *but only if* $\tau_{\text{mix}}(G)$ *is small*. In particular, MST can be solved in $\tau_{\text{mix}}(G) \cdot 2^{O(\sqrt{\log n})}$ rounds in CONGEST [78]. This shows that the $\tilde{\Omega}(\sqrt{n})$ lower bound [128, 133] can be bypassed in networks with small

$\tau_{\text{mix}}(G)$.

A natural question to ask is whether or not this line of research [77, 78] can be extended to a broader class of graphs (that may have high $\tau_{\text{mix}}(G)$), or even general graphs. In Chapter 7, we show that this is in fact doable. Based on this approach we improve the state-of-the-art algorithms for triangle detection and enumeration.

The main technical novelty underlying in this result is a distributed graph partitioning algorithm. We show that in $\tilde{O}(n^{1-\delta})$ rounds we can partition the edge set of the network $G = (V, E)$ into three parts $E = E_m \cup E_s \cup E_r$ such that

- Each connected component induced by E_m has minimum degree $\Omega(n^\delta)$ and conductance $\Omega(1/\text{poly}\log(n))$. As a consequence the mixing time of a random walk within the component is $O(\text{poly}\log(n))$.
- The subgraph induced by E_s has arboricity at most n^δ.
- $|E_r| \leq |E|/6$.

Our triangle detection and enumeration algorithm is based on the following generic framework, which we believe is of interest beyond this work. Roughly, we deal with the set E_s by an algorithm that is efficient for low-arboricity graphs, and deal with the set E_r using recursive calls. For each connected component induced by E_m, we are able to simulate CONGESTED-CLIQUE algorithms with small overhead by applying a routing algorithm due to Ghaffari, Kuhn, and Su [77].

1.3 Publications that Constitute this Thesis

This thesis is based on a subset of my publications written during my PhD work at the University of Michigan.[5] Each of these research works is a result of a collaborative effort, and the credit of each work is shared by all collaborators.

Sections 2.1–2.4 of Chapter 2 are based on the publication [39]:

- Y.-J. Chang, T. Kopelowitz, and S. Pettie. "An exponential separation between randomized and deterministic complexity in the LOCAL model." In *Proceedings*

[5] My works on contention resolution [36], energy efficient distributed computing [35, 40], and the complexity of $(\Delta + 1)$-vertex coloring in CONGESTED-CLIQUE and other models [37] are not included in this thesis.

of the IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pages 615–624, 2016.

Sections 2.5–2.6 of Chapter 2 and Chapter 3 are based on the publication [42]:

- Y.-J. Chang and S. Pettie. "A time hierarchy theorem for the LOCAL model." In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 156–167, 2017.

Chapter 4 is based on the publication [13]:

- A. Balliu, S. Brandt, Y.-J. Chang, D. Olivetti, M. Rabie, and J. Suomela. "The distributed complexity of locally checkable problems on paths is decidable." *ArXiv e-prints 1811.01672*, 2018.

Chapter 5 is based on the publication [38]:

- Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto. "The complexity of distributed edge coloring withsmall palettes." In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2633–2652, 2018.

Chapter 6 is based on the publication [41]:

- Y.-J. Chang, W. Li, and S. Pettie. "An optimal distributed $(\Delta + 1)$-coloring algorithm?" In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 445–456, 2018.

Chapter 7 is based on the publication [43]:

- Y.-J. Chang, S. Pettie, and H. Zhang. "Distributed Triangle Detection via Expander Decomposition." In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 821–840, 2019.

Chapter 2

Complexity Landscape of LCLs on General Graphs

2.1 Overview

The objective of this chapter is to understand the value of *random bits* in the LOCAL model (i.e., how much advantage we can gain from switching from DetLOCAL to RandLOCAL?) and establish *gaps* in the possible complexities in DetLOCAL and RandLOCAL models.

Value of Random Bits. One of the central problems of theoretical computer science is to determine the value of *random bits*. If the distinction is between computable vs. incomputable functions, random bits are provably useless in centralized models (Turing machines) [137]. However, this is not true in the distributed world! The celebrated Fischer-Lynch-Patterson theorem [60] states that asynchronous deterministic agreement is impossible with one unannounced failure, yet it is possible to accomplish with probability 1 using randomization. See Ben-Or [26] and [30, 97, 130]. There are also a number of basic symmetry breaking tasks that are trivially impossible to solve by identical, synchronized, deterministic processes, for example, medium access control to an Ethernet-like channel.

Early work in the LOCAL models suggested that randomness is of limited help. Naor [115] showed that Linial's $\Omega(\log^* n)$ lower bound [107] for 3-vertex coloring the ring holds even in RandLOCAL. Naor and Stockmeyer [116] proved that the class of problems solvable by $O(1)$-round algorithms is the same in RandLOCAL and DetLOCAL. See also [59] for a generalization of this result. However, in the intervening decades we have seen dozens of examples of symmetry breaking algorithms for RandLOCAL that are substantially faster

than their counterparts in DetLOCAL; see [23] for an extensive survey or Table 2.1 for a glimpse at three archetypal problems: maximal independent set (MIS), maximal matching, and $(\Delta+1)$-vertex coloring.

Problem	Model and Result	
MIS	DetLOCAL:	$O\Big(\min\Big\{\Delta+\log^* n,\ 2^{O(\sqrt{\log n})}\Big\}\Big)$ [21, 120]
	RandLOCAL:	$O\Big(\log\Delta+2^{O(\sqrt{\log\log n})}\Big)$ [68]
	Lower Bound:	$\Omega\Big(\min\Big\{\sqrt{\log n/\log\log n},\ \log\Delta/\log\log\Delta+\log^* n\Big\}\Big)$ [102, 107, 115]
Maximal Matching	DetLOCAL:	$O\big(\min\big\{\Delta+\log^* n,\ \log^2\Delta\log n\big\}\big)$ [63, 119]
	RandLOCAL:	$O\big(\log\Delta+\log^3\log n\big)$ [23, 63]
	Lower Bound:	$\Omega\Big(\min\Big\{\sqrt{\log n/\log\log n},\ \log\Delta/\log\log\Delta+\log^* n\Big\}\Big)$ [102, 107, 115]
$(\Delta+1)$**-vertex Coloring**	DetLOCAL:	$O\Big(\min\Big\{\sqrt{\Delta\log\Delta}\log^*\Delta+\log^* n,\ 2^{O(\sqrt{\log n})}\Big\}\Big)$ [20, 65, 120]
	RandLOCAL:	$2^{O(\sqrt{\log\log n})}$ [41]
	Lower Bound:	$\Omega(\log^* n)$ [107, 115]

Table 2.1: The impact of "graph shattering" on three archetypical symmetry breaking problems.

Graph Shattering. The randomized algorithms in Table 2.1 are exponentially faster than their deterministic counterparts *in two ways*. Their dependence on Δ is exponentially faster and their dependence on n is usually identical to the best deterministic complexity, but for $\text{poly}(\log n)$-size instances. For example, $2^{O(\sqrt{\log n})}$ becomes $2^{O(\sqrt{\log\log n})}$, and $O(\log^3 n)$ becomes $O(\log^3\log n)$. This second phenomenon is no coincidence! It is a direct result of the *graph shattering* approach to symmetry breaking used in [23] and further in [27, 44, 55, 68, 71, 86, 100, 129]. The idea is to apply a randomized procedure that fixes some fragment of the output (e.g., part of the MIS is fixed, part of the coloring is fixed, etc.), thereby effectively removing a large fraction of the vertices from further consideration. If it can be shown that the connected components in the subgraph still under consideration have size $\text{poly}(\log n)$, one can revert to the best available *deterministic* algorithm and solve the problem on each component of the "shattered" graph in parallel.

Lower Bounds in the LOCAL Model. Until recently, the main principle used to prove lower bounds in the LOCAL model was *indistinguishability*. The first application of this principle was by Linial [107] himself, who argued that any algorithm for coloring degree-Δ

trees either uses $\Omega(\Delta/\log\Delta)$ colors or takes $\Omega(\log_\Delta n)$ time. The proof is as follows (i) in $o(\log_\Delta n)$ time, a vertex *cannot always distinguish* whether the input graph G is a tree or a graph with girth $\Omega(\log_\Delta n)$, (ii) for all Δ and all n, there exists a degree-Δ graph with girth $\Omega(\log_\Delta n)$ and chromatic number $\chi = \Omega(\Delta/\log\Delta)$, hence[6] (iii) any $o(\log_\Delta n)$-time algorithm for coloring trees could also color such a graph, and therefore must use at least χ colors.

A more subtle indistinguishability argument was used by Kuhn, Moscibroda, and Wattenhofer [102], who showed that $O(1)$-approximate vertex cover, maximal matching, MIS, and several other problems have $\Omega(\min\{\log\Delta/\log\log\Delta, \sqrt{\log n/\log\log n}\})$ lower bounds. Bar-Yehuda, Censor-Hillel, and Schwartzman [24] showed that a $(2+\epsilon)$-approximate vertex cover can be found in $O(\log\Delta/\log\log\Delta)$ time, matching the above lower bound.

By its nature, indistinguishability is not very good at separating randomized and deterministic complexities. Brandt et al. [31] developed a lower bound technique that explicitly incorporates error probabilities and proved that several problems on graphs with constant Δ take $\Omega(\log\log n)$ time in RandLOCAL (with error probability $1/\text{poly}(n)$) such as *sinkless orientation*, *sinkless coloring*, and Δ-vertex coloring. Refer to Section 2.3 for definitions of these problems. Since the existence of a sinkless orientation can be proved with the Lovász local lemma (LLL), this gave $\Omega(\log\log n)$ lower bounds on distributed algorithms for the constructive LLL. See [44, 68] for upper bounds on the distributed LLL.

Organization. In Section 2.2, we establish a derandomization result that leads to the necessity of graph shattering. In Section 2.3, we prove that Δ-vertex coloring takes $\Omega(\log_\Delta \log n)$ time in RandLOCAL and $\Omega(\log_\Delta n)$ time in DetLOCAL. In Section 2.4, we prove the DetLOCAL $\omega(\log^* n)$—$o(\log n)$ gap and the RandLOCAL $\omega(\log^* n)$—$o(\log\log n)$ gap. In Section 2.5, we prove the RandLOCAL $\omega(T_{\text{LLL}})$—$o(\log n)$ gap. In Section 2.6, we review Naor and Stockmeyer's characterization of $O(1)$-time LCL algorithms, using Ramsey theory, and explain how it implies gaps in the complexity hierarchy that depend on the graph topology.

[6] Linial [107] actually only used the existence of Δ-regular graphs with high girth and chromatic number $\Omega(\sqrt{\Delta})$. See [28] for constructions with chromatic number $\Omega(\Delta/\log\Delta)$.

2.1.1 New Results on the Value of Random Bits

We exhibit an exponential separation between RandLOCAL and DetLOCAL for several *specific* symmetry breaking problems. More generally, we give new connections between the randomized and deterministic complexities of all *locally checkable labeling* problems, a class that includes essentially any natural symmetry breaking problem.

Separation of RandLOCAL and DetLOCAL. We extend Brandt et al.'s [31] randomized lower bound as follows: Δ-vertex coloring Δ-regular graphs takes $\Omega(\log_\Delta \log n)$ time in RandLOCAL and $\Omega(\log_\Delta n)$ time in DetLOCAL. The hard graphs in this lower bound have girth $\Omega(\log_\Delta n)$, so by the indistinguishability principle, these lower bounds also apply to Δ-vertex coloring trees. Combining this new $\Omega(\log_\Delta n)$ lower bound with the known $O(\log_\Delta \log n + \log^* n)$-time algorithm in RandLOCAL from [129], we obtain an exponential separation of RandLOCAL and DetLOCAL for the problem of Δ-vertex coloring trees.

Graph Shattering is Necessary. We prove the following *derandomization* result. The RandLOCAL complexity for any LCL problem on instances of size n is at least its DetLOCAL complexity on instances of size $\sqrt{\log n}$. This *reverses* the implication proved above. For example, if we begin with a proof that Δ-vertex coloring takes $\Omega(\log_\Delta n)$ time in DetLOCAL, then we conclude that it must take $\Omega(\log_\Delta \log n)$ time in RandLOCAL.

2.1.2 New Results on the Complexity Gaps

We prove that there exist large "gaps" in the spectrum of possible complexities in DetLOCAL and RandLOCAL. More specifically, we establish the following three gaps for bounded-degree general graphs.[7]

- DetLOCAL $\omega(\log^* n)$—$o(\log n)$ gap.
- RandLOCAL $\omega(\log^* n)$—$o(\log \log n)$ gap.
- RandLOCAL $\omega(T_{\text{LLL}})$—$o(\log n)$ gap.

The notion T_{LLL} refers to the complexity of the distributed LLL problem.

[7] The $\omega(\log n)$—$n^{o(1)}$ gap on bounded degree trees is left to Chapter 3.

The DetLOCAL $\omega(\log^* n)$—$o(\log n)$ Gap. The proof for the DetLOCAL $\omega(\log^* n)$—$o(\log n)$ gap actually shows that any $f(\Delta) + o(\log_\Delta n)$ time algorithm for an LCL problem can be transformed in a black box way to run in $O(f(\Delta)(\log^* n - \log^* \Delta + 1))$ time (when $f(\Delta) \geq 1$). Thus, on bounded-degree graphs, there are no deterministic time bounds between $\omega(\log^* n)$ and $o(\log n)$. Any $\omega(\log^* n)$ lower bound for bounded degree graphs (in either RandLOCAL or DetLOCAL) *immediately* implies an $\Omega(\log n)$ lower bound in DetLOCAL.

This reduction can be parameterized in many different ways. For example, if one were to develop a deterministic $O(\sqrt{\log n / \log\log n})$-time MIS or maximal matching algorithm—matching one of the KMW [102] lower bounds—it immediately implies an $O((\log \Delta / \log\log \Delta) \cdot (\log^* n - \log^* \Delta + 1))$-time MIS/maximal matching algorithm, which almost matches the *other* KMW lower bound. We show that any $O(\log^{1-\frac{1}{k+1}} n)$-time DetLOCAL algorithm for an LCL problem can be transformed to run in $O(\log^k \Delta(\log^* n - \log^* \Delta + 1))$-time.

The RandLOCAL $\omega(\log^* n)$—$o(\log\log n)$ Gap. Combining the above reduction with the proof of the necessity of graph shattering, we obtain the RandLOCAL $\omega(\log^* n)$—$o(\log\log n)$ gap. Any randomized $O(1) + o(\log_\Delta \log n)$-round algorithm can be *derandomized* to run in deterministically $O(1)+o(\log_\Delta n)$ rounds, and hence can be transformed to run in $O(\log^* n)$ rounds.

The RandLOCAL $\omega(T_{\text{LLL}})$—$o(\log n)$ Gap. The RandLOCAL $\omega(T_{\text{LLL}})$—$o(\log n)$ gap is obtained by a new method that is able to solve any sub-logarithmic time problem using a distributed LLL algorithm. Roughly speaking, the proof shows that any sub-logarithmic time RandLOCAL algorithm $\mathcal{A}$ can be re-implemented in a canonical way using the LLL.

Speedup Implications of Naor & Stockmeyer. We also revisit Naor and Stockmeyer's characterization of $O(1)$-time LOCAL algorithms for LCL problems [116] and calculate the complexity gaps that are directly implied by their proof. For n-paths/cycles we see a $\omega(1)$—$o(\log^* n)$ complexity gap, for $(\sqrt{n} \times \sqrt{n})$-grids/tori an $\omega(1)$—$o(\sqrt{\log^* n})$ gap, and for bounded-degree trees and bounded-degree general graphs, an $\omega(1)$—$o(\log(\log^* n))$

complexity gap.

$\omega(1)$—$o(\log^* n)$	for n-paths/cycles.
$\omega(1)$—$o\left(\sqrt{\log^* n}\right)$	for $(\sqrt{n} \times \sqrt{n})$-grids/tori.
$\omega(1)$—$o(\log(\log^* n))$	for bounded degree trees or bounded degree general graphs.

These gaps hold in both DetLOCAL and RandLOCAL.

2.2 The Necessity of Graph Shattering

Theorem 2.1 establishes that the graph shattering technique [23] is optimal and unavoidable in RandLOCAL. In particular, the randomized complexity of any symmetry breaking problem always hinges on its deterministic complexity.

Theorem 2.1. *Let $\mathcal{P}$ be an LCL problem. Define $\mathsf{Det}_{\mathcal{P}}(n, \Delta)$ to be the complexity of the optimal deterministic algorithm for $\mathcal{P}$ in the* DetLOCAL *model and define $\mathsf{Rand}_{\mathcal{P}}(n, \Delta)$ to be its complexity in the* RandLOCAL *model, with global error probability $1/n$. Then, for sufficiently large n,*

$$\mathsf{Det}_{\mathcal{P}}(n, \Delta) \;\leq\; \mathsf{Rand}_{\mathcal{P}}(2^{n^2}, \Delta).$$

Proof. Let $\mathcal{A}_{\mathsf{Rand}}$ be a randomized algorithm for $\mathcal{P}$. Each vertex running $\mathcal{A}_{\mathsf{Rand}}$ generates a string of $r(n, \Delta)$ random bits and proceeds for $t(n, \Delta)$ rounds, where r and t are two arbitrary functions. The probability that the algorithm fails in any way is at most $1/n$. Our goal is to convert $\mathcal{A}_{\mathsf{Rand}}$ into a deterministic algorithm $\mathcal{A}_{\mathsf{Det}}$ in the DetLOCAL model. Let $G = (V, E)$ be the network on which $\mathcal{A}_{\mathsf{Det}}$ runs. Initially each $v \in V$ knows $n = |V|, \Delta$, and a unique $\mathrm{ID}(v) \in \{0,1\}^{c \log n}$. Let $\mathcal{G}_{n,\Delta}$ be the set of *all* n-vertex graphs with unique vertex IDs in $\{0,1\}^{c \log n}$ and maximum degree at most Δ. Since c is a constant, for sufficiently large n,

$$|\mathcal{G}_{n,\Delta}| \leq 2^{\binom{n}{2} + cn \log n} \ll 2^{n^2} \stackrel{\text{def}}{=} N,$$

regardless of Δ.

Imagine simulating $\mathcal{A}_{\mathsf{Rand}}$ on a graph $G' \in \mathcal{G}_{n,\Delta}$ whose vertices are given input parameters (N, Δ), that is, we imagine G' is disconnected from the remaining $N - n$ vertices. The probability that $\mathcal{A}_{\mathsf{Rand}}$ fails on an N-vertex graph is at most $1/N$, so the probability that any vertex in G' witnesses a failure is also certainly at most $1/N$.

Suppose we select a function $\phi : \{0,1\}^{c \log n} \to \{0,1\}^{r(N,\Delta)}$ uniformly at random from the space of all such functions. Define $\mathcal{A}_{\mathsf{Det}}[\phi]$ to be the *deterministic* algorithm that simulates $\mathcal{A}_{\mathsf{Rand}}$ for $t(N,\Delta)$ steps, where the string of random bits generated by v is fixed to be $\phi(\mathrm{ID}(v))$. We shall call ϕ a *bad* function if $\mathcal{A}_{\mathsf{Det}}[\phi]$ fails to compute the correct answer on some member of $\mathcal{G}_{n,\Delta}$. By the union bound,

$$\begin{aligned}
\Pr_{\phi}(\phi \text{ is bad}) &\leq \sum_{G' \in \mathcal{G}_{n,\Delta}} \Pr_{\phi}(\mathcal{A}_{\mathsf{Det}}[\phi] \text{ errs on } G') \\
&= \sum_{G' \in \mathcal{G}_{n,\Delta}} \Pr(\mathcal{A}_{\mathsf{Rand}} \text{ errs on } G', \text{ with input parameters } (N,\Delta)) \\
&\leq |\mathcal{G}_{n,\Delta}| / N \ < \ 1.
\end{aligned}$$

Thus, there exists some good ϕ. Any ϕ can be encoded as a long bit-string $\langle \phi \rangle \stackrel{\text{def}}{=} \phi(0)\phi(1)\cdots\phi(2^{c \log n} - 1)$. Define $\phi^\star$ to be the good function for which $\langle \phi^\star \rangle$ is lexicographically first.

The algorithm $\mathcal{A}_{\mathsf{Det}}$ is as follows. Each vertex v, given input parameters (n,Δ), first computes $N = 2^{n^2}, t(N,\Delta), r(N,\Delta)$, then performs the simulations of $\mathcal{A}_{\mathsf{Rand}}$ necessary to compute $\phi^\star$. Once $\phi^\star$ is computed it executes $\mathcal{A}_{\mathsf{Det}}[\phi^\star]$ for $t(N,\Delta)$ rounds. By definition, $\mathcal{A}_{\mathsf{Det}}[\phi^\star]$ never errs when run on any member of $\mathcal{G}_{n,\Delta}$. □

Remark 2.1. *Theorem 2.1 works equally well when t and r are functions of n, Δ, and possibly other quantitative global graph parameters. For example, the time may depend on measures of local sparsity (as in [55]), arboricity/degeneracy (as in [19, 23]), or neighborhood growth (as in [136]).*

Remark 2.2. *The role of the LCL assumption in the proof of Theorem 2.1 is to make sure that $\mathcal{P}$ does not depend on n. This rules out some silly tasks. For example, if $\mathcal{P}$ were the task that asks each vertex v to report the number of vertices in $N^{\sqrt{\log n}}(v)$, then clearly $\mathcal{P}$ is not an LCL, and the* RandLOCAL *and* DetLOCAL *complexities of $\mathcal{P}$ are both $\Theta(\sqrt{\log n})$.*

Naor and Stockmeyer [116] proved that the class of truly local ($O(1)$-time) problems in RandLOCAL and DetLOCAL is identical for bounded Δ. Theorem 2.1 offers a slight improvement over the Naor-Stockmeyer derandomization, since $\log^* n$ and $\log^*(\sqrt{\log n})$ differ by a constant.

Corollary 2.1. *Any* RandLOCAL *algorithm for an LCL taking* $t(n) = 2^{O(\log^* n)}$ *time can be derandomized without asymptotic penalty. The corresponding* DetLOCAL *algorithm runs in* $O(t(n))$ *time.*

2.3 Lower bounds for Δ-vertex coloring Δ-regular Trees

In this section we prove that on Δ-regular graphs with girth $\Omega(\log_\Delta n)$, Δ-vertex coloring takes $\Omega(\log_\Delta \log n)$ time in RandLOCAL and $\Omega(\log_\Delta n)$ time in DetLOCAL. Since the girth of the graphs used to prove these lower bounds is $\Omega(\log_\Delta n)$, by the indistinguishability principle they also apply to the problem of Δ-vertex coloring trees.

Brandt et al. [31] considered the following problems.

Δ-Sinkless Coloring. Given a Δ-regular graph $G = (V, E)$ and a proper Δ-edge coloring of E using colors in $\{1, 2, \ldots, \Delta\}$, find a Δ-vertex coloring of V using colors in $\{1, 2, \ldots, \Delta\}$ such that there is no edge $\{u, v\} \in E$ for which u, v, and $\{u, v\}$ all have the same color.

Δ-Sinkless Orientation. Given a Δ-regular graph $G = (V, E)$ and a proper Δ-edge coloring of E, find an orientation of the edges such that all vertices have out-degree ≥ 1.

Observe that both Δ-Sinkless Coloring and Δ-Sinkless Orientation are LCL graph problems with $r = 1$. For Sinkless Orientation $\Sigma = \{\rightarrow, \leftarrow\}^\Delta$ encodes the directions of all edges incident to a vertex, and the radius $r = 1$ is necessary and sufficient to verify that the orientations declared by both endpoints of an edge are consistent.

Brandt et al. [31] proved $\Omega(\log \log n)$ lower bounds on RandLOCAL algorithms that have a $1/\text{poly}(n)$ probability of failure, for sinkless coloring and sinkless orientation of 3-regular graphs. We say that a sinkless coloring algorithm $\mathcal{A}$ has failure probability p if, for *each* individual edge $e = \{u, v\}$, the probability that $\text{Color}^\star(u) = \text{Color}^\star(v) = \text{Color}^\star(\{u, v\})$ is at most p. Thus, by the union bound, the *global* probability of failure is at most $p|E|$. We say a that sinkless orientation algorithm $\mathcal{A}$ has failure probability p if, for each $v \in V$, the probability that v is a sink is at most p. We say that monochromatic edges and sinks are *forbidden configurations* for sinkless coloring and sinkless orientation, respectively.

The following two lemmas are proven in [31] for $\Delta = 3$. It is straightforward to go through the details of the proof and track the dependence on Δ.

Lemma 2.1 ([31])**.** *Let $G = (V, E, \psi)$ be a Δ-regular graph with girth g that is equipped with a proper Δ-edge coloring ψ. Suppose that there is a* RandLOCAL *algorithm $\mathcal{A}$ for Δ-sinkless coloring taking $t < \frac{g-1}{2}$ rounds such that for each $e \in E$, $\mathcal{A}$ outputs a forbidden configuration at e with probability at most p. Then there is a* RandLOCAL *algorithm $\mathcal{A}'$ for Δ-sinkless orientation taking t rounds such that for each $v \in V$, $\mathcal{A}'$ outputs a forbidden configuration at v with probability at most $2\Delta p^{1/3}$.*

Lemma 2.2 ([31])**.** *Let $G = (V, E, \psi)$ be a Δ-regular graph with girth g that is equipped with a proper Δ-edge coloring ψ. Suppose that there is a* RandLOCAL *algorithm $\mathcal{A}'$ for sinkless orientation taking $t < \frac{g-1}{2}$ rounds such that for each $v \in V$, $\mathcal{A}'$ outputs a forbidden configuration at v with probability at most p. Then there is a* RandLOCAL *algorithm $\mathcal{A}$ for Δ-sinkless coloring taking $t - 1$ rounds such that for each $e \in E$, $\mathcal{A}$ outputs a forbidden configuration at e with probability at most $4p^{1/(\Delta+1)}$.*

The following theorem generalizes Corollary 25 in [31] to allow non-constant Δ and arbitrary failure probability p.

Theorem 2.2. *Any* RandLOCAL *algorithm for Δ-vertex coloring a graph with degree at most Δ and error probability p takes at least $t = \min\{\epsilon \log_{3(\Delta+1)} \ln(1/p),\ \epsilon \log_\Delta n\} - 1$ rounds for a sufficiently small constant $\epsilon > 0$.*

Proof. We assume that $\epsilon \log_{3(\Delta+1)} \ln(1/p) \geq 1$, since otherwise the theorem is trivial as $t < 0$. For any $\Delta \geq 3$ there exists a bipartite Δ-regular graph with girth $\Omega(\log_\Delta n)$; see [29, 48]. Such graphs are trivially Δ-edge colorable. Moreover, any Δ-vertex coloring of such a graph is also a valid Δ-sinkless coloring. Applying Lemmas 2.1 and 2.2 we conclude that any t-round Δ-sinkless coloring algorithm with error probability p can be transformed into a $(t-1)$-round Δ-sinkless coloring algorithm with error probability $4(2\Delta)^{\frac{1}{\Delta+1}} p^{\frac{1}{3(\Delta+1)}} < 7p^{\frac{1}{3(\Delta+1)}}$. Iterating this process t times, it follows that there exists a 0-round Δ-sinkless coloring algorithm with failure probability $O(p^{(\frac{1}{3(\Delta+1)})^t})$. Notice that

$$p^{(\frac{1}{3(\Delta+1)})^t} \leq p^{(\frac{1}{3(\Delta+1)})^{\epsilon \log_{3(\Delta+1)} \ln(1/p)}} = p^{(\ln(1/p))^{-\epsilon}} = \exp(-(\ln(1/p))^{1-\epsilon}).$$

Because the graph is Δ-regular and the vertices undifferentiated by IDs, any 0-round RandLOCAL algorithm colors each vertex independently according to the same distribution.

The probability that any vertex is involved in a forbidden configuration (a monochromatic edge) is therefore at least $1/\Delta^2$. Since $\epsilon \log_{3(\Delta+1)} \ln(1/p) \geq 1$ we have $\Delta < \ln(1/p)$, but

$$\frac{1}{\Delta^2} \geq \exp(-2 \ln \ln(1/p)) \gg \exp\left(-(\ln(1/p))^{1-\epsilon}\right).$$

This is a contradiction since we obtain a 0-round Δ-sinkless coloring algorithm with failure probability less than $1/\Delta^2$. Thus, there is no RandLOCAL Δ-sinkless coloring algorithm that takes t-rounds and errs with probability p. □

Corollary 2.2 is an immediate consequence of Theorem 2.2.

Corollary 2.2. *Any* RandLOCAL *algorithm for Δ-vertex coloring a graph with global error probability* $1/\mathrm{poly}(n)$ *takes* $\Omega(\log_\Delta \log n)$ *time.*

Theorem 2.2 does not immediately extend to DetLOCAL. It is tempting to feel that setting $p = 0$ yields a $\Omega(\log_\Delta n)$ DetLOCAL lower bound. But this is not a correct inference. Recall that in the DetLOCAL model vertices are initially endowed with $O(\log n)$-bit IDs whereas in RandLOCAL they are undifferentiated, and the naive way of generating such IDs in RandLOCAL has failure probability $1/\mathrm{poly}(n)$.

Theorem 2.3. *Any* DetLOCAL *algorithm that Δ-vertex colors degree-Δ graphs with girth* $\Omega(\log_\Delta n)$ *or degree-Δ trees requires* $\Omega(\log_\Delta n)$ *time.*

Proof. Let $\mathcal{A}_{\mathsf{Det}}$ be a DetLOCAL algorithm that Δ-vertex colors a graph in $t = t(n, \Delta)$ rounds and G be the input graph. We construct a RandLOCAL algorithm $\mathcal{A}_{\mathsf{Rand}}$ taking $O(t)$ rounds as follows. Before the first round each vertex locally generates a random n-bit ID. Assume for the time being that these IDs are unique, and therefore constitute a 2^n-vertex coloring of G. Let $G' = (V, \{\{u, v\} \mid \mathrm{dist}_G(u, v) \leq 2t + 1\})$. The maximum degree Δ' in G' is clearly less than n.

We explain the choice of the parameter $2t+1$. Whether an edge $e = \{u, v\}$ is monochromatic depends on the colors of u and v, and this depends on the graph topology and the IDs in the subgraph induced by $N^t(u) \cup N^t(v)$. If all vertices in $N^t(u) \cup N^t(v)$ have distinct IDs, u and v must be colored differently. Notice that the maximum distance within $N^t(u) \cup N^t(v)$ is $2t + 1$. In general, for LCLs of radius r, a deterministic algorithm works correctly as long as all vertices within distance $2t + 2r$ have distinct IDs.

We apply one step of Linial's recoloring algorithm of Theorem 1.1 to G' and obtain a coloring with palette size $O(\Delta'^2 \log(2^n)) = O(n^3)$. A step of Linial's algorithm in G' is

simulated in G using $O(t)$ time. Using these colors as $(3\log n + O(1))$-bit IDs, we simulate $\mathcal{A}_{\mathsf{Det}}$ in G for t steps. Since no vertex can see two vertices with the same ID, this algorithm necessarily behaves as if all IDs are unique. Observe that because $\mathcal{A}_{\mathsf{Det}}$ is deterministic, the only way $\mathcal{A}_{\mathsf{Rand}}$ can err is if the initial n-bit IDs fail to be unique. This occurs with probability $p < n^2/2^n$. By Theorem 2.2, $\mathcal{A}_{\mathsf{Rand}}$ takes $\Omega(\min\{\log_\Delta \log(1/p), \log_\Delta n\}) = \Omega(\log_\Delta n)$ time. □

2.4 Gaps in Deterministic and Randomized Time Complexity

The Time Hierarchy Theorem informally says that a Turing machine can solve more problems given more time. A similar question can be asked in the setting of distributed computation. For example, are there natural or contrived problems with DetLOCAL complexity $\Theta((\log^* n)^2)$, $\Theta(\log\log n)$, or $\Theta(\sqrt{\log n})$, when $\Delta = O(1)$? In this section, we demonstrate a general technique that allows one to speedup deterministic algorithms in the DetLOCAL model. Based on this technique, we demonstrate the existence of a "gap" in possible DetLOCAL and RandLOCAL complexities, answering the above question in the negative.

A graph class is *hereditary* if it is closed under removing vertices and edges. Examples of hereditary graph classes are general graphs, forests, bounded arboricity graphs, triangle-free graphs, and planar graphs. We prove that, for graphs with constant Δ, the DetLOCAL complexity of *any* LCL problem on a *hereditary* graph class is either $\Omega(\log n)$ or $O(\log^* n)$. Moreover, if the hereditary graph class is also closed under taking disjoint union, then the RandLOCAL complexity of *any* LCL problem is either $\Omega(\log\log n)$ or $O(\log^* n)$.

Notations. Throughout this section, β is the universal constant from Theorem 1.1 (Linial's $\beta\Delta^2$-vertex coloring algorithm). We only consider LCLs without input labels, we write $\Sigma_{\text{out}} = \Sigma$. All results can be extended to LCLs with input labels in a straightforward manner.

Theorem 2.4. *Let $\mathcal{P}$ be an LCL graph problem with parameters r, Σ, and $\mathcal{C}$, and let $\mathcal{A}$ be a* DetLOCAL *algorithm for solving $\mathcal{P}$. Suppose that the runtime of $\mathcal{A}$ on an n-vertex graph taken from a hereditary graph class is at most $f(\Delta) + \epsilon\log_\Delta n$, where $f(\Delta) \geq 1$ and $\epsilon = \frac{1}{4+4\log\beta+4r}$ is a constant. Then there exists a* DetLOCAL *algorithm $\mathcal{A}'$ that solves $\mathcal{P}$ on the same instances in $O\left(f(\Delta)(\log^* n - \log^*\Delta + 1)\right)$ time.*

Proof. Notice that for any instance of $\mathcal{P}$ with n vertices and ID length ℓ, it must be that $\ell \geq \log n$ and so the running time of $\mathcal{A}$ on such instances is bounded by $T(\Delta, \ell) \leq f(\Delta) + \frac{\epsilon \ell}{\log \Delta}$.

Let $G = (V, E)$ be a graph in a hereditary graph class. The algorithm $\mathcal{A}'$ on G works as follows. Let $\tau = 1 + \log \beta$ be a constant. We use Linial's coloring technique to produce short IDs of length ℓ' that are distinct within distance $4f(\Delta) + 2\tau + 2r$. Let $G' = (V, E')$ be the graph with

$$E' = \left\{ \{u, v\} \in V^2 \;\middle|\; \mathrm{dist}_G(u, v) \leq 4f(\Delta) + 2\tau + 2r, \text{ and } u \neq v \right\}.$$

The maximum degree in G' is clearly at most $\Delta^{4f(\Delta)+2\tau+2r}$. Each vertex $u \in V$ simulates G' by collecting $N^{4f(\Delta)+2\tau+2r}(u)$ in $O(f(\Delta) + \tau + r)$ time.

We simulate the algorithm of Theorem 1.1 on G' by treating each of the ℓ-bit IDs of vertices in V as a color. This produces a $\beta \cdot \Delta^{8f(\Delta)+4\tau+4r}$-coloring, which is equivalent to identifiers of length $\ell' = (8f(\Delta) + 4\tau + 4r) \log \Delta + \log \beta$. Although these identifiers are not globally unique, they are distinct in $N^{2f(\Delta)+\tau+r}(u)$ for each vertex $u \in V$. The time complexity of this process is

$$(4f(\Delta) + 2\tau + 2r) \cdot O\left(\log^* n - \log^* \Delta + 1\right).$$

Finally, we apply $\mathcal{A}$ on G while implicitly assuming that the graph size is $2^{\ell'}$ and using the shorter IDs. The runtime of this execution of $\mathcal{A}$ is:

$$\begin{aligned}
f(\Delta) + \frac{\epsilon \ell'}{\log \Delta} &= f(\Delta) + \frac{\epsilon((8f(\Delta) + 4\tau + 4r) \log \Delta + \log \beta)}{\log \Delta} \\
&= (1 + 8\epsilon) f(\Delta) + 1 + \frac{\epsilon \log \beta}{\log \Delta} && \epsilon(4\tau + 4r) = 1 \\
&\leq (1 + 8\epsilon) f(\Delta) + \tau && \log \Delta \geq 1, \epsilon < 1 \\
&\leq 2f(\Delta) + \tau. && 8\epsilon = \frac{2}{\tau + r} \leq 1
\end{aligned}$$

Whether the output labeling of $u \in V$ is legal depends on the labeling of the vertices in $N^r(u)$, which depends on the graph structure and the IDs in $N^{2f(\Delta)+\tau+r}(u)$. Due to the hereditary property of the graph class under consideration, for each $u \in V$, $N^{2f(\Delta)+\tau+r}(u)$ is isomorphic to a subgraph of some $2^{\ell'}$-vertex graph in the same class. Moreover, the shortened IDs in $N^{2f(\Delta)+\tau+r}(u)$ are distinct. Therefore, it is guaranteed that the output of the simulation is a legal labeling.

The total time complexity is

$$\begin{aligned}&(4f(\Delta)+2\tau+2r)\cdot O(\log^* n-\log^*\Delta+1)+2f(\Delta)+\tau\\&=O\left(f(\Delta)(\log^* n-\log^*\Delta+1)\right).\end{aligned}$$

□

Combining Theorem 2.4 with Corollary 2.2 and setting $f(\Delta)=O(1)$ provides a new proof of Theorem 2.3 for small enough Δ. To see this, notice that any lower bound for the RandLOCAL model with error probability $1/\text{poly}(n)$ can be adapted to DetLOCAL since we can randomly pick $O(\log n)$-bit IDs that are distinct with probability $1-1/\text{poly}(n)$. From Theorem 2.2 any DetLOCAL algorithm that Δ-colors a degree-Δ tree requires $\Omega(\log_\Delta \log n)$ time. However, Theorem 2.4 states that any DetLOCAL algorithm running in $O(1)+o(\log_\Delta n)$ time can be sped up to run in $O\left(\log^* n-\log^*\Delta+1\right)$ time. This contradicts the lower bound whenever $\log_\Delta \log n \gg \log^* n-\log^*\Delta+1$. Hence Δ-coloring a degree-Δ tree takes $\Omega(\log_\Delta n)$ time in DetLOCAL for small enough Δ such that $\log_\Delta \log n \gg \log^* n-\log^*\Delta+1$.

Another consequence of Theorem 2.4 is that the deterministic time complexity of a problem can either be solved very efficiently (i.e. in $O\left(f(\Delta)(\log^* n-\log^*\Delta+1)\right)$ time) or requires $\Omega(f(\Delta)+\log_\Delta n)$ time, which is at least the order of the diameter when the underlying graph is a complete regular tree. Such a consequence is the strongest when Δ is small. For example, if Δ is a constant, Theorem 2.4 implies the following corollary.

Corollary 2.3. *The* DetLOCAL *complexity of any LCL problem on any hereditary graph class that has constant* Δ *is either* $\Omega(\log n)$ *or* $O(\log^* n)$.

A simple adaptation of the proof of Theorem 2.4 shows an even stronger dichotomy when $\Delta=2$.

Theorem 2.5. *The* DetLOCAL *time complexity of any LCL problem on any hereditary graph class with* $\Delta=2$ *is either* $\Omega(n)$ *or* $O(\log^* n)$.

We remark that an intuitive explanation of the time complexity requirement in Theorems 2.4 and 2.5 is that the diameter of a graph with maximum degree Δ is at least $\Omega(\log_\Delta n)$ for $\Delta\geq 3$ and $\Omega(n)$ when $\Delta=2$. To ensure that the algorithm is indeed "local", it must not explore the entire graph.

Combining Theorem 2.4 and Theorem 2.1 also yields a gap in the complexities of the RandLOCAL model.

Theorem 2.6. *Let $\mathcal{P}$ be an LCL graph problem with parameters r, Σ, and $\mathcal{C}$, and let $\mathcal{A}$ be a* RandLOCAL *algorithm for solving $\mathcal{P}$. Let $\mathcal{G}$ be a hereditary graph class that is closed under taking disjoint union. Suppose that the runtime of $\mathcal{A}$ on an n-vertex graph taken from $\mathcal{G}$ is at most $f(\Delta) + (\epsilon/2)\log_\Delta \log n$, where $f(\Delta) \geq 1$ and $\epsilon = \frac{1}{4+4\log\beta+4r}$ is a constant. Then there exists a* DetLOCAL *algorithm $\mathcal{A}'$ that solves $\mathcal{P}$ on the same instances in $O\left(f(\Delta)(\log^* n - \log^* \Delta + 1)\right)$ time.*

Proof. Since $\mathcal{G}$ is closed under taking disjoint union, by Theorem 2.1, for sufficiently large n, the deterministic complexity of $\mathcal{P}$ is at most $f(\Delta) + (\epsilon/2)\log_\Delta \log 2^{n^2} = f(\Delta) + \epsilon \log_\Delta n$. Since $\mathcal{G}$ is a hereditary graph class, by Theorem 2.4, there exists a DetLOCAL algorithm $\mathcal{A}'$ that solves $\mathcal{P}$ in $O\left(f(\Delta)(\log^* n - \log^* \Delta + 1)\right)$ time. □

Setting $\Delta = O(1)$ gives us the following corollary.

Corollary 2.4. *Let $\mathcal{G}$ be any hereditary graph class with a constant degree bound Δ that is closed under taking disjoint union. The* RandLOCAL *complexity of any LCL problem on $\mathcal{G}$ is either $\Omega(\log\log n)$ or $O(\log^* n)$. Moreover, any $O(\log^* n)$ algorithm can be implemented in* DetLOCAL*.*

Given a $O(\sqrt{\log n})$-time deterministic algorithm, one may feel that it is possible to use Theorem 2.4 to improve the time complexity to $O(\log^* n)$ since $\sqrt{\log n} = o(\log_\Delta n)$ for the case $\Delta = \exp(o(\sqrt{\log n}))$. However, the class of graphs with $\Delta = \exp(o(\sqrt{\log n}))$ is not hereditary, and so Theorem 2.4 does not apply. Nonetheless, Linial's coloring technique can be made to speed up algorithms with time complexity of the form $f(\Delta) + g(n)$.

Theorem 2.7. *Let $\mathcal{P}$ be an LCL graph problem with parameters r, Σ, and $\mathcal{C}$, and let $\mathcal{A}$ be a* DetLOCAL *algorithm for solving $\mathcal{P}$. Suppose that the runtime of the algorithm $\mathcal{A}$ on an n-vertex graph taken from a hereditary graph class is at most $O(\log^k \Delta + \log^{\frac{k}{k+1}} n)$. Then there exists a deterministic algorithm $\mathcal{A}'$ that solves $\mathcal{P}$ on the same instances in $O(\log^k \Delta(\log^* n - \log^* \Delta + 1))$ time.*

Proof. Let the ID length be ℓ, then it must be that $\ell \geq \log n$ and so the running time of $\mathcal{A}$ on such instances is bounded by $\epsilon_1 \log^k \Delta + \epsilon_2 \ell^{\frac{k}{k+1}}$, for some constants ϵ_1, ϵ_2.

We set $\tau = \epsilon \log^k \Delta$, with the parameter ϵ to be determined. Similar to the proof of Theorem 2.4, the algorithm $\mathcal{A}'$ first produces shortened ID that are distinct for vertices within distance $2\tau + 2r$, and then simulates $\mathcal{A}$ on the shortened IDs in τ rounds.

Let $G' = (V, E')$ be the graph with

$$E' = \left\{ \{u, v\} \in V^2 \;\middle|\; \mathrm{dist}_G(u, v) \leq 2\tau + 2r, \text{ and } u \neq v \right\}.$$

The maximum degree in G' is at most $\Delta^{2\tau+2r}$. Each vertex $u \in V$ simulates G' by collecting $N^{2\tau+2r}(u)$ in $O(\tau + r)$ time.

We simulate the algorithm of Theorem 1.1 on G' by treating each of the ℓ-bit IDs of vertices in V as a color. This produces a $\beta \cdot \Delta^{4\tau+4r}$-coloring, which is equivalent to identifiers of length $\ell' = (4\tau + 4r) \log \Delta + \log \beta$. Although these identifiers are not globally unique, they are distinct in $N^{\tau+r}(u)$ for each vertex $u \in V$. The time complexity of this process is

$$(2\tau + 2r) \cdot O\left(\log^* n - \log^* \Delta + 1\right).$$

Finally, we apply $\mathcal{A}$ on G while implicitly assuming that the graph size is $2^{\ell'}$ and using the shorter IDs. By setting ϵ as a large enough number such that $\epsilon_1 + \epsilon_2 \left(4(\epsilon + r + \log \beta)\right)^{\frac{k}{k+1}} \leq \epsilon$, the runtime of this execution of $\mathcal{A}$ is

$$\begin{aligned}
\epsilon_1 \log^k \Delta + \epsilon_2 \left(\ell'\right)^{\frac{k}{k+1}} &= \epsilon_1 \log^k \Delta + \epsilon_2 \left((4\tau + 4r) \log \Delta + \log \beta\right)^{\frac{k}{k+1}} \\
&\leq \epsilon_1 \log^k \Delta + \epsilon_2 \left(4(\epsilon \log^k \Delta + r + \log \beta) \log \Delta\right)^{\frac{k}{k+1}} \\
&\leq \epsilon_1 \log^k \Delta + \epsilon_2 \left(4(\epsilon + r + \log \beta) \log^{k+1} \Delta\right)^{\frac{k}{k+1}} \\
&= \left(\epsilon_1 + \epsilon_2 \left(4(\epsilon + r + \log \beta)\right)^{\frac{k}{k+1}}\right) \log^k \Delta \\
&\leq \epsilon \log^k \Delta \\
&= \tau.
\end{aligned}$$

Whether the output labeling of $u \in V$ is legal depends on the labeling of the vertices in $N^r(u)$, which depends on the graph structure and the IDs in $N^{\tau+r}(u)$. Due to the hereditary property of the graph class under consideration, for each $u \in V$, $N^{\tau+r}(u)$ is isomorphic to a subgraph of some $2^{\ell'}$-vertex graph in the same class. Moreover, the shortened IDs in $N^{\tau+r}(u)$ are distinct. Therefore, it is guaranteed that the output of the simulation is a legal labeling.

The total time complexity is at most

$$(2\tau + 2r) \cdot O(\log^* n - \log^* \Delta + 1) + \tau = O(\log^k \Delta (\log^* n - \log^* \Delta + 1)). \qquad \square$$

A Note about MIS Lower Bounds. Kuhn, Moscibroda, and Wattenhofer [102] demonstrated that for a variety of problems (including MIS) there is a lower bound of

$$\Omega(\min\{\log\Delta/\log\log\Delta, \sqrt{\log n/\log\log n}\})$$

rounds. The lower bound graph they used to prove such these result has $\log\Delta/\log\log\Delta = O(\sqrt{\log n/\log\log n})$. The proof framework of Theorem 2.7 can be used to show that if there is a deterministic algorithm $\mathcal{A}$ for MIS that runs in $O(\sqrt{\log n/\log\log n})$ time, then there is another deterministic algorithm $\mathcal{A}'$ running in $O(\log\Delta/\log\log\Delta)\cdot(\log^* n - \log^*\Delta + 1))$ time. Let the runtime of $\mathcal{A}$ be $\epsilon_1\sqrt{\ell/\log\ell}$ for ID length ℓ. Set $\tau = \epsilon\log\Delta/\log\log\Delta$. Then $\ell' = (4\tau + 4r)\log\Delta + \log\beta = \Theta(\log^2\Delta/\log\log\Delta)$. The runtime of $\mathcal{A}$ for ID length ℓ' is $\epsilon_1\sqrt{\ell/\log\ell} = \epsilon_1\cdot\sqrt{\epsilon/\log\epsilon}\cdot O(\log\Delta/\log\log\Delta) < \tau$, by choosing a large enough constant ϵ.

Interestingly, Barenboim, Elkin, Pettie, and Schneider [23] showed that an MIS algorithm in RandLOCAL running in $O(\log^k\Delta + f(n))$-time implied another RandLOCAL algorithm running in $O(\log^k\lambda + \log^{1-\frac{1}{k+1}} n + f(n))$ time on graphs of arboricity λ. This is analogous to Theorem 2.7 but in the reverse direction.

2.5 A Gap in the RandLOCAL Complexity Hierarchy

We *define* $T_{LLL}(n, d, c)$ to be the RandLOCAL time to compute a point in the probability space avoiding all bad events (w.h.p.), under a "polynomial" LLL criterion of the form

$$pd^c < 1. \tag{2.1}$$

It is conceivable that the distributed complexity of the LLL is sensitive to the criterion used and depends on c. However, for our purpose (Theorem 2.8), *any* constant c is enough. In the subsequent discussion, we slightly abuse the notation to denote $T_{LLL}(n, d)$ as the distributed complexity of the LLL, where c is allowed to be an arbitrary constant. Earlier results [31, 44] imply that $T_{LLL}(n, d)$ is $\Omega(\log_{\log(1/p)}\log n)$, $\Omega(\log^* n)$, and $O(\log_{1/epd^2} n)$.

In this section we prove an automatic speedup theorem for RandLOCAL sublogarithmic algorithms. We do *not* assume that $\Delta = O(1)$ in this section. Theorem 2.8 considers algorithms that run in "sublogarithmic" time in RandLOCAL. The term *sublogarithmic* is insufficiently detailed, for two reasons. First, asymptotic notation is not always well defined

when there are multiple free parameters (e.g., n and Δ). Second, and more importantly, the proof of Theorem 2.8 considers what happens when n gets very small, rather than $n \to \infty$. It is for these reasons that Theorem 2.8 assumes the running time can be written in a specific form.

Theorem 2.8. *Suppose that* A *is a* RandLOCAL *algorithm that solves some LCL problem* $\mathcal{P}$ *(w.h.p.), in* $T_\Delta(n)$ *time. For any sufficiently small constant* $\epsilon > 0$ *and some function* C*, suppose* $T_\Delta(n)$ *is upper bounded by* $C(\Delta) + \epsilon \log_\Delta n$*. It is possible to transform* A *into a new* RandLOCAL *algorithm* A' *that solves* $\mathcal{P}$ *(w.h.p.) in* $O(C(\Delta) \cdot T_{LLL}(n, \Delta^{O(C(\Delta))}))$ *time.*

Proof. Suppose that A has a local probability of failure $1/n$, that is, for any $v \in V(G)$, the probability that $N^r(v)$ is inconsistent with $\mathcal{P}$ is $1/n$, where r is the radius of $\mathcal{P}$. Once we settle on the LLL criterion exponent c in (2.1), we fix $\epsilon = O((2c)^{-1})$. Define $n^\star$ as the minimum value for which

$$t^\star = T_\Delta(n^\star) < (1/2c) \cdot \log_\Delta n^\star - r.$$

It follows that $t^\star = O(C(\Delta))$ and $n^\star = \Delta^{O(C(\Delta))}$.

The algorithm A' applied to an n-vertex graph G works as follows. Imagine an experiment where we run A, but lie to the vertices, telling them that "n" $= n^\star$. Any $v \in V(G)$ will see a $t^\star$-neighborhood $N^{t^\star}(v)$ that is consistent with some $n^\star$-vertex graph. However, the probability of the *bad event* that $N^r(v)$ is incorrectly labeled is $1/n^\star$, not $1/\mathrm{poly}(n)$, as desired. We now show that this system of bad events satisfies the LLL criterion (2.1). Define the following events, graph, and quantities:

$$\begin{aligned}
\mathcal{E}_v &: \text{the event that } N^r(v) \text{ is incorrectly labeled} \\
&\quad \text{according to } \mathcal{P} \\
\mathcal{X} &= \{\mathcal{E}_v \mid v \in V(G)\} && \text{the set of bad events} \\
G_{\mathcal{X}} &= (\mathcal{X}, \{(\mathcal{E}_u, \mathcal{E}_v) \mid N^{r+t^\star}(u) \cap N^{r+t^\star}(v) \neq \emptyset\}) && \text{the dependency graph} \\
d &\leq \Delta^{2(r+t^\star)} \\
p &= 1/n^\star
\end{aligned}$$

The event $\mathcal{E}_v$ is determined by the labeling of $N^r(v)$ and the label of each $v' \in N^r(v)$ is determined by $N^{t^\star}(v')$, hence $\mathcal{E}_v$ is determined by (the data stored in, and random bits generated by) vertices in $N^{r+t^\star}(v)$. Clearly $\mathcal{E}_v$ is independent of any $\mathcal{E}_u$ for which

$N^{r+t^\star}(u) \cap N^{r+t^\star}(v) = \emptyset$, which justifies the definition of the edge set of $G_{\mathcal{X}}$. Since the maximum degree in G is Δ, the maximum degree d in $G_{\mathcal{X}}$ is less than $\Delta^{2(r+t^\star)}$. By definition of A, $\Pr(\mathcal{E}_v) \leq 1/n^\star = p$. This system satisfies LLL criterion (2.1) since, by definition of $t^\star$,

$$pd^c = p\Delta^{2c(r+t^\star)} < (1/n^\star) \cdot n^\star = 1.$$

The algorithm A' now simulates a constructive LLL algorithm on $G_{\mathcal{X}}$ in order to find a labeling such that no bad event occurs. Since a virtual edge $(\mathcal{E}_u, \mathcal{E}_v)$ exists iff u and v are at distance at most $2(r + t^\star) = O(C(\Delta))$, any RandLOCAL algorithm in $G_{\mathcal{X}}$ can be simulated in G with $O(C(\Delta))$ slowdown. Thus, A' runs in $O(C(\Delta) \cdot T_{LLL}(n, \Delta^{O(C(\Delta))}))$ time. □

Theorem 2.8 shows that when $\Delta = O(1)$, $o(\log n)$-time RandLOCAL algorithms can be *sped up* to run in $O(T_{LLL}(n, O(1)))$ time. Another consequence of this same technique is that sublogarithmic RandLOCAL algorithms with *large messages* can be converted to (possibly slightly slower) algorithms with small messages. The statement of Theorem 2.9 reflects the use of a particular distributed LLL algorithm, namely [44, Corollary 1 and Algorithm 2]. It may be improvable using future distributed LLL technology.

The LLL algorithm of [44] works under the assumption that $epd^2 < 1$, and that each bad event $A \in \mathcal{X}$ is associated with a unique ID. The algorithm starts with a random assignment to the variables $\mathcal{V}$. In each iteration, let $\mathcal{F}$ be the set of bad events that occur under the current variable assignment; let $\mathcal{I}$ be the subset of $\mathcal{F}$ such that $A \in \mathcal{I}$ if and only if $\mathrm{ID}(A) < \mathrm{ID}(B)$ for each $B \in \mathcal{F}$ such that $\mathrm{vbl}(A) \cap \mathrm{vbl}(B) \neq \emptyset$. The next variable assignment is obtained by *resampling* all variables in $\bigcup_{A \in \mathcal{I}} \mathrm{vbl}(A)$. After $O(\log_{1/epd^2} n)$ iterations, no bad event occurs with probability $1 - 1/\mathrm{poly}(n)$.

Theorem 2.9. *Let* A *be a* $(C(\Delta) + \epsilon \log_\Delta n)$*-time* RandLOCAL *algorithm that solves some LCL problem* $\mathcal{P}$ *with high probability, where* $\epsilon > 0$ *is a sufficiently small constant. Each vertex locally generates* $r_\Delta(n)$ *random bits and sends* $m_\Delta(n)$*-bit messages. It is possible to transform* A *into a new* RandLOCAL *algorithm* A' *that solves* $\mathcal{P}$ *(w.h.p.) in* $O(\log_\Delta n)$ *time, where each vertex generates* $O(\log n + r_\Delta(\zeta) \cdot \log_\zeta n)$ *random bits, and sends* $O(\min\{\log(|\Sigma_{\mathrm{out}}|) \cdot \Delta^{O(1)} + m_\Delta(\zeta) + \zeta,\ r_\Delta(\zeta) \cdot \zeta\})$*-bit messages, where* $\zeta = \Delta^{O(C(\Delta))}$ *depends on* Δ*.*

Proof. We continue to use the notation and definitions from Theorem 2.8, and fix $c = 3$ in the LLL criterion (2.1). Since $d = \Omega(\Delta^{O(C(\Delta))}) = \Omega(\zeta)$ and we selected $t^\star$ w.r.t. $c = 3$ (i.e., LLL criterion $pd^3 < 1$), we have $1/epd^2 = \Omega(\zeta)$. If A' uses the LLL algorithm of [44],

each vertex $v \in V(G)$ will first generate an $O(\log n)$-bit unique identifier $\mathrm{ID}(\mathcal{E}_v)$ (which costs $O(\log n)$ random bits) and generate $r_\Delta(n^\star) \cdot O(\log_{1/epd^2} n) = O(r_\Delta(\zeta) \cdot \log_\zeta n)$ random bits throughout the computation. Thus, the total number of random bits per vertex is $O(\log n + r_\Delta(\zeta) \cdot \log_\zeta n)$.

In each resampling step of A', in order for v to tell whether $\mathcal{E}_v \in \mathcal{I}$, it needs the following information: (i) $\mathrm{ID}(\mathcal{E}_u)$ for all $u \in N^{2(r+t^\star)}(v)$, and (ii) whether $\mathcal{E}_u$ occurs under the current variable assignment, for all $u \in N^{2(r+t^\star)}(v)$. We now present two methods to execute one resampling step of A'; they both take $O(C(\Delta))$ time using a message size that depends on Δ but is independent of n. There are $O(\log_{1/epd^2} n) = O(\log_\zeta n) = O(\frac{\log_\Delta n}{C(\Delta)})$ resampling steps, so the total time is $O(\log_\Delta n)$, independent of the function C.

Method 1 Before the LLL algorithm proper begins, we do the following preprocessing step. Each vertex v gathers up all IDs and random bits in its $3(t^\star + r)$-neighborhood. This takes $O((\log n + r_\Delta(\zeta) \cdot \log_\zeta n) \cdot \zeta / b)$ time with b-bit messages (recall that $\Delta^{O(t^\star+r)} = \Delta^{O(C(\Delta))} = \zeta$). In particular, the runtime can be made $O(\log_\Delta n)$ if we set $b = O(r_\Delta(\zeta) \cdot \zeta)$.

During the LLL algorithm, each vertex u owns one random variable: an $r_\Delta(n^\star)$-bit string V_u. In order for v to tell whether $\mathcal{E}_u$ occurs for each $u \in N^{2(r+t^\star)}(v)$ under the current variable assignment, it only needs to know how many times each V_u, $u \in N^{3(r+t^\star)}(v)$, has been resampled. Whether the output labeling of $u \in N^{2(r+t^\star)}(v)$ is locally consistent depends on the output labeling of vertices in $N^r(u)$, which depends on the random bits and the graph topology within $N^{r+t^\star}(u) \subseteq N^{3(r+t^\star)}(v)$. Given the graph topology, IDs, and the random bits within $N^{3(r+t^\star)}(v)$, the vertex v can locally simulate A and decides whether $\mathcal{E}_v \in \mathcal{I}$.

Thus, in each iteration of the LLL algorithm, each vertex v simply needs to alert its $3(r + t^\star)$-neighborhood whether V_v is resampled or not. This can be accomplished in $O(r + t^\star) = O(C(\Delta))$ time with ζ-bit messages.

Method 2 In the second method, vertices keep their random bits private. Similar to the first method, we do a preprocessing step to let each vertex gathers up all IDs in its $2(t^\star + r)$-neighborhood. This can be done in $O(\log_\Delta n)$ time using ζ-bit messages.

During the LLL algorithm, in order to tell which subset of bad events $\{\mathcal{E}_v\}_{v \in V(G)}$ occur under the current variable assignment, all vertices simulate A for $t^\star$ rounds, sending $m_\Delta(n^\star)$-bit messages. After the simulation, for a vertex v to tell whether $\mathcal{E}_v$ occurs, it needs to gather the output labeling of the vertices in $N^r(v)$. This can be done in $r = O(1)$

rounds, sending $\log(|\Sigma_{\text{out}}|) \cdot \Delta^{O(1)}$-bit messages. Note that an output label can be encoded as a $\log(|\Sigma_{\text{out}}|)$-bit string. We do not assume that Δ is constant so $|\Sigma_{\text{out}}|$, which may depend on Δ but not directly on n, is also not constant. One example is the $O(\Delta)$ vertex coloring problem.

Next, for a vertex v to tell whether $\mathcal{E}_v \in \mathcal{I}$, it needs to know whether $\mathcal{E}_u$ occurs for all $u \in N^{2(r+t^\star)}(v)$. This information can be gathered in $O(C(\Delta))$ time using messages of size $O(\zeta)$. To summarize, the required message size is $O(\log(|\Sigma_{\text{out}}|) \cdot \Delta^{O(1)} + m_\Delta(\zeta) + \zeta)$. □

An interesting corollary of Theorem 2.9 is that when $\Delta = O(1)$, randomized algorithms with unbounded length messages can be simulated with 1-bit messages.

Corollary 2.5. *Let $\mathcal{P}$ be any LCL problem. When $\Delta = O(1)$, any $o(\log n)$ algorithm solving $\mathcal{P}$ w.h.p. using* unbounded length messages *can be made to run in $O(\log n)$ time with 1-bit messages.*

2.6 Speedup Implications of Naor & Stockmeyer

Let $\mathcal{A}$ be any $T(n)$-round DetLOCAL algorithm. Let η and η' be any two *order-indistinguishable* assignments of distinct IDs to $N^{T(n)}(v)$, i.e., for $u, w \in N^{T(n)}(v)$, $\eta(u) > \eta(w)$ if and only if $\eta'(u) > \eta'(w)$. If, for every possible input graph fragment induced by $N^{T(n)}(v)$, the output label of v is identical under every pair of order-indistinguishable η, η', then $\mathcal{A}$ is *order-invariant*.

Suppose that there exists a number $n' = O(1)$ such that $\Delta^{T(n')+r} < n'$. If $\mathcal{A}$ is order-invariant then it can be turned into an $O(1)$-round DetLOCAL algorithm $\mathcal{A}'$, since we can pretend that the total number of vertices is n' instead of n.

Naor and Stockmeyer [116] proved that any DetLOCAL algorithm that takes $\tau = O(1)$ rounds on a bounded degree graph can be turned into an order-invariant τ-round DetLOCAL algorithm. A more careful analysis shows that the proof still works when τ is a slowly growing function of n.

2.6.1 Requirements for Automatic Speedup

The multicolor hypergraph Ramsey number $R(p, m, c)$ is the minimum number such that the following holds. Let H be a complete p-uniform hypergraph of at least $R(p, m, c)$ vertices. Then any c-edge-coloring of H contains a monochromatic clique of size m.

Given the number $\tau \geq 2$, the three parameters p, m, and c are selected as follows. (See the proof of [116, Lemma 3.2] for more details.)

- The number p is the maximum number of vertices in $N^{\tau}(v)$, over all vertices $v \in V(G)$ and all graphs G under consideration. For paths/cycles, $p = 2\tau + 1$. For grids/tori, $p \leq 2(\tau+1)^2$. For trees or general graphs, $p \leq \Delta^{\tau}$.
- The number m is the maximum number of vertices in $N^{\tau+r}(v)$, over all vertices $v \in V(G)$ and all graphs G under consideration. E.g., for paths/cycles, $p = 2\tau+2r+1$ and for general graphs, $p \leq \Delta^{\tau+r}$.
- The number z counts the distinguishable radius-τ centered subgraphs, disregarding IDs. For example, for LCLs on the n-cycle without input labels or port numbering, $z = 1$, whereas with input labels and port numbering it is $(2|\Sigma_{\text{in}}|)^{2\tau+1}$ since each vertex has one of $|\Sigma_{\text{in}}|$ input labels and 2 port numberings. In general z is less than $2^{\binom{\Delta^{\tau}}{2}} \cdot (\Delta!|\Sigma_{\text{in}}|)^{p}$.
- The number c is defined as $|\Sigma_{\text{out}}|^{p!z}$. Intuitively, we can use a number in $[c]$ to encode a function that maps a radius-τ centered subgraph, whose vertices are equipped with distinct vertex IDs drawn from some set S with cardinality p, to an output label in Σ_{out}.

Recall that vertices in DetLOCAL have $O(\log n)$-bit IDs, i.e., they can be viewed as elements of $[n^k]$ for some $k = O(1)$. Naor and Stockmeyer's proof implies that, as long as $n^k \geq R(p,m,c)$, any DetLOCAL τ-round algorithm on a bounded degree graph can be turned into an order-invariant τ-round DetLOCAL algorithm, which then implies an $O(1)$-round DetLOCAL algorithm.

2.6.2 Automatic Speedup Theorems

According to the proof of [84, §1, Theorem 2], we have:

$$\begin{aligned}
&\text{For } p = 1, \quad && R(p,m,c) = c(m-1)+1 \\
&\text{For } p > 1, \quad && R(p,m,c) \leq 2c^{x} \\
& && \text{where } x = \sum_{i=p-1}^{R(p-1,m,c)-1} \binom{i+1}{p-1} < R(p-1,m,c)^{p}
\end{aligned}$$

Therefore, $\log^*(R(p,m,c)) \le p + \log^* m + \log^* c + O(1)$.

Observe that in all scenarios described in Section 2.6.1, if the running time τ satisfies $\tau = \tau(n) = \omega(1)$, we have $\log^* m + \log^* c = o(p)$. Therefore, having $p \le \epsilon \log^* n$ for some small enough constant ϵ suffices to meet the condition $n^k \ge R(p,m,c)$. We conclude that the complexity of any LCL problem (with or without input labels and port numbering) in the LOCAL model never falls in the following gaps:

$\omega(1)$—$o(\log^* n)$	for n-paths/cycles.
$\omega(1)$—$o\left(\sqrt{\log^* n}\right)$	for $(\sqrt{n} \times \sqrt{n})$-grids/tori.
$\omega(1)$—$o(\log(\log^* n))$	for bounded degree trees or bounded degree general graphs.

By Corollary 2.1, the DetLOCAL and RandLOCAL complexities of any LCL problem are asymptotically the same if they are at most $2^{O(\log^* n)}$. Therefore, the above gaps not only apply to DetLOCAL but also RandLOCAL.

Due to the "Stepping-Up Lemma" (see [84, §4, Lemma 17]), we have a lower bound $\log^*(R(p,m,2)) = \Omega(p)$ (for any p, m). Therefore, Naor and Stockmeyer's approach *alone* cannot give an $\omega(1)$—$o(\log^* n)$ gap for bounded degree trees. However, for a certain class of LCL problems on $(\sqrt{n} \times \sqrt{n})$-grids/tori, the gap can be widened to $\omega(1)$—$o(\log^* n)$ [31, p. 2]. The following proof is due to J. Suomela (personal communication).

Theorem 2.10 (J. Suomela)**.** *Let $\mathcal{P}$ be any LCL problem on $(\sqrt{n}\times\sqrt{n})$-grids/tori that does not refer to input labels or port-numbering. The* DetLOCAL *and* RandLOCAL *complexity of $\mathcal{P}$ is either $O(1)$ or $\Omega(\log^* n)$.*

Proof. Given a $(\sqrt{n} \times \sqrt{n})$-torus G, we associate each vertex $v \in V(G)$ with a coordinate (α, β), where $\alpha, \beta \in \{0, \dots, \sqrt{n} - 1\}$. We consider the following special way to generate unique $2k \log n$-bit IDs. Let ϕ_x and ϕ_y be two functions mapping integers in $\{0, \dots, \sqrt{n}-1\}$ to integers in $\{0, \dots, n^k - 1\}$. We additionally require that $\phi_x(0) < \cdots < \phi_x(\sqrt{n} - 1) < \phi_y(0) < \cdots < \phi_y(\sqrt{n} - 1)$. If v is at position (α, β), it has ID $\phi_x(\alpha) \cdot n^k + \phi_y(\beta)$. Notice that the IDs of all vertices in $N^\tau(v)$ can be deduced from just $4\tau + 2$ numbers: $\phi_x(i)$, $i \in [\alpha - \tau, \alpha + \tau]$ and $\phi_y(j)$, $j \in [\beta - \tau, \beta + \tau]$.

Suppose that the complexity of $\mathcal{P}$ is $o(\log^* n)$. Let $\mathcal{A}$ be any τ-round DetLOCAL algorithm for solving $\mathcal{P}$, where $\tau = o(\log^* n)$. Notice that the algorithm $\mathcal{A}$ works correctly even when we restrict ourselves to the above special ID assignment. Our goal is to show that $\mathcal{P}$ is actually *trivial* in the sense that there exists an element $\sigma \in \Sigma_{\text{out}}$ such that labeling all

vertices by σ gives a legal labeling, assuming w.l.o.g. that $\sqrt{n} > 2r + 1$. Thus, $\mathcal{P}$ can be solved in $O(1)$ rounds.

In subsequent discussion, we let v be any vertex whose position is (α, β), where $\tau + r \leq \alpha \leq (\sqrt{n} - 1) - (\tau + r)$ and $\tau + r \leq \beta \leq (\sqrt{n} - 1) - (\tau + r)$. That is, v is sufficiently far from the places where the coordinates wrap around.

Given $\mathcal{A}$, we construct a function f as follows. Let $S = (s_1, \ldots, s_{4\tau+2})$ be a vector of $4\tau + 2$ numbers in $\{0, \ldots, n^k - 1\}$ such that $s_l < s_{l+1}$ for each $l \in [4\tau + 2]$. Then $f(S) \in \Sigma_{\text{out}}$ is defined as the output labeling of v resulting from executing $\mathcal{A}$ with the following ID assignment of vertices in $N^\tau(v)$. We set $\phi_x(\alpha - \tau - 1 + i) = s_i$ for each $i \in [2\tau + 1]$ and set $\phi_y(\beta - \tau - 1 + j) = s_{j+2\tau+1}$ for each $j \in [2\tau + 1]$. Recall that $\mathcal{P}$ does not use port-numbering and input labeling, so the output labeling of v depends only on IDs of vertices in $N^\tau(v)$.

We set $p = 4\tau + 2$, $m = 4\tau + 4r + 2$, and $c = |\Sigma_{\text{out}}|$. Notice that the calculation of the parameter c here is different from the original proof of Naor and Stockmeyer. Since we already force that $\phi_x(0) < \cdots < \phi_x(\sqrt{n} - 1) < \phi_y(0) < \cdots < \phi_y(\sqrt{n} - 1)$, we do not need to consider all $p!$ permutations of the set S.

We have $R(p, m, c) \ll n^k$ (since $p = o(\log^* n)$). Thus, there exists a set S' of m distinct numbers in $\{0, \ldots, n^k\}$ such that the following is true. We label these m numbers $\phi_x(i)$, $i \in [\alpha - \tau - r, \alpha + \tau + r]$, and $\phi_y(j)$, $j \in [\beta - \tau - r, \beta + \tau + r]$ by the set S' such that $\phi_x(\alpha - \tau - r) < \ldots < \phi_x(\alpha + \tau + r) < \phi_y(\beta - \tau - r) < \ldots < \phi_y(\beta + \tau + r)$. Then the output labels of all vertices in $N^r(v)$ assigned by $\mathcal{A}$ are identical.

Therefore, there exists an element $\sigma \in \Sigma_{\text{out}}$ such that labeling all vertices by σ yields a legal labeling of G. Thus, $\mathcal{P}$ can be solved in $O(1)$ rounds.

On grids, the proof above shows that the LCL $\mathcal{P}$ admits a labeling where all interior vertices (those at distance greater than r from the boundary) can be labeled uniformly by some $\sigma \in \Sigma_{\text{out}}$ and every other vertex can be labeled according to an $O(1)$-round order-invariant algorithm.

Similarly, by Corollary 2.1, the $\omega(1)$—$o(\log^* n)$ gap given in this proof applies to both DetLOCAL and RandLOCAL. □

2.6.3 Discussion

It still remains an outstanding open problem whether the gap for other cases can also be widened to $\omega(1)$—$o(\log^* n)$.

The proof of Theorem 2.10 extends easily to d-dimensional tori, but does not extend to bounded degree trees, since there is a non-trivial problem that can be solved in $O(1)$ rounds on a subset of bounded degree trees (see the proof of Theorem 2.10 for the definition of a trivial problem). A *weak coloring* is a coloring in which every vertex is colored differently than at least one neighbor. Naor and Stockmeyer [116] showed that on any graph class in which all vertex degrees are odd, *weak* $2^{O(\Delta \log \Delta)}$*-coloring* can be solved in 2 rounds and *weak* 2*-coloring* can be solved in $O(\log^* \Delta)$ rounds in DetLOCAL. This problem is *non-trivial* in the sense that coloring all vertices by the same color is not a legal solution. Since the d-dimensional torus is Δ-regular, $\Delta = 2d$, we infer that the complexity of weak $O(1)$-coloring on Δ-regular graphs is $\Theta(\log^* n)$ for every fixed even number $\Delta \geq 2$.

Theorem 2.10 also does not extend to LCL problems that use input labels or port-numbering. If either input labels or port-numbering are allowed, then one can construct a non-trivial LCL problem that can be solved in $O(1)$ rounds even on cycle graphs. An *orientation* of a vertex $v \in V(G)$ is defined as a port-number in $[\deg(v)]$, indicating a vertex in $N(v)$ that v is pointed towards. An ℓ*-orientation* of a cycle G is an orientation of all vertices in G meeting the following conditions. If $|V(G)| \leq \ell$, then all vertices in G are oriented to the same direction, i.e., no two vertices point toward each other. If $|V(G)| > \ell$, then each vertex $v \in V(G)$ belongs to a path P such that (i) all vertices in P are oriented to the same direction (no two point to each other), and (ii) the number of vertices in P is at least ℓ. Notice that ℓ-orientation, $\ell = O(1)$, is an LCL that refers to port-numbering. We show that in $O(1)$ rounds we can compute an ℓ-orientation of G for any constant ℓ.

Theorem 2.11. *Let G be a cycle graph and ℓ be a constant. There is a* DetLOCAL *algorithm that computes an ℓ-orientation of G in $O(1)$ rounds.*

Proof. This is a known result. See [85, Fact 5.2] or [63, Lemma 14 (Rounding Lemma), Case B] for a sketch of the proof. For the sake of completeness, we present a full proof. We first show how to compute a 2-orientation of a cycle G in $O(1)$ rounds, and then we extend it to any constant ℓ.

Computing a 2-orientation We assume $|V(G)| \geq 3$. A DetLOCAL $O(1)$-round algorithm to compute a 2-orientation is described as follows. First, each vertex $v \in V(G)$ computes an arbitrary orientation. With respect to this orientation of G, define sets V_1, V_2, V_3 as follows.

- $v \in V_1$ if and only if there exists $u \in N(v)$ such that u and v are oriented to the same direction.
- $v \in V_2$ if and only if there exists $u \in N(v) \setminus V_1$ such that u and v are oriented toward each other.
- $V_3 = V(G) \setminus (V_1 \cup V_2)$. Observe that for each $v \in V_3$, there exists $u \in N(v) \cap V_1$.

A 2-orientation is obtained by re-orienting the vertices in V_2 and V_3. The vertices in V_2 are partitioned into unordered pairs such that $u, v \in V_2$ are paired-up if and only if (i) $\{u, v\} \in E(G)$ and (ii) u and v are oriented toward each other. For each pair $\{u, v\}$, reverse the orientation of any one of $\{u, v\}$. For each vertex $v \in V_3$, let u be any neighbor of v such that $u \in V_1$, and re-orient v to the orientation of u.

Computing an ℓ-orientation We define an $O(1)$-round DetLOCAL algorithm $\mathcal{A}_\ell$ that computes an ℓ-orientation. It makes recursive calls to $\mathcal{A}_{\lceil \ell/2 \rceil}$. In what follows, we assume $\ell \geq 3$ and $|V(G)| \geq 3$.

First, execute $\mathcal{A}_{\lceil \ell/2 \rceil}$ to obtain a $\lceil \ell/2 \rceil$-orientation of G. With respect to this orientation of G, define the following terminologies. Let $\mathscr{P}$ be the set of all maximal-size connected subgraphs in G such that all constituent vertices are oriented to the same direction. Notice that if $\mathscr{P}$ contains a cycle, then $\mathscr{P} = \{G\}$. Otherwise $\mathscr{P}$ contains only paths. Define $\mathscr{P}_1$ as the subset of $\mathscr{P}$ such that $P \in \mathscr{P}_1$ if and only if the number of vertices in P is at least ℓ. Define $\mathscr{P}_2$ as the subset of $\mathscr{P} \setminus \mathscr{P}_1$ such that $P \in \mathscr{P}_2$ if and only if there exists another path $P' \in \mathscr{P} \setminus \mathscr{P}_1$ meeting the following condition. There exist an endpoint u of P and an endpoint v of P' such that $\{u, v\} \in E(G)$, and u and v are oriented toward each other. Define $\mathscr{P}_3 = \mathscr{P} \setminus (\mathscr{P}_1 \cup \mathscr{P}_2)$. Observe that each $P \in \mathscr{P}_3$ is adjacent to a path in $\mathscr{P}_1$.

The paths in $\mathscr{P}_2$ are partitioned into unordered pairs such that $P, P' \in \mathscr{P}_2$ are paired-up if and only if there exist an endpoint u of P and an endpoint v of P' such that $\{u, v\} \in E(G)$, and u and v are oriented toward each other. For each pair $\{P, P'\}$, reverse the orientation of all the vertices in any one of $\{P, P'\}$. For each path $P \in \mathscr{P}_3$, let $P' \in \mathscr{P}_1$ be any path adjacent to P, and re-orient P to the orientation of P'.

The round complexity of $\mathcal{A}_\ell$ satisfies the recurrence $T(\ell) = T(\lceil \ell/2 \rceil) + O(\ell)$, which is $O(\ell)$. □

Notice that even though orienting all vertices in the cycle to the same direction gives a legal labeling, ℓ-orientation is still a non-trivial LCL problem. Consider a subpath

(v_1, v_2, v_3, v_4) in the cycle. Suppose that the port-number of (v_2, v_3) stored at v_2 is 1, but the port-number of (v_3, v_4) stored at v_3 is 2. Then we need to label v_2 and v_3 differently (1 and 2, respectively) in order to orient them in the same direction '$\rightarrow$'.

Lastly, we remark that for the case the given $(\sqrt{n} \times \sqrt{n})$-torus is *oriented* in the sense that the input port-numberings all agree with a fixed N/S/E/W orientation [32], then there is no non-trivial LCL problem solvable in $O(1)$ time.

Chapter 3

Complexity Landscape of LCLs on Trees

3.1 Overview

In this chapter we study the LOCAL complexity landscape on bounded degree trees. We establish a new (deterministic and randomized) complexity gap for bounded degree trees and a new infinite hierarchy of coloring problems with polynomial time complexities.

An Infinite Hierarchy of Coloring Problems. We define an infinite class of LCL problems called *Hierarchical* $2\frac{1}{2}$*-Coloring*. A correctly colored graph can be confirmed by simply checking the neighborhood of each vertex, so this problem fits into the class of LCL problems. However, the complexity of the k-level Hierarchical $2\frac{1}{2}$-Coloring problem is $\Theta(n^{1/k})$, for $k \in \mathbb{Z}^+$. The upper bound holds in DetLOCAL on general graphs, and the lower bound holds even on degree-3 trees in RandLOCAL.

This result is interesting in that this is the first time hierarchy-type theorem for the distributed LOCAL model. In contrast to paths/cycles and grids/tori, trees and general graphs support an *infinite* number of natural problem complexities.

A Complexity Gap on Bounded Degree Trees. We prove that on the class of bounded degree trees, no LCL has complexity in the range $\omega(\log n)$—$n^{o(1)}$. Specifically, any $n^{o(1)}$-time RandLOCAL algorithm can be converted to an $O(\log n)$-time DetLOCAL algorithm. Moreover, given the description of an LCL problem $\mathcal{P}$, it is *decidable* whether the RandLOCAL complexity of $\mathcal{P}$ is $n^{\Omega(1)}$ or the DetLOCAL complexity of $\mathcal{P}$ is $O(\log n)$. It

turns out that this gap is maximal. That is, we cannot extend it lower than $\omega(\log n)$ [39, 107], nor higher than $n^{o(1)}$.

Commentary. All the existing automatic speedup theorems are quite different in terms of proof techniques. Naor and Stockmeyer's approach is based on Ramsey theory. The speedup theorems of Chapter 2 use the fact that $o(\log_\Delta n)$ algorithms on general graphs (and $o(n)$ algorithms on n-cycles/paths and $o(\sqrt{n})$ algorithms on $(\sqrt{n} \times \sqrt{n})$-grids/tori) cannot "see" the whole graph, and can therefore be efficiently tricked into thinking the graph has constant size.

Our new $n^{o(1)} \to O(\log n)$ speedup theorem introduces an entirely new set of techniques based on classic automata theory. We show that any LCL problem gives rise to a regular language that represents partial labelings of the tree that can be consistently extended to total labelings. By applying the pumping lemma for regular languages, we can "pump" the input tree into a much larger tree that behaves similar to the original tree. The advantage of creating a larger imaginary tree is that each vertex can (mentally) simulate the behavior of an $n^{o(1)}$-time algorithm on the *imaginary* tree, merely by inspecting its $O(\log n)$-neighborhood in the *actual* tree. Moreover, because the pumping operation preserves properties of the original tree, a labeling of the imaginary tree can be efficiently converted to a labeling of the original tree.

Organization. In Section 3.2 we introduce Hierarchical $2\frac{1}{2}$-Coloring and prove that the k-level variant of this problem has complexity $\Theta(n^{1/k})$. In Section 3.3 we prove the $n^{o(1)} \to O(\log n)$ speedup theorem for bounded degree trees.

3.2 An Infinitude of Complexities: Hierarchical $2\frac{1}{2}$-Coloring

In this section we give an infinite sequence $(\mathcal{P}_k)_{k \in \mathbb{Z}^+}$ of LCL problems, where the complexity of $\mathcal{P}_k$ is precisely $\Theta(n^{1/k})$.[8] The upper bound holds on general graphs in DetLOCAL and the lower bound holds in RandLOCAL, even on degree-3 trees. Informally, the task of $\mathcal{P}_k$ is to 2-color (with $\{\mathbf{a}, \mathbf{b}\}$) certain specific subgraphs of the input graph. Some vertices

[8] Brandt et al. [32, Appendix A.3] described an LCL that has complexity $\Theta(\sqrt{n})$ on general graphs, but not trees. It may be possible to generalize their LCL to any complexity of the form $\Theta(n^{1/k})$.

are *exempt* from being colored (in which case they are labeled $\mathbf{X}$), and in addition, it is possible to *decline* to 2-color certain subgraphs, by labeling them $\mathbf{D}$.

There are no input labels. The output label set is $\Sigma_{\text{out}} = \{\mathbf{a}, \mathbf{b}, \mathbf{D}, \mathbf{X}\}$. The problem $\mathcal{P}_k$ is an LCL defined by the following rules.

Levels. Subsequent rules depend on the *levels* of vertices. Let V_i, $i \in \{1, \ldots, k+1\}$, be the set of vertices on level i, defined as follows.

$$\begin{aligned} G_1 &= G \\ G_i &= G_{i-1} - V_{i-1}, && \text{for } i \in [2, k+1] \\ V_i &= \{v \in V(G_i) \mid \deg_{G_i}(v) \leq 2\}, && \text{for } i \in [1, k] \\ V_{k+1} &= V(G_{k+1}) && \text{(the remaining vertices)} \end{aligned}$$

Remember that vertices know their degrees, so a vertex in V_1 deduces this with 0 rounds of communication. In general the level of v can be calculated from information in $N^k(v)$.

Exemption. A vertex labeled $\mathbf{X}$ is called *exempt*. No V_1 vertex is labeled $\mathbf{X}$; all V_{k+1} vertices are labeled $\mathbf{X}$. Any V_i vertex is labeled $\mathbf{X}$ iff it is adjacent to a lower level vertex labeled $\mathbf{a}, \mathbf{b}$, or $\mathbf{X}$. Define $X_i \subseteq V_i$ to be the set of level i exempt vertices.

Two-Coloring. Vertices not covered by the exemption rule are labeled one of $\mathbf{a}, \mathbf{b}, \mathbf{D}$.

- Any vertex in V_i, $i \in [1, k]$, labeled $\mathbf{a}$ has no neighbor in V_i labeled $\mathbf{a}$ or $\mathbf{D}$.
- Any vertex in V_i, $i \in [1, k]$, labeled $\mathbf{b}$ has no neighbor in V_i labeled $\mathbf{b}$ or $\mathbf{D}$.
- Any vertex in $V_k - X_k$ with exactly 0 or 1 neighbors in $V_k - X_k$ must be labeled $\mathbf{a}$ or $\mathbf{b}$.

Commentary. The Level rule implies that the graph induced by V_i consists of paths and cycles. The Two-Coloring rule implies that each component of non-exempt vertices in the graph induced by $V_i - X_i$ must either (a) be labeled uniformly by $\mathbf{D}$ or (b) be properly 2-colored by $\{\mathbf{a}, \mathbf{b}\}$. Every *path* in $V_k - X_k$ must be properly 2-colored, but *cycles* in $V_k - X_k$ are allowed to be labeled uniformly by $\mathbf{D}$. This last provision is necessary to ensure that *every* graph can be labeled according to $\mathcal{P}_k$ since there is no guarantee that cycles in $V_k - X_k$ are bipartite.

Remark 3.1. *As stated $\mathcal{P}_k$ is an LCL with an alphabet size of 4 and a radius k, since the coloring rules refer to levels, which can be deduced by looking up to radius k. On the other hand, we can also represent $\mathcal{P}_k$ as an LCL with radius 1 and alphabet size $4k$ by including a vertex's level in its output label. A correct level assignment can be verified within radius 1. For example, level 1 vertices are those with degree at most 2, and a vertex is labeled $i \in [2, k]$ iff all but at most 2 neighbors have levels less than i.*

Theorem 3.1. *The* DetLOCAL *complexity of $\mathcal{P}_k$ on general graphs is $O(n^{1/k})$.*

Proof. The algorithm fixes the labeling of $V_1, \ldots, V_k, V_{k+1}$ in order, according to the following steps. Assume that all vertices in $V_1, \ldots, V_{i-1}$ have already been labeled.

- Compute X_i according to the Exemption rule. (E.g., $X_1 = \emptyset$, $X_{k+1} = V_{k+1}$.)
- Each path in the subgraph induced by $V_i - X_i$ calculates its length. If it contains at most $\lceil 2n^{1/k} \rceil$ vertices, it properly 2-colors itself with $\{\mathbf{a}, \mathbf{b}\}$; longer paths and cycles in $V_i - X_i$ label themselves uniformly by $\mathbf{D}$.

This algorithm correctly solves $\mathcal{P}_k$ provided that it never labels a path in $V_k - X_k$ with $\mathbf{D}$. Let U_i be the subgraph induced by those vertices in $V_1 \cup \cdots \cup V_i$ labeled $\mathbf{D}$. Consider a connected component C in U_i whose V_i-vertices are arranged in a path (not a cycle). We argue by induction that C has at least $2n^{i/k}$ vertices. This is clearly true in the base case $i = 1$: if a path component of U_1 were colored $\mathbf{D}$, it must have more than $\lceil 2n^{1/k} \rceil$ vertices. Now assume the claim is true for $i - 1$ and consider a component C of U_i. If the V_i-vertices in C form a path, it must have length greater than $2n^{1/k}$. Each vertex in that path must be adjacent to an endpoint of a V_{i-1} path. Since V_{i-1} paths have two endpoints, the V_i path is adjacent to at least $\lceil 2n^{1/k} \rceil / 2 \geq n^{1/k}$ components in U_{i-1}, each of which has size at least $2n^{(i-1)/k}$, by the inductive hypothesis. Thus, the size of C is at least $n^{1/k} \cdot 2n^{(i-1)/k} + 2n^{1/k} > 2n^{i/k}$. Because there are at most n vertices in the graph, it is impossible for V_k vertices arranged in a path to be colored $\mathbf{D}$. □

Theorem 3.2. *The* RandLOCAL *complexity of $\mathcal{P}_k$ on trees with maximum degree $\Delta = 3$ is $\Omega(n^{1/k})$.*

Proof. Fix an integer parameter x and define a sequence of graphs $(H_i)_{1 \leq i \leq k}$ as follows. Each H_i has a *head* and a *tail*.

- H_1 is a path (or *backbone*) of length x. One end of the path is the head and the other end the tail.

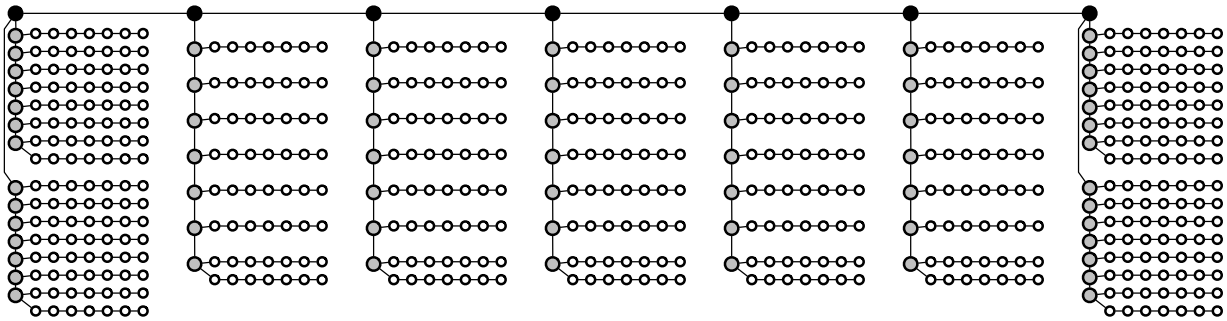

Figure 3.1: The graph H_k with parameters $k = 3, x = 7$.

- To construct H_i, $i \in [2, k-1]$, begin with a *backbone* path $(v_1, v_2, \ldots, v_x)$, with head v_1 and tail v_x. Form $x+1$ copies $(H_{i-1}^{(j)})_{1 \le j \le x+1}$ of H_{i-1}, where $v^{(j)}$ is the head of $H_{i-1}^{(j)}$. Connect $v^{(j)}$ to v_j by an edge, for $j \in [1, x]$, and also connect $v^{(x+1)}$ to v_x by an edge.

- H_k is constructed exactly as above, except that we generate $x+2$ copies of H_{k-1} and connect the heads of two copies of H_{k-1} to both v_1 and v_x. See Figure 3.1 for an example with $k = 3$. White vertices are in V_1, gray in V_2, and black in V_3. $V_4 = V_{k+1}$ is empty.

We make several observations about the construction of H_k. First, it is a tree with maximum degree 3. Second, when decomposing $V(H_k)$ into levels $(V_1, \ldots, V_k, V_{k+1})$, V_i is precisely the union of the backbones in all copies of H_i, and $V_{k+1} = \emptyset$. Third, the number of vertices in H_k is $\Theta(x^k)$, so a $o(n^{1/k})$ algorithm for $\mathcal{P}_k$ must run in $o(x)$ time on H_k.

Consider a RandLOCAL algorithm $\mathcal{A}$ solving $\mathcal{P}_k$ on H_k within $t < x/5 - O(1)$ time, that fails with probability p_{fail}. If $\mathcal{A}$ is a good algorithm then $p_{\text{fail}} \le 1/|V(H_k)|$. However, we will now show that p_{fail} is constant, independent of $|V(H_k)|$.

Define $\mathcal{E}_i$ to be the event that $X_i \neq \emptyset$ and $p_i = \Pr(\mathcal{E}_i)$. By an induction from $i = 2$ to k, we prove that $p_i \le 2(i-1) \cdot p_{\text{fail}}$.

Base Case. We first prove that

$$\Pr\left(H_k \text{ is not correctly colored according to } \mathcal{P}_k \mid \mathcal{E}_2\right) \ge 1/2.$$

Conditioning on $\mathcal{E}_2$ means that $X_2 \neq \emptyset$. Fix any $v \in X_2$ and let P be a copy of H_1 (a path) adjacent to v. In order for $v \in X_2$, it must be that P is properly 2-colored with $\{\mathbf{a}, \mathbf{b}\}$.

Since $t < x/5 - O(1)$, there exist two vertices u and u' in P such that

1. $N^t(u)$, $N^t(u')$, and $N^t(v)$ are disjoint sets,
2. the subgraphs induced by $N^t(u)$ and $N^t(u')$ are isomorphic, and
3. the distance between u and u' is odd.

Let $p_{\mathbf{a}}$ and $p_{\mathbf{b}}$ be the probabilities that u/u' is labeled $\mathbf{a}$ and $\mathbf{b}$, respectively. A proper 2-coloring of P assigns u and u' different colors, and that occurs with probability $2p_{\mathbf{a}}p_{\mathbf{b}} \leq 2p_{\mathbf{a}}(1-p_{\mathbf{a}}) \leq 1/2$. Moreover, this holds independent of the random bits generated by vertices in $N^t(v)$. The algorithm fails unless u, u' have different colors, thus $p_{\text{fail}} \geq p_2/2$, and hence $p_2 \leq 2 \cdot p_{\text{fail}}$.

Inductive Step. Let $3 \leq i \leq k$. The inductive hypothesis states that $p_{i-1} \leq 2(i-2) \cdot p_{\text{fail}}$. By a proof similar to the base case, we have that:

$$\Pr\left(H_k \text{ is not correctly colored according to } \mathcal{P}_k \;\middle|\; \mathcal{E}_i \backslash \mathcal{E}_{i-1}\right) \geq 1/2.$$

We are conditioning on $\mathcal{E}_i \backslash \mathcal{E}_{i-1}$. If this event is empty, then $p_i \leq p_{i-1} \leq 2(i-2) \cdot p_{\text{fail}}$ and the induction is complete. On the other hand, if $\mathcal{E}_i \backslash \mathcal{E}_{i-1}$ holds then there is some $v \in X_i$ adjacent to a copy of H_{i-1} with backbone path P, where $P \cap X_{i-1} = \emptyset$. In other words, if H_k is colored according to $\mathcal{P}_k$ then P must be properly 2-colored with $\{\mathbf{a}, \mathbf{b}\}$. The argument above shows this occurs with probability at least $1/2$. Thus,

$$p_{\text{fail}} = \Pr(H_k \text{ is incorrectly colored}) \geq \Pr(\mathcal{E}_i \backslash \mathcal{E}_{i-1})/2 \geq (p_i - p_{i-1})/2,$$

or $p_i \leq 2p_{\text{fail}} + p_{i-1} \leq 2(i-1)p_{\text{fail}}$, completing the induction.

Finally, let P be the path induced by vertices in V_k. The probability that $\mathcal{E}_k$ holds ($P \cap X_k \neq \emptyset$) is $p_k \leq 2(k-1) \cdot p_{\text{fail}}$. On the other hand, we have $\Pr(H_k \text{ not colored correctly} \mid \overline{\mathcal{E}_k}) \geq 1/2$ by the argument above, hence $p_{\text{fail}} \geq (1-p_k)/2$, or $p_k \geq 1 - 2p_{\text{fail}}$. Combining the upper and lower bounds on p_k we conclude that $p_{\text{fail}} \geq (2k)^{-1}$ is constant, independent of $|V(H_k)|$. Thus, algorithm $\mathcal{A}$ cannot succeed with high probability. $\square$

3.3 A Complexity Gap on Bounded Degree Trees

In this section we prove an $n^{o(1)} \to O(\log n)$ speedup theorem for LCL problems on bounded degree trees. The progression of definitions and lemmas in Sections 3.3.2–3.3.13 is *logical*, but obscures the high level structure of the proof. Section 3.3.1 gives an informal tour of the proof and its key ideas. Throughout, $\mathcal{P}$ is a radius-r LCL and $\mathcal{A}$ is an $n^{o(1)}$-time algorithm for $\mathcal{P}$ on bounded degree trees.

3.3.1 A Tour of the Proof

Consider this simple way to decompose a tree in $O(\log n)$ time, inspired by Miller and Reif [109]. Iteratively remove paths of degree-2 vertices (*compress*) and vertices with degree 0 or 1 (*rake*). Vertices removed in iteration i are at *level* i. If $O(\log n)$ *rakes* alone suffice to decompose a tree then it has $O(\log n)$ diameter and any LCL can be solved in $O(\log n)$ time on such a graph. Thus, we mainly have to worry about the situation where *compress* removes very long ($\omega(1)$-length) paths.

The first observation is that it is easy to split up long degree-2 paths of level-i vertices into constant length paths, by artificially promoting a well-spaced subset of level-i vertices to level $i+1$. Thus, we have a situation that looks like this (see Figure 3.2): level-i vertices are arranged in an $O(1)$-length path, each the root of a subtree of level-$(< i)$ vertices (colored subtrees in the figure) that were removed in previous rake/compress steps, and bookended by level-$(> i)$ vertices (black in the figure). Call the subgraph between the bookends H.

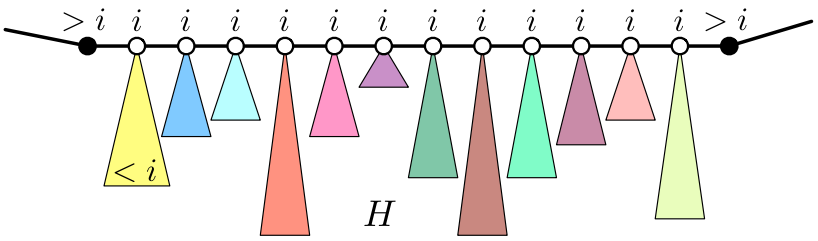

Figure 3.2: A constant length path resulting from spliting up long degree-2 paths of level-i vertices.

In our approach it is the level-$(> i)$ vertices that are in charge of coordinating the labeling

of level-($\leq i$) vertices in their purview. In this diagram, H is in the purview of both black bookends. We only have one tool available for computing a labeling of this subgraph: an $n^{o(1)}$-time RandLOCAL algorithm $\mathcal{A}$ that works w.h.p. What would happen if we *simulated* $\mathcal{A}$ on the vertices of H? The simulation would fail catastrophically of course, since it needs to look up to an $n^{o(1)}$ radius, to parts of the graph far outside of H.

The colored subtrees are unbounded in terms of size and depth. Nonetheless, they fall into a *constant* number of equivalence classes in the following sense. The *class* of a rooted tree is the set of all labelings of the r-neighborhood of its root that can be extended to total labelings of the tree that are consistent with $\mathcal{P}$ (see Figure 3.3).

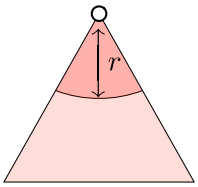

Figure 3.3: Class of a rooted tree.

In other words, the large and complex graph H can be succinctly encoded as a simple class vector $(c_1, c_2, \ldots, c_\ell)$, where c_j is the class of the jth colored tree. Consider the set of all labelings of H that are consistent with $\mathcal{P}$. This set can also be succinctly represented by listing the labelings of the r-neighborhoods of the bookends that can be extended to all of H, while respecting $\mathcal{P}$. The set of these partial labelings defines the *type* of H. We show that H's type can be computed by a finite automaton that reads the class vector $(c_1, \ldots, c_\ell)$ one character at a time. By the pigeonhole principle, if ℓ is sufficiently large then the automaton loops, meaning that $(c_1, \ldots, c_\ell)$ can be written as $x \circ y \circ z$, which has the same type as every $x \circ y^j \circ z$, for all $j \geq 1$. This *pumping lemma for trees* lets us dramatically expand the size of H without affecting its type, i.e., how it interacts with the outside world beyond the bookends.

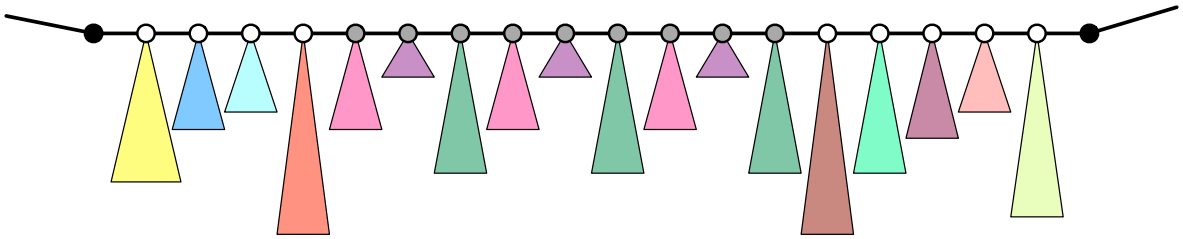

Figure 3.4: Pumping lemma for trees.

Figure 3.4 illustrates the pumping lemma with a substring of $|y| = 3$ trees (rooted at gray vertices) repeated $j = 3$ times. Now let us reconsider the simulation of $\mathcal{A}$. If we first pump H to be long enough, and then simulate $\mathcal{A}$ on the middle section of pumped-H, $\mathcal{A}$ must, according to its $n^{o(1)}$ time bound, compute a labeling *without needing any information outside of pumped-H*, i.e., beyond the bookends. Thus, we can use $\mathcal{A}$ to *pre-commit* to a labeling of a small (radius-r) subgraph of pumped-H. Given this pre-commitment, the left and right bookends no longer need to coordinate their activities: everything left (right) of the pre-committed zone is now in the purview of the left (right) bookend. Interestingly, these manipulations (tree surgery and pre-commitments) can be repeated for each i, yielding a hierarchy of *imaginary* trees such that a proper labeling at one level of the hierarchy implies a proper labeling at the previous level.

Roadmap This short proof sketch has been simplified to the point that it is riddled with small inaccuracies. Nonetheless, it does accurately capture the difficulties, ideas, and techniques used in the actual proof. In Section 3.3.2 we formally define the notion of a partially labeled graph, i.e., one with certain vertices pre-commited to their output labels. Section 3.3.3 defines a surgical "cut-and-paste" operation on graphs. Section 3.3.4 defines a partition of the vertices of a subgraph H, which differentiates between vertices that "see" the outside graph, and those that see only H. Section 3.3.5 defines an equivalence relation on graphs that, intuitively, justifies surgically replacing a subgraph with an equivalent graph. Sections 3.3.6 and 3.3.7 explore properties of the equivalence relation. Section 3.3.8 introduces the pumping lemma for trees, and Section 3.3.9 defines a specialized Rake/Compress-style graph decomposition. Section 3.3.10 presents the operations Extend (which pumps a subtree) and Label (which pre-commits a small partial labeling)

in terms of a black-box *labeling function* f. Section 3.3.11 defines the set of all (partially labeled) trees that can be encountered, by considering the interplay between the graph decomposition, Extend, and Label. It is important that for each tree encountered, its partial labeling can be extended to a complete labeling consistent with $\mathcal{P}$; whether this actually holds depends on the choice of black-box f. Section 3.3.12 shows that $\mathcal{P}$ can be solved in $O(\log n)$ time, given a *feasible labeling function* f. Section 3.3.13 shows how a feasible f can be extracted from any $n^{o(1)}$-time algorithm $\mathcal{A}$.

3.3.2 Partial Labeled Graphs

A *partially labeled graph* $\mathcal{G} = (G, \mathcal{L})$ is a graph G together with a function $\mathcal{L} : V(G) \to \Sigma_{\text{out}} \cup \{\perp\}$. The vertices in $\mathcal{L}^{-1}(\perp)$ are *unlabeled*. A *complete labeling* $\mathcal{L}' : V(G) \to \Sigma_{\text{out}}$ for $\mathcal{G}$ is one that labels all vertices and is consistent with $\mathcal{G}$'s partial labeling, i.e., $\mathcal{L}'(v) = \mathcal{L}(v)$ whenever $\mathcal{L}(v) \neq \perp$. A *legal* labeling is a complete labeling that is *locally consistent* for all $v \in V(G)$, i.e., the labeled subgraph induced by $N^r(v)$ is consistent with the LCL $\mathcal{P}$. Here $N^r(v)$ is the set of all vertices within distance r of v.

All graph operations can be extended naturally to partially labeled graphs. For instance, a subgraph of a partially labeled graph $\mathcal{G} = (G, \mathcal{L})$ is a pair $\mathcal{H} = (H, \mathcal{L}')$ such that H is a subgraph of G, and $\mathcal{L}'$ is $\mathcal{L}$ restricted to the domain $V(H)$. With slight abuse of notation, we usually write $\mathcal{H} = (H, \mathcal{L})$.

3.3.3 Graph Surgery

Let $\mathcal{G} = (G, \mathcal{L})$ be a partially labeled graph, and let $\mathcal{H} = (H, \mathcal{L})$ be a subgraph of $\mathcal{G}$. The *poles* of $\mathcal{H}$ are those vertices in $V(H)$ that are adjacent to some vertex in the outside graph $V(G) - V(H)$. We define an operation Replace that surgically removes $\mathcal{H}$ and replaces it with some $\mathcal{H}'$.

Replace. Let $S = (v_1, \ldots, v_p)$ be a list of the poles of $\mathcal{H}$ and let $S = (v'_1, \ldots, v'_p)$ be a designated set of poles in some partially labeled graph $\mathcal{H}'$. The partially labeled graph $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, (\mathcal{H}, S), (\mathcal{H}', S'))$ is constructed as follows. Beginning with $\mathcal{G}$, replace $\mathcal{H}$ with $\mathcal{H}'$, and replace any edge $\{u, v_i\}$, $u \in V(G) - V(H)$, with $\{u, v'_i\}$. If the poles S, S' are clear from context, we may also simply write $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, \mathcal{H}, \mathcal{H}')$. Writing $\mathcal{G}' = (G', \mathcal{L}')$ and $\mathcal{H}' = (H', \mathcal{L}')$, there is a natural 1-1 correspondence between the vertices in $V(G) - V(H)$ and $V(G') - V(H')$. See Figure 3.5.

In the proof of our $n^{o(1)} \to O(\log n)$ speedup thereom we only consider unipolar and bipolar graphs ($p \in \{1, 2\}$) but for maximum generality we define everything w.r.t. graphs having $p \geq 1$ poles.

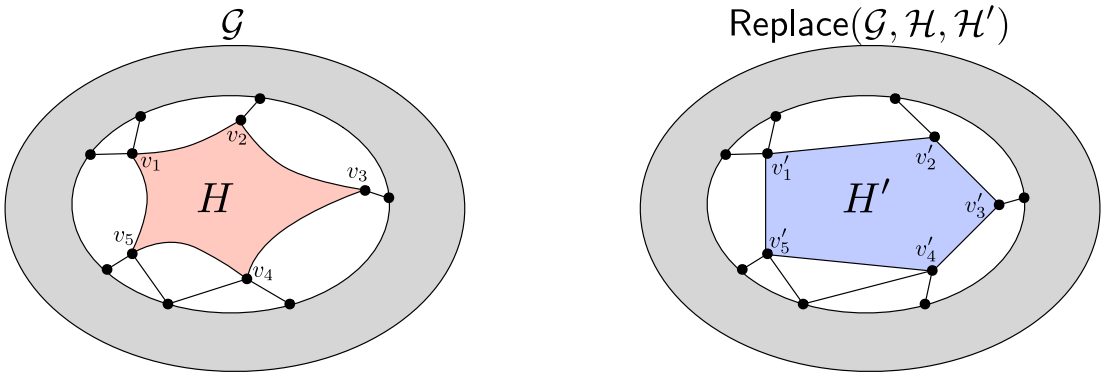

Figure 3.5: The operation Replace.

Given a legal labeling $\mathcal{L}_\diamond$ of $\mathcal{G}$, we would like to know whether there is a legal labeling $\mathcal{L}'_\diamond$ of $\mathcal{G}'$ that agrees with $\mathcal{L}_\diamond$, i.e., $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(v')$ for each $v \in V(G) - V(H)$ and the corresponding $v' \in V(G') - V(H')$. Our goal is to define an equivalence relation $\overset{\star}{\sim}$ on partially labeled graphs (with designated poles) so that the following is true: if $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$, then such a legal labeling $\mathcal{L}'_\diamond$ must exist, regardless of the choice of $\mathcal{G}$ and $\mathcal{L}_\diamond$. Observe that since $\mathcal{P}$ has radius r, the interface between $V(H)$ (or $V(H')$) and the rest of the graph only occurs around the $O(r)$-neighborhoods of the poles of $\mathcal{H}$ (or $\mathcal{H}'$). This motivates us to define a certain partition of $\mathcal{H}$'s vertices that depends on its poles and r.

3.3.4 A Tripartition of the Vertices

Let $\mathcal{H} = (H, \mathcal{L})$ be a partially labeled graph with poles $S = (v_1, \ldots, v_p)$. Define $\xi(\mathcal{H}, S) = (D_1, D_2, D_3)$ to be a tripartition of $V(H)$, where

$$D_1 = \bigcup_{v \in S} N^{r-1}(v),$$
$$D_2 = \bigcup_{v \in D_1} N^r(v) - D_1,$$
$$\text{and } D_3 = V(H) - (D_1 \cup D_2).$$

See Figure 3.6 for an illustration. In the figure, the partition $\xi(\mathcal{H}, S) = (D_1, D_2, D_3)$, D_1 is the set of vertices in $V(H)$ within radius $r-1$ of S, D_2 are those within radius $2r-1$ of S, excluding D_1, and D_3 is the rest of $V(H)$. When $\mathcal{H}$ is embedded in some larger graph $\mathcal{G}$, D_0 denotes the remaining vertices in $V(G) - V(H)$.

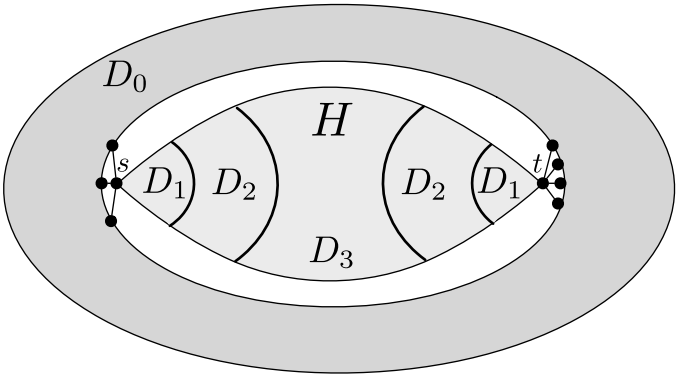

Figure 3.6: A partially labeled subgraph $\mathcal{H}$ with poles $S = (s, t)$, embedded in a larger graph $\mathcal{G}$.

Consider the partition $\xi(\mathcal{H}, S) = (D_1, D_2, D_3)$ of a partially labeled graph $\mathcal{H} = (H, \mathcal{L})$. Let $\mathcal{L}_* : D_1 \cup D_2 \to \Sigma_{\text{out}}$ assign output labels to $D_1 \cup D_2$. We say that $\mathcal{L}_*$ is *extendible* (to all of $V(H)$) if there exists a complete labeling $\mathcal{L}_\diamond$ of H such that $\mathcal{L}_\diamond$ agrees with $\mathcal{L}$ where it is defined, agrees with $\mathcal{L}_*$ on $D_1 \cup D_2$, and is locally consistent with $\mathcal{P}$ on all vertices in $D_2 \cup D_3$.[9]

3.3.5 An Equivalence Relation on Graphs

Consider two partially labeled graphs $\mathcal{H}$ and $\mathcal{H}'$ with poles $S = (v_1, \ldots, v_p)$ and $S' = (v'_1, \ldots, v'_p)$, respectively. Let $\xi(\mathcal{H}, S) = (D_1, D_2, D_3)$ and $\xi(\mathcal{H}', S') = (D'_1, D'_2, D'_3)$. Define $\mathcal{Q} = (Q, \mathcal{L})$ and $\mathcal{Q}' = (Q', \mathcal{L}')$ as the subgraphs of $\mathcal{H}$ and $\mathcal{H}'$ induced by the vertices in $D_1 \cup D_2$ and $D'_1 \cup D'_2$, respectively.

The relation $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$ holds if and only if there is a 1-1 correspondence $\phi : (D_1 \cup D_2) \to (D'_1 \cup D'_2)$ meeting the following conditions.

[9] We are not concerned whether $\mathcal{L}_\diamond$ is consistent with $\mathcal{P}$ for vertices in D_1. Ultimately, $\mathcal{H}$ will be a subgraph of a larger graph $\mathcal{G}$. Since the r-neighborhoods of vertices in D_1 will intersect $V(G) - V(H)$, the labeling of H does not provide enough information to tell if these vertices' r-neighborhoods will be consistent with $\mathcal{P}$. See Figure 3.6.

Isomorphism. The two graphs Q and Q' are isomorphic under ϕ. Moreover, for each $v \in D_1 \cup D_2$ and its corresponding vertex $v' = \phi(v) \in D'_1 \cup D'_2$, (i) $\mathcal{L}(v) = \mathcal{L}'(v')$, (ii) if the underlying LCL problem has input labels, then the input labels of v and v' are the same, and (iii) v is the ith pole in S iff v' is the ith pole in S'.

Extendibility. Let $\mathcal{L}_*$ be *any* assignment of output labels to vertices in $D_1 \cup D_2$ and let $\mathcal{L}'_*$ be the corresponding labeling of $D'_1 \cup D'_2$ under ϕ. Then $\mathcal{L}_*$ is extendible to $V(H)$ if and only if $\mathcal{L}'_*$ is extendible to $V(H')$.

Notice that there could be many 1-1 correspondences between $D_1 \cup D_2$ and $D'_1 \cup D'_2$ that satisfy the isomorphism requirement, though only some subset may satisfy the extendibility requirement due to differences in the topology and partial labeling of D_3 and D'_3. Any ϕ meeting both requirements is a *witness* of the relation $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$.

3.3.6 Properties of the Equivalence Relation

Let us consider the graph $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, (\mathcal{H}, S), (\mathcal{H}', S'))$, and the two partitions $\xi(\mathcal{H}, S) = (D_1, D_2, D_3)$ and $\xi(\mathcal{H}', S') = (D'_1, D'_2, D'_3)$. Let $D_0 = V(G) - V(H)$ and $D'_0 = V(G') - V(H')$ be the remaining vertices in G and G', respectively.

If $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$ then there exists a 1-1 correspondence $\phi : (D_0 \cup D_1 \cup D_2) \to (D'_0 \cup D'_1 \cup D'_2)$ such that (i) ϕ restricted to D_0 is the natural 1-1 correspondence between D_0 and D'_0 and (ii) ϕ restricted to $D_1 \cup D_2$ witnesses the relation $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$. Such a 1-1 correspondence ϕ is called *good*. We have the following lemma.

Lemma 3.1. *Let $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, (\mathcal{H}, S), (\mathcal{H}', S'))$. Consider the two partitions $\xi(\mathcal{H}, S) = (D_1, D_2, D_3)$ and $\xi(\mathcal{H}', S') = (D'_1, D'_2, D'_3)$ and let $D_0 = V(G) - V(H)$ and $D'_0 = V(G') - V(H')$. Suppose that $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$, so there is a* good *1-1 correspondence $\phi : (D_0 \cup D_1 \cup D_2) \to (D'_0 \cup D'_1 \cup D'_2)$. Let $\mathcal{L}_\diamond$ be a complete labeling of $\mathcal{G}$ that is locally consistent for all vertices in $D_2 \cup D_3$. Then there exists a complete labeling $\mathcal{L}'_\diamond$ of $\mathcal{G}'$ such that the following conditions are met.*

Condition 1. *$\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(v')$ for each $v \in D_0 \cup D_1 \cup D_2$ and its corresponding vertex $v' = \phi(v) \in D'_0 \cup D'_1 \cup D'_2$. Moreover, if $\mathcal{L}_\diamond$ is locally consistent for v, then $\mathcal{L}'_\diamond$ is locally consistent for v'.*

Condition 2. *$\mathcal{L}'_\diamond$ is locally consistent for all vertices in $D'_2 \cup D'_3$.*

Proof. We construct $\mathcal{L}'_\diamond$ as follows. First of all, for each $v \in D_0 \cup D_1 \cup D_2$, fix $\mathcal{L}'_\diamond(\phi(v)) = \mathcal{L}_\diamond(v)$. It remains to show how to assign output labels to vertices in D'_3 to meet Conditions 1 and 2.

Let $\mathcal{L}_*$ be $\mathcal{L}_\diamond$ restricted to the domain $D_1 \cup D_2$. Similarly, let $\mathcal{L}'_*$ be $\mathcal{L}'_\diamond$ restricted to $D'_1 \cup D'_2$. Due to the fact that $\mathcal{L}_\diamond$ is locally consistent for all vertices in $D_2 \cup D_3$, the labeling $\mathcal{L}_*$ is extendible to all of $\mathcal{H}$. Since $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$, the labeling $\mathcal{L}'_*$ must also be extendible to all of $\mathcal{H}'$. Thus, we can set $\mathcal{L}'_\diamond(v')$ for all $v' \in D'_3$ in such a way that $\mathcal{L}'_\diamond$ is locally consistent for all vertices in $D'_2 \cup D'_3$. Therefore, Condition 2 is met.

To see that (the second part of) Condition 1 is also met, observe that for $v \in D_0 \cup D_1$, $N^r(v) \subseteq D_0 \cup D_1 \cup D_2$. Therefore, if $\mathcal{L}_\diamond$ is locally consistent for $v \in D_0 \cup D_1$, then $\mathcal{L}'_\diamond$ is locally consistent for $\phi(v)$ since they have the same radius-r neighborhood view. Condition 2 already guarantees that $\mathcal{L}'_\diamond$ is locally consistent for all $v' \in D'_2$.[10] □

Theorem 3.3 provides a user-friendly corollary of Lemma 3.1, which does not mention the tripartition ξ.

Theorem 3.3. *Let $\mathcal{G} = (G, \mathcal{L})$ and $\mathcal{H} = (H, \mathcal{L})$ be a subgraph $\mathcal{G}$. Suppose $\mathcal{H}'$ is a graph for which $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$ and let $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, (\mathcal{H}, S), (\mathcal{H}', S'))$. We write $\mathcal{G}' = (G', \mathcal{L}')$ and $\mathcal{H}' = (H', \mathcal{L}')$. Let $\mathcal{L}_\diamond$ be a complete labeling of $\mathcal{G}$ that is locally consistent for all vertices in H. Then there exists a complete labeling $\mathcal{L}'_\diamond$ of $\mathcal{G}'$ such that the following conditions are met.*

- *For each $v \in V(G) - V(H)$ and its corresponding $v' \in V(G') - V(H')$, we have $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(v')$. Moreover, if $\mathcal{L}_\diamond$ is locally consistent for v, then $\mathcal{L}'_\diamond$ is locally consistent for v'.*
- *$\mathcal{L}'_\diamond$ is locally consistent for all vertices in H'.*

Theorem 3.3 has several useful consequences. If $\mathcal{L}_\diamond$ is a legal labeling of $\mathcal{G}$, then the output labeling $\mathcal{L}'_\diamond$ of $\mathcal{G}'$ guaranteed by Theorem 3.3 is also legal. Observe that setting $\mathcal{G} = \mathcal{H}$ in Theorem 3.3 implies $\mathcal{G}' = \mathcal{H}'$. Suppose that $\mathcal{H}$ admits a legal labeling. For *any* $(\mathcal{H}', S')$ such that $(\mathcal{H}', S') \overset{\star}{\sim} (\mathcal{H}, S)$, the partially labeled graph $\mathcal{H}'$ also admits a legal labeling. Thus, whether $\mathcal{H}$ admits a legal labeling is determined by the equivalence class of $(\mathcal{H}, S)$ (for any choice of S).

[10] It is this lemma that motivates our definition of the *tri*partition $\xi(\mathcal{H}, S)$. It is not clear how an analogue of Lemma 3.1 could be proved using the seemingly more natural *bi*partition, i.e., by collapsing D_1, D_2 into one set.

Roughly speaking, Theorem 3.4 shows that the equivalence class of $(\mathcal{G}, X)$ is preserved after replacing a subgraph $\mathcal{H}$ of $\mathcal{G}$ by another partially labeled graph $\mathcal{H}'$ such that $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$.

Theorem 3.4. *Let $\mathcal{G} = (G, \mathcal{L})$, and let $\mathcal{H} = (H, \mathcal{L})$ be a subgraph of $\mathcal{G}$. Suppose $\mathcal{H}'$ is a graph that satisfies $(\mathcal{H}, S) \overset{\star}{\sim} (\mathcal{H}', S')$ for some pole lists S, S'. Let $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, (\mathcal{H}, S), (\mathcal{H}', S'))$ be a partially labeled graph. Designate a set $X \subseteq (V(G) - V(H)) \cup S$ as the poles of $\mathcal{G}$, listed in some order, and let X' be the corresponding list of vertices in $\mathcal{G}'$. It follows that $(\mathcal{G}, X) \overset{\star}{\sim} (\mathcal{G}', X')$.*

Proof. Consider the partitions $\xi(\mathcal{H}, S) = (B_1, B_2, B_3)$, $\xi(\mathcal{H}', S') = (B'_1, B'_2, B'_3)$, $\xi(\mathcal{G}, X) = (D_1, D_2, D_3)$, and $\xi(\mathcal{G}', X') = (D'_1, D'_2, D'_3)$. We write $B_0 = V(G) - V(H)$ and $B'_0 = V(G') - V(H')$. Let ϕ be any good 1-1 correspondence from $B_0 \cup B_1 \cup B_2$ to $B'_0 \cup B'_1 \cup B'_2$. Because $X \subseteq B_0 \cup S$, we have $D_1 \cup D_2 \subseteq B_0 \cup B_1 \cup B_2$ and $D'_1 \cup D'_2 \subseteq B'_0 \cup B'_1 \cup B'_2$. To show that $(\mathcal{G}, X) \overset{\star}{\sim} (\mathcal{G}', X')$, it suffices to prove that ϕ (restricted to the domain $D_1 \cup D_2$) is a witness to the relation $(\mathcal{G}, X) \overset{\star}{\sim} (\mathcal{G}', X')$.

Let $\mathcal{L}_* : (D_1 \cup D_2) \to \Sigma_{\text{out}}$ and $\mathcal{L}'_*$ be the corresponding labeling of $D'_1 \cup D'_2$. All we need to do is show that $\mathcal{L}_*$ is extendible to all of $V(G)$ if and only if $\mathcal{L}'_*$ is extendible to all of $V(G')$. Since we can also write $\mathcal{G} = \mathsf{Replace}(\mathcal{G}', (\mathcal{H}', S'), (\mathcal{H}, S))$, it suffices to show just one direction, i.e., if $\mathcal{L}_*$ is extendible then $\mathcal{L}'_*$ is extendible.

Suppose that $\mathcal{L}_*$ is extendible. Then there exists an output labeling $\mathcal{L}_\diamond$ of $\mathcal{G}$ such that (i) for each $v \in D_1 \cup D_2$, we have $\mathcal{L}_*(v) = \mathcal{L}_\diamond(v)$, and (ii) $\mathcal{L}_\diamond$ is locally consistent for all vertices in $D_2 \cup D_3$. Observe that $D_2 \cup D_3 \supseteq B_2 \cup B_3$. By Lemma 3.1, there exists a complete labeling $\mathcal{L}'_\diamond$ of $\mathcal{G}'$ such that the two conditions in Lemma 3.1 are met. We show that this implies that $\mathcal{L}'_*$ is extendible.

Lemma 3.1 guarantees that $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(\phi(v))$ for each $v \in B_0 \cup B_1 \cup B_2$ and its corresponding vertex $\phi(v) \in B'_0 \cup B'_1 \cup B'_2$. Since $D'_1 \cup D'_2 \subseteq B'_0 \cup B'_1 \cup B'_2$, we have $\mathcal{L}'_*(v') = \mathcal{L}'_\diamond(v')$ for each $v' \in D'_1 \cup D'_2$.

Since $\mathcal{L}_\diamond$ is locally consistent for all vertices in $D_2 \cup D_3$, Lemma 3.1 guarantees that $\mathcal{L}'_\diamond$ is locally consistent for all vertices in $D'_2 \cup D'_3$. More precisely, due to Condition 1, $\mathcal{L}'_\diamond$ is locally consistent for all vertices in $(D'_2 \cup D'_3) - B'_3$; due to Condition 2, $\mathcal{L}'_\diamond$ is locally consistent for all vertices in $B'_2 \cup B'_3$.

Thus, $\mathcal{L}'_*$ is extendible, as the complete labeling $\mathcal{L}'_\diamond$ of $\mathcal{G}'$ satisfies: (i) for each $v' \in D'_1 \cup D'_2$, we have $\mathcal{L}'_*(v') = \mathcal{L}'_\diamond(v')$, and (ii) $\mathcal{L}'_\diamond$ is locally consistent for all vertices in $D'_2 \cup D'_3$. □

3.3.7 The Number of Equivalence Classes

An important feature of $\overset{\star}{\sim}$ is that it has a *constant* number of equivalence classes, for any fixed number p of poles. Which constant is not important, but we shall work out an upper bound nonetheless.[11]

Consider a partially labeled graph $\mathcal{H}$ with poles $S = (v_1, \ldots, v_p)$. Let $\xi(\mathcal{H}, S) = (D_1, D_2, D_3)$ and define $\mathcal{Q} = (Q, \mathcal{L})$ to be the subgraph of $\mathcal{H}$ induced by $D_1 \cup D_2$. Observe that the equivalence class of $(\mathcal{H}, S)$ is determined by (i) the topology of Q (including its input labels from Σ_{in}, if $\mathcal{P}$ has input labels), (ii) the locations of the poles $S \subseteq V(Q)$ in Q, and (iii) the subset of all output labelings of $V(Q) = D_1 \cup D_2$ that are extendible.

The number of vertices in $D_1 \cup D_2$ is at most $p\Delta^{2r}$. The total number of distinct graphs of at most $p\Delta^{2r}$ vertices (with input labels from Σ_{in} and a set of p designated poles) is at most $2^{\binom{p\Delta^{2r}}{2}} |\Sigma_{\text{in}}|^{p\Delta^{2r}}$. The total number of output labelings of $D_1 \cup D_2$ is at most $|\Sigma_{\text{out}}|^{p\Delta^{2r}}$. Therefore, the total number of equivalence classes of graphs with p poles is at most $2^{\binom{p\Delta^{2r}}{2}} |\Sigma_{\text{in}}|^{p\Delta^{2r}} 2^{|\Sigma_{\text{out}}|^{p\Delta^{2r}}}$, which is constant whenever $\Delta, r, |\Sigma_{\text{in}}|, |\Sigma_{\text{out}}|$, and p are.

3.3.8 A Pumping Lemma for Trees

In this section we consider partially labeled trees with one and two poles; they are called *unipolar* (or *rooted*) and *bipolar*, respectively. Let $\mathcal{T} = (T, \mathcal{L})$ be a unipolar tree with pole list $S = (z)$, $z \in V(T)$ being the root. Define $\mathsf{Class}(\mathcal{T})$ to be the equivalence class of $(\mathcal{T}, S)$ w.r.t. $\overset{\star}{\sim}$. Notice that whether a partially labeled rooted tree $\mathcal{T}$ admits a legal labeling is determined by $\mathsf{Class}(\mathcal{T})$ (Theorem 3.3). We say that a class is *good* if each partially labeled rooted tree in the class admits a legal labeling; otherwise the class is *bad*. We write $\mathscr{C}$ to denote the set of all classes. Notice that $|\mathscr{C}|$ is constant. The following lemma is a specialization of Theorem 3.4.

Lemma 3.2. *Let $\mathcal{T}$ be a partially labeled rooted (unipolar) tree, and let $\mathcal{T}'$ be a rooted subtree of $\mathcal{T}$, whose leaves are also leaves of $\mathcal{T}$. Let $\mathcal{T}''$ be another partially labeled rooted tree such that $\mathsf{Class}(\mathcal{T}') = \mathsf{Class}(\mathcal{T}'')$. Then replacing $\mathcal{T}'$ with $\mathcal{T}''$ does not alter the class of $\mathcal{T}$.*

Let $\mathcal{H} = (H, \mathcal{L})$ be a bipolar tree with poles $S = (s, t)$. The unique oriented path in H

[11] For the sake of simplicity, in the calculation we assume that the underlying LCL problem does not refer to port-numbering. It is straightforward to see that even if port-numbering is taken into consideration, the number of equivalence classes (for any fixed p) is still a constant.

from s to t is called the *core path* of $\mathcal{H}$. It is more convenient to express a bipolar tree as a *sequence of rooted/unipolar trees*, as follows. The partially labeled bipolar tree $\mathcal{H} = (\mathcal{T}_i)_{i\in[k]}$ is formed by arranging the roots of unipolar trees $(\mathcal{T}_i)$ into a path $P = (v_1, \ldots, v_k)$, where v_i is the root/pole of $\mathcal{T}_i$. The two poles of $\mathcal{H}$ are $s = v_1$ and $t = v_k$, so P is the core path of $\mathcal{H}$. Define $\mathsf{Type}(\mathcal{H})$ as the equivalence class of $(\mathcal{H}, S = (s,t))$ w.r.t. $\overset{\star}{\sim}$. The following lemma follows from Theorem 3.4.

Lemma 3.3. *Let $\mathcal{H}$ be a partially labeled bipolar tree with poles (s,t). Let $\mathcal{T}$ be $\mathcal{H}$, but regarded as a unipolar tree rooted at s. Then $\mathsf{Class}(\mathcal{T})$ is determined by $\mathsf{Type}(\mathcal{H})$. If we write $\mathcal{H} = (\mathcal{T}_i)_{i\in[k]}$, then $\mathsf{Type}(\mathcal{H})$ is determined by $\mathsf{Class}(\mathcal{T}_1), \ldots, \mathsf{Class}(\mathcal{T}_k)$.*

Let $\mathcal{G} = (G, \mathcal{L})$ be a partially labeled graph, and let $\mathcal{H} = (H, \mathcal{L})$ be a bipolar subtree of $\mathcal{G}$ with poles (s,t). Let $\mathcal{H}'$ be another partially labeled bipolar tree. Recall that $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, \mathcal{H}, \mathcal{H}')$ is defined as the partially labeled graph resulting from replacing the subgraph $\mathcal{H}$ with $\mathcal{H}'$ in $\mathcal{G}$. We write $\mathcal{G}' = (G', \mathcal{L}')$ and $\mathcal{H}' = (H', \mathcal{L}')$. The following lemmas follow from Theorems 3.3 and 3.4.

Lemma 3.4. *Consider $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, \mathcal{H}, \mathcal{H}')$. If $\mathsf{Type}(\mathcal{H}') = \mathsf{Type}(\mathcal{H})$ and $\mathcal{G}$ admits a legal labeling $\mathcal{L}_\diamond$, then $\mathcal{G}'$ admits a legal labeling $\mathcal{L}'_\diamond$ such that $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(v')$ for each vertex $v \in V(G) - V(H)$ and its corresponding $v' \in V(G') - V(H')$.*

Lemma 3.5. *Suppose that $\mathcal{G} = (\mathcal{T}_i)_{i\in[k]}$ is a partially labeled bipolar tree, $\mathcal{H} = (\mathcal{T}_i, \ldots, \mathcal{T}_j)$ is a bipolar subtree of $\mathcal{G}$, and $\mathcal{H}'$ is some other partially labeled bipolar tree with $\mathsf{Type}(\mathcal{H}') = \mathsf{Type}(\mathcal{H})$. Then $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, \mathcal{H}, \mathcal{H}')$ is a partially labeled bipolar tree and $\mathsf{Type}(\mathcal{G}') = \mathsf{Type}(\mathcal{G})$.*

Lemma 3.6. *Let $\mathcal{H} = (\mathcal{T}_i)_{i\in[k]}$ and $\mathcal{H}' = (\mathcal{T}_i)_{i\in[k+1]}$ be identical to $\mathcal{H}$ in its first k trees. Then $\mathsf{Type}(\mathcal{H}')$ is a function of $\mathsf{Type}(\mathcal{H})$ and $\mathsf{Class}(\mathcal{T}_{k+1})$.*

Lemma 3.6 is what allows us to bring classical automata theory into play. Suppose that we somehow computed and stored $c_i = \mathsf{Class}(\mathcal{T}_i)$ at the root of $\mathcal{T}_i$. Lemma 3.6 implies that a finite automaton walking along the core path of $\mathcal{H}' = (\mathcal{T}_i)_{i\in[k+1]}$, can compute $\mathsf{Type}(\mathcal{H}')$, by reading the vector $(c_1, \ldots, c_{k+1})$ one character at a time. The number of states in the finite automaton depends only on the number of types (which is constant) and is independent of $k+1$ and the size of the individual trees $(\mathcal{T}_i)$. Define $\ell_{\text{pump}} = O(1)$ as the number of states in this finite automaton. The following *pumping lemma* for bipolar trees is analogous to the pumping lemma for regular languages.

Lemma 3.7. *Let $\mathcal{H} = (\mathcal{T}_1, \ldots, \mathcal{T}_k)$, with $k \geq \ell_{\text{pump}}$. We regard each $\mathcal{T}_i$ in the string notation $\mathcal{H} = (\mathcal{T}_1, \ldots, \mathcal{T}_k)$ as a character. Then $\mathcal{H}$ can be decomposed into three substrings $\mathcal{H} = x \circ y \circ z$ such that (i) $|xy| \leq \ell_{\text{pump}}$, (ii) $|y| \geq 1$, and (iii) $\mathsf{Type}(x \circ y^j \circ z) = \mathsf{Type}(\mathcal{H})$ for each non-negative integer j.*

We will use Lemma 3.7 to expand the length of the core path of a bipolar tree to be close to a desired *target length* w. The specification for the function Pump is as follows.

Pump. Let $\mathcal{H} = (\mathcal{T}_i)_{i \in [k]}$ be a partially labeled bipolar tree with $k \geq \ell_{\text{pump}}$. The function $\mathsf{Pump}(\mathcal{H}, w)$ produces a partially labeled bipolar tree $\mathcal{H}' = (\mathcal{T}'_i)_{i \in [k']}$ such that (i) $\mathsf{Type}(\mathcal{H}) = \mathsf{Type}(\mathcal{H}')$, (ii) $k' \in [w, w + \ell_{\text{pump}}]$, and (iii) if we let $Z = \{\mathcal{T}_i\}_{i \in [k]}$ (resp., $Z' = \{\mathcal{T}'_i\}_{i \in [k']}$) be the *set* of rooted trees appearing in the tree list of $\mathcal{H}$ (resp., $\mathcal{H}'$), then $Z' = Z$.

By Lemma 3.7, such a function Pump exists.

3.3.9 Rake & Compress Graph Decomposition

In this section we describe an $O(\log n)$-round DetLOCAL algorithm to decompose the vertex set $V(G)$ of a tree into the disjoint union $V_1 \cup \cdots \cup V_L$, $L = O(\log n)$. Our algorithm is inspired by Miller and Reif's *parallel tree contraction* [109]. We first describe the decomposition algorithm then analyze its properties.

Fix the constant $\ell = 2(r + \ell_{\text{pump}})$, where r and ℓ_{pump} depend on the LCL problem $\mathcal{P}$. In the *postprocessing* step of the decomposition algorithm we compute an $(\ell, 2\ell)$-*independent set*, in $O(\log^* n)$ time [107], defined as follows.

Definition 3.1. *Let P be a path. A subset $I \subset V(P)$ is called an* (α, β)-independent set *if the following conditions are met: (i) I is an independent set, and I does not contain either endpoint of P, and (ii) each connected component induced by $V(P) - I$ has at least α vertices and at most β vertices, unless $|V(P)| < \alpha$, in which case $I = \emptyset$.*

The Decomposition Algorithm. The algorithm begins with $U = V(G)$ and $i = 1$, repeats Steps 1–3 until $U = \emptyset$, then executes the *Postprocessing* step.

1. For each $v \in U$:

a) Compress. If v belongs to a path P such that $|V(P)| \geq \ell$ and $\deg_U(u) = 2$ for each $u \in V(P)$, then tag v with i_C.

b) Rake. If $\deg_U(v) = 0$, then tag v with i_R. If $\deg_U(v) = 1$ and the unique neighbor u of v in U satisfies either (i) $\deg_U(u) > 1$ or (ii) $\deg_U(u) = 1$ and $\mathrm{ID}(v) > \mathrm{ID}(u)$, then tag v with i_R.

2. Remove from U all vertices tagged i_C or i_R.

3. $i \leftarrow i + 1$.

Postprocessing Step. Initialize V_i as the set of all vertices tagged i_C or i_R. At this point the graph induced by V_i consists of *unbounded length* paths, but we prefer constant length paths. For each edge $\{u, v\}$ such that v is tagged i_R and u is tagged i_C, promote v from V_i to V_{i+1}. For each path P that is a connected component induced by vertices tagged i_C, compute an $(\ell, 2\ell)$-independent set I_P of P, and then promote every vertex in I_P from V_i to V_{i+1}. Notice that the set V_i in the graph decomposition is analogous to (but clearly different from) the set V_i defined in the Hierarchical $2\frac{1}{2}$-coloring problem from Section 3.2.

Properties of the Decomposition. As we show below, $L = O(\log n)$ iterations suffice, i.e., $V(G) = V_1 \cup \cdots \cup V_L$. The following properties are easily verified.

- Define G_i as the graph induced by vertices at level i or above: $\bigcup_{j=i}^{L} V_j$. For each $v \in V_i$, $\deg_{G_i}(v) \leq 2$.
- Define $\mathscr{P}_i$ as the set of connected components (paths) induced by vertices in V_i that contain more than one vertex. For each $P \in \mathscr{P}_i$, $\ell \leq |V(P)| \leq 2\ell$ and $\deg_{G_i}(v) = 2$ for each vertex $v \in V(P)$.
- The graph G_L contains only isolated vertices, i.e., $\mathscr{P}_L = \emptyset$.

As a consequence, each vertex $v \in V_i$ falls into exactly one of two cases: (i) v has $\deg_{G_i}(v) \leq 1$ and has no neighbor in V_i, or (ii) v has $\deg_{G_i}(v) = 2$ and is in some path $P \in \mathscr{P}_i$.

Analysis. We prove that for $L = O(\log_{1+1/\ell} n) = O(\log n)$, L iterations of the graph decomposition routine suffices to decompose any n-vertex tree. Each iteration of the routine takes $O(1)$ time, and the $(\ell, 2\ell)$-independent set computation at the end takes $O(\log^* n)$ time, so $O(\log n)$ time suffices in DetLOCAL.

Let W be the vertices of a connected component induced by U at the beginning of the ith iteration. In general, the graph induced by U is a forest, but it is simpler to analyze a single connected component W. We claim that at least a constant $\Omega(1/\ell)$ fraction of vertices in W are eliminated (i.e., tagged i_C or i_R) in the ith iteration. The proof of the claim is easy for the special case of $\ell = 1$, as follows. If W is not a single edge, then all $v \in W$ with $\deg_U(v) \leq 2$ are eliminated. Since the degree of at least half of the vertices in a tree is at most 2, the claim follows. In general, degree-2 paths of length less than ℓ are not eliminated quickly. If one endpoint of such a path is a leaf, vertices in the path are peeled off by successive *Rake* steps.

Assume w.l.o.g. that $|W| > 2(\ell + 1)$. Define $W_1 = \{v \in W \mid \deg_U(v) = 1\}$, $W_2 = \{v \in W \mid \deg_U(v) = 2\}$, and $W_3 = \{v \in W \mid \deg_U(v) \geq 3\}$.

Case 1: $|W_2| \geq \frac{\ell|W|}{\ell+1}$. The number of connected components induced by vertices in W_2 is at most $|W_1| + |W_3| - 1 < \frac{|W|}{\ell+1}$. The number of vertices in W_2 that are not tagged i_C during Compress is less than $\frac{(\ell-1)|W|}{\ell+1}$. Therefore, at least $\frac{\ell|W|}{\ell+1} - \frac{(\ell-1)|W|}{\ell+1} = \frac{|W|}{\ell+1}$ vertices are tagged i_C by Compress.

Case 2: $|W_2| < \frac{\ell|W|}{\ell+1}$. In any tree $|W_1| > |W_3|$, so $|W_1| > \frac{|W_1|+|W_3|}{2} = \frac{|W|-|W_2|}{2} \geq \frac{|W|}{2(\ell+1)}$. Therefore, at least $\frac{|W|}{2(\ell+1)}$ vertices are tagged i_R by Rake.

Hence the claim follows.

3.3.10 Extend and Label Operations

In this section we define three operations Extend, Label, and Duplicate-Cut which are used extensively in Sections 3.3.11 and 3.3.12. All these operations are graph-theoretic operations, and they are not implemented in a distributed manner.

The operation Extend is parameterized by a target length $w \geq \ell = 2(r + \ell_{\text{pump}})$. The operation Label is parameterized by a function f which takes a partially labeled bipolar tree $\mathcal{H}$ as input, and assigns output labels to the vertices in $v \in N^{r-1}(e)$, where e is the middle edge in the core path of $\mathcal{H}$.[12] The function f will be constructed in Section 3.3.13.

[12]By definition, if $e = \{x, y\}$ then $N^{r-1}(e) = N^{r-1}(x) \cup N^{r-1}(y)$.

Let $\mathcal{H} = (\mathcal{T}_1, \ldots, \mathcal{T}_x)$ be a partially labeled bipolar tree with $x \geq \ell$. Let $(v_1, \ldots, v_x)$ be the core path of $\mathcal{H}$ and $e = \{v_{\lfloor x/2 \rfloor}, v_{\lfloor x/2 \rfloor + 1}\}$ be the middle edge of the core path. It is guaranteed that all vertices in $N^{r-1}(e)$ in $\mathcal{H}$ are not already assigned output labels. The partially labeled bipolar tree $\mathcal{H}' = \mathsf{Label}(\mathcal{H})$ is defined as the result of assigning output labels to vertices in $N^{r-1}(e)$ by the function f.[13]

Extend. Let $\mathcal{H} = (\mathcal{T}_1, \ldots, \mathcal{T}_x)$ be a partially labeled bipolar tree with $x \in [\ell, 2w]$. The partially labeled bipolar tree $\mathcal{H}' = \mathsf{Extend}(\mathcal{H})$ is defined as follows. Consider the decomposition $\mathcal{H} = \mathcal{X} \circ \mathcal{Y} \circ \mathcal{Z}$, where $\mathcal{Y} = (\mathcal{T}_{\lfloor x/2 \rfloor - r + 1}, \ldots, \mathcal{T}_{\lfloor x/2 \rfloor + r})$. Then $\mathcal{H}' = \mathsf{Pump}(\mathcal{X}, w) \circ \mathcal{Y} \circ \mathsf{Pump}(\mathcal{Z}, w)$.

Intuitively, the goal of the operation Extend is to extend the length of the core path of $\mathcal{H}$ while preserving the type of $\mathcal{H}$, due to Lemma 3.5. Suppose that the number of vertices in the core path of $\mathcal{H}$ is in the range $[\ell, 2\ell]$. The prefix $\mathcal{X}$ and suffix $\mathcal{Z}$ are stretched to lengths in the range $[w, w + \ell_{\text{pump}}]$, and the middle part $\mathcal{Y}$ has length $2r$, so the core path of $\mathcal{H}'$ has length in the range $[2(w + r), 2(w + r + \ell_{\text{pump}})]$.

The reason that the Extend operation does not modify the middle part $\mathcal{Y}$ is to ensure that (given any labeling function f) the type of $\mathcal{H}' = \mathsf{Extend}(\mathsf{Label}(\mathcal{H}))$ is invariant over all choices of the parameter w.[14] We have the following lemma.

Lemma 3.8. *Let $\mathcal{G} = (G, \mathcal{L})$ be a partially labeled graph and $\mathcal{H} = (H, \mathcal{L})$ be a bipolar subtree of $\mathcal{G}$ with poles (s, t). Let $\tilde{\mathcal{H}}$ be another partially labeled bipolar tree with $\mathsf{Type}(\tilde{\mathcal{H}}) = \mathsf{Type}(\mathcal{H})$ and $\mathcal{H}' = \mathsf{Extend}(\mathsf{Label}(\tilde{\mathcal{H}}))$. If $\mathcal{G}' = \mathsf{Replace}(\mathcal{G}, \mathcal{H}, \mathcal{H}')$ admits a legal labeling $\mathcal{L}'_\diamond$, then $\mathcal{G}$ admits a legal labeling $\mathcal{L}_\diamond$ such that $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(v')$ for each vertex $v \in V(G) - V(H)$ and its corresponding vertex $v' \in V(G') - V(H')$.*

Proof. Recall that the operation Extend guarantees that

$$\mathsf{Type}(\mathsf{Extend}(\tilde{\mathcal{H}})) = \mathsf{Type}(\tilde{\mathcal{H}}) = \mathsf{Type}(\mathcal{H}).$$

Define $\mathcal{H}'' = \mathsf{Extend}(\tilde{\mathcal{H}})$ and $\mathcal{G}'' = \mathsf{Replace}(\mathcal{G}, \mathcal{H}, \mathcal{H}'')$. Observe that the graph $\mathcal{H}' = \mathsf{Extend}(\mathsf{Label}(\tilde{\mathcal{H}}))$ can be seen as the result of fixing the output labels of some unlabeled

[13] Note that the neighborhood function is evaluated w.r.t. H. In particular, the set $N^{r-1}(e)$ contains the vertices $v_{\lfloor x/2 \rfloor - r + 1}, \ldots, v_{\lfloor x/2 \rfloor + r}$ of the core path, and also contains parts of the trees $\mathcal{T}_{\lfloor x/2 \rfloor - r + 1}, \ldots, \mathcal{T}_{\lfloor x/2 \rfloor + r}$.

[14] Notice that Extend is applied *after* Label. Thus, the vertices that are assigned output labels during Label must be within the middle part $\mathcal{Y}$, no part of which is modified during Extend.

vertices in $\mathcal{H}'' = \mathsf{Extend}(\tilde{\mathcal{H}})$. Therefore, $\mathcal{L}'_\diamond$ is also a legal labeling of $\mathcal{G}''$. By Lemma 3.4, the desired legal labeling $\mathcal{L}_\diamond$ of $\mathcal{G} = \mathsf{Replace}(\mathcal{G}'', \mathcal{H}'', \mathcal{H})$ can be obtained from the legal labeling $\mathcal{L}'_\diamond$ of $\mathcal{G}''$. □

In addition to Extend and Label, we also modify trees using the Duplicate-Cut operation, defined below.

Duplicate-Cut. Let $\mathcal{G} = (G, \mathcal{L})$ be a partially labeled graph and $\mathcal{H} = (H, \mathcal{L})$ be a bipolar subtree with poles (s, t). Suppose that $\mathcal{H}$ is connected to the rest of $\mathcal{G}$ via two edges $\{u, s\}$ and $\{v, t\}$. The partially labeled graph $\mathcal{G}' = \mathsf{Duplicate\text{-}Cut}(\mathcal{G}, \mathcal{H})$ is formed by (i) duplicating $\mathcal{H}$ and the edges $\{u, s\}, \{v, t\}$ so that u and v are attached to both copies of $\mathcal{H}$, (ii) removing the edge that connects u to one copy of $\mathcal{H}$, and removing the edge from v to the other copy of $\mathcal{H}$.

Later on we will see that both poles of a bipolar tree are responsible for computing the labeling of the tree. On the other hand, we do not want the poles to have to communicate too much. As Lemma 3.9 shows, the Duplicate-Cut operation (in conjunction with Extend and Label) allows both poles to work independently and cleanly integrate their labelings afterward.

Lemma 3.9. *Let $\mathcal{H} = \mathsf{Extend}(\mathsf{Label}(\tilde{\mathcal{H}}))$ for some partially labeled bipolar tree $\tilde{\mathcal{H}}$. If $\mathcal{G}' = \mathsf{Duplicate\text{-}Cut}(\mathcal{G}, \mathcal{H})$ admits a legal labeling $\mathcal{L}'_\diamond$, then $\mathcal{G}$ admits a legal labeling $\mathcal{L}_\diamond$ such that $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(v')$ for each vertex $v \in V(G) - V(H)$ and a particular corresponding vertex v' in $\mathcal{G}'$.*

Proof. Let $\mathcal{G}' = (G', \mathcal{L}')$. We write $\mathcal{H} = (\mathcal{T}_1, \ldots, \mathcal{T}_x)$. Let $(v_1, \ldots, v_x)$ be the core path of $\mathcal{H}$, where $s = v_1$ and $t = v_x$ are the two poles of $\mathcal{H}$. Let $\{u, s\}$ and $\{v, t\}$ be the two edges that connect H two the rest of G. Let $e = \{v_j, v_{j+1}\}$ be the edge in the core path of $\mathcal{H}$ such that the output labels of vertices in $N^{r-1}(e)$ in $\mathcal{H}$ were fixed by Label.[15] We write $\mathcal{H}_u$ (resp., $\mathcal{H}_v$) to denote the copy of $\mathcal{H}$ in $\mathcal{G}'$ that attaches to u (resp., v). Define a mapping ϕ from $V(G)$ to $V(G')$ as follows.

- For $z \in V(G) - V(H)$, $\phi(z)$ is the corresponding vertex in G'.
- For $z \in \bigcup_{i=1}^{j} \mathcal{T}_i$, $\phi(z)$ is the corresponding vertex in H_u.

[15] Since Pump usually does not extend $\mathcal{X}$ and $\mathcal{Z}$ by precisely the same amount, the edge e is generally not *exactly* in the middle.

- For $z \in \bigcup_{i=j+1}^{x} \mathcal{T}_i$, $\phi(z)$ is the corresponding vertex in H_v.

We set $\mathcal{L}_\diamond(z) = \mathcal{L}'_\diamond(\phi(z))$ for each $z \in V(G)$. It is straightforward to verify that the distance-r neighborhood view (with output labeling $\mathcal{L}_\diamond$) of each vertex $z \in V(G)$ is the same as the distance-r neighborhood view (with output labeling $\mathcal{L}'_\diamond$) of its corresponding vertex $\phi(z)$ in G'. Thus, $\mathcal{L}_\diamond$ is a legal labeling. □

Notice that in the proof of Lemma 3.9, the only property of $\mathcal{H}$ that we use is that $N^{r-1}(e)$ was assigned output labels in the application of $\mathsf{Label}(\tilde{\mathcal{H}})$.

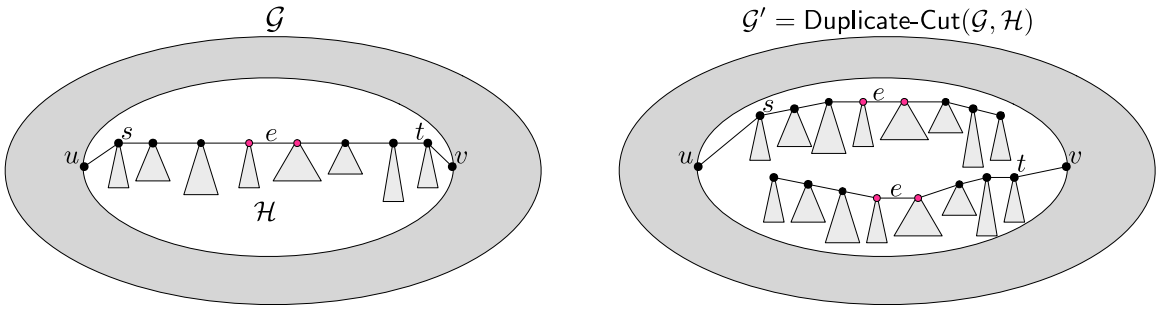

Figure 3.7: Illustration of the Duplicate-Cut operation.

See Figure 3.7 for an illustration of the Duplicate-Cut operation. **Left:** A bipolar subtree $\mathcal{H}$ is attached to the rest of the graph $\mathcal{G}$ via edges $\{u, s\}, \{v, t\}$. The pink vertices have been pre-committed to output labels by Label ($r = 1$). **Right:** The Duplicate-Cut operation duplicates $\mathcal{H}$ and attaches one copy to u and the other to v.

3.3.11 A Hierarchy of Partially Labeled Trees

In this section we construct several sets of partially labeled unipolar and bipolar trees—$\{\mathscr{T}_i\}$, $\{\mathscr{H}_i\}$, and $\{\mathscr{H}_i^+\}$, $i \in \mathbb{Z}^+$—using the operations Extend and Label. If each member of $\mathscr{T}^\star = \bigcup_i \mathscr{T}_i$ admits a legal labeling, then we can use these trees to design an $O(\log n)$-time DetLOCAL algorithm for $\mathcal{P}$. Each $\mathcal{T} \in \mathscr{T}^\star$ is partially labeled in the following restricted manner. The tree $\mathcal{T} = (T, \mathcal{L})$ has a set of *designated edges* such that $\mathcal{L}(v) \neq \perp$ is defined if and only if $v \in N^{r-1}(e)$ for some designated edge e; these vertices were issued labels by some invocation of Label.

The sets of bipolar trees $\{\mathscr{H}_i\}_{i \in \mathbb{Z}^+}$ and $\{\mathscr{H}_i^+\}_{i \in \mathbb{Z}^+}$ and unipolar trees $\{\mathscr{T}_i\}_{i \in \mathbb{Z}^+}$ are defined inductively. In the base case we have $\mathscr{T}_1 = \{\mathcal{T}\}$, where $\mathcal{T}$ is the unique unlabeled, single-vertex, unipolar tree.

- $\mathscr{T}$ **Sets:** For each $i > 1$, $\mathscr{T}_i$ consists of all partially labeled rooted trees $\mathcal{T}$ formed in the following manner. The root z of $\mathcal{T}$ has degree $0 \leq \deg(z) \leq \Delta$. Each child of z is either (i) the root of a partially labeled rooted tree $\mathcal{T}'$ from $\mathscr{T}_{i-1}$ (having degree at most $\Delta - 1$ in $\mathcal{T}'$), or (ii) one of the two poles of a bipolar tree $\mathcal{H}$ from $\mathscr{H}_{i-1}^+$.
- $\mathscr{H}$ **Sets:** For each $i \geq 1$, $\mathscr{H}_i$ contains all partially labeled bipolar trees $\mathcal{H} = (\mathcal{T}_j)_{j \in [x]}$ such that $x \in [\ell, 2\ell]$, and for each $j \in [x]$, $\mathcal{T}_j \in \mathscr{T}_i$, where the root of $\mathcal{T}_j$ has degree at most $\Delta - 2$ in $\mathcal{T}_j$. For example, since $\mathscr{T}_1$ contains only the single-vertex unlabeled tree, $\mathscr{H}_1$ is the set of all bipolar, unlabeled paths with between ℓ and 2ℓ vertices.
- $\mathscr{H}^+$ **Sets:** For each $i \geq 1$, $\mathscr{H}_i^+$ is constructed by the following procedure. If $i = 1$, initialize $\mathscr{H}_1^+ \leftarrow \emptyset$, otherwise initialize $\mathscr{H}_i^+ \leftarrow \mathscr{H}_{i-1}^+$. Consider each $\mathcal{H} \in \mathscr{H}_i$ in some canonical order. If there does *not* already exist a partially labeled bipolar tree $\tilde{\mathcal{H}}$ such that $\mathsf{Type}(\tilde{\mathcal{H}}) = \mathsf{Type}(\mathcal{H})$ and $\mathsf{Extend}(\mathsf{Label}(\tilde{\mathcal{H}})) \in \mathscr{H}_i^+$, then update $\mathscr{H}_i^+ \leftarrow \mathscr{H}_i^+ \cup \{\mathsf{Extend}(\mathsf{Label}(\mathcal{H}))\}$.

Observe that whereas $\{\mathscr{T}_i\}$ and $\{\mathscr{H}_i\}$ grow without end, and contain arbitrarily large trees, the cardinality of $\mathscr{H}_i^+$ is at most the total number of types, which is constant.[16] This is due to the observation that whenever we add a new partially labeled bipolar tree $\mathsf{Extend}(\mathsf{Label}(\mathcal{H}))$ to $\mathscr{H}_i^+$, it is guaranteed that there is no other partially labeled bipolar tree $\mathsf{Extend}(\mathsf{Label}(\tilde{\mathcal{H}})) \in \mathscr{H}_i^+$ such that $\mathsf{Type}(\tilde{\mathcal{H}}) = \mathsf{Type}(\mathcal{H})$. The property that $|\mathscr{H}_i^+|$ is constant is crucial in the proof of Lemma 3.17. Lemmas 3.10–3.13 reveal some useful properties of these sets.

Lemma 3.10. *We have (i) $\mathscr{T}_1 \subseteq \mathscr{T}_2 \subseteq \cdots$, (ii) $\mathscr{H}_1 \subseteq \mathscr{H}_2 \subseteq \cdots$, and (iii) $\mathscr{H}_1^+ \subseteq \mathscr{H}_2^+ \subseteq \cdots$.*

Proof. By construction, we already have $\mathscr{H}_1^+ \subseteq \mathscr{H}_2^+ \subseteq \cdots$. Due to the construction of $\mathscr{H}_i$ from the set $\mathscr{T}_i$, it is guaranteed that if $\mathscr{T}_j \subseteq \mathscr{T}_{j+1}$ holds then $\mathscr{H}_j \subseteq \mathscr{H}_{j+1}$ holds as well. Thus, it suffices to show that $\mathscr{T}_1 \subseteq \mathscr{T}_2 \subseteq \cdots$. This is proved by induction.

For the base case, we have $\mathscr{T}_1 \subseteq \mathscr{T}_2$ because $\mathscr{T}_2$ also contains $\mathcal{T} \in \mathscr{T}_1$, the unlabeled, single-vertex, unipolar tree.

For the inductive step, suppose that we already have $\mathscr{T}_1 \subseteq \mathscr{T}_2 \subseteq \cdots \subseteq \mathscr{T}_i$, $i \geq 2$. Then we show that $\mathscr{T}_i \subseteq \mathscr{T}_{i+1}$. Observe that the set $\mathscr{T}_{i+1}$ contains all partially labeled rooted

[16] However, it is not necessarily true that $\mathscr{H}_i^+$ contains at most one bipolar tree of each type. The Extend operation is type-preserving, but this is not true of Label: $\mathsf{Type}(\mathsf{Label}(H))$ may not equal $\mathsf{Type}(H)$, so it is possible that $\mathscr{H}_i^+$ contains two members of the same type.

trees constructed by attaching partially labeled trees from the sets $\mathscr{H}_i^+$ and $\mathscr{T}_i$ to the root vertex. We already know that $\mathscr{H}_{i-1}^+ \subseteq \mathscr{H}_i^+$, and by the inductive hypothesis we have $\mathscr{T}_{i-1} \subseteq \mathscr{T}_i$. Thus, each $\mathcal{T} \in \mathscr{T}_i$ must also appear in the set $\mathscr{T}_{i+1}$. □

If $\mathscr{T}$ and $\mathscr{H}$ are arbitrary sets of unipolar and bipolar trees, we define $\mathsf{Class}(\mathscr{T}) = \{\mathsf{Class}(\mathcal{T}) \mid \mathcal{T} \in \mathscr{T}\}$ and $\mathsf{Type}(\mathscr{H}) = \{\mathsf{Type}(\mathcal{H}) \mid \mathcal{H} \in \mathscr{H}\}$ to be the set of classes and types appearing among them.

Lemma 3.11. *Define $k^\star = |\mathscr{C}|$, where $\mathscr{C}$ is the set of all classes. Then we have $\mathsf{Class}(\mathscr{T}^\star) = \mathsf{Class}(\mathscr{T}_{k^\star})$.*

Proof. For each $i > 1$, $\mathsf{Class}(\mathscr{T}_i)$ depends only on $\mathsf{Type}(\mathscr{H}_{i-1}^+)$ and $\mathsf{Class}(\mathscr{T}_{i-1})$, due to Lemmas 3.2 and 3.3. Let i^* be the smallest index such that $\mathsf{Class}(\mathscr{T}_{i^*}) = \mathsf{Class}(\mathscr{T}_{i^*+1})$. Then we have $\mathsf{Type}(\mathscr{H}_{i^*}) = \mathsf{Type}(\mathscr{H}_{i^*+1})$ and as a consequence, $\mathscr{H}_{i^*}^+ = \mathscr{H}_{i^*+1}^+$. This implies that $\mathsf{Class}(\mathscr{T}_{i^*+1}) = \mathsf{Class}(\mathscr{T}_{i^*+2})$. By repeating the same argument, we conclude that for each $j \geq i^*$, we have $\mathsf{Class}(\mathscr{T}_j) = \mathsf{Class}(\mathscr{T}_{i^*}) = \mathsf{Class}(\mathscr{T}^\star)$. Since $\mathscr{T}_1 \subseteq \mathscr{T}_2 \subseteq \cdots$ (Lemma 3.10), we have $i^* \leq |\mathscr{C}|$. □

Lemma 3.12. *For each i, $\mathsf{Class}(\mathscr{T}_i)$ does not depend on the parameter w used in the operation Extend.*

Proof. Let $\mathcal{H} = (\mathcal{T}_1, \ldots, \mathcal{T}_x)$ be any partially labeled bipolar tree with $x \geq 2r + 2\ell_{\text{pump}}$. The type of $\mathcal{H}' = \mathsf{Extend}(\mathcal{H})$ is invariant over all choices of the parameter w. Thus, by induction, the sets $\mathsf{Class}(\mathscr{T}_i)$, $\mathsf{Type}(\mathscr{H}_i)$, and $\mathsf{Type}(\mathscr{H}_i^+)$ are also invariant over the choice of w. □

Lemma 3.13. *The maximum number of vertices of a tree in $\mathscr{T}_i$, over all choices of labeling function f, is at most λ^{i-1}, where $\lambda = 2\Delta(r + w + \ell_{\text{pump}})$.*

Proof. For any i, we write t_i (resp., h_i) to denote the maximum number of vertices of a tree in $\mathscr{T}_i$ (resp., $\mathscr{H}_i^+$). By the definition of these sets, we have the following formulas, which together imply that $t_i \leq \lambda^{i-1}$, where $\lambda = 2\Delta(r + w + \ell_{\text{pump}})$.

$$t_1 = 1. \tag{3.1}$$

$$t_i \leq \Delta \max\{t_{i-1}, h_{i-1}\}, \qquad \text{for } i > 1. \tag{3.2}$$

$$h_i \leq (2(w + \ell_{\text{pump}}) + 2r)t_i, \qquad \text{for } i \geq 1. \tag{3.3}$$

We explain the numbers in the upper bound on h_i. The operation Extend takes $\mathcal{H} = \mathcal{X} \circ \mathcal{Y} \circ \mathcal{Z}$ as an input, and returns $\mathcal{H}' = \mathsf{Pump}(\mathcal{X}, w) \circ \mathcal{Y} \circ \mathsf{Pump}(\mathcal{Z}, w)$; the length of the core path of $\mathcal{Y}$ is $2r$; the length of the core path of both $\mathsf{Pump}(\mathcal{X}, w)$ and $\mathsf{Pump}(\mathcal{Z}, w)$ is at most $w + \ell_{\text{pump}}$.

Notice that Formula 3.3 is not tight in the sense that we actually have $\mathscr{H}_{i^*}^{+} = \mathscr{H}_{i^*+1}^{+} = \cdots$, i.e., the sequence (h_i) stops growing as $i \geq i^*$. However, even for $i \geq i^*$, the sequence (t_i) still grows exponentially in view of Formula 3.2. □

Feasible Labeling Function. In view of Lemma 3.12, $\mathsf{Class}(\mathscr{T}^\star)$ depends only on the choice of the labeling function f used by Label. We call a function f *feasible* if implementing Label with f makes each tree in $\mathsf{Class}(\mathscr{T}^\star)$ good, i.e., its partial labeling can be extended to a complete and legal labeling. In Section 3.3.12 we show that given a feasible function, we can generate a DetLOCAL algorithm to solve $\mathcal{P}$ in $O(\log n)$-time. In Section 3.3.13, we show that (i) a feasible function can be derived from any $n^{o(1)}$-time RandLOCAL algorithm for $\mathcal{P}$, and (ii) the existence of a feasible function is decidable. These results together imply the $\omega(\log n)$—$n^{o(1)}$ gap. Moreover, given an LCL problem $\mathcal{P}$ on bounded degree trees, it is decidable whether the RandLOCAL complexity of $\mathcal{P}$ is $n^{\Omega(1)}$ or the DetLOCAL complexity of $\mathcal{P}$ is $O(\log n)$.

3.3.12 An $O(\log n)$-time DetLOCAL Algorithm from a Feasible Labeling Function

In this section, we show that given a feasible function f for the LCL problem $\mathcal{P}$, it is possible to design an $O(\log n)$-time DetLOCAL algorithm for $\mathcal{P}$ on bounded degree trees.

Regardless of f, the algorithm begins by computing the graph decomposition $V(G) = V_1 \cup \cdots \cup V_L$, with $L = O(\log n)$; see Section 3.3.9. We let the three infinite sequences $\{\mathscr{H}_i\}_{i \in \mathbb{Z}^+}$, $\{\mathscr{H}_i^{+}\}_{i \in \mathbb{Z}^+}$, and $\{\mathscr{T}_i\}_{i \in \mathbb{Z}^+}$ be constructed with respect to a feasible function f and a sufficiently large parameter w. We will choose w to be large enough so that a feasible function exists. Notice that the operation Extend already requires $w \geq \ell = 2(r + \ell_{\text{pump}})$.

A Sequence of Partially Labeled Graphs. We define below a sequence of partially labeled graphs $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_L$, where $\mathcal{R}_1$ is the unlabeled tree G (the underlying communications network), and $\mathcal{R}_{i+1}$ is constructed from $\mathcal{R}_i$ using the graph operations Extend, Label, and Duplicate-Cut. An alternative, and helpful way to visualize $\mathcal{R}_i$ is that it is obtained

by stripping away some vertices of G, and then grafting on some *imaginary* subtrees to its remaining vertices. Formally, the graph $\mathcal{R}_i$ is formed by taking G_i (the subforest induced by $\bigcup_{j=i}^{L} V_j$, defined in Section 3.3.9), and identifying each vertex $u \in V(G_i)$ with the root of a partially labeled imaginary tree $\mathcal{T}_{u,i} \in \mathscr{T}_i$ (defined within the proof of Lemma 3.14). Since G_L consists solely of isolated vertices, $\mathcal{R}_L$ is the disjoint union of trees drawn from $\mathscr{T}_L$.

Once each vertex $v \in V(G_i) = \bigcup_{j=i}^{L} V_j$ in the communication network G knows $\mathcal{T}_{v,i}$, we are able to simulate the imaginary graph $\mathcal{R}_i$ in the communication network G. In particular, a *legal labeling of* $\mathcal{R}_i$ is represented by storing the entire output labeling of the (imaginary) tree $\mathcal{T}_{v,i}$ at the (real) vertex $v \in V(G_i)$.

The official, inductive construction of $\mathcal{R}_i$ is described in the proof of Lemma 3.14. We remark that the "pre-commitment" of output labeling specified by the function f during the operation Label (in the construction of $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_L$) is used only in the *imaginary* trees. This does not directly lead to any *real* vertices committing to specific output labels.

Lemma 3.14. *Suppose that a feasible function f is given. The partially labeled graphs $\mathcal{R}_1, \ldots, \mathcal{R}_L$ and partially labeled trees $\{\mathcal{T}_{v,i} \mid v \in V(G_i), i \in [L]\}$ can be constructed in $O(\log n)$ time meeting the following conditions.*

1. *For each $i \in [1, L]$, each vertex $v \in V(G_i) = \bigcup_{j=i}^{L} V_j$ knows $\mathcal{T}_{v,i} \in \mathscr{T}_i$.*
2. *For each $i \in [2, L]$, given a legal labeling of $\mathcal{R}_i$, a legal labeling of $\mathcal{R}_{i-1}$ can be computed in $O(1)$ time.*

Proof. Part (1) of the lemma is proved by induction.

Base Case. Define $\mathcal{R}_1 = G$. This satisfies the lemma since $\mathcal{T}_{v,1} \in \mathscr{T}_1$ must be the unlabeled single-vertex tree, for each $v \in V(G)$.

Inductive Step. We can assume inductively that $\mathcal{R}_{i-1}$ and $\{\mathcal{T}_{v,i-1} \mid v \in V(G_{i-1})\}$ have been defined and satisfy the lemma. The set $\mathscr{P}_{i-1}$ was defined in Section 3.3.9. Each $P \in \mathscr{P}_{i-1}$ is a path such that $\deg_{G_{i-1}}(v) = 2$ for each vertex $v \in V(P)$ and $|V(P)| \in [\ell, 2\ell]$. Fix a path $P = (v_1, \ldots, v_x) \in \mathscr{P}_{i-1}$. The bipolar graphs $\mathcal{H}_P$ and $\mathcal{H}_P^+$ are defined as follows.

- Define $\mathcal{H}_P$ to be the partially labelled bipolar tree $(\mathcal{T}_{v_1,i-1}, \ldots, \mathcal{T}_{v_x,i-1})$. Notice that $\mathcal{H}_P$ is a subgraph of $\mathcal{R}_{i-1}$. Since $\mathcal{T}_{v_j,i-1} \in \mathscr{T}_{i-1}$, for each $j \in [x]$, it follows that $\mathcal{H}_P \in \mathscr{H}_{i-1}$.

- Construct $\mathcal{H}_P^+$ as follows. Select the unique member $\tilde{\mathcal{H}} \in \mathscr{H}_{i-1}$ such that (i) $\mathsf{Type}(\tilde{\mathcal{H}}) = \mathsf{Type}(\mathcal{H}_P)$ and (ii) $\mathsf{Extend}(\mathsf{Label}(\tilde{\mathcal{H}})) \in \mathscr{H}_{i-1}^+$; and then set $\mathcal{H}_P^+ = \mathsf{Extend}(\mathsf{Label}(\tilde{\mathcal{H}})) \in \mathscr{H}_{i-1}^+$. Due to the way we define $\mathscr{H}_{i-1}^+$, such a graph $\tilde{\mathcal{H}} \in \mathscr{H}_{i-1}$ must exist, as $\mathcal{H}_P \in \mathscr{H}_{i-1}$.

The partially labeled graph $\mathcal{R}_i$ is constructed from $\mathcal{R}_{i-1}$ with the following three-step procedure. See Figure 3.8 for a schematic example of how these steps work.

Step 1. Define $\mathcal{R}'_{i-1}$ as the result of applying the following operations on $\mathcal{R}_{i-1}$. For each $v \in V_{i-1}$ such that $\mathcal{T}_{v,i-1}$ is a connected component of $\mathcal{R}_{i-1}$, remove $\mathcal{T}_{v,i-1}$. Notice that a tree $\mathcal{T}_{v,i-1}$ is a connected component of $\mathcal{R}_{i-1}$ if and only if v's neighborhood in G contains only vertices at lower levels: $V_1, \ldots, V_{i-2}$.

Step 2. Define $\mathcal{R}_{i-1}^+$ by the following procedure. (i) Initialize $\tilde{\mathcal{G}} \leftarrow \mathcal{R}'_{i-1}$. (ii) For each $P \in \mathscr{P}_{i-1}$, do $\tilde{\mathcal{G}} \leftarrow \mathsf{Replace}(\tilde{\mathcal{G}}, \mathcal{H}_P, \mathcal{H}_P^+)$. (iii) Set $\mathcal{R}_{i-1}^+ \leftarrow \tilde{\mathcal{G}}$.

Step 3. Define $\mathcal{R}_i$ by the following procedure. (i) Initialize $\tilde{\mathcal{G}} \leftarrow \mathcal{R}_{i-1}^+$. (ii) For each $P \in \mathscr{P}_{i-1}$, do $\tilde{\mathcal{G}} \leftarrow \mathsf{Duplicate\text{-}Cut}(\tilde{\mathcal{G}}, \mathcal{H}_P^+)$. (iii) Set $\mathcal{R}_i \leftarrow \tilde{\mathcal{G}}$.

After Steps 1–3, for $v \in V(G_i)$, $\mathcal{T}_{v,i}$ is now defined to be the tree in $\mathcal{R}_i - (V(G_i) - \{v\})$ rooted at v. Notice that the two copies of $\mathcal{H}_P^+$ generated during Step 3(ii) become subtrees of $\mathcal{T}_{u,i}$ and $\mathcal{T}_{v,i}$, where u and v are the two vertices in $V(G_i)$ adjacent to the two endpoints of P in the graph G.

Example. See Figure 3.8 for an illustration of the procedure constructing $\mathcal{R}_i$.

Top: In this example v was a vertex in a long degree-2 path tagged $(i-1)_C$ by the decomposition procedure, and subsequently promoted to V_i. Black vertices are in V_i (or above); white vertices are in V_{i-1}; gray vertices are in V_{i-2} or below. The paths $P_0 = (s_0, \ldots, t_0)$ and $P_1 = (s_1, \ldots, t_1)$ adjacent to v have constant length, between ℓ and 2ℓ. The colored subtrees grafted onto white and gray vertices are imaginary subtrees formed in the construction of $\mathcal{R}_{i-1}$.

Middle: The graph is transformed by finding the graph $\tilde{\mathcal{H}}_b \in \mathscr{H}_{i-1}^+$, $b \in \{0,1\}$ that has the same type as $\mathcal{H}_{P_b}$, and replacing $\mathcal{H}_{P_b}$ with $\mathcal{H}_{P_b}^+ = \mathsf{Extend}(\mathsf{Label}(\tilde{\mathcal{H}}_b))$. The vertices receiving pre-committed labels are indicated in pink ($r = 1$).

Bottom: We duplicate $\mathcal{H}_{P_b}^+$, $b \in \{0,1\}$, and attach one of the copies of each duplicate to v. (The copies of $\mathcal{H}_{P_b}^+$ attached to v', v'' are not shown.) The tree $\mathcal{T}_{v,i}$ is the resulting tree rooted at v. Since each subtree of v is in $\mathscr{T}_{i-1}$ or $\mathscr{H}_{i-1}^+$, it follows that $\mathcal{T}_{v,i} \in \mathscr{T}_i$.

In this case v had no neighbors at higher levels ($i+1$ and above), so $\mathcal{T}_{v,i}$ is a connected component of $\mathcal{R}_i$. Thus, v can locally compute a legal labelling of $\mathcal{T}_{v,i}$.

We now need to verify that $\mathcal{R}_i$ satisfies all the claims of the lemma. Given the partially labeled graph $\mathcal{R}_i$, the partially labeled trees $\mathcal{T}_{v,i}$ for all $v \in V(G_i)$ are uniquely determined. According to the construction of $\mathcal{R}_i$, each connected component of $\mathcal{R}_i - V(G_i)$ must be an imaginary tree that is either (i) some $\mathcal{T}_{v,j}$, where $v \in V_j$ and $j \in \{1, \ldots, i-1\}$ or (ii) a copy of $\mathcal{H}_P^+$, where $P \in \mathscr{P}_j$ and $j \in \{1, \ldots, i-1\}$. By induction (and Lemma 3.10), for $v \in V_1 \cup \cdots \cup V_j$ and $j \in \{1, \ldots, i-1\}$, we have $\mathcal{T}_{v,j} \in \mathscr{T}_j \subseteq \mathscr{T}_{i-1}$; for each $P \in \mathscr{P}_j$ where $j \in \{1, \ldots, i-1\}$, we have $\mathcal{H}_P^+ \in \mathscr{H}_j^+ \subseteq \mathscr{H}_{i-1}^+$. According to the inductive definition of $\mathscr{T}_i$, for each $v \in V(G_i)$ we have $\mathcal{T}_{v,i} \in \mathscr{T}_i$. This concludes the induction of Part (1).

We now turn to the proof of Part (2) of the lemma. Suppose that we have a legal labeling of $\mathcal{R}_i$, where the labeling of $\mathcal{T}_{v,i}$ is stored in $v \in V(G_i)$. We show how to compute a legal labeling of $\mathcal{R}_{i-1}$ in $O(1)$ time as follows. Starting with any legal labeling $\mathcal{L}_1$ of $\mathcal{R}_i$, we compute a legal labeling $\mathcal{L}_2$ of $\mathcal{R}_{i-1}^+$, a legal labeling $\mathcal{L}_3$ of $\mathcal{R}_{i-1}'$, and finally a legal labeling $\mathcal{L}_4$ of $\mathcal{R}_{i-1}$. Throughout the process, the labels of all vertices in $\bigcup_{j=i}^{L} V_j$ are stable under $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$, and $\mathcal{L}_4$. Recall that $\mathcal{R}_i$, $\mathcal{R}_{i-1}^+$, $\mathcal{R}_{i-1}'$, and $\mathcal{R}_{i-1}$ are all *imaginary*. "Time" refers to communications rounds in the *actual* network G, not any imaginary graph.

From $\mathcal{L}_1$ to $\mathcal{L}_2$. Let s, t be the poles of $\mathcal{H}_P^+$ and u, v be the vertices outside of $\mathcal{H}_P^+$ in $\mathcal{R}_{i-1}^+$ adjacent to s, t, respectively. At this point u and v have legal labelings of $\mathcal{T}_{u,i}$ and $\mathcal{T}_{v,i}$, both trees of which contain a copy of $\mathcal{H}_P^+$. Using Lemma 3.9 we integrate the labelings of $\mathcal{T}_{u,i}$ and $\mathcal{T}_{v,i}$ to fix a single legal labeling $\mathcal{L}_2$ of $\mathcal{H}_P^+$ in $\mathcal{R}_{i-1}^+$.[17]

From $\mathcal{L}_2$ to $\mathcal{L}_3$. A legal labeling $\mathcal{L}_3$ of $\mathcal{R}_{i-1}'$ is obtained by applying Lemma 3.8. For each $P \in \mathscr{P}_{i-1}$, the labeling $\mathcal{L}_3$ on $\mathcal{H}_P$ in $\mathcal{R}_{i-1}'$ can be determined from the labeling $\mathcal{L}_2$ of $\mathcal{H}_P^+$ in $\mathcal{R}_{i-1}^+$. In greater detail, suppose s, t are the poles of $\mathcal{H}_P / \mathcal{H}_P^+$, which know $\mathcal{L}_2$ on the $(2r-1)$-neighborhood of $\{s, t\}$ in $\mathcal{H}_P^+$. By Lemma 3.8, there exists a legal labeling $\mathcal{L}_3$ on $\mathcal{H}_P$, which can be succinctly encoded by fixing $\mathcal{L}_3$ on the $(2r-1)$-neighborhoods of the *roots* of each unipolar tree on the core path $(s = v_1, \ldots, v_x = t)$ of $\mathcal{H}_P$. Thus, once s, t calculate $\mathcal{L}_3$, they can transmit the relevant information with constant-length messages to the roots $v_1, \ldots, v_x$. At this point each $v_j \in V_{i-1}$ can locally compute an extension of its labeling to all of $\mathcal{T}_{v_j, i-1}$.

[17] It is not necessary to physically store the entire $\mathcal{L}_2$ on $\mathcal{H}_P^+$. To implement the following steps, it suffices that s, t both know what $\mathcal{L}_2$ is on the subgraph induced by the $(2r-1)$-neighborhood of $\{s, t\}$ in $\mathcal{H}_P^+$.

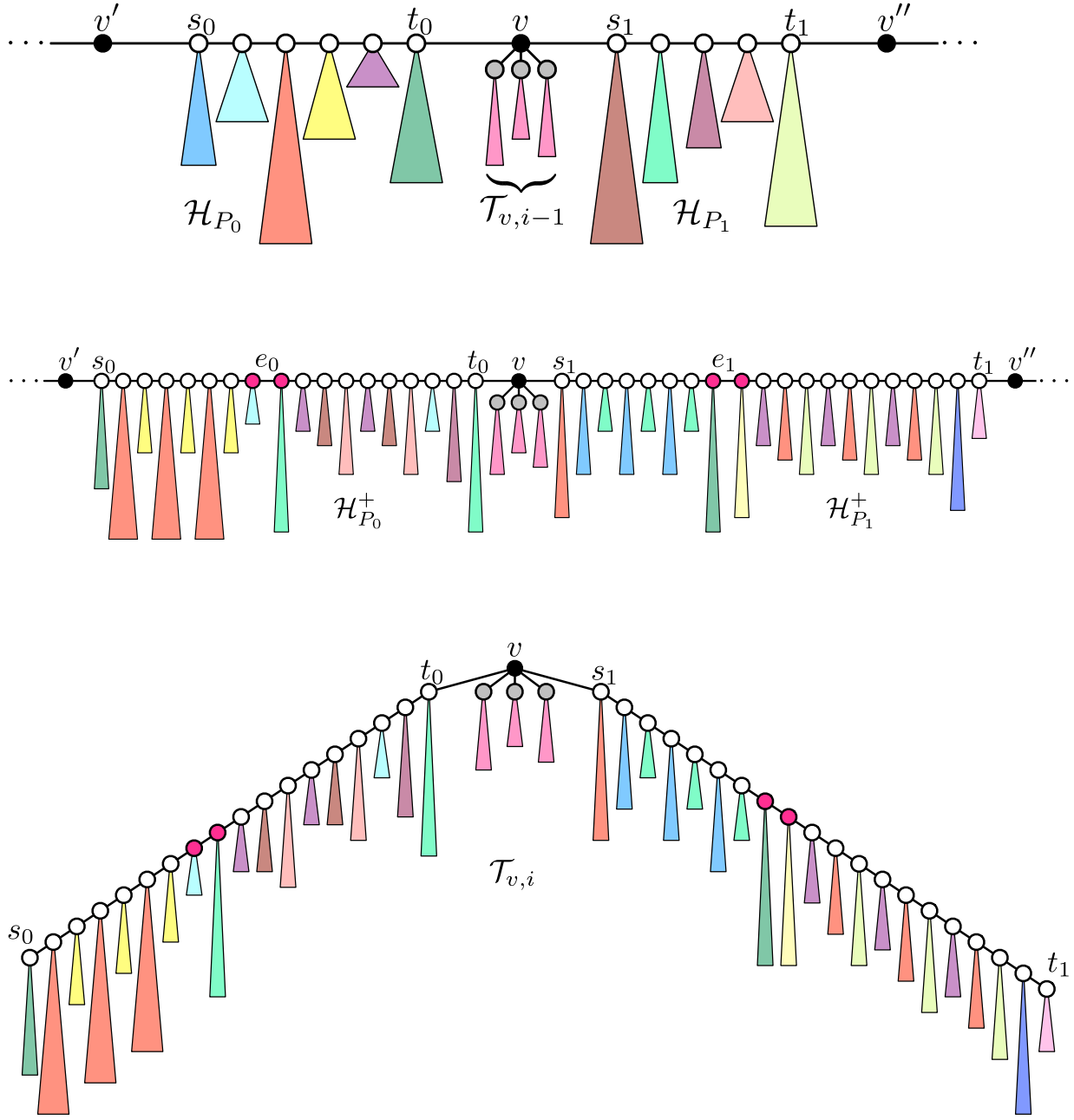

Figure 3.8: Construction of the partially labeled graph $\mathcal{R}_i$.

From $\mathcal{L}_3$ to $\mathcal{L}_4$. Notice that $\mathcal{R}_{i-1}$ is simply the disjoint union of $\mathcal{R}'_{i-1}$—for which we already have a legal labeling $\mathcal{L}_3$—and each $\mathcal{T}_{v,i-1}$ that is a connected component of $\mathcal{R}_{i-1}$. A legal labeling $\mathcal{L}_4$ of $\mathcal{T}_{v,i-1}$ is computed locally at v, which is guaranteed to exist since $\mathcal{T}_{v,i-1} \in \mathscr{T}_{i-1}$.

This concludes the proof of the lemma. □

Lemma 3.15. *Let $\mathcal{P}$ be any LCL problem on trees with $\Delta = O(1)$. Given a feasible function f, the LCL problem $\mathcal{P}$ can be solved in $O(\log n)$ time in* DetLOCAL.

Proof. First compute a graph decomposition in $O(\log n)$ time. Given the graph decomposition, for each $i \in [L]$, each vertex $v \in V_i$ computes the partially labeled rooted trees $\mathcal{T}_{v,j}$ for all $j \in [1, i]$; this can be done in $O(\log n)$ rounds. Since f is feasible, each partially labeled tree in $\mathscr{T}^\star$ admits a legal labeling. Therefore, $\mathcal{R}_L$ admits a legal labeling, and such a legal labeling can be computed without communication by the vertices in V_L. Starting with any legal labeling of $\mathcal{R}_L$, legal labelings of $\mathcal{R}_{L-1}, \ldots, \mathcal{R}_1 = G$ can be computed in $O(\log n)$ additional time, using Lemma 3.14(2). □

3.3.13 Existence of Feasible Labeling Function

In Lemmas 3.16 and 3.17 we show *two* distinct ways to arrive at a feasible labeling function. In Lemma 3.16 we assume that we are given the code of a RandLOCAL algorithm $\mathcal{A}$ that solves $\mathcal{P}$ in $n^{o(1)}$ time with at most $1/n$ probability of failure. Using $\mathcal{A}$ we can extract a feasible labeling function f.[18] Lemma 3.16 suffices to prove our $n^{o(1)} \to O(\log n)$ speedup theorem but, because it needs the code of $\mathcal{A}$, it is insufficient to answer a more basic question. Given the description of an LCL $\mathcal{P}$, is $\mathcal{P}$ solvable in $O(\log n)$ time on trees or not? Lemma 3.17 proves that this question is, in fact, decidable, which serves to highlight the delicate boundary between decidable and undecidable problems in LCL complexity [32, 116].

We briefly discuss some ideas behind the way we construct f. One natural attempt to assigning the output labels during Label is by simulating the given $n^{o(1)}$-time RandLOCAL algorithm $\mathcal{A}$. If we choose w to be sufficiently large (depending on n), then we can still force the runtime of the simulation to be less than w. This gives us a feasible function f

[18] The precise running time of $\mathcal{A}$ influences the w parameter used by Extend. For example, if $\mathcal{A}$ runs in $O(\log^2 n)$ time then w will be smaller than if $\mathcal{A}$ runs in $n^{1/\log\log\log n}$ time.

that is *randomized*, which is enough for the purpose of establishing the $\omega(\log n)$—$n^{o(1)}$ gap in RandLOCAL.

In Lemma 3.16, we derandomize the above process with a choice of w *independent* of the size of the underlying graph n, thereby establishing the $\omega(\log n)$—$n^{o(1)}$ gap in DetLOCAL. In Lemma 3.17, we show that our construction of f leads to a decidability result.

Lemma 3.16. *Suppose that there exists a* RandLOCAL *algorithm $\mathcal{A}$ that solves $\mathcal{P}$ in $n^{o(1)}$ time on n-vertex bounded degree trees, with local probability of failure at most $1/n$. Then there exists a feasible function f.*

Proof. Define $\beta = |\Sigma_{\text{out}}|^{\Delta^r}$ to be an upper bound on the number of distinct output labelings of $N^{r-1}(e)$, where e is any edge in any graph of maximum degree Δ. Define N as the maximum number of vertices of a tree in $\mathscr{T}_{k^\star}$ over *all* choices of labeling function f. As $\Delta, r, \ell_{\text{pump}}$, and $k^\star$ are all constants, we have $N = w^{O(1)}$; see Lemma 3.13. Define t to be the running time of $\mathcal{A}$ on a $(\beta N + 1)$-vertex tree. Notice that t depends on N, which depends on w.

Choices of w and f. We select w to be sufficiently large such that $w \geq 4(r + t)$. Such a w exists since $\mathcal{A}$ runs in $n^{o(1)}$ time on an n-vertex graph, and in our case n is polynomial in w. By our choice of w, the labeled parts of $\mathcal{T} = (T, \mathcal{L}) \in \mathscr{T}_{k^\star}$ are spread far apart. In particular, (i) the sets $N^{(r-1)+t}(e)$ for all designated edges e in $\mathcal{T}$ are disjoint, (ii) for each vertex $v \in V(T)$, there is at most one designated edge e such that the set $N^{r+t}(v)$ intersects $N^{r-1+t}(e)$.

Let the function f be defined as follows. Take any bipolar tree $\mathcal{H} = (H, \mathcal{L}')$ with middle edge e on its core path. The output labels of $N^{r-1}(e)$ are assigned by selecting the *most probable labeling* that occurs when running $\mathcal{A}$ on the tree $\mathcal{H}' = \mathsf{Extend}(\mathcal{H})$, while pretending that the underlying graph has $\beta N + 1$ vertices. Notice that even though $\mathcal{A}$ is a randomized algorithm, there is no randomness involved in the definition of the labeling function f; that is, given the description of $\mathcal{A}$, the function f is defined *deterministically*. In the subsequent discussion, we will use the fact that the most probable labeling occurs with probability at least $|\Sigma_{\text{out}}|^{-\Delta^r} = 1/\beta$.

Proof Idea. To show that f is good, all we need is to show that each member of $\mathscr{T}_{k^\star}$ admits a legal labeling. In what follows, consider *any* partially labeled rooted tree $\mathcal{T} =$

$(T, \mathcal{L}) \in \mathscr{T}_{k^\star}$, where the set $\mathscr{T}_{k^\star}$ is constructed with the parameter w and function f. We prove that $\mathcal{T}$ admits a legal labeling $\mathcal{L}_\diamond$.

Suppose that we execute $\mathcal{A}$ on T while pretending that the total number of vertices is $\beta N + 1$. Let v be any vertex in T. According to $\mathcal{A}$'s specs, the probability that the output labeling of $N^r(v)$ is inconsistent with $\mathcal{P}$ is at most $1/(\beta N + 1)$. However, it is not guaranteed that the output labeling resulting from $\mathcal{A}$ is also consistent with $\mathcal{T}$, since $\mathcal{T}$ is partially labeled. To handle the partial labeling of $\mathcal{T}$, our strategy is to consider a modified distribution of random bits generated by vertices in T that forces any execution of $\mathcal{A}$ to agree with $\mathcal{L}$, wherever it is defined. We will later see that with an appropriately chosen distribution of random bits, the outcome of $\mathcal{A}$ is a legal labeling of $\mathcal{T}$ with positive probability.

Modified Distribution of Random Bits. Suppose that an execution of $\mathcal{A}$ on a $(\beta N + 1)$-vertex graph needs a b-bit random string for each vertex. For each designated edge e, let U_e be the set of all assignments of b-bit strings to vertices in $N^{(r-1)+t}(e)$. Define S_e as the subset of U_e such that $\rho \in S_e$ if and only if the following is true. Suppose that the b-bit string of each $u \in N^{(r-1)+t}(e)$ is $\rho(u)$. Using the b-bit string $\rho(u)$ for each $u \in N^{(r-1)+t}(e)$, the output labeling of the vertices in $N^{r-1}(e)$ resulting from executing $\mathcal{A}$ is the same as the output labeling specified by $\mathcal{L}$. According to our choice of f, we must have $|S_e|/|U_e| \geq 1/\beta$.

Define the modified distribution $\mathcal{D}$ of b-bit random strings to the vertices in T as follows. For each designated edge e, the b-bit strings of the vertices in $N^{(r-1)+t}(e)$ are chosen uniformly at random from the set S_e. For the remaining vertices, their b-bit strings are chosen uniformly at random.

Legal Labeling $\mathcal{L}_\diamond$ Exists. Suppose that $\mathcal{A}$ is executed on T with the modified distribution of random bits $\mathcal{D}$. Then it is guaranteed that $\mathcal{A}$ outputs a complete labeling that is consistent with $\mathcal{T}$. Of course, the probability that $\mathcal{A}$ outputs an *illegal* labeling under $\mathcal{D}$ may be larger than under uniform randomness. We need to show that $\mathcal{A}$ nonetheless succeeds with non-zero probability.

Consider any vertex $v \in V(T)$. The probability that $N^r(v)$ is inconsistent with $\mathcal{P}$ is at most $\beta/(\beta N+1)$ under distribution $\mathcal{D}$, as explained below. Due to our choice of w, the set $N^{r+t}(v)$ intersects at most one set $N^{r-1+t}(e)$ where e is a designated edge. Let U_v be the set of all assignments of b-bit strings to vertices in $N^{r+t}(v)$. For each $\rho \in U_v$, the probability that ρ occurs in an execution of $\mathcal{A}$ is $1/|U_v|$ if all random bits are chosen uniformly at

random, and is at most $\beta/|U_v|$ under $\mathcal{D}$. Thus, the probability that $\mathcal{A}$ (using distribution $\mathcal{D}$) labels $N^r(v)$ incorrectly is at most $\beta/(\beta N+1)$. The total number of vertices in T is at most N. Thus, by the union bound, the probability that the output labeling of $\mathcal{A}$ (using $\mathcal{D}$) is not a legal labeling is $\beta N/(\beta N+1) < 1$. Thus, $\mathcal{T} = (T, \mathcal{L})$ admits a legal labeling $\mathcal{L}_\diamond$. □

Lemma 3.17. *Given an LCL problem $\mathcal{P}$ on bounded degree graphs, it is decidable whether there exists a feasible function f.*

Proof. Throughout the construction of the three infinite sequences $\{\mathscr{H}_i\}_{i\in\mathbb{Z}^+}$, $\{\mathscr{H}_i^+\}_{i\in\mathbb{Z}^+}$, and $\{\mathscr{T}_i\}_{i\in\mathbb{Z}^+}$, the number of distinct applications of the operation Label is constant, as $|\mathscr{H}_i^+|$ is at most the total number of types.

Therefore, the number of distinct candidate functions f that need to be examined is finite. For each candidate labeling function f (with any parameter $w \geq \ell$), in bounded amount of time we can construct the set $\mathscr{T}_{k^\star}$, as $k^\star = |\mathscr{C}|$ is a constant. By examining the classes of the partially labeled rooted trees in $\mathscr{T}_{k^\star}$ we can infer whether the function f is feasible (Lemma 3.11). Thus, deciding whether there exists a feasible function f can be done in bounded amount of time. □

Combining Lemmas 3.15, 3.16, and 3.17, we obtain the following theorem.

Theorem 3.5. *Let $\mathcal{P}$ be any LCL problem on trees with $\Delta = O(1)$. If there exists a* RandLOCAL *algorithm $\mathcal{A}$ that solves $\mathcal{P}$ in $n^{o(1)}$ rounds, then there exists a* DetLOCAL *algorithm $\mathcal{A}'$ that solves $\mathcal{P}$ in $O(\log n)$ rounds. Moreover, given a description of $\mathcal{P}$, it is decidable whether the* RandLOCAL *complexity of $\mathcal{P}$ is $n^{\Omega(1)}$ or the* DetLOCAL *complexity of $\mathcal{P}$ is $O(\log n)$.*

Discussion. To better understand Theorem 3.5, we consider some concrete examples. What would happen if we tried to apply the speedup theorem to the hierarchical $2\frac{1}{2}$-coloring $\mathcal{P}_2$ defined in Section 3.2? Since the complexity of $\mathcal{P}_2$ is $\Theta(\sqrt{n})$, there does not exist a feasible function f for $\mathcal{P}_2$. In principle, one can write a program to test whether a feasible function f exists for a given LCL; but it is not hard to see that there is no feasible function for $\mathcal{P}_2$. Recall that $\mathscr{H}_1$ is the set of all bipolar, unlabeled paths with between ℓ and 2ℓ vertices. The partial labeling in $\mathscr{H}_1^+$ must not involve **a** and **b**, since the usage of these colors will make some members in $\mathscr{T}_2$ to have no legal labeling, due to the two-coloring

rule. For example, consider a path $\mathcal{H} = \mathcal{H}_1 \circ \mathcal{H}_2 \circ \mathcal{H}_3$, where both $\mathcal{H}_1$ and $\mathcal{H}_3$ are colored by **a** and **b**, and $\mathcal{H}_2$ is unlabeled. Let $\mathcal{H}_2'$ be the path resulting from contracting one edge in $\mathcal{H}_2$, and let $\mathcal{H}' = \mathcal{H}_1 \circ \mathcal{H}_2' \circ \mathcal{H}_3$. If $\mathcal{H}$ admits a legal labeling, then $\mathcal{H}'$ must not have a legal labeling. Therefore, if there is a feasible function f for $\mathcal{P}_2$, then it must color all level 1 vertices **D**, since no V_1 vertex can be labeled **X** by the exemption rule. This coloring strategy clearly does not work (i.e., this does not give us an $O(\log n)$ time algorithm), since this requires each level 2 path (whose length can be $\Theta(n)$) to solve a 2-coloring problem.

Let us consider another problem. The problem of 3-coloring a 3-regular tree can be solved in $O(\log n)$ time, and so it admits a feasible function f. It is not hard to see that *any* function f that does a proper 3-coloring is feasible, i.e., the partial proper 3-coloring of any trees in $\mathscr{T}^\star$ can be completed to a full proper 3-coloring. For example, consider the above paths $\mathcal{H}$ and $\mathcal{H}'$, but here $\mathcal{H}_1$ and $\mathcal{H}_3$ are properly 3-colored. As long as $\mathcal{H}_2$ contains at least 2 vertices, both $\mathcal{H}$ and $\mathcal{H}'$ admit a proper 3-coloring.

Chapter 4

Decidability of LCL Complexity

4.1 Overview

Consider the following fundamental decision problem: Decide whether a given LCL problem can be solved in $O(1)$ rounds. Unfortunately, even for 2D grids this problem is undecidable [32, 116]. To our best knowledge, the problem is only known to be decidable on paths/cycles *without* input labels. In fact, we have the following characterization. An LCL problem $\mathcal{P}$ can be solved in $\Theta(1)$ rounds on cycles if and only if there exists $\alpha \in \Sigma_{\text{out}}$ such that labeling all vertices with α yields a legal labeling for all cycles of length at least $2r + 2$ [116].[19] An LCL problem $\mathcal{P}$ requires $\Theta(n)$ rounds on cycles if and only if there are infinitely many numbers n such that no legal labeling exists on an n-vertex cycle [32].

In this chapter we extend this decidability result to cycles with input labels. We show that the two gaps $\omega(1)$—$o(\log^* n)$ and $\omega(\log^* n)$—$o(n)$ for LCL problems *with input labels* on paths and cycles are decidable.[20] More specifically, given a specification of an LCL problem $\mathcal{P}$, there is a sequential algorithm that outputs a description of an asymptotically optimal deterministic LOCAL algorithm for $\mathcal{P}$, as well as its time complexity.[21]

[19] The number $2r+2$ is the minimum number which ensures that the subgraph induced by $N^r(v)$ is a path, for each v in the cycle. Some people define LCL in a slightly different manner that local consistency of a labeling on a vertex v depends on the subgraph induced by $N^r(v)$ excluding all edges $e = \{u_1, u_2\}$ such that $\text{dist}(u_1, v) = \text{dist}(u_2, v) = r$. In such a definition, the number can be reduced to $2r - 1$.

[20] We only prove the statements for the case of cycles, as the analogous results for cycles and paths follows as a simple corollary. We can use a special input label $\perp$ to effectively remove some vertices in the underlying cycle graph. Furthermore, having a promise that the input is a path does not change the time complexity of an LCL problem. By the principle of indistinguishably, if a problem can be solved by an $o(n)$-time algorithm $\mathcal{A}$ on paths, the same algorithm $\mathcal{A}$ also applies to cycles.

[21] Randomness does not affect the asymptotic time complexity for paths and cycles.

The proof of Theorem 4.1 is in Section 4.3; the proof of Theorem 4.2 is in Section 4.6.

Theorem 4.1. *For any LCL problem $\mathcal{P}$ on cycle graphs, its deterministic* LOCAL *complexity is either $\Omega(n)$ or $O(\log^* n)$. Moreover, there is an algorithm that decides whether $\mathcal{P}$ has complexity $\Omega(n)$ or $O(\log^* n)$ on cycle graphs; for the case the complexity is $O(\log^* n)$, the algorithm outputs a description of an $O(\log^* n)$-round deterministic* LOCAL *algorithm that solves $\mathcal{P}$.*

We allow the possibility for the LCL problem $\mathcal{P}$ to have no valid solution on some graphs. In this case, our algorithm for Theorem 4.1 is guaranteed to decide that the time complexity of $\mathcal{P}$ is $\Omega(n)$. We do not distinguish between (i) LCLs that are solvable in $\Theta(n)$ time on all instances and (ii) LCLs that are unsolvable on some instances.

Theorem 4.2. *For any LCL problem $\mathcal{P}$ on cycle graphs, its deterministic* LOCAL *complexity is either $\Omega(\log^* n)$ or $O(1)$. Moreover, there is an algorithm that decides whether $\mathcal{P}$ has complexity $\Omega(\log^* n)$ or $O(1)$ on cycle graphs; for the case the complexity is $O(1)$, the algorithm outputs a description of an $O(1)$-round deterministic* LOCAL *algorithm that solves $\mathcal{P}$.*

All the above gaps can be proved using the known tools in Chapter 2. For example, the (randomized and deterministic) $\omega(1)$—$o(\log^* n)$ gap can be established using Ramsey theory. However, these proofs are non-constructive by nature. The main contribution of this chapter is to re-prove these gaps using pumping lemmas in such a way similar to the proof of the $\omega(\log n)$—$n^{o(1)}$ gap on bounded degree trees in Chapter 3. The main advantage of such proofs is that they come with a decidability result.

Notation. For convenience, in this section, a directed path P with input labels is alternatively described as a string in Σ_{in}^k, where $k > 0$ is the number of vertices in P. Similarly, an output labeling $\mathcal{L}$ of P is alternatively described as a string in Σ_{out}^k. In subsequent discussion, we freely switch between the graph-theoretic notation and the string notation. Given an output labeling $\mathcal{L}$ of P, we say that $\mathcal{L}$ is *locally consistent* at v if the input and output labeling assigned to $N^r(v)$ is acceptable for v. Note that $N^r(v)$ refers to the radius-r neighborhood of v. Given two integers $a \leq b$, the notation $[a, b]$ represents the set of all integers $\{a, a+1, \ldots, b\}$. Given a string w, denote w^R as the reverse of w.

Overview. Before we proceed, we briefly discuss the high level idea of the proofs. The main tool underlying the proofs is the "pumping lemma" which we developed in Chapter 3. Intuitively, we classify the set of all input-labeled paths into a finite number of equivalence classes satisfying the following property. Let P be a subpath of G, and let P' be another path that is of the same equivalence class as P. Given a complete legal labeling of G, if we let G' be the result of replacing P with P', then it is always possible to extend this partial labeling of G' to a complete legal labeling by appropriately labeling P'. The pumping lemma guarantees that for any path P whose length is at least the pumping constant ℓ_{pump}, and for any number $x \geq \ell_{\text{pump}}$, there is another path P' of length at least x and P' is of the same equivalence class as P.

Informally, in the proof of Theorem 4.1, we show that any LCL problem $\mathcal{P}$ solvable in $o(n)$ rounds can be solved in $O(\log^* n)$ rounds in the following canonical way based on a "feasible labeling function" f. Intuitively, a labeling function f is feasible if for any given independent set I that is sufficiently well-spaced, we can apply f to assign the output labels to each $v \in I$ and its nearby neighbors locally such that this partial labeling can always be extended to a complete legal labeling. The $\omega(\log^* n)$—$o(n)$ gap and the decidability result follows from these two claims.

- If there is an $o(n)$-round algorithm $\mathcal{A}$ that solves $\mathcal{P}$, then a feasible function f exists. This is proved by first create an imaginary graph where some paths are extended using pumping lemmas, and then apply a simulation of $\mathcal{A}$ on the imaginary graph.
- Whether a feasible function exists is decidable. Intuitively, this is due to the fact that the number of equivalence classes is finite.

The proof of Theorem 4.2 is a little more complicated since the time budget is only $O(1)$, so we cannot even afford to find an MIS. To solve this issue, we decompose the cycle graph G into paths with unrepetitive patterns and paths with repetitive patterns, in $O(1)$ rounds. For paths with unrepetitive patterns, we are able to compute a sufficiently well-spaced MIS in $O(1)$ rounds by making use of the irregularity of the input patterns. Paths with repetitive patterns are similar to the paths without input labels, and we will show that we can always label them by repetitive output patterns, given that the underlying LCL problem is $o(\log^* n)$-time solvable.

Organization. Section 4.2 established some pumping lemmas for paths, which lays down the foundation for the proofs of the two main theorems. In Section 4.3, we prove Theo-

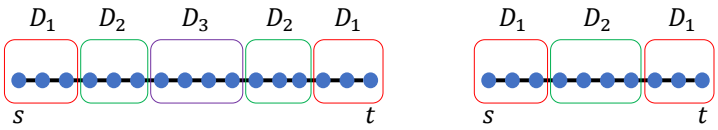

Figure 4.1: Illustration of the tripartition $\xi(P) = (D_1, D_2, D_3)$ with $r = 3$.

rem 4.1. Section 4.4 considers an $O(1)$-round algorithm that partitions a cycle into paths that are short or have a repeated input pattern. Section 4.5 defines a *feasible function* whose existence characterizes the $O(1)$-round solvable LCL problems. In Section 4.6, we prove Theorem 4.2.

4.2 Pumping Lemmas for Paths

Let $P = (s, \ldots, t)$ be a directed path, where each vertex has an input label from Σ_{in}. The *tripartition* of the vertices $\xi(P) = (D_1, D_2, D_3)$ is defined as follows:

$$\begin{aligned} D_1 &= N^{r-1}(s) \cup N^{r-1}(t), \\ D_2 &= \left(N^{2r-1}(s) \cup N^{2r-1}(t)\right) \setminus D_1, \\ D_3 &= P \setminus (D_1 \cup D_2). \end{aligned}$$

See Figure 4.1 for an illustration. More specifically, suppose $P = (u_1, \ldots, u_k)$, and let $i \in [1, k]$. Then we have:

- $u_i \in D_1$ if and only if $i \in [1, r] \cup [k - r + 1, k]$.
- $u_i \in D_2$ if and only if $i \in [r + 1, 2r] \cup [k - 2r + 1, k - r]$.
- $u_i \in D_3$ if and only if $i \notin [1, 2r] \cup [k - 2r + 1, k]$.

Let $\mathcal{L}\colon D_1 \cup D_2 \to \Sigma_{\text{out}}$ assign output labels to $D_1 \cup D_2$. We say that $\mathcal{L}$ is *extendible* w.r.t. P if there exists a complete labeling $\mathcal{L}_\diamond$ of P such that $\mathcal{L}_\diamond$ agrees with $\mathcal{L}$ on $D_1 \cup D_2$, and $\mathcal{L}_\diamond$ is locally consistent at all vertices in $D_2 \cup D_3$.

An Equivalence Class. We define an equivalence class $\overset{\star}{\sim}$ for the directed paths (i.e., the set of all non-empty strings in Σ_{in}^*), as follows.

Consider two directed paths $P = (u_1, \ldots, u_x)$ and $P' = (v_1, \ldots, v_y)$, and let $\xi(P) = (D_1, D_2, D_3)$ and $\xi(P') = (D'_1, D'_2, D'_3)$. Consider the following natural 1-to-1 correspondence $\phi\colon (D_1 \cup D_2) \to (D'_1 \cup D'_2)$ defined as $\phi(u_i) = v_i$ and $\phi(u_{x-i+1}) = v_{y-i+1}$ for each $i \in [1, 2r]$. The 1-to-1 correspondence is well-defined so long as (i) $x = y$ or (ii) $x \geq 4r$ and $y \geq 4r$. We have $P \overset{\star}{\sim} P'$ if and only if the following two statements are met:

- **Isomorphism:** The 1-to-1 correspondence is ϕ well-defined, and for each $u_i \in D_1 \cup D_2$, the input label of u_i is identical to the input label of $\phi(u_i)$.
- **Extendibility:** Let $\mathcal{L}$ be *any* assignment of output labels to vertices in $D_1 \cup D_2$, and let $\mathcal{L}'$ be the corresponding output labeling of $D'_1 \cup D'_2$ under ϕ. Then $\mathcal{L}$ is extendible w.r.t. P if and only if $\mathcal{L}'$ is extendible w.r.t. P'.

Note that for the special case of $x \leq 4r$, we have $P \overset{\star}{\sim} P'$ if and only if P is identical to P'.

Define $\mathsf{Type}(P)$ as the equivalence class of P w.r.t. $\overset{\star}{\sim}$. The following technical lemma is analogous to Lemma 3.1 in a specialized setting. We only use this lemma to prove the lemmas in Section 4.2.

Lemma 4.1. *Let G be a path graph or a cycle graph where all vertices have input labels from Σ_{in}. Let P be a directed subpath of G, and let P' be another directed path such that $\mathsf{Type}(P') = \mathsf{Type}(P)$. We write $\xi(P) = (D_1, D_2, D_3)$ and $\xi(P') = (D'_1, D'_2, D'_3)$. Let $\mathcal{L}_\diamond$ be any complete labeling of G such that $\mathcal{L}_\diamond$ is locally consistent at all vertices in $D_2 \cup D_3$. Let $G' = \mathsf{Replace}(G, P, P')$ be the graph resulting from replacing P with P' in G. Then there exists a complete labeling $\mathcal{L}'_\diamond$ of G' such that the following two conditions are met.*

1. *For each $v \in (V(G) \setminus V(P)) \cup (D_1 \cup D_2)$ and its corresponding $v' \in (V(G') \setminus V(P')) \cup (D'_1 \cup D'_2)$, we have $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(v')$. Moreover, if $v \in (V(G) \setminus V(P)) \cup D_1$ and $\mathcal{L}_\diamond$ is locally consistent at v, then $\mathcal{L}'_\diamond$ is locally consistent at v'.*
2. *$\mathcal{L}'_\diamond$ is locally consistent at all vertices in $D'_2 \cup D'_3$.*

Proof. The labeling $\mathcal{L}'_\diamond(v')$ of G' for each $v' \in (V(G') \setminus V(P')) \cup (D'_1 \cup D'_2)$ is chosen "naturally" as follows. For each $v' \in V(G') \setminus V(P')$, we set $\mathcal{L}'_\diamond(v') = \mathcal{L}_\diamond(v)$ for its corresponding vertex $v \in V(G) \setminus V(P)$. For each $v' \in D'_1 \cup D'_2$, we set $\mathcal{L}'_\diamond(v') = \mathcal{L}_\diamond(v)$ for its corresponding vertex $v \in D_1 \cup D_2$ such that $\phi(v) = v'$ in the definition of $\overset{\star}{\sim}$. At this point, it is

clear that if $v \in (V(G) \setminus V(P)) \cup D_1$ has a locally consistent labeling under $\mathcal{L}_\diamond$, then its corresponding vertex $v' \in (V(G') \setminus V(P')) \cup D'_1$ also has a locally consistent labeling under $\mathcal{L}'_\diamond$, so Condition 1 holds.

Now, the labeling $\mathcal{L}'_\diamond$ is only undefined for vertices in D'_3. We show that we can complete the labeling in such a way that is locally consistent at all vertices in $D'_2 \cup D'_3$. Denote $\mathcal{L}$ as $\mathcal{L}_\diamond$ restricted to $D_1 \cup D_2$. Since $\mathcal{L}_\diamond$ is locally consistent at all vertices in P, the labeling $\mathcal{L}$ is extendible w.r.t. P. Note that if we let $\mathcal{L}'$ be $\mathcal{L}_\diamond$ restricted to $D'_1 \cup D'_2$, then according to the way we define $\mathcal{L}'_\diamond$, the two labeling $\mathcal{L}'$ and $\mathcal{L}$ are identical under the 1-to-1 correspondence ϕ specified in the definition of $\overset{\star}{\sim}$. That is, for each $v' \in D'_1 \cup D'_2$, we have $\mathcal{L}'(v') = \mathcal{L}(v)$ for its corresponding vertex $v \in D_1 \cup D_2$ such that $\phi(v) = v'$. Since $P \overset{\star}{\sim} P'$, the labeling $\mathcal{L}'$ must be extendible w.r.t. P'. That is, there is a way to assign $\mathcal{L}'_\diamond(v')$ for each $v' \in D'_3$ such that all vertices in $D'_2 \cup D'_3$ have locally consistent labelings under $\mathcal{L}'_\diamond$, so Condition 2 holds. □

One useful consequence of this lemma is that if we start with a path or a cycle G with a legal labeling, after replacing its subpath P with another one P' having the same type as P, then it is always possible to assign output labeling to P' to get a legal labeling without changing the already-assigned output labels of vertices outside of P'.

Lemma 4.2. *Let G be a path graph or a cycle graph where all vertices have input labels from Σ_{in}. Let P be a directed subpath of G, and let P' be another directed path such that $\mathsf{Type}(P') = \mathsf{Type}(P)$. Let $\mathcal{L}_\diamond$ be complete labeling of G that is locally consistent at all vertices in P. Let $G' = \mathsf{Replace}(G, P, P')$ be the graph resulting from replacing P with P' in G. Then there exists a legal labeling $\mathcal{L}'_\diamond$ of G' such that the following two conditions are met.*

1. *For each $v \in V(G) \setminus V(P)$ and its corresponding $v' \in V(G') \setminus V(P')$, we have $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(v')$. Moreover, if $\mathcal{L}_\diamond$ is locally consistent at $v \in V(G) \setminus V(P)$, then $\mathcal{L}'_\diamond$ is locally consistent at v'.*
2. *$\mathcal{L}'_\diamond$ is locally consistent at all vertices in P'.*

Proof. We write $\xi(P') = (D'_1, D'_2, D'_3)$. Condition 1 in this lemma is implied by Condition 1 in Lemma 4.1. To see that Condition 2 in this lemma holds, notice that in this lemma we additionally require that $\mathcal{L}_\diamond$ is locally consistent at all vertices in P. Therefore, Condition 1 of Lemma 4.1 implies that $\mathcal{L}'_\diamond$ is locally consistent at all vertices in D'_1. This observation,

together with Condition 2 of Lemma 4.1, implies that $\mathcal{L}'_\diamond$ is locally consistent at all vertices in P'. □

The following lemma is analogous to Theorem 3.4 in a specialized setting. We only use this lemma in Section 4.2.

Lemma 4.3. *Let $P = (v_1, \ldots, v_k)$, and let $P' = (v_1, \ldots, v_{k-1})$. Let the input label of v_k be α. Then $\mathsf{Type}(P)$ is a function of α and $\mathsf{Type}(P')$.*

Proof. We prove the following stronger statement. Let G be a directed path, and let H be a directed subpath of G. Suppose H' is another directed path satisfying $\mathsf{Type}(H) = \mathsf{Type}(H')$. Let $G' = \mathsf{Replace}(G, H, H')$ be the result of replacing H with H' in G. Then we claim that $\mathsf{Type}(G) = \mathsf{Type}(G')$. The lemma is a corollary of this claim.

Consider the tripartitions $\xi(H) = (B_1, B_2, B_3)$, $\xi(H') = (B'_1, B'_2, B'_3)$, $\xi(G) = (D_1, D_2, D_3)$, and $\xi(G') = (D'_1, D'_2, D'_3)$. We write $B_0 = V(G) \setminus V(H)$ and $B'_0 = V(G') \setminus V(H')$.

Let $\phi^\star$ be the natural 1-to-1 correspondence from $B_0 \cup B_1 \cup B_2$ to $B'_0 \cup B'_1 \cup B'_2$. Note that $D_1 \cup D_2 \subseteq B_0 \cup B_1 \cup B_2$ and $D'_1 \cup D'_2 \subseteq B'_0 \cup B'_1 \cup B'_2$. Also, the 1-to-1 correspondence between $D_1 \cup D_2$ and $D'_1 \cup D'_2$ given by $\phi^\star$ is exactly the 1-to-1 correspondence ϕ specified in the requirement of $G \overset{\star}{\sim} G'$.

Let $\mathcal{L}\colon (D_1 \cup D_2) \to \Sigma_{\text{out}}$ and let $\mathcal{L}'$ be the corresponding output labeling of $D'_1 \cup D'_2$, under the 1-to-1 correspondence ϕ. To show that $G \overset{\star}{\sim} G'$, all we need to do is show that $\mathcal{L}$ is extendible w.r.t. G if and only if $\mathcal{L}'$ is extendible w.r.t. G'. Since we can also write $G = \mathsf{Replace}(G', H', H)$, it suffices to show just one direction, i.e., if $\mathcal{L}$ is extendible then $\mathcal{L}'$ is extendible.

Suppose $\mathcal{L}$ is extendible. Then there exists an output labeling $\mathcal{L}_\diamond$ of G such that (i) for each $v \in D_1 \cup D_2$, we have $\mathcal{L}_\diamond(v) = \mathcal{L}(v)$, and (ii) $\mathcal{L}_\diamond$ is locally consistent at all vertices in $D_2 \cup D_3$. Since $D_2 \cup D_3 \supseteq B_2 \cup B_3$, we can apply Lemma 4.1, which shows that there exists a complete labeling $\mathcal{L}'_\diamond$ of G' such that the two conditions in Lemma 4.1 are met. We argue that this implies that $\mathcal{L}'$ is extendible. We verify that (i) $\mathcal{L}'(v') = \mathcal{L}'_\diamond(v')$ for each $v' \in D'_1 \cup D'_2$, and (ii) $\mathcal{L}'_\diamond$ is locally consistent at all vertices in $D'_2 \cup D'_3$.

- Condition 1 of Lemma 4.1 guarantees that $\mathcal{L}_\diamond(v) = \mathcal{L}'_\diamond(\phi^\star(v))$ for each $v \in (V(G) \setminus V(H)) \cup (B_1 \cup B_2) = B_0 \cup B_1 \cup B_2$ and its corresponding vertex $\phi^\star(v) \in B'_0 \cup B'_1 \cup B'_2$. Since $D'_1 \cup D'_2 \subseteq B'_0 \cup B'_1 \cup B'_2$, we have $\mathcal{L}'(v') = \mathcal{L}'_\diamond(v')$ for each $v' \in D'_1 \cup D'_2$.

- The fact that $\mathcal{L}_\diamond$ is locally consistent at all vertices in $D_2 \cup D_3$, together with Condition 1 in Lemma 4.1, guarantees that $\mathcal{L}'_\diamond$ is locally consistent at all vertices in $(D'_2 \cup D'_3) \setminus B'_3$. Condition 2 in Lemma 4.1 guarantees that $\mathcal{L}'_\diamond$ is locally consistent at all vertices in $B'_2 \cup B'_3$. Therefore, $\mathcal{L}'_\diamond$ is locally consistent at all vertices in $D'_2 \cup D'_3$, as required. □

The number of types can be upper bounded as follows.

Lemma 4.4. *The number of equivalence classes of $\overset{\star}{\sim}$ (i.e., types) is at most $|\Sigma_{\text{in}}|^{4r} 2^{|\Sigma_{\text{out}}|^{4r}}$.*

Proof. Let P be a directed path, and let $\xi(P) = (D_1, D_2, D_3)$. Then $\mathsf{Type}(P)$ is determined by the following information.

- The input labels in $D_1 \cup D_2$. Note that there are at most $|\Sigma_{\text{in}}|^{4r}$ possible input labeling of $D_1 \cup D_2$.
- A length-x binary string indicating the extendibility of each possible output labeling of $D_1 \cup D_2$, where $x = |\Sigma_{\text{out}}|^{4r}$.

Therefore, the number of equivalence classes of $\overset{\star}{\sim}$ is at most $|\Sigma_{\text{in}}|^{4r} 2^{|\Sigma_{\text{out}}|^{4r}}$. □

Define ℓ_{pump} as the total number of types. Observe that Lemma 4.3 implies that $\mathsf{Type}(P)$ can be computed by a finite automaton whose number of states is the total number of types, which is a constant independent of P. Thus, we have the following two *pumping lemmas* which allow us to extend the length of a given directed path P while preserving the type of P. The following two lemmas follow from the standard pumping lemma for regular language.

Lemma 4.5. *Let $P \in \Sigma_{\text{in}}^k$ with $k \geq \ell_{\text{pump}}$. Then P can be decomposed into three substrings $P = x \circ y \circ z$ such that (i) $|xy| \leq \ell_{\text{pump}}$, (ii) $|y| \geq 1$, and (iii) for each non-negative integer i, $\mathsf{Type}(x \circ y^i \circ z) = \mathsf{Type}(P)$.*

Lemma 4.6. *For each $w \in \Sigma_{\text{in}}^{>0}$, there exist two positive integers a and b such that $a + b \leq \ell_{\text{pump}}$, and $\mathsf{Type}(w^{ai+b})$ is invariant for each non-negative integer i.*

4.3 The $\omega(\log^* n)$—$o(n)$ Gap

In this section we show that the $\omega(\log^* n)$—$o(n)$ gap is decidable. More specifically, we show that an LCL problem $\mathcal{P}$ can be solved in $O(\log^* n)$ rounds if and only if there exists a *feasible function*, which is defined as follows.

Input: A directed path $P = w_1 \circ S \circ w_2$, where $|w_1| \in [\ell_{\text{pump}}, 2\ell_{\text{pump}}]$, $|w_2| \in [\ell_{\text{pump}}, 2\ell_{\text{pump}}]$, and $|S| = 2r$. The decomposition $P = w_1 \circ S \circ w_2$ is considered part of the input.

Output: A string $\mathcal{L} \in \Sigma_{\text{out}}^{2r}$ that represents the output labeling of S.

Requirement: Any such function f is said to be *feasible* if the following requirement is met for any paths S_1, S_2 and w_a, w_b, w_c, w_d such that $|S_1| = |S_2| = 2r$ and $\{|w_a|, |w_b|, |w_c|, |w_d|\} \subseteq [\ell_{\text{pump}}, 2\ell_{\text{pump}}]$. Let $P = w_a \circ S_1 \circ w_b \circ w_c \circ S_2 \circ w_d$, and consider the following assignment of output labels to $S_1 \cup S_2$.

- Either label S_1 by $f(w_a \circ S_1 \circ w_b)$ or label S_1^R by $f(w_b^R \circ S_1^R \circ w_a^R)$.
- Either label S_2 by $f(w_c \circ S_2 \circ w_d)$ or label S_2^R by $f(w_d^R \circ S_2^R \circ w_c^R)$.

It is required that given this partial labeling of P, the middle part $w_b \circ w_c$ can be assigned output labels in such a way that the labeling of (i) the last r vertices of S_1, (ii) all vertices in $w_b \circ w_c$, and (iii) the first r vertices of S_2 are locally consistent.

Lemma 4.7. *If a feasible function f exists, then there is an $O(\log^* n)$-round deterministic* LOCAL *algorithm for $\mathcal{P}$ on cycles.*

Proof. Given that the number of vertices n is at least some large enough constant, in $O(\log^* n)$ rounds we can compute the decomposition $V = A \cup B$ such that each connected component of A has size $2r$, and each connected component of B has size within $[2\ell_{\text{pump}}, 4\ell_{\text{pump}}]$. This can be done using a ruling set computation. We further decompose each connected component P of B into two paths $P = P_1 \circ P_2$ in such a way that the size of both P_1 and P_2 are within the range $[\ell_{\text{pump}}, 2\ell_{\text{pump}}]$. We write $\mathscr{P}$ to denote the set of all these paths.

Let S be a connected component of A, and let w_1 and w_2 be its two neighboring paths in $\mathscr{P}$ so that $(w_1 \circ S \circ w_2)$ is a subpath of the underlying graph G. The output labels of S is assigned either by labeling S with $f(w_1 \circ S \circ w_2)$ or by labeling S^R with $f(w_2^R \circ S^R \circ w_1^R)$. At this moment, all components of A have been assigned output labels using f. By the feasibility of f, each connected component of B is able to label itself output labels in such a way that the labeling of all vertices are locally consistent. □

Lemma 4.8. *If there is an $o(n)$-round deterministic* LOCAL *algorithm $\mathcal{A}$ for $\mathcal{P}$ on cycles, then a feasible function f exists.*

Proof. Fix s to be some sufficiently large number, and fix $n = 8(s + \ell_{\text{pump}}) + 2(2r)$. We select s to be large enough so that the runtime of $\mathcal{A}$ is smaller than $0.1s$.

For any given directed path w with $|w| \in [\ell_{\text{pump}}, 2\ell_{\text{pump}}]$, we fix w^+ as the result of applying the pumping lemma (Lemma 4.5) on w so that the following two conditions are met: (i) $|w^+| \in [s, s + \ell_{\text{pump}}]$ and (ii) $\mathsf{Type}(w) = \mathsf{Type}(w^+)$.

Constructing a Feasible Function f by Simulating $\mathcal{A}$. The function $f(w_1 \circ S \circ w_2)$ is constructed by simulating a given $o(n)$-round deterministic LOCAL algorithm for $\mathcal{P}$. The output labeling given by $f(w_1 \circ S \circ w_2)$ is exactly the result of simulating $\mathcal{A}$ on the path $P = w_1^+ \circ S \circ w_2^+$ while assuming the number of vertices of the underlying graph is n. Remember that the round complexity of $\mathcal{A}$ is $o(n)$ on n-vertex graphs. By setting s to be large enough, the runtime of $\mathcal{A}$ can be made smaller than $0.1s$. Thus, the calculation of $f(w_1 \circ S \circ w_2)$ only depends on the IDs and the input labels of (i) the last $0.1s$ vertices in w_1^+, (ii) all vertices in S, and (iii) the first $0.1s$ vertices in w_2^+. In the calculation of $f(w_1 \circ S \circ w_2)$, the IDs of the vertices that participate in the simulation of $\mathcal{A}$ are chosen arbitrarily so long as they are distinct.

Feasibility of f. Now we verify that the function f constructed above is feasible. Consider any choices of paths S_1, S_2 and w_a, w_b, w_c, w_d such that $|S_1| = |S_2| = 2r$ and $\{|w_a|, |w_b|, |w_c|, |w_d|\} \subseteq [\ell_{\text{pump}}, 2\ell_{\text{pump}}]$. Define $P = w_a \circ S_1 \circ w_b \circ w_c \circ S_2 \circ w_d$, and let G be the cycle graph formed by connecting the two ends of the path P. To show that f is feasible, we need to consider the following four ways of assigning output labels to $S_1 \cup S_2$.

1. Label S_1 by $f(w_a \circ S_1 \circ w_b)$; label S_2 by $f(w_c \circ S_2 \circ w_d)$.
2. Label S_1 by $f(w_a \circ S_1 \circ w_b)$; label S_2^R by $f(w_d^R \circ S_2^R \circ w_c^R)$.
3. Label S_1^R by $f(w_b^R \circ S_1^R \circ w_a^R)$; label S_2 by $f(w_c \circ S_2 \circ w_d)$.
4. Label S_1^R by $f(w_b^R \circ S_1^R \circ w_a^R)$; label S_2^R by $f(w_d^R \circ S_2^R \circ w_c^R)$.

For each of the above four partial labelings of P, we need to show that the middle part $w_b \circ w_c$ can still be assigned output labels in such a way that the labeling of (i) the last r vertices of S_1, (ii) all vertices in $w_b \circ w_c$, and (iii) the first r vertices of S_2 are locally consistent.

Proof of the First Case. In what follows, we focus on the first case, i.e., the partial labeling is given by labeling S_1 by $f(w_a \circ S_1 \circ w_b)$ and labeling S_2 by $f(w_c \circ S_2 \circ w_d)$; the proof for the other three cases are analogous. In this case, we define $P' = w_a^+ \circ S_1 \circ w_b^+ \circ w_c^+ \circ S_2 \circ w_d^+$, and let G' be the cycle graph formed by connecting the two ends of P'. Note that the

number of vertices in G' is at most $8(s + \ell_{\text{pump}}) + 2(2r) = n$. All we need to do is to find an output labeling $\mathcal{L}$ of G such that the following conditions are satisfied.

(a) The output labels of S_1 is given by $f(w_a \circ S_1 \circ w_b)$.

(b) The output labels of S_2 is given by $f(w_c \circ S_2 \circ w_d)$.

(c) The labeling of (i) the last r vertices of S_1, (ii) all vertices in $w_b \circ w_c$, and (iii) the first r vertices of S_2 are locally consistent.

We first generate an output labeling $\mathcal{L}'$ of G' by executing $\mathcal{A}$ on G' under the following ID assignment. The IDs of (i) the last $0.1s$ vertices in w_a^+, (ii) all vertices in S_1, and (iii) the first $0.1s$ vertices in w_b^+ are chosen as the ones used in the definition of $f(w_a \circ S_1 \circ w_b)$. Similarly, the IDs of (i) the last $0.1s$ vertices in w_c^+, (ii) all vertices in S_2, and (iii) the first $0.1s$ vertices in w_d^+ are chosen as the ones used in the definition of $f(w_c \circ S_2 \circ w_d)$. The IDs of the rest of the vertices are chosen arbitrarily so long as when we run $\mathcal{A}$ on G', no vertex sees two vertices with the same ID. Due to the way we define f, the output labeling $\mathcal{L}'$ of the subpath S_1 is exactly given by $f(w_a \circ S_1 \circ w_b)$, and the output labeling $\mathcal{L}'$ of S_2 is exactly $f(w_c \circ S_2 \circ w_d)$. Due to the correctness of $\mathcal{A}$, $\mathcal{L}'$ is a legal labeling.

We transform the output labeling $\mathcal{L}'$ of G' to a desired output labeling $\mathcal{L}$ of G. Remember that G is the result of replacing the for subpaths w^+ of G' by w, and we have $\mathsf{Type}(w^+) = \mathsf{Type}(w)$. In view of Lemma 4.2, there is a legal labeling $\mathcal{L}$ of G such that all vertices in S_1 and S_2 are labeled the same as in G'. Therefore, the labeling $\mathcal{L}$ satisfies the above three conditions (a), (b), and (c).

The Other Cases. We briefly discuss how we modify the proof to deal with the other three cases. For example, consider the second case, where the partial labeling is given by labeling S_1 by $f(w_a \circ S_1 \circ w_b)$ and labeling S_2^R by $f(w_d^R \circ S_2^R \circ w_c^R)$. In this case, the path P' is defined as

$$P' = w_a^+ \circ S_1 \circ w_b^+ \circ \left((w_c^R)^+\right)^R \circ S_2^R \circ \left((w_d^R)^+\right)^R.$$

During the ID assignment of G', the IDs of (i) the last $0.1s$ vertices in w_c^+, (ii) all vertices in S_2, and (iii) the first $0.1s$ vertices in w_d^+ are now chosen as the ones used in the definition of $f(w_d^R \circ S_2^R \circ w_c^R)$. Using such an ID assignment, the output labeling $\mathcal{L}'$ of S_2^R as the result of executing $\mathcal{A}$ on G' will be exactly the same as the output labeling given by $f(w_d^R \circ S_2^R \circ w_c^R)$. The rest of the proof is the same. □

Theorem 4.1 follows from the above two lemmas. The decidability result is due to the

simple observation that whether a feasible function exists is decidable.

4.4 Partitioning a Cycle

In the following sections, we prove the decidability result associated with the $\omega(1)$—$o(\log^* n)$ gap. In this proof, we also define a feasible function, prove its decidability, and show the existence given an $o(\log^* n)$-time algorithm. The main challenge here is that a ruling set cannot be computed in $O(1)$ time. To solve this issue, we decompose a cycle into paths with unrepetitive patterns and paths with repetitive patterns. For paths with unrepetitive patterns, we are able to compute a ruling set in $O(1)$ time by making use of the irregularity of the input patterns. Paths with repetitive patterns are similar to the paths without input labels, and we show that we can always label them by repetitive output patterns, given that the problem is $o(\log^* n)$-time solvable.

Section 4.4 considers an $O(1)$-round algorithm that partitions a cycle into long paths of repeated input pattern or short paths. Section 4.5 defines a *feasible function* whose existence characterizes the $O(1)$-round solvable LCL problems. In Section 4.6, we prove Theorem 4.2.

Partitioning an Undirected Cycle into Directed Paths. Let G be a cycle graph. An orientation of the vertices in G is called *ℓ-orientation* if the following condition is met. If $|V(G)| \leq \ell$, then all vertices in G are oriented[22] to the same direction. If $|V(G)| > \ell$, then each vertex $v \in V(G)$ belongs to a path P such that (i) all vertices in P are oriented to the same direction, and (ii) the number of vertices in P is at least ℓ. In $O(1)$ rounds we can compute an ℓ-orientation of G for any constant ℓ. The following lemma is a restatement of Theorem 2.11.

Lemma 4.9. *Let G be a cycle graph. Let ℓ be a constant. There is a deterministic* LOCAL *algorithm that computes an ℓ-orientation of G in $O(1)$ rounds.*

In this section, we will use a generalization of an ℓ-orientation that satisfies an additional requirement that the input labels of each directed path P in the decomposition with $|V(P)| > 2\ell_{\text{width}}$ (where $2\ell_{\text{width}}$ is a threshold) must form a periodic string (whose period length is at most ℓ_{pattern}).

[22] An orientation of a vertex v is an assignment to one of its neighbor. This can be specified using port-numbering. See Section 2.6.3.

A string $w \in \Sigma_{\text{in}}^*$ is called *primitive* if w cannot be written as x^i for some $x \in \Sigma_{\text{in}}^*$ and $i \geq 2$. Let G be a cycle graph or a path graph where each vertex $v \in V(G)$ has an input label from Σ_{in}. We define an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-*partition* as a partition of G into a set of connected subgraphs $\mathscr{P}$ meeting the following criteria. We assume $|V(G)| > 2\ell_{\text{width}}$ and $\ell_{\text{pattern}} \geq \ell_{\text{width}}$.

Direction and Minimum Length: For each $P \in \mathscr{P}$, the vertices in P are oriented to the same direction, and $|V(P)| \geq \ell_{\text{width}}$.

Short Paths: Define $\mathscr{P}_{\text{short}}$ as the subset of $\mathscr{P}$ that contains paths having at most $2\ell_{\text{width}}$ vertices. For each directed path $P = (v_1, \ldots, v_k) \in \mathscr{P}_{\text{short}}$, each vertex v_i in P knows its rank i.

Long Paths: Define $\mathscr{P}_{\text{long}} = \mathscr{P} \setminus \mathscr{P}_{\text{short}}$. Then the input labeling of the vertices in P is of the form w^k for some primitive string $w \in \Sigma_{\text{in}}^*$ such that $|w| \leq \ell_{\text{pattern}}$ and $k \geq \ell_{\text{count}}$. Moreover, each vertex v in P knows the string w.

Note that $\mathscr{P}$ may contain a cycle. This is possible only when G is a cycle where the input labeling is a repetition (at least ℓ_{count} times) of a primitive string $w \in \Sigma_{\text{in}}^*$ of length at most ℓ_{pattern}. In this case, we must have $\mathscr{P} = \mathscr{P}_{\text{long}} = \{G\}$. Otherwise, $\mathscr{P}$ contains only paths.

The goal of this section is to show that an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition can be found in $O(1)$ rounds. First of all, in Lemma 4.10 we demonstrate how we can break symmetry in $O(1)$ rounds given that the underlying graph is directed and the input labels does not form long periodic strings. Let G be a path or a cycle. A set $I \subseteq V(G)$ is called an (α, β)-*independent set* if the following conditions are met: (i) I is an independent set, and I does not contain either endpoint of G (if G is a path), and (ii) each connected component induced by $V \setminus I$ has at least α vertices and at most β vertices, unless $|V| \leq \alpha$, in which case we allow $I = \emptyset$. Note that finding an (α, β)-independent set takes $O(\log^* n)$ rounds in general, but in Lemma 4.10 we show that by leveraging the "irregularity" of input labels, we can do this in $O(1)$ rounds on directed paths or cycles without periodic patterns.

Lemma 4.10. *Let γ and ℓ be any two constants with $\ell \geq \gamma$. Let G be a directed cycle or a directed path that does not contain any subpath of the form w^x, with $|w| \leq \gamma$ and $|w^x| \geq \ell$. There is a deterministic* LOCAL *algorithm that computes an $(\gamma, 2\gamma)$-independent set I of G in $O(1)$ rounds.*

Proof. For the case G is a directed path $P = (s, \ldots, t)$, define V' as the set of vertices in G whose distance to t is at least $\ell - 1$. For the case G is a directed cycle, define $V' = V(G)$. In what follows, we focus on finding an $(\gamma, 2\gamma)$-independent set I' of the vertices in V'. Extending the set I' to produce the desired independent set I can be done with extra $O(1)$ rounds.

Recall that G is directed. Define the color of a vertex $v \in V'$ by the sequence of the ℓ input labels of v and the $\ell - 1$ vertices following v in G. For each vertex $v \in V'$, there is no other vertex within distance γ to v having the same color as v, since otherwise we can find a subpath whose input labels form a string w^x, with $|w| \leq \gamma$ and $|w^x| \geq \ell$. By applying the standard procedure that computes an MIS from a coloring, within $O(1)$ rounds a $(\gamma, 2\gamma)$-independent set I' can be obtained. □

Using Lemma 4.10, we first show that an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition can be found in $O(1)$ rounds for the case G is *directed*. That is, all vertices in G are initially oriented to the same direction, and we are allowed to re-orient the vertices.

Lemma 4.11. *Let G be a directed cycle or a directed path where each vertex $v \in V(G)$ has an input label from Σ_{in}, and $|V(G)| > 2\ell_{\text{width}}$. Let $\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}}$ be three constants such that $\ell_{\text{pattern}} \geq \ell_{\text{width}}$. There is a deterministic* LOCAL *algorithm that computes an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition in $O(1)$ rounds*

Proof. Let $(w_1, w_2, \ldots, w_k)$ be *any* ordering of the primitive strings in Σ_{in}^* of length at most ℓ_{pattern}. First, construct a set of subgraphs $\mathscr{P}_{\text{long}}$ as follows. Initialize $U = V(G)$ and $\mathscr{P}_{\text{long}} = \emptyset$. For $i = 1$ to k, execute the following procedure. Let S_i be the set of maximal-size connected subgraphs formed by vertices in U such that the input labels form the string w_i^x with $x \geq \ell_{\text{count}} + 2\ell_{\text{width}}$. Each vertex $v \in U$ in $O(1)$ rounds checks if v belongs to a subgraph in S_i; if so, remove v from U. For each $P \in S_i$, define P' as follows. If P is a cycle, then $P' = P$. If P is a path, then P' is the result of removing all vertices that are within distance $\ell_{\text{width}}|w_i| - 1$ to an endpoint in P. Note that each vertex v in P knows whether v belongs to P'. Define $S_i' = \{P' | P \in S_i\}$, and then update $\mathscr{P}_{\text{long}} \leftarrow \mathscr{P}_{\text{long}} \cup S_i'$.

It is straightforward to verify that each path or cycle $P \in \mathscr{P}_{\text{long}}$ satisfies the requirement in the definition of $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition. Define the set of subgraphs $\mathscr{P}_{\text{irreg}}$ as the connected components of the vertices not in any subgraph in $\mathscr{P}_{\text{long}}$. Define $\ell = (\ell_{\text{pattern}} + 2\ell_{\text{width}}) \cdot \ell_{\text{count}}$. By our construction, the input labeling in each subgraph $P \in \mathscr{P}_{\text{irreg}}$ does not contain any substring w^x, with $1 \leq |w| \leq \ell_{\text{pattern}}$ and $|w^x| \geq \ell$. An

$(\ell_{\text{pattern}}, 2\ell_{\text{pattern}})$-independent set of each $P \in \mathscr{P}_{\text{irreg}}$ can be computed using Lemma 4.10 in $O(1)$ rounds. Observe that each subgraph $P \in \mathscr{P}_{\text{irreg}}$ has at least ℓ_{width} vertices. Given an $(\ell_{\text{pattern}}, 2\ell_{\text{pattern}})$-independent set of a subgraph $P \in \mathscr{P}_{\text{irreg}}$, in $O(1)$ rounds P can be partitioned into subpaths, each of which contains at least ℓ_{pattern} vertices and at most $2\ell_{\text{pattern}}$ vertices. This finishes the construction of an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition. □

Combining Lemma 4.11 and Lemma 4.9, we are able to construct an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition in $O(1)$ rounds for undirected graphs.

Lemma 4.12. *Let G be a cycle or a path where each vertex $v \in V(G)$ has an input label from Σ_{in}, and $|V(G)| > 2\ell_{\text{width}}$. Let $\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}}$ be three constants such that $\ell_{\text{pattern}} \geq \ell_{\text{width}}$. There is a deterministic* LOCAL *algorithm that computes an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition in $O(1)$ rounds*

Proof. The algorithm is as follows. Compute an ℓ-orientation of G by Lemma 4.9 in $O(1)$ rounds with $\ell = 2\ell_{\text{width}} + 1$. For each maximal-length connected subgraph P where each constituent vertex is oriented to the same direction, find an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition of P in $O(1)$ rounds by Lemma 4.11. □

4.5 Feasible Function

The goal of this section is to define a *feasible function* whose existence characterizes the $O(1)$-round solvable LCL problems. With respect to an LCL problem $\mathcal{P}$ and a function f which takes a string $w \in \Sigma_{\text{in}}^k$ with $1 \leq k \leq \ell_{\text{pump}}$ as input, and returns a string $f(w) \in \Sigma_{\text{out}}^k$, we define some partially or completely labeled path graphs which are used in the definition of a feasible function.

Completely Labeled Graph $\mathcal{G}_{w,z}$: Let $w \in \Sigma_{\text{in}}^*$ be any string of length at least 1 and at most ℓ_{pump}. Let z be any non-negative integer. Define $\mathcal{G}_{w,z} = (G_{w,z}, \mathcal{L})$ as follows. The graph $G_{w,z}$ is a path of the form $w^r \circ w^z \circ w^r$. The labeling $\mathcal{L}$ is a complete labeling of the form $f(w)^{z+2r}$. Define $\mathsf{Mid}(G_{w,z})$ as the middle subpath w^z of $G_{w,z}$.

Partially Labeled Graph $\mathcal{G}_{w_1,w_2,S}$: Let $w_1, w_2 \in \Sigma_{\text{in}}^*$ be any two strings of length at least 1 and at most ℓ_{pump}. Let $S \in \Sigma_{\text{in}}^*$ be any string (can be empty). Define $\mathcal{G}_{w_1,w_2,S} = (G_{w_1,w_2,S}, \mathcal{L})$ as follows. The graph $G_{w_1,w_2,S}$ is the path of the form $w_1^{\ell_{\text{pump}}+2r} \circ S \circ w_2^{\ell_{\text{pump}}+2r}$. The labeling $\mathcal{L}$ is a partial labeling of $G_{w_1,w_2,S}$ which fixes the output labels

of the first $2r|w_1|$ and the last $2r|w_2|$ vertices by $f(w_1)^{2r}$ and $f(w_2)^{2r}$, respectively. Define $\mathsf{Mid}(G_{w_1,w_2,S})$ as the middle subpath $w_1^{\ell_{\text{pump}}+r} \circ S \circ w_2^{\ell_{\text{pump}}+r}$ of $G_{w_1,w_2,S}$.

Feasible Function: We call f a *feasible function* if the following conditions are met: (i) For each $\mathcal{G}_{w,z} = (G_{w,z}, \mathcal{L})$, the complete labeling $\mathcal{L}$ is locally consistent at all vertices in $\mathsf{Mid}(G_{w,z})$. (ii) Each partially labeled graph $\mathcal{G}_{w_1,w_2,S}$ admits a complete labeling $\mathcal{L}_\diamond$ that is locally consistent at all vertices in $\mathsf{Mid}(\mathcal{G}_{w_1,w_2,S})$.

Lemma 4.13. *Given an LCL problem $\mathcal{P}$ on cycle graphs. It is decidable whether there is a feasible function.*

Proof. Note that it is not immediate from its definition as to whether a feasible function exists is decidable, since there appears to be *infinitely* many graphs $\mathcal{G}_{w,z}$ and $\mathcal{G}_{w_1,w_2,S}$ needed to be examined. However, the following simple observations show that it suffices to check only a constant number of these graphs.

- If the complete labeling $\mathcal{L}$ of $\mathcal{G}_{w,1} = (G_{w,1}, \mathcal{L})$ is locally consistent at all vertices in $\mathsf{Mid}(G_{w,1})$, then for all $z \geq 1$, the complete labeling $\mathcal{L}$ of $\mathcal{G}_{w,z} = (G_{w,z}, \mathcal{L})$ is also locally consistent at all vertices in $\mathsf{Mid}(G_{w,z})$.
- If $\mathcal{G}_{w_1,w_2,S}$ admits a complete labeling $\mathcal{L}_\diamond$ that is locally consistent at all vertices in $\mathsf{Mid}(\mathcal{G}_{w_1,w_2,S})$, then for each S' such that $\mathsf{Type}(S) = \mathsf{Type}(S')$, the partially labeled graph $\mathcal{G}_{w_1,w_2,S'}$ also admits a complete labeling $\mathcal{L}_\diamond$ that is locally consistent at all vertices in $\mathsf{Mid}(\mathcal{G}_{w_1,w_2,S'})$. This is due to Lemma 4.2.

Therefore, to decide whether a function f is feasible, we only need to check all possible $\mathcal{G}_{w,z}$ and $\mathcal{G}_{w_1,w_2,S}$. For each w we only need to consider the graph $\mathcal{G}_{w,z}$ with $z = 1$. For each w_1 and w_2, we do not need to go over all S; we only need to consider (i) the empty string $S = \emptyset$, and (ii) for each type τ, a string $S \in \Sigma_{\text{in}}^*$ such that $\mathsf{Type}(S) = \tau$. By Lemma 4.5, for each type τ, there exists $P \in \Sigma_{\text{in}}^x$ with $x \leq \ell_{\text{pump}}$ such that $\mathsf{Type}(P) = \tau$. Therefore, a string S with $\mathsf{Type}(S) = \tau$ can be found in bounded amount of time; also notice that the number of types is bounded; see Lemma 4.4. □

For the rest of this section, we show that as long as the deterministic LOCAL complexity of $\mathcal{P}$ is $o(\log^* n)$ on cycle graphs, there exists a feasible function f. In Lemma 4.14 we show how to extract a function f from a given $o(\log^* n)$-round deterministic LOCAL algorithm $\mathcal{A}$, and then in Lemma 4.15 we prove that such a function f is feasible. Intuitively, Lemma 4.14 shows that there exists an ID-assignment such that when we run $\mathcal{A}$ on a subpath whose

input labeling is a repetition of a length-k pattern w, the output labeling is also a repetition of a length-k pattern w'. The function f will be defined as $f(w) = w'$.

Lemma 4.14. *Let $\mathcal{A}$ be any deterministic* LOCAL *algorithm that solves $\mathcal{P}$ in $t(n) = o(\log^* n)$ rounds. Then there is a number n' and function f which takes a string $w \in \Sigma_{\text{in}}^k$ with $1 \leq k \leq \ell_{\text{pump}}$ as input, and returns a string $f(w) \in \Sigma_{\text{out}}^k$ meeting the following condition. For any $P = w^i \circ w^{2r+1} \circ w^i$ such that $|w^i| \geq t(n')$ and $1 \leq |w| \leq \ell_{\text{pump}}$, there is an assignment of distinct $\Theta(\log n')$-bit IDs to the vertices in P such that the following is true. Simulating $\mathcal{A}$ on P while assuming that the total number of vertices in the underlying graph is n' yields the output labeling $f(w)^{2r+1}$ for the middle subpath w^{2r+1}.*

Proof. In this proof we assume that there is no such a number n'. Then we claim that using $\mathcal{A}$ it is possible to obtain a deterministic LOCAL algorithm for MIS on an n-vertex directed cycle G *without input labeling*, in $O(t(n)) + O(1) = o(\log^* n)$ rounds. This contradicts the well-known $\Omega(\log^* n)$ lower bound for MIS [107].

Let G be an n-vertex directed cycle without input labeling. The MIS algorithm on G is described as follows. Let $w \in \Sigma_{\text{in}}^k$ with $1 \leq k \leq \ell_{\text{pump}}$ be chosen such that for any function f, the string $f(w) \in \Sigma_{\text{out}}^k$ does not satisfy the conditions stated in the lemma for the number $n' = nk$. Define G' as the graph resulting from replacing each vertex $v \in V(G)$ with a path w. We can simulate the imaginary graph G' in the communication network G by letting each vertex $v \in V(G)$ simulate a path w.

We execute the algorithm $\mathcal{A}$ on G' while assuming that the total number of vertices is n'. The execution takes $t(n') = O(t(n))$ rounds. For each vertex $v \in V(G)$, define the color of v as the sequence of the output labels of the path w^{2r} simulated by the vertex v and the $2r - 1$ vertices following v in the directed cycle G. This gives us a proper $O(1)$-coloring, since otherwise there must exist a subpath $P = w^{2r+1}$ of G' such that the output labeling of P is of the form y^{2r+1} for some y, contradicting our choice of w. Using the standard procedure of computing an MIS from a coloring, with extra $O(1)$ rounds, an MIS of G can be obtained.

Note that there is a subtle issue about how we set the IDs of vertices in $V(G')$. The following method is guaranteed to output distinct IDs. Let $v \in V(G)$, and let $u_1, \ldots, u_k$ be the vertices in $V(G')$ simulated by v. Then we may use $\text{ID}(u_i) = k \cdot \text{ID}(v) + i$. □

Lemma 4.15. *Suppose that the deterministic* LOCAL *complexity of $\mathcal{P}$ is $o(\log^* n)$ on cycle graphs. Then there exists a feasible function f.*

Proof. Let $\mathcal{A}$ be any deterministic LOCAL algorithm that solves $\mathcal{P}$ in $t(n) = o(\log^* n)$ rounds. Let n' and f be chosen to meet the conditions in Lemma 4.14 for $\mathcal{A}$. The goal of the proof is to show that f is a feasible function. According to the conditions specified in Lemma 4.14 for the function f, we already know that the complete labeling $\mathcal{L}$ of each $\mathcal{G}_{w,z} = (G_{w,z}, \mathcal{L})$ is locally consistent at all vertices in $\mathsf{Mid}(G_{w,z})$. Therefore, all we need to do is the following. For each partially labeled graph $\mathcal{G}_{w_1,w_2,S}$, find a complete labeling $\mathcal{L}_\diamond$ that is locally consistent at all vertices in $\mathsf{Mid}(\mathcal{G}_{w_1,w_2,S})$.

Given the three parameters w_1, w_2, and S, define G as the cycle resulting from linking the two ends of the path $w_1^{\ell_{\text{pump}}} \circ w_1^{2r+1} \circ w_1^{\ell_{\text{pump}}} \circ S \circ w_2^{\ell_{\text{pump}}} \circ w_2^{2r+1} \circ w_2^{\ell_{\text{pump}}}$. Define $\mathcal{L}$ as the partial labeling of G which fixes the output labeling of the two subpaths w_1^{2r+1} and w_2^{2r+1} by $f(w_1)^{2r+1}$ and $f(w_2)^{2r+1}$, respectively. We write P_1^{mid} and P_2^{mid} to denote the two subpaths w_1^{2r+1} and w_2^{2r+1}, respectively.

In what follows, we show that the partially labeled graph $\mathcal{G} = (G, \mathcal{L})$ admits a legal labeling $\mathcal{L}_\diamond$. Since $\mathcal{G}_{w_1,w_2,S}$ is a subgraph of $\mathcal{G} = (G, \mathcal{L})$, such a legal labeling $\mathcal{L}_\diamond$ is also a complete labeling of $\mathcal{G}_{w_1,w_2,S}$ that is locally consistent at all vertices in $\mathsf{Mid}(\mathcal{G}_{w_1,w_2,S})$.

For the rest of the proof, we show the existence of $\mathcal{L}_\diamond$. This will be established by applying a pumping lemma. Define the graph G' as the result of the following operations on G.

- Replace the two subpaths $w_1^{\ell_{\text{pump}}}$ by w_1^x, where the number x is chosen such that $x|w_1| \geq 2t(n') + r$, and $\mathsf{Type}(w_1^{\ell_{\text{pump}}}) = \mathsf{Type}(w_1^x)$.
- Replace the two subpaths $w_2^{\ell_{\text{pump}}}$ by w_2^y, where the number y is chosen such that $y|w_2| \geq 2t(n') + r$, and $\mathsf{Type}(w_2^{\ell_{\text{pump}}}) = \mathsf{Type}(w_2^y)$.

The existence of the numbers x and y above is guaranteed by Lemma 4.6. The IDs of vertices in G' are assigned as follows. For $i = 1, 2$, select the IDs of the vertices in $\bigcup_{v \in P_i^{\text{mid}}} N^{t(n')}(v)$ in such a way that the output labeling of P_i^{mid} resulting from executing $\mathcal{A}$ on G' while assuming that the total number of vertices is n' is $f(w_i)^{2r+1}$. The existence of such an ID assignment is guaranteed by Lemma 4.14. For all remaining vertices in G', select their IDs in such a way that all vertices in $N^{r+t(n')}(v)$ receive distinct IDs, for each $v \in V(G')$. This ensures that the outcome of executing $\mathcal{A}$ on G' while assuming that the total number of vertices is n' is a legal labeling.

Let $\mathcal{L}'_\diamond$ be the legal labeling of G' resulting from executing $\mathcal{A}$ with the above IDs while pretending that the total number of vertices is n'. Note that $\mathcal{L}'_\diamond$ must label P_1^{mid} and P_2^{mid} by $f(w_1)^{2r+1}$ and $f(w_2)^{2r+1}$, respectively. A desired legal labeling $\mathcal{L}_\diamond$ of $\mathcal{G}$ can be obtained from

the legal labeling $\mathcal{L}'_\diamond$ of G' by applying Lemma 4.2, as we have $\mathsf{Type}(w_1^{\ell_{\text{pump}}}) = \mathsf{Type}(w_1^x)$ and $\mathsf{Type}(w_2^{\ell_{\text{pump}}}) = \mathsf{Type}(w_2^y)$. □

4.6 The $\omega(1)$—$o(\log^* n)$ Gap

In this section we prove that it is decidable whether a given LCL problem $\mathcal{P}$ has complexity $\Omega(\log^* n)$ or $O(1)$ on cycle graphs.

Lemma 4.16. *Let f be any feasible function. Let G be any cycle graph. Let $\mathscr{P}$ be any set of disjoint subgraphs in G such that the input labeling of each $P \in \mathscr{P}$ is of the form w^x such that $x \geq 2\ell_{\text{pump}} + 2r$, and $w \in \Sigma_{\text{in}}^k$ is a string with $1 \leq k \leq \ell_{\text{pump}}$. For each $P \in \mathscr{P}$, define the subgraph P' as follows. If P is a cycle, define $P' = P$. If P is a path, write $P = w^{\ell_{\text{pump}}} \circ w^i \circ w^{\ell_{\text{pump}}}$, and define P' as the middle subpath w^i. Let $\mathcal{L}$ be a partial labeling of G defined as follows. For each $P = w^x \in \mathscr{P}$, fix the output labels of each subpath w of P' by $f(w)$. Then $\mathcal{G} = (G, \mathcal{L})$ admits a legal labeling $\mathcal{L}_\diamond$.*

Proof. Define V_1 as the set of all vertices such that $v \in V_1$ if v belongs to the middle subpath w^j of some path $P = w^{\ell_{\text{pump}}} \circ w^r \circ w^j \circ w^r \circ w^{\ell_{\text{pump}}} \in \mathscr{P}$. By the definition of feasible function, $\mathcal{L}$ is already locally consistent at all vertices in V_1. Thus, all we need to do is to construct a complete labeling $\mathcal{L}_\diamond$ of $\mathcal{G} = (G, \mathcal{L})$, and argue that $\mathcal{L}_\diamond$ is locally consistent at all vertices in $V_2 = V(G) \setminus V_1$.

There are two easy special cases. If $\mathscr{P} = \emptyset$, then no output label of any vertex in G is fixed, and so $\mathcal{G}$ trivially admits a legal labeling. If $\mathscr{P}$ contains a cycle, then $\mathscr{P} = \{G\}$, and hence $\mathcal{L}$ is already a legal labeling as $V_1 = V(G)$.

In subsequent discussion, we restrict ourselves to the case that $\mathscr{P}$ is non-empty and contains only paths. The output labeling $\mathcal{L}_\diamond$ is constructed as follows. Define $\mathscr{P}_{\text{unlabeled}}$ as the maximal-length subpaths of G that are not assigned any output labels by $\mathcal{L}$. A path $P \in \mathscr{P}_{\text{unlabeled}}$ must be of the form $w_1^{\ell_{\text{pump}}} \circ S \circ w_2^{\ell_{\text{pump}}}$, where $w_1, w_2 \in \Sigma_{\text{in}}^*$ are two strings of length at least 1 and at most ℓ_{pump}, and $S \in \Sigma_{\text{in}}^*$ can be any string (including the empty string). Given $P \in \mathscr{P}_{\text{unlabeled}}$, we make the following definitions.

- Define P^+ as the subpath of G that includes P and the $r|w_1|$ vertices preceding P, and the $r|w_2|$ vertices following P in the graph G. Note that the set V_2 is exactly the union of vertices in P^+ for all $P \in \mathscr{P}_{\text{unlabeled}}$.

- Define P^{++} as the subpath of G that includes P and the $2r|w_1|$ vertices preceding P, and the $2r|w_2|$ vertices following P in the graph G. The path P^{++} must be of the form $w_1^{\ell_{\text{pump}}+2r} \circ S \circ w_2^{\ell_{\text{pump}}+2r}$, and the labeling $\mathcal{L}$ already fixes the output labels of the first $2r|w_1|$ and the last $2r|w_2|$ vertices of P^{++} by $f(w_1)^{2r}$ and $f(w_2)^{2r}$, respectively.

Observe that the path $P^{++} = w_1^{\ell_{\text{pump}}+2r} \circ S \circ w_2^{\ell_{\text{pump}}+2r}$ together with the labeling $\mathcal{L}$ is exactly the partially labeled graph $\mathcal{G}_{w_1,w_2,S}$. We assign the output labels to the vertices in P by the labeling $\mathcal{L}_\diamond$ guaranteed in the definition of feasible function. It is ensured that the labeling of all vertices within P^+ are locally consistent. By doing so for each $P \in \mathscr{P}_{\text{unlabeled}}$, we obtain a desired complete labeling that is locally consistent at all vertices in V_2. □

Lemma 4.17. *Suppose that there is a feasible function f for the LCL problem $\mathcal{P}$. Then there is an $O(1)$-round deterministic* LOCAL *algorithm $\mathcal{A}$ on cycle graphs.*

Proof. The first step of the algorithm $\mathcal{A}$ is to compute an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition in $O(1)$ rounds by Lemma 4.12. We set $\ell_{\text{count}} = 2\ell_{\text{pump}} + 2r$ and $\ell_{\text{width}} = \ell_{\text{pattern}} = \ell_{\text{pump}}$. We assume $|V(G)| > 2\ell_{\text{width}}$. Recall that an $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition decomposes the cycle G into two sets of disjoint subgraphs $\mathscr{P}_{\text{short}}$ and $\mathscr{P}_{\text{long}}$.

Define G' as the graph resulting from applying the following operations on G. For each $P \in \mathscr{P}_{\text{short}}$, replace the path P by the path $P^* = x \circ y^i \circ z$ such that $i = \ell_{\text{count}}$, $1 \le |y| \le \ell_{\text{pattern}}$, and the type of P^* is the same as the type of P. The path P^* is obtained via Lemma 4.5. Note that each path $P \in \mathscr{P}_{\text{short}}$ has at least $\ell_{\text{width}} = \ell_{\text{pump}}$ vertices and at most $2\ell_{\text{width}} = 2\ell_{\text{pump}}$ vertices. Define $\mathscr{P}^*$ as the set of all P^* such that $P \in \mathscr{P}_{\text{short}}$. The graph G' is simulated in the communication graph G by electing a leader for each path $P \in \mathscr{P}_{\text{short}}$ to simulate P^*.

Calculate a partial labeling $\mathcal{L}'$ of G' using the feasible function f as follows. Recall $\ell_{\text{count}} = 2\ell_{\text{pump}} + 2r$. For each $P^* = x \circ y^{\ell_{\text{pump}}} \circ y^{2r} \circ y^{\ell_{\text{pump}}} \circ z \in \mathscr{P}^*$, label the middle subpath y^{2r} by the function f. For each $P = w^{\ell_{\text{pump}}} \circ w^i \circ w^{\ell_{\text{pump}}} \in \mathscr{P}_{\text{long}}$, label the middle subpath w^i by $f(w)^i$. Even though a path $P \in \mathscr{P}_{\text{long}}$ can have $\omega(1)$ vertices, this step can be done locally in $O(1)$ rounds due to the following property of $(\ell_{\text{width}}, \ell_{\text{count}}, \ell_{\text{pattern}})$-partition. All vertices in a path $P \in \mathscr{P}_{\text{long}}$ agree with the same direction and know the primitive string w.

By Lemma 4.16, the remaining unlabeled vertices in G' can be labeled to yield a legal labeling of G'. This can be done in $O(1)$ rounds since the connected components formed by unlabeled vertices have at most $O(1)$ vertices. Given any valid labeling of G', a legal

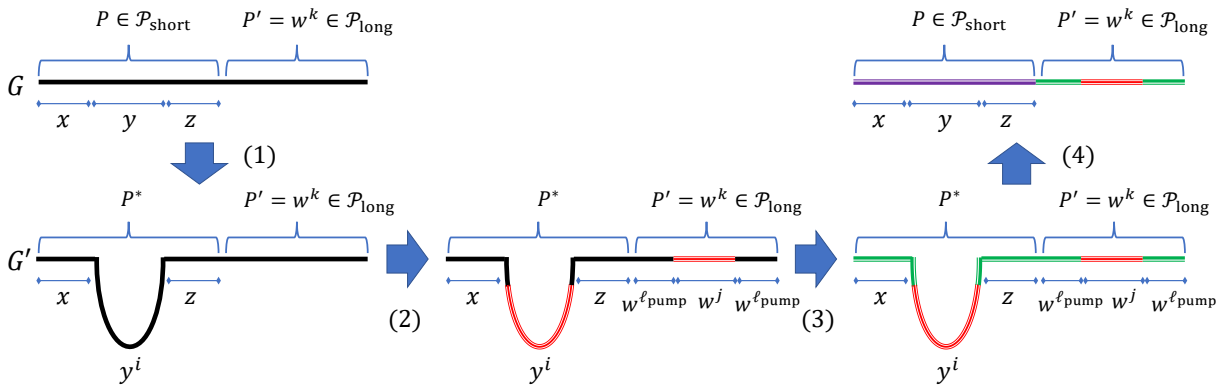

Figure 4.2: Illustration of Lemma 4.17.

labeling of G can be obtained by applying Lemma 4.2 in $O(1)$ rounds. Remember that $\mathsf{Type}(P) = \mathsf{Type}(P^*)$ for each $P \in \mathscr{P}_{\text{short}}$, and G' is exactly the result of replacing each $P \in \mathscr{P}_{\text{short}}$ by P^*. □

See Figure 4.2 for an illustration of Lemma 4.17: (1) applying a pumping lemma to extend each path $P \in \mathscr{P}_{\text{short}}$; (2) labeling the middle subpath y^{2r} of $P^* = x \circ y^{\ell_{\text{pump}}} \circ y^{2r} \circ y^{\ell_{\text{pump}}} \circ z \in \mathscr{P}^*$ and the middle subpath w^j of $P' = w^{\ell_{\text{pump}}} \circ w^i \circ w^{\ell_{\text{pump}}} \in \mathscr{P}_{\text{long}}$ by the function f; (3) the remaining unlabeled vertices in G' can be labeled to yield a legal labeling of G' by Lemma 4.16; (4) since $\mathsf{Type}(P) = \mathsf{Type}(P^*)$ for each $P \in \mathscr{P}_{\text{short}}$, we can recover a legal labeling of G by re-labeling vertices in each $P \in \mathscr{P}_{\text{short}}$.

Combining Lemma 4.13, Lemma 4.15, and Lemma 4.17, we have proved Theorem 4.2. That is, for any LCL problem $\mathcal{P}$ on cycle graphs, its deterministic LOCAL complexity is either $\Omega(\log^* n)$ or $O(1)$. Moreover, there is an algorithm that decides whether $\mathcal{P}$ has complexity $\Omega(\log^* n)$ or $O(1)$ on cycle graphs; for the case the complexity is $O(1)$, the algorithm outputs a description of an $O(1)$-round deterministic LOCAL algorithm that solves $\mathcal{P}$.

Chapter 5

The Complexity of Distributed Edge Coloring

5.1 Overview

Recall that a *k-edge coloring* of a graph $G = (V, E)$ is a function $\phi : E \to \{1, \ldots, k\}$ such that edges sharing an endpoint are colored differently; the parameter k is called the *palette size*. In this chapter, we study edge coloring problems in the LOCAL model. The distributed complexity of computing a k-edge coloring depends heavily on the value of k, relative to the maximum degree Δ, and whether vertices can generate random bits.

5.1.1 Edge Coloring Algorithms

In this section, we review previous edge coloring algorithms in descending order by palette size; see Table 5.1 for a summary.

Edge coloring can be interpreted as a *vertex* coloring problem on the *line graph* $L(G)$, in which edges becomes vertices and two edges are adjacent if they share an endpoint; the line graph has maximum degree $\hat{\Delta} = 2\Delta - 2$. Applied to $L(G)$, Linial's [107] vertex coloring algorithm will compute an $O(\hat{\Delta}^2)$-edge coloring in $O(\log^* n - \log^* \hat{\Delta} + 1)$ time. Using the fastest deterministic $(\hat{\Delta}+1)$-vertex coloring algorithms [65, 120], $(2\Delta-1)$-edge coloring is solved in $\min\{2^{O(\sqrt{\log n})}, \tilde{O}(\sqrt{\Delta}) + O(\log^* n)\}$ time. Barenboim, Elkin, and Maimon [22] gave deterministic algorithms for $(2^k\Delta)$-edge coloring ($k \geq 2$) in $\tilde{O}(k\Delta^{1/2k}) + O(\log^* n)$ time.

Barenboim, Elkin, Pettie, and Schneider [23] proved that $O(\log \Delta)$ iterations of the

natural randomized $(2\Delta - 1)$-edge coloring algorithm effectively *shatters* the graph into uncolored components of $n' = \text{poly}(\log n)$ vertices; then we can employ a deterministic list coloring algorithm to color these components in $2^{O(\sqrt{\log n'})} = 2^{O(\sqrt{\log \log n})}$ time [120]. Thus, the total time complexity is $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$.

Elkin, Pettie, and Su [55] proved that when $\Delta > (\log n)^{1+\gamma}$ (for some constant γ), $(2\Delta - 1)$-edge coloring can be solved in $O(\log^* n)$ time in RandLOCAL. Fischer, Ghaffari, and Kuhn [62] proved that $(2\Delta - 1)$-edge coloring can be solved in $O(\log^7 \Delta \log n)$ time in DetLOCAL. This was later improved to $O(\log^2 n \log^4 \Delta)$ by Ghaffari, Harris, and Kuhn [69]. Together with [23] and [55], this implies an $O((\log \log n)^6)$-time RandLOCAL algorithm. Using a slightly larger palette of $(2 + \epsilon)\Delta$ colors, $\epsilon > 1/\log \Delta$, Ghaffari et al. [70] (improving [**GhaffariS17**]) gave an $O(\epsilon^{-1} \log^2 \Delta \log \log \Delta (\log \log \log \Delta)^{1.71} \log n)$-time DetLOCAL edge coloring algorithm.

Below the Greedy Threshold. The number "$2\Delta - 1$" arises because it is the *smallest* palette size with the property that any partial coloring can be extended to a total coloring, by the trivial greedy algorithm. Below the *greedy threshold* $2\Delta - 1$, iterative coloring algorithms must be more careful in how they proceed. In particular, at intermediate stages in the algorithm, edges must keep their available palettes relatively large compared to the size of their uncolored neighborhood.

Using the *Rödl nibble* technique, Dubhashi, Grable, and Panconesi [52] gave a RandLOCAL algorithm for $(1 + \epsilon)\Delta$-edge coloring in $O(\log n)$ time, provided that Δ is sufficiently large, e.g., even when ϵ is constant, $\Delta > (\log n)^{1+\gamma}$. Elkin, Pettie, and Su [55] gave RandLOCAL algorithms for $(1 + \epsilon)\Delta$-edge coloring that are faster when Δ is large and work for *all* Δ via a reduction to the distributed Lovász local lemma (LLL); see Section 5.1.3 for a discussion of the distributed LLL. The $(1 + \epsilon)\Delta$-edge coloring problem is solved in $O(\log^* n \cdot \lceil \frac{\log n}{\Delta^{1-o(1)}} \rceil)$ time. The running time of the Dubhashi-Grable-Panconesi and Elkin-Pettie-Su algorithms depend *polynomially* on ϵ^{-1}. In both algorithms it is clear that ϵ need not be constant, but it is not self-evident how small it can be made *as a function of* Δ.

The $\lceil \frac{\log n}{\Delta^{1-o(1)}} \rceil$-factor in the time complexity is due to the application of the Chung-Pettie-Su LLL algorithm [44]. If Δ is sufficiently small, this algorithm can be sped up using faster LLL algorithms for small degree graphs [61, 69].

Limits to Coloring Strategies. A natural limit for randomized coloring strategies is a $(\Delta + O(\sqrt{\Delta}))$-size palette. This is the threshold at which we have a constant probability of being able to color e, given a *random* feasible coloring of its neighborhood. Edge coloring with this palette size was achieved in 1987 by Karloff and Shmoys [95] in the context of parallel (PRAM) algorithms, but has never been achieved in the LOCAL model.

We cannot hope to use fewer than $\Delta + 1$ colors on general graphs. Vizing [142] proved that $\Delta + 1$ suffices for any graph, and Holyer [88] proved that it is NP-hard to tell if a graph is Δ-colorable. The best *sequential* $(\Delta + 1)$-edge coloring algorithms [7, 67] run in $O(\min\{\Delta m \log n,\, m\sqrt{n \log n}\})$ time and are not suited for implementation in the LOCAL model. When the palette size is $\Delta + o(\sqrt{\Delta})$, a natural way to solve the problem [7, 67] is to begin with any maximal partial coloring, and then iteratively *recolor* portions of the graph (e.g., along "augmenting paths") so that at least one uncolored edge can be legally colored. This approach was successfully employed by Panconesi and Srinivasan [122] in their distributed algorithm for Brooks' theorem, which states that any graph with $\Delta \geq 3$ having no $(\Delta + 1)$-cliques is Δ-vertex colorable. They proved that for any partial coloring, there exists an augmenting path with length $O(\log_\Delta n)$, and that given a $(\Delta + 1)$-vertex coloring, a Δ-vertex coloring could be computed in $O(\log^2 n \log_\Delta n)$ additional time (this complexity was later improved in [75]).

5.1.2 Lower Bounds

Linial's $\Omega(\log^* n)$ lower bound for $O(1)$-coloring the ring [107, 115] implies that $f(\Delta)$-edge coloring also cannot be computed in $o(\log^* n)$ time, for any function f. To the best of our knowledge, none of the other published lower bounds applies directly to the edge coloring problem. Kuhn, Moscibroda, and Wattenhofer's $\Omega(\min\{\frac{\log \Delta}{\log \log \Delta}, \sqrt{\frac{\log n}{\log \log n}}\})$ lower bounds apply to MIS and maximal matching, but not to any vertex or edge coloring problem. Linial's $\Omega(\log_\Delta n)$ lower bound [107] (see [129, p. 265]) on $o(\Delta / \ln \Delta)$-vertex coloring trees does not imply anything for edge coloring trees. The lower bounds of Brandt et al. [31] (RandLOCAL $\Omega(\log_\Delta \log n)$) and the Chapter 2 of this thesis (DetLOCAL $\Omega(\log_\Delta n)$) for *sinkless orientation* and Δ*-vertex coloring trees* do not naturally generalize to edge coloring. In fact, Brandt et al.'s lower bound technique requires that the input graph be Δ-regular and *come equipped* with a Δ-edge coloring.

Palette Size	Time	(Rand)	Notes	References
$f(\Delta)$	$\Omega(\log^* n)$	R	$\Delta = O(1)$	[107, 115]
$O(\Delta^2)$	$O(\log^* n - \log^* \Delta + 1)$	$\star$	Vertex coloring $L(G)$	[107]
$\Delta^{1+\epsilon}$	$O(\log \Delta \log n)$		Vertex coloring $L(G)$	[18]
$O(\Delta \log n)$	$O(\log^4 n)$			[47]
$t(2\Delta - 2)$	$(\Delta/t)^{O(1)} \cdot O(\log n)$		Vertex coloring $L(G)$	[18]
$2^k\Delta$	$\tilde{O}(k\Delta^{1/2k}) + O(\log^* n)$	$\star$	$k \geq 2$	[**GhaffariS17**, 22]
$(2+\epsilon)\Delta$	$O(\epsilon^{-3} \log^{11} n)$			[**GhaffariS17**]
	$O(\epsilon^{-1} \log \Delta^{2+o(1)} \log n)$	$\star$	$\epsilon > 1/\log \Delta$	[70]
$2\Delta - 1$	$2^{O(\sqrt{\log n})}$		Vertex coloring $L(G)$	[120]
	$\tilde{O}(\sqrt{\Delta}) + O(\log^* n)$	$\star$	Vertex coloring $L(G)$	[65]
	$O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$	R	Vertex coloring $L(G)$	[23]
	$O(\log^* n)$	R$\star$	$\Delta > (\log n)^{1+o(1)}$	[55]
	$2^{O(\sqrt{\log \log n})}$	R		[55]
	$O(\log^4 \Delta \log^2 n)$	$\star$		[69]
	$O((\log \log n)^6)$	R$\star$		[23]+[55]+[69]
$2\Delta - 2$	$\Omega(\log_\Delta \log n)$	R		**new**
	$\Omega(\log_\Delta n)$			**new**
1.6Δ	$O(\log n)$	R	$\Delta > \log^{1+o(1)} n$	[121]
$(1+\epsilon)\Delta$	$O(\epsilon^{-1} \log \epsilon^{-1} + \log n)$	R	$\Delta > (\log n)^{1+\gamma(\epsilon)}$	[52]
	$O\left((\epsilon^{-2} \log \epsilon^{-1} + \log^* \Delta) \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil\right)$	R	$\Delta > \Delta_\epsilon$	[55]
	$O\left(\log \epsilon^{-1} \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil + \log^* n\right)$	R$\star$	$\epsilon\Delta > (\log n)^{1+o(1)}$	**new**
	$O\left(\log \epsilon^{-1} \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil + (\log \log n)^{3+o(1)}\right)$	R$\star$	$\Delta > \Delta_\epsilon$	**new**
$\Delta + \tilde{O}(\sqrt{\Delta})$	$O\left(\log \Delta \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil + (\log \log n)^{3+o(1)}\right)$	R$\star$		**new**
$\Delta + 1$	$\mathsf{diameter}(G)$	$\star$		[142]

Table 5.1: A history of notable edge coloring algorithms and lower bounds, in descending order by palette size. Some $(2\Delta - 1)$-edge coloring algorithms that follow from vertex coloring $L(G)$, such as [12, 17, 21, 103], have been omitted for brevity. RandLOCAL algorithms are marked with R; all others work in DetLOCAL. Those algorithms that are the "best" in any sense are marked with a $\star$.

Criterion	Time	Rand/Det	Notes	Reference
$ep(d+1) < 1$	$O(\mathsf{MIS} \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[112]
	$O(\mathsf{WeakMIS} \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[44]
	$O(\log d \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[68]+[44]
$epd^2 < 1$	$O(\log_{1/epd^2} n)$	Rand	also asymmetric criterion	[44]
$\text{poly}(d)2^d < 1$	$O(\log n / \log\log n)$	Rand		[44]
$p(ed)^\lambda < 1$	$O(n^{1/\lambda} \cdot 2^{O(\sqrt{\log n})})$	Det	Any $\lambda \geq 1$	[61]
$p(ed)^{4\lambda} < 1$	$O(d^2 + (\log n)^{1/\lambda} \cdot 2^{O(\sqrt{\log\log n})})$	Rand	Any $\lambda \geq 8$	[61]
$p(ed)^{32} < 1$	$2^{O(\sqrt{\log\log n})}$	Rand	Requires $d < (\log\log n)^{1/5}$	[61]
$p20000d^8 < 1$	$\exp^{(i)}\left(O\left(\sqrt{\log^{(i+1)} n}\right)\right)$	Rand	Requires $d < 2^{\sqrt{\log^{(i+1)} n}}$	[69]
$p(ed)^{d^2+1} < 1$	$O(d^2 + \log^* n)$	Det		[61]

Lower Bounds (apply to tree-structured instances)

$p \cdot f(d) < 1$	$\Omega(\log^* n)$	Rand	Any f	[44]
$p \cdot f(d) \leq 1$	$\Omega(\log_{\log(1/p)} \log n)$	Rand	Any $f(d) \leq 2^d$	[31]
$p \cdot f(d) \leq 1$	$\Omega(\log_d n)$	Det	Any $f(d) \leq 2^d$	Theorem 2.3

LLL for Tree-Structured Instances

$p(ed)^2 < 1$	$O(\log n)$	Det		**new**
$p(ed)^\lambda < 1$	$O(\max\{\log_\lambda n, \frac{\log n}{\log\log n}\})$	Det	$\lambda \geq 2$	**new**
$p(ed)^\lambda < 1$	$O(\max\{\log_\lambda \log n, \frac{\log\log n}{\log\log\log n}\})$	Rand	$\lambda \geq 2(4^r + 8r)$	**new**

Table 5.2: A survey of distributed LLL algorithms (with a symmetric LLL criterion). $\mathsf{MIS} = O(\min\{d + \log^* n, \log d + 2^{O(\sqrt{\log\log n})}\})$ [21, 68] is the complexity of computing a maximal independent set in a graph with maximum degree d. $\mathsf{WeakMIS} = O(\log d)$ [68] is the task of finding an independent set I such that the probability that v is not in/adjacent to I is $1/\text{poly}(d)$. All lower bounds apply even to tree-structured instances. We do not optimize the LLL criterion $\lambda \geq 2(4^r + 8r)$.

5.1.3 Distributed Lovász Local Lemma

Randomized coloring algorithms in the LOCAL model are often composed of $O(1)$-round routines that commit to a partial coloring, whose local probability of failure is small, as a function of Δ. Using a distributed Lovász local lemma (LLL) algorithm, we can guarantee global success with probability $1 - 1/\text{poly}(n)$ (using a randomized LLL algorithm) or even 1 (using a deterministic LLL algorithm). Table 5.2 summarizes distributed LLL algorithms under different symmetric criteria $p \cdot f(d) < 1$, where p is the local probability of failure and d is the maximum degree in the dependency graph. In distributed coloring algorithms it is typical to see $d = \text{poly}(\Delta)$ and $p = \exp(-d^{\Omega(1)})$, i.e., *any* polynomial LLL criterion of the form $p(ed)^c < 1$ where $c = O(1)$ is good enough.

We conjecture that the RandLOCAL complexity of the LLL under some polynomial LLL criterion is $O(\log \log n)$, matching the Brandt et al. [31] lower bound. If this conjecture were true, due to the necessity of graph shattering (Theorem 2.1), an optimal randomized LLL algorithm should be structured as follows. It must combine an $O(\log n)$-time deterministic LLL algorithm and an $O(\log \log n)$-time randomized *graph shattering* routine to break the dependency graph into $\text{poly}(\log n)$-size LLL instances. Fischer and Ghaffari [61] exhibited a deterministic $n^{1/\lambda + o(1)}$-time algorithm for LLL criterion $p(ed)^\lambda < 1$, and an $O(d^2 + \log^* n)$ routine to shatter the dependency graph into $\text{poly}(\log n)$-size components. More recently, Ghaffari, Harris, and Kuhn [69] developed a generic derandomization method for the LOCAL model that implies randomized LLL algorithms with time $\exp^{(i)}(O(\sqrt{\log^{(i+1)} n}))$ for sufficiently small d.

5.1.4 New Results

We present new upper and lower bounds on the complexity of edge coloring in the regimes between palette size Δ and $2\Delta - 2$, i.e., strictly below the "greedy" threshold $2\Delta - 1$.

Round Elimination. Our first result is a lower bound on $(2\Delta - 2)$-edge coloring using a simplified version of Brandt et al.'s [31] *round elimination* technique. Roughly speaking, their idea is to convert any randomized t-round algorithm with local error probability p into a $(t-1)$-round algorithm with error probability $\approx p^{1/\Delta}$. By iterating the procedure they obtain a 0-round algorithm with error probability $\approx p^{\Delta^t}$. If any 0-round algorithm must have constant probability of failure, then $t = \Omega(\log_\Delta \log p^{-1})$. By setting $p = 1/\text{poly}(n)$ we get $\Omega(\log_\Delta \log n)$ RandLOCAL lower bounds for some problems, e.g., sinkless orientation.

We present a much simplified round elimination technique that *appears* to give quantitatively worse bounds, but which can be *automatically* strengthened to match those of [31]. Rather than try to shave one round off the running time of *every* processor, it is significantly simpler to do it piecemeal, which leads us to the useful concept of an *irregular* time profile. Suppose that the graph is initially k-edge colored, k being at least $2\Delta - 1$ so as not to trivialize the problem. An algorithm has irregular time profile $\mathbf{t} = (t_1, \ldots, t_k)$ if edges with input color i choose their output color by examining only their t_i-neighborhood. In our simplified round-elimination technique, we show that any algorithm with time profile $(\underbrace{t, t, \cdots, t}_{i}, \underbrace{t-1, \cdots, t-1}_{k-i})$ and error probability p can be transformed into one with time profile $(\underbrace{t, t, \cdots, t}_{i-1}, \underbrace{t-1, \cdots, t-1}_{k-i+1})$ and error probability $O(p^{1/3})$, *only* by changing the algorithm for edges initially colored i. By iterating this process we arrive at $\Omega(\Delta^{-1} \log\log p^{-1})$ lower bounds, which has a weaker dependence on Δ than [31]. By following the proofs in Chapter 2, any randomized lower bound of this type implies $\Omega(\log_\Delta n)$ lower bounds in DetLOCAL, and hence $\Omega(\log_\Delta \log n)$ lower bounds in RandLOCAL.

Faster $(1+\epsilon)\Delta$-edge Coloring. The $(1+\epsilon)\Delta$-edge coloring algorithms of [52, 55] are slow (with a polynomial dependence on ϵ^{-1}) and have limits on how small ϵ can be, as a function of Δ. We prove that the most "natural" randomized algorithm converges exponentially faster with ϵ^{-1} and can achieve palette sizes close to the minimum of $\Delta + \tilde{O}(\sqrt{\Delta})$ allowed by the nibble method. In particular, for any $\epsilon = \tilde{\Omega}(1/\sqrt{\Delta})$, $(1+\epsilon)\Delta$-edge coloring is reducible to $O(\log \epsilon^{-1})$ instances of the Lovász local lemma with local failure probability $\exp(-\epsilon^2 \Delta^{1-o(1)})$, plus one instance of $O(\Delta)$-edge coloring, which can be solved quickly using [23, 55, 70]. When $\epsilon^2 \Delta \gg \log n$ the error is $1/\mathrm{poly}(n)$; otherwise we can invoke a distributed LLL algorithm [44, 61, 112]. The $\left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil$-factor in Table 5.1 is due to the $O(\log_{1/epd^2} n)$-time LLL algorithm of [44], with $1/epd^2 = \exp(\epsilon^2 \Delta^{1-o(1)})$.

Upper Bounds on Trees. Our lower bound on $(2\Delta - 2)$-edge coloring applies even to trees. In order to adapt our randomized $(1+\epsilon)\Delta$-edge coloring algorithms to trees, we need a special LLL algorithm for *tree structured* dependency graphs. Using the framework of Fischer and Ghaffari [61], we give a deterministic $O(\max\{\log_\lambda n, \log n / \log\log n\})$-time LLL algorithm for such instances under criterion $p(ed)^\lambda < 1$, $\lambda \geq 2$. The algorithm is based on a special network decomposition algorithm for tree-structured graphs, in which one color class

has diameter $O(\log_\lambda n)$ while the other color classes have diameter 0. We also present a new graph shattering routine for tree-structured LLL instances that runs in time $O(\log_\lambda \log n)$, improving the $O(d^2 + \log^* n)$-time shattering routine of [61] when d is not too small. (The new graph shattering method can be viewed as an algorithm that computes the final state of a certain contagion dynamic exponentially faster than simulating the actual contagion.) By composing these results we obtain a randomized $O(\max\{\log_\lambda \log n, \log\log n/\log\log\log n\})$ LLL algorithm for trees, which essentially matches the lower bound of [31]. See Table 5.2.

A Distributed Vizing's Theorem? Suppose that a distributed $(\Delta+1)$-edge coloring algorithm begins with a partial coloring and iteratively recolors subgraphs, always increasing the subset of colored edges. If this algorithm works correctly given *any* partial coloring, we prove that it takes $\Omega(\Delta \log n)$ time in any LOCAL model, and more generally, $(\Delta + c)$-coloring takes $\Omega(\frac{\Delta}{c} \log n)$ time. This establishes a quantitative difference between the "locality" of Vizing's theorem and Brooks' theorem [122].

Organization. In Section 5.2 we give lower bounds on $(2\Delta - 2)$-edge coloring. In Section 5.3 we give a randomized $(1+\epsilon)\Delta$-edge coloring algorithm, which requires a distributed LLL algorithm when $\epsilon^2\Delta$ is sufficiently small. Some details in Section 5.3 are left to Section 5.4.

In Section 5.5 we give new LLL algorithms for tree-structured dependency graphs. In Section 5.6 we present new network decomposition algorithms for trees, which are used in Section 5.5. In Section 5.7 we prove some bounds on the problems of Δ- and $(\Delta+1)$-edge coloring trees. In Section 5.8 we give lower bounds on a class of "recoloring" algorithms for Vizing's theorem.

Remark 5.1. *After the initial publication of this work in [38], we learned that Molloy and Reed [111] also obtained a bound of $\Delta + O(\sqrt{\Delta}\log^4 \Delta)$ on the palette size for edge coloring. Their algorithm was more general in that it extends to k-uniform hypergraphs (with palette size $\Delta + O(\Delta^{1-1/k}\log^4 \Delta)$) and list-coloring. The main difference between our work and theirs [111] is the analysis. We use a concentration bound [53, Equation (8.5)] that takes into account the variance of each variable. The analysis of [111] is based on Talagrand's concentration inequality. Our result is slightly better in terms of the polylog-factor, and it also improves the existential bound on the palette size for list edge coloring. Specifically, if each edge is given a list of $(1+\epsilon)\Delta$ with $\epsilon = \omega((\log^{2.5}\Delta)/\sqrt{\Delta})$ colors, then the graph admits a proper list edge coloring.*

5.2 Lower Bound for $(2\Delta - 2)$-Edge Coloring

The *sinkless orientation* problem is to direct the edges such that no vertex has out-degree zero. Since this problem becomes *harder* with fewer edges, we let Δ denote the *minimum* degree in this problem, whereas in the edge coloring problem Δ is still the maximum degree. We first observe that sinkless orientation on 2-vertex colored bipartite graphs is reducible to $(2\Delta - 2)$-edge coloring.

Theorem 5.1. *Suppose $\mathcal{A}_{e.c.}$ is a t-round $(2\Delta-2)$-edge coloring algorithm with local failure probability p. There is a $(t+1)$-round sinkless orientation algorithm $\mathcal{A}_{s.o.}$ for 2-vertex colored graphs with minimum degree Δ whose local failure probability is p.*

Proof. $\mathcal{A}_{e.c.}$ produces a proper *partial* $(2\Delta - 2)$-edge coloring $\phi : E \to \{1, \ldots, 2\Delta - 2, \perp\}$ such that for all $v \in V$, $\Pr[\exists (u, v) : \phi(u, v) = \perp] \leq p$, i.e., a vertex errs if not all of its edges are colored. Suppose we are given a bipartite graph $G = (V, E)$ with a 2-coloring $V \to \{0, 1\}$ and minimum degree Δ. In the first round of $\mathcal{A}_{s.o.}$, each vertex selects Δ of its incident edges arbitrarily and notifies the other endpoint whether it was selected. Let $G' = (V, E')$ be the subgraph of edges selected by *both* endpoints. The algorithm $\mathcal{A}_{s.o.}$ runs $\mathcal{A}_{e.c.}$ on G' for t rounds to get a partial coloring $\phi : E' \to \{1, \ldots, 2\Delta - 2, \perp\}$, then orients the edges as follows. Recall that the underlying graph G is 2-vertex colored. Let $e = \{u_0, u_1\} \in E$ be an edge with u_j colored $j \in \{0, 1\}$. If both u_0 and u_1 do not select e, then e is oriented arbitrarily. Otherwise, $\mathcal{A}_{s.o.}$ orients e as follows.

$$\mathcal{A}_{s.o.}(\{u_0, u_1\}) = \begin{cases} 0 \to 1 & \text{if } \{u_0, u_1\} \in E' \text{ and } \phi(u_0, u_1) \in \{1, 2, \ldots, \Delta - 1, \perp\}, \\ & \quad \text{or if only } u_0 \text{ selected } \{u_0, u_1\}. \\ 0 \leftarrow 1 & \text{if } \{u_0, u_1\} \in E' \text{ and } \phi(u_0, u_1) \in \{\Delta, \ldots, 2\Delta - 2\}, \\ & \quad \text{or if only } u_1 \text{ selected } \{u_0, u_1\}. \end{cases}$$

The only way a vertex v can be a sink is if (i) v has degree exactly Δ in G', (ii) v is colored 1, and (iii) each edge e incident to v has $\phi(e) \in \{1, 2, \ldots, \Delta - 1, \perp\}$. Criterion (iii) only occurs with probability at most p. □

Thus, any lower bound for sinkless orientation on 2-vertex colored graphs also applies to $(2\Delta - 2)$-edge coloring. Define $\mathcal{T}_\Delta$ to be an infinite Δ-regular tree whose vertices are properly 2-colored by $\{0, 1\}$ and whose edges are assigned a proper $(2\Delta - 1)$-coloring

uniformly at random. One could generate such a coloring as follows. Pick an edge and assign it a random color, then iteratively pick any vertex u with one incident edge colored, choose $\Delta - 1$ colors at random from the $\binom{2\Delta-2}{\Delta-1}$ possibilities, then assign them to u's remaining uncolored edges uniformly at random. Randomized algorithms that run on $\mathcal{T}_\Delta$ know the edge coloring and how it was generated. Thus, the probability of *failure* depends on the random bits generated by the algorithm, and those used to generate the edge coloring.

For simplicity we suppose that the *edges* host processors, and that two edges can communicate if they are adjacent in the line graph $L(\mathcal{T}_\Delta)$. Define $N^t(e)$ to be all edges within distance t of e in the line graph; we also use $N^t(e)$ to refer to *all information* stored in the processors within $N^t(e)$; this includes edge coloring, vertex coloring, and the random bits.

Recall that an algorithm has irregular time profile $\mathbf{t} = (t_1, \ldots, t_k)$ if edges with input color i decide their output by examining only their t_i-neighborhood. By definition, a time-t algorithm has time profile $(t, t, t, \ldots, t)$.

Lemma 5.1 (Round Elimination Lemma)**.** *Suppose $\mathcal{A}_{s.o.}$ is a sinkless orientation algorithm for $\mathcal{T}_\Delta$ with error probability p and time profile $(\underbrace{t, t, \ldots, t}_{i}, t-1, \ldots, t-1)$, i.e., edges colored $\{1, \ldots, i\}$ halt after t rounds and the others after $t-1$ rounds. There exists a sinkless orientation algorithm $\mathcal{A}'_{s.o.}$ with error probability $3p^{1/3}$ and time profile $(\underbrace{t, t, \ldots, t}_{i-1}, t-1, \ldots, t-1)$.*

Proof. Only edges colored i modify their algorithm; all others behave identically under $\mathcal{A}'_{s.o.}$ and $\mathcal{A}_{s.o.}$. Let $e_0 = \{u_0, u_1\}$ be an edge colored i with u_j colored $j \in \{0, 1\}$ and let the remaining edges incident to u_0 and u_1 be $\{e_1, \ldots, e_{\Delta-1}\}$ and $\{e_\Delta, \ldots, e_{2\Delta-2}\}$, respectively. Consider the following two events regarding the output of $\mathcal{A}_{s.o.}$.

$$\mathcal{E}_0 : \forall j \in [1, \Delta-1], \mathcal{A}_{s.o.}(e_j) = 0 \leftarrow 1 \qquad \text{I.e., } u_0 \text{ has outdegree 0 in } G - \{e_0\}$$
$$\mathcal{E}_1 : \forall j \in [\Delta, 2\Delta-2], \mathcal{A}_{s.o.}(e_j) = 0 \rightarrow 1 \qquad \text{I.e., } u_1 \text{ has outdegree 0 in } G - \{e_0\}$$

If both events hold, then either u_0 or u_1 must be a sink, so

$$\Pr[\mathcal{E}_0 \cap \mathcal{E}_1] \leq 2p \tag{5.1}$$

On edge e_0, $\mathcal{A}'_{s.o.}$ runs for $t-1$ rounds and determines whether the following events occur.

$$\mathcal{E}_0^\star : \left[\Pr[\mathcal{E}_0 \mid N^{t-1}(e_0)] \geq p^{1/3}\right], \qquad \mathcal{E}_1^\star : \left[\Pr[\mathcal{E}_1 \mid N^{t-1}(e_0)] \geq p^{1/3}\right]$$

Notice that if we inspect $N^{t-1}(e_0)$, and condition on the information seen in $N^{t-1}(e_0)$, the events $\mathcal{E}_0$ and $\mathcal{E}_1$ become independent, since they now depend on disjoint sets of random variables. Specifically, $\mathcal{E}_0$ depends on $\bigcup_{j\in[1,\Delta-1]} N^t(e_j)\backslash N^{t-1}(e_0)$ and $\mathcal{E}_1$ depends on $\bigcup_{j\in[\Delta,2\Delta-2]} N^t(e_j)\backslash N^{t-1}(e_0)$. Thus,

$$\Pr[\mathcal{E}_0 \cap \mathcal{E}_1 \mid N^{t-1}(e_0)] = \Pr[\mathcal{E}_0 \mid N^{t-1}(e_0)] \cdot \Pr[\mathcal{E}_1 \mid N^{t-1}(e_0)] \tag{5.2}$$

Since $\mathcal{E}_0^\star, \mathcal{E}_1^\star$ are determined by $N^{t-1}(e_0)$, (5.2) implies that $\Pr[\mathcal{E}_0 \cap \mathcal{E}_1 \,|\, \mathcal{E}_0^\star \cap \mathcal{E}_1^\star] \geq p^{2/3}$, and with (5.1) we deduce that

$$\Pr[\mathcal{E}_0^\star \cap \mathcal{E}_1^\star] \leq 2p^{1/3} \tag{5.3}$$

The algorithm $A'_{s.o.}$ orients e_0 as follows.

$$\mathcal{A}'_{s.o.}(e_0) = \begin{cases} 0 \to 1 & \text{if } \mathcal{E}_0^\star \text{ holds} \\ 0 \leftarrow 1 & \text{otherwise} \end{cases}$$

We now calculate the failure probabilities of u_0 and u_1.

$$\begin{aligned}
\Pr[u_0 \text{ is a sink}] &= \Pr[\overline{\mathcal{E}_0^\star} \cap \mathcal{E}_0] \\
&\leq \Pr[\mathcal{E}_0 \mid \overline{\mathcal{E}_0^\star}] \;\leq\; p^{1/3}, && \text{by definition of } \mathcal{E}_0^\star \\
\Pr[u_1 \text{ is a sink}] &= \Pr[\mathcal{E}_0^\star \cap \mathcal{E}_1] \\
&\leq \Pr[\mathcal{E}_0^\star \cap \mathcal{E}_1^\star] + \Pr[\mathcal{E}_1 \cap \overline{\mathcal{E}_1^\star}] \\
&\leq 2p^{1/3} + p^{1/3} \;=\; 3p^{1/3}, && \text{by (5.3) and the definition of } \mathcal{E}_1^\star.
\end{aligned}$$

The failure probability of the remaining vertices (those not incident to any edge colored i) is the same under $\mathcal{A}_{s.o.}$ and $\mathcal{A}'_{s.o.}$. □

Lemma 5.2. *Any sinkless orientation algorithm for $\mathcal{T}_\Delta$ with local error probability p has time complexity $\Omega(\Delta^{-1} \log\log p^{-1})$.*

Proof. Let $\mathcal{A}_{s.o.}$ be a t-round algorithm with error probability p, i.e., it has time profile

$(t, t, \ldots, t)$. Applying Lemma 5.1 $t(2\Delta-1)$ times we get an algorithm $\mathcal{A}'_{s.o.}$ with time profile $(0, 0, \ldots, 0)$ and error probability $p_0 = O(p^{3^{-t(2\Delta-1)}})$. We now claim that p_0 must also be at least $8^{-\Delta}$. Any 0-round orientation algorithm can be characterized by a real vector $(q_1, \ldots, q_{2\Delta-1})$, where q_i is the probability that an edge colored i is oriented as $0 \to 1$. Without loss of generality, suppose that $q_1, \ldots, q_\Delta \geq 1/2$. Fix any $v \in V(\mathcal{T}_\Delta)$ labeled 1. The probability that v is a sink is at least the probability that its edges are initially colored $\{1, \ldots, \Delta\}$ and that they are all oriented away from v, hence $p_0 \geq \binom{2\Delta-1}{\Delta}^{-1} \cdot 2^{-\Delta} \geq 2^{-3\Delta}$. Combining the upper and lower bounds on p_0 we have

$$2^{3\Delta} \geq p_0^{-1} = \Omega((p^{-1})^{3^{-t(2\Delta-1)}})$$

and taking logs twice we have

$$\log(3\Delta) \geq \log\log p^{-1} - t(2\Delta - 1)\log 3 - O(1)$$

which implies that $t = \Omega(\Delta^{-1} \log\log p^{-1})$. □

Theorem 5.2. *Even on 2-vertex colored trees or 2-vertex colored graphs of girth $\Omega(\log_\Delta n)$, sinkless orientation and $(2\Delta - 2)$-edge coloring require $\Omega(\log_\Delta \log n)$ time in* RandLOCAL *and $\Omega(\log_\Delta n)$ time in* DetLOCAL.

Proof. Consider any sinkless orientation or $(2\Delta - 2)$-edge coloring algorithm with local probability of failure p. Lemma 5.2 applies to any vertex v and any radius t such that $N^t(v)$ is consistent with a subgraph of $\mathcal{T}_\Delta$. Thus, on degree-Δ trees or graphs of girth $\Omega(\log_\Delta n)$ [29, 48], we get $\Omega(\min\{\Delta^{-1} \log\log p^{-1}, \log_\Delta n\})$ lower bounds. Following the same proof as Theorem 2.3, this implies an $\Omega(\log_\Delta n)$ lower bound in DetLOCAL, which also implies an $\Omega(\log_\Delta \log n)$ lower bound in RandLOCAL. In other words, the weak RandLOCAL lower bound $\Omega(\Delta^{-1} \log\log n)$ implied by Lemma 5.2 *automatically* implies a stronger lower bound. □

5.3 Randomized Edge Coloring Algorithm

Elkin, Pettie, and Su [55] showed that for any *constant* $\epsilon > 0$, there is a number Δ_ϵ such that for $\Delta > \Delta_\epsilon$, $\Delta(1+\epsilon)$-edge coloring can be solved in

$$O(T_{LLL}(n, \text{poly}(\Delta), \exp(-\epsilon^2\Delta/\text{poly}(\log\Delta))) + T^*(n, O(\Delta)))$$

rounds in the RandLOCAL model, where

$T_{LLL}(n, d, p)$ is the RandLOCAL complexity for constructive LLL with the parameters d and p on an n-vertex dependency graph.

$T^*(n, \Delta')$ is the RandLOCAL complexity for $5\Delta'$-edge coloring on an n-vertex graph of maximum degree Δ'.

It is unclear to what extent the algorithm of [55] (or its predecessor [52]) still works if we allow $\epsilon = o(1)$. For instance, it is unknown whether $(\Delta + \Delta^{0.7})$-edge coloring can be solved in RandLOCAL.

Challenges to Reducing the Number of Colors. The analysis of our algorithm is substantially more involved than all previous edge coloring algorithms [52, 55, 121]. Here we give a short technical review of the types of issues faced in distributed edge coloring.

Previous algorithms [52, 55] are based on the *Rödl Nibble* method. In each round, every uncolored edge nominates itself to be colored with probability $O(\epsilon)$ and remains idle otherwise; a self-nominated edge picks a free color from its available palette and *permanently* colors itself if the colors selected by adjacent edges do not conflict with it. The goal is to show that natural quantities (palette size, degree of vertices in the uncolored graph, etc.) are sharply concentrated around their expectations. The first issue is finding the right concentration bound. Chernoff bounds are insufficient for several reasons, one of which is the need for independence (or negative dependence [53, 54]) between the events of interest. Azuma's inequality and variants fall short due to the weakness of Lipschitz properties (bounded differences).[23] The algorithm of Dubhashi, Grable, and Panconesi [52] used a specialized concentration inequality of Grable [83], whereas our algorithm and that of Elkin, Pettie, and Su [55] use one [53, Thm. (8.5)] that is syntactically closer to Chernoff/Hoeffding/Azuma-type inequalities. It is restated as Theorem 5.4 in Section 5.4.

The purpose of the "self-nomination" step in [52, 55] is to simplify certain aspects of the analysis. For example, the probability that an edge is successfully colored, *conditioned on it nominating itself*, is a very high $1-O(\epsilon)$. Because of this, we can afford to toss out any color

[23] This can be seen by considering the problem of bounding the c-degree of a vertex v (the number of edges incident to v with color c in their palettes). This quantity potentially depends on the choices of $\Omega(\Delta^3)$ edges within distance 3 of v, and each such choice could affect v's c-degree by 1 or more. The sum of these Lipschitz constants completely dwarfs the expected c-degree, which makes Azuma-type inequalities inapplicable.

c from e's palette if any nominated edge e' adjacent to e selects c — regardless of whether e' successfully colors itself. This type of subtle change generally makes things simpler. Some events which would ordinarily be dependent become independent, and some variables (e.g., a vertex's c-degree) now depend on $\Theta(\Delta^2)$ variables rather than $\Theta(\Delta^3)$. The downside of this approach is that $\Omega(\epsilon^{-1})$ steps are necessary to color a large fraction of the graph, and *with each coloring step* the quantities we are monitoring (c-degree, palette size, etc.) deviate further from their expectations. When ϵ^{-1} is polynomial in Δ, the accumulated deviation errors make it impossible to achieve palette sizes as small as $\Delta + \tilde{O}(\sqrt{\Delta})$.

Our algorithm is more "natural" than [52, 55]. Roughly speaking, in each step each edge chooses a color uniformly at random from its available palette and permanently colors itself if there are no local conflicts. I.e., we dispense with the low probability self-nomination step. Let p_i be a lower bound on the palette size after i such steps, and d_i, t_i be upper bounds on uncolored degree and c-degree of any vertex, respectively. It is straightforward to show that if everything behaves precisely according to expectation, the (d_i) sequence shrinks by a $(1 - e^{-2})$ factor in each step and both $(p_i), (t_i)$ shrink by a $(1 - e^{-2})^2$ factor. In reality these quantities *do* deviate from their expectations, and even tiny, $(1 + o(1))$-factor deviations compound themselves and spin out of control. One reason our analysis is more complex than [52, 55] is that we look at concentration up to *lower order terms*. For example, although $p_i \approx t_i$, we bound $\beta_i = \frac{p_i}{t_i} - 1$, which captures accumulated errors beyond the leading constants.

As in [55], we obtain good concentration in d_i, p_i, t_i with probability $1 - \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta)$, which is $1 - 1/\mathrm{poly}(n)$ if Δ and ϵ are sufficiently large. If not, we must invoke a Lovász Local Lemma algorithm to make sure each random coloring experiment introduces bounded deviation errors in d_i, p_i, t_i. A constant fraction of the edges are colored in each step. For many parameter regimes the running time is dominated by $O(\log \epsilon^{-1})$ calls to an LLL algorithm.

In this section, we prove the following theorem, which improves upon the algorithm of [52, 55].

Theorem 5.3. *Let $\epsilon = \omega\left(\frac{\log^{2.5}\Delta}{\sqrt{\Delta}}\right)$ be a function of Δ. If $\Delta > \Delta_\epsilon$ is sufficiently large there is a* RandLOCAL *algorithm for $(1+\epsilon)\Delta$-edge coloring in time*

$$O\left(\log(1/\epsilon)\right) \cdot T_{LLL}\left(n, \mathrm{poly}(\Delta), \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta)\right) \; + \; T^*\left(n, O(\epsilon\Delta)\right).$$

Notice that $\exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta) = \exp(-\omega(\log\Delta))$, so we may use a distributed LLL algorithm under any criterion $p(ed)^\lambda < 1$. There is an inherent tradeoff between the palette size and the runtime in Theorem 5.3. Selecting smaller ϵ allows us to use fewer colors, but it leads to a higher $p = \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta)$, which may increase the runtime of the LLL algorithm.

Runtime of $5\Delta'$-edge Coloring. It is known that $T^*(n,\Delta')$ is at most $O(\log\Delta')$ plus the DetLOCAL complexity of $3\Delta'$-edge coloring on poly$(\log n)$-size graphs. This is achieved by applying the $(\tilde{\Delta}+1)$-vertex coloring algorithm of [23] to the line graph, where $\tilde{\Delta} = 2\Delta'-2$ is the maximum degree of the line graph.

For the special case of $\Delta' = \log^{1+\Omega(1)} n$, $(2\Delta'-1)$-edge coloring can be solved in RandLOCAL $O(\log^* n)$ rounds [55]. The state-of-the-art DetLOCAL algorithm [70] for $(2+x)\Delta'$-edge coloring has complexity

$$O(\log^2\Delta'\cdot x^{-1}\cdot\log\log\Delta'\cdot\log^{1.71}\log\log\Delta'\cdot\log n)$$

for any $x > 1/\log\Delta'$. Thus, combining [23, 55, 70] with $x = 1$, we have

$$T^*(n,\Delta') = O(\log^3\log n\cdot\log\log\log n\cdot\log^{1.71}\log\log\log n) = (\log\log n)^{3+o(1)}.$$

This is achieved as follows. If $\Delta' = \Omega(\log^2 n)$, we run the $O(\log^* n)$-time RandLOCAL algorithm of [55]. Otherwise, we run the RandLOCAL graph shattering phase of [23] (using the first $2\Delta'$ colors) followed by the DetLOCAL algorithm of [70] (using the remaining $3\Delta'$ colors) on each component.

Runtime on Trees. Consider running our algorithm on a tree with palette size $(1+\epsilon)\Delta$, where $\epsilon = \Omega\left(\frac{\log^{2.5+x}\Delta}{\sqrt{\Delta}}\right)$, for some positive constant x. Then the LLL parameters are $d = \text{poly}(\Delta)$ and $p = \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta)$ in Theorem 5.3, which satisfy the criterion $p(ed)^\lambda < 1$ with $\lambda = \Omega(\log^x\Delta)$. Using our randomized LLL algorithm for trees (Section 5.5), we have

$$T_{LLL}\left(n, \text{poly}(\Delta), \exp(-\epsilon^2\Delta/\log^{4+o(1)})\right) = O\left(\max\{\tfrac{\log\log n}{\log\log\log n}, \log_{\log\Delta}\log n\}\right).$$

We claim that $T^*(n,\Delta') = O(\log^*\Delta' + \log_{\Delta'}\log n)$ on trees. This is achieved as follows. First, do a $O(\log^*\Delta')$-time randomized procedure to partially color the graph using the

first $2\Delta'$ colors so that the remaining uncolored components have size $\text{poly}(\log n)$. This can be done using the algorithm of [55] without invoking any distributed LLL algorithm. Then, apply our deterministic $O(\log_{\Delta'} \tilde{n})$-time algorithm for Δ'-edge coloring trees (Section 5.7) to each uncolored component separately, using a set of Δ' fresh colors.

To sum up, the time complexity of $(1+\epsilon)\Delta$-edge coloring trees is

$$\begin{aligned} &O\left(\log(1/\epsilon) \cdot \max\{\tfrac{\log\log n}{\log\log\log n}, \log_{\log\Delta} \log n\} + \log^* \Delta + \log_\Delta \log n\right) \\ &= O\left(\log(1/\epsilon) \cdot \max\{\tfrac{\log\log n}{\log\log\log n}, \log_{\log\Delta} \log n\}\right). \end{aligned}$$

This matches our $\Omega(\log_\Delta \log n)$ lower bound (Section 5.2) when $1/\epsilon, \Delta = O(1)$.

5.3.1 The Algorithm

Our algorithm has two phases. The goal of the first phase is to color a subset of the edges using the colors from $\mathcal{C}_1 \stackrel{\text{def}}{=} \{1, \ldots, \Delta(1+\xi)\}$ such that the subgraph induced by the uncolored edges has degree less than $\Delta' = \frac{1}{5}(\epsilon - \xi)\Delta = \Theta(\epsilon\Delta)$. The first phase consists of $O(\log(1/\epsilon))$ executions of a distributed Lovász Local Lemma algorithm. The second phase colors the remaining edges using the colors from $\mathcal{C}_2 \stackrel{\text{def}}{=} \{\Delta(1+\xi)+1, \ldots, \Delta(1+\epsilon)\}$ using the fastest available coloring algorithm, which takes $T^*(n, \Delta')$ time.

Algorithm. In what follows we focus on the first phase. We write G_i to denote the graph induced by the set of uncolored edges at the beginning of the ith iteration. Each edge e in G_i has a palette $\Psi_i(e) \subseteq \mathcal{C}_1$. We write $\deg_i(v)$ to denote the number of edges incident to v in G_i and $\deg_{c,i}(v)$ to denote the number of edges incident to v that have color c in their palettes. For the base case, we set $G_1 = G$ and $\Psi_i(e) = \mathcal{C}_1$ for all edges. In the graph G_i we maintain the following invariant $\mathcal{H}_i$.

Invariant $\mathcal{H}_i$: For each edge e, vertex v, and color c, we have:

$$\begin{aligned} \deg_i(v) &\leq d_i, \\ \deg_{c,i}(v) &\leq t_i, \\ |\Psi_i(e)| &\geq p_i. \end{aligned}$$

Parameters. Given two numbers $\eta \geq 1$ and $\xi \in (0, \epsilon)$ (which are functions of Δ), we define three sequences of numbers $\{d_i\}$, $\{t_i\}$, and $\{p_i\}$ as follows.

Base case ($i = 1$):

$$d_1 \stackrel{\text{def}}{=} \Delta \qquad t_1 \stackrel{\text{def}}{=} \Delta \qquad p_1 \stackrel{\text{def}}{=} \Delta(1+\xi)$$

Inductive step ($i > 1$):

$$\begin{aligned}
&d_i \stackrel{\text{def}}{=} (1+\delta_{i-1})d^{\diamond}_{i-1} \quad && d^{\diamond}_{i-1} \stackrel{\text{def}}{=} d_{i-1} \cdot \left(1 - (1-1/p_{i-1})^{2(t_{i-1}-1)}\right) \\
&t_i \stackrel{\text{def}}{=} (1+\delta_{i-1})t^{\diamond}_{i-1} \quad && t^{\diamond}_{i-1} \stackrel{\text{def}}{=} t_{i-1} \cdot \left(1 - \frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}\right)\left(1 - (1-1/p_{i-1})^{2t_{i-1}}\right) \\
&p_i \stackrel{\text{def}}{=} (1-\delta_{i-1})p^{\diamond}_{i-1} \quad && p^{\diamond}_{i-1} \stackrel{\text{def}}{=} p_{i-1} \cdot \left(1 - \frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}\right)^2
\end{aligned}$$

Drifts (all i):

$$\delta_i \stackrel{\text{def}}{=} \frac{\beta_i}{\eta} \qquad \beta_i \stackrel{\text{def}}{=} \frac{p_i}{t_i} - 1 \qquad \text{(Notice that } \beta_1 = \xi\text{)}$$

The choice of parameters are briefly explained as follows. Consider an ideal situation where $\deg_{i-1}(v) = d_{i-1}$, $\deg_{c,i-1}(v) = t_{i-1}$, and $|\Psi_{i-1}(e)| = p_{i-1}$ for all c, e, and v. Consider a very simple experiment called OneShotColoring in which each uncolored edge attempts to color itself by selecting a color uniformly at random from its available palette. An edge e successfully colors itself with probability $(1-1/p_{i-1})^{2(t_{i-1}-1)}$, since there are $2(t_{i-1}-1)$ edges competing with e for $c \in \Psi_{i-1}(e)$, and each of these $2(t_{i-1}-1)$ edges selects c with probability $1/p_{i-1}$. Thus, by linearity of expectation, the expected degree of v after OneShotColoring is $d^{\diamond}_{i-1}$, and the parameter d_i is simply $d^{\diamond}_{i-1}$ with some slack. The parameters $\{t^{\diamond}_{i-1}, t_i, p^{\diamond}_{i-1}, p_i\}$ carry analogous meanings. The term β_i represents the *second-order* error. We need control over $\{\beta_i\}$ since it influences the growth of the three sequences $\{d_i\}$, $\{t_i\}$, and $\{p_i\}$.

For the base case, it is straightforward to see that we have $\deg_1(v) = \Delta$, $\deg_{c,1}(v) = \Delta$, and $|\Psi_1(e)| = \Delta(1+\xi)$, and thus G_1 satisfies the invariant $\mathcal{H}_1$. For the inductive step, given that $\mathcal{H}_i$ is met in G_i, we use a distributed LLL algorithm (based on OneShotColoring) to color a subset of edges in G_i so that the next graph G_{i+1} induced by the uncolored edges satisfies $\mathcal{H}_{i+1}$. We repeat this procedure until the *terminating condition* $d_i \le \frac{1}{5}(\epsilon - \xi)\Delta$ is met, and then we proceed to the second phase.

Analysis. Recall that $\epsilon = \omega(\frac{\log^{2.5}\Delta}{\sqrt{\Delta}})$. We set η to be any function of Δ that is $\omega(\log\Delta)$ such that $\epsilon \geq \frac{\eta^{2.5}}{\sqrt{\Delta}}$. We set $\xi = \frac{\epsilon}{6\eta}$. The following lemma shows that under certain criteria, the parameters $\{d_i\}$, $\{t_i\}$, $\{p_i\}$, and $\{\beta_i\}$ are very close to their "ideal" values. The proof is deferred to Section 5.3.3.

Lemma 5.3. *Consider an index $i > 1$. Suppose $\min\{d_{i-1}, t_{i-1}, p_{i-1}\} = \omega(\log\Delta)$, $\beta_{i-1} = o(1/\log\Delta)$, and $\delta_{i-1} = o(\beta_{i-1}/\log\Delta)$. Then the following four equations hold.*

$$\begin{aligned}
d_i &= d_{i-1}\cdot(1\pm o(1/\log\Delta))(1-e^{-2})\\
t_i &= t_{i-1}\cdot(1\pm o(1/\log\Delta))(1-e^{-2})^2\\
p_i &= p_{i-1}\cdot(1\pm o(1/\log\Delta))(1-e^{-2})^2\\
\beta_i &= \beta_{i-1}\cdot(1\pm o(1/\log\Delta))/(1-e^{-2})
\end{aligned}$$

Based on Lemma 5.3, we have the following lemma.

Lemma 5.4. *Let $i^\star = O(\log(1/\epsilon)) = O(\log\Delta)$ be the largest index such that $\beta_{i^\star-1} \leq 1/\eta$. Then the following four equations hold for any $1 < i \leq i^\star$.*

$$\begin{aligned}
d_i &= (1\pm o(1/\log\Delta))^{i-1}\Delta(1-e^{-2})^{i-1} = (1\pm o(1))\Delta(1-e^{-2})^{i-1}\\
t_i &= (1\pm o(1/\log\Delta))^{i-1}\Delta(1-e^{-2})^{2(i-1)} = (1\pm o(1))\Delta(1-e^{-2})^{2(i-1)}\\
p_i &= (1\pm o(1/\log\Delta))^{i-1}\Delta(1-e^{-2})^{2(i-1)} = (1\pm o(1))\Delta(1-e^{-2})^{2(i-1)}\\
\beta_i &= (1\pm o(1/\log\Delta))^{i-1}\xi/(1-e^{-2})^{i-1} = (1\pm o(1))\xi/(1-e^{-2})^{i-1}
\end{aligned}$$

Proof. To prove the lemma, it suffices to show that the condition of Lemma 5.3 is met for all indices $1 < i \leq i^\star$. We prove this by an induction on i. By the induction hypothesis the four equations hold at index $i-1$. We show that the condition of Lemma 5.3 is met for the index i, and so the four equations also hold for index i. Due to $1/\eta = o(1/\log\Delta)$, we already have $\beta_{i-1} = o(1/\log\Delta)$ and $\delta_{i-1} = o(\beta_{i-1}/\log\Delta)$. It remains to prove that

$\min\{d_{i-1}, t_{i-1}, p_{i-1}\} = \omega(\log \Delta)$.

$$
\begin{aligned}
&\min\{d_{i-1}, t_{i-1}, p_{i-1}\} \\
&\quad \geq (1 \pm o(1))\Delta(1 - e^{-2})^{2(i-1)} && \text{(Induction hypothesis for } d_{i-1}, t_{i-1}, p_{i-1}) \\
&\quad = (1 \pm o(1))\Delta(1 - e^{-2})^{2(i-2)}(1 - e^{-2})^2 \\
&\quad = (1 \pm o(1))\Delta \cdot \left(\frac{(1 - e^{-2} \pm o(1))\xi}{\beta_{i-1}}\right)^2 && \text{(Induction hypothesis for } \beta_{i-1}) \\
&\quad \geq (1 - e^{-2} \pm o(1))\xi^2\eta^2\Delta && (\beta_{i-1} \leq 1/\eta) \\
&\quad = \Omega(\eta^5) && (\xi = \Omega(\frac{\eta^{1.5}}{\sqrt{\Delta}})) \\
&\quad = \omega(\log \Delta) && \square
\end{aligned}
$$

It remains to show that (i) the number of iterations it takes to reach the terminating condition is $O(\log 1/\epsilon)$, and (ii) in each iteration, in $T_{LLL}\left(n, \text{poly}(\Delta), \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta)\right)$ time, invariant $\mathcal{H}_i$ can be maintained. By Lemma 5.4, we have:

$$
\begin{aligned}
d_{i^\star} &= (1 \pm o(1))\Delta(1 - e^{-2})^{i^\star - 1} && \text{(Lemma 5.4 for } d_{i^\star}) \\
&= (1 \pm o(1))\Delta \cdot \xi/\beta_{i^\star} && \text{(Lemma 5.4 for } \beta_{i^\star}) \\
&\leq (1 \pm o(1))\xi\eta\Delta && (\beta_{i^\star} > 1/\eta)
\end{aligned}
$$

For our choices of η and ξ, we have $d_{i^\star} \approx \xi\eta\Delta = \frac{\epsilon\Delta}{6}$. Thus, the terminating condition $d_i \leq \frac{1}{5}(\epsilon - \xi)\Delta$ must be reached before the $i^\star$-iteration (since $\frac{1}{5}(\epsilon - \xi)\Delta > \frac{\epsilon\Delta}{6}$). The number of iterations it takes to reach the terminating condition is $O(\log 1/\epsilon)$ by Lemma 5.4 for d_i.

For each $1 < i \leq i^\star$, we have:

$$
\begin{aligned}
&\delta_i^2 \cdot \min\{d_i, t_i, p_i\} \\
&\quad = \beta_i^2 t_i/\eta^2 && \text{(Definition of } \delta_i) \\
&\quad = (1 \pm o(1)) \cdot \left(\xi/(1 - e^{-2})^{i-1}\right)^2 \cdot \left(\Delta(1 - e^{-2})^{2(i-1)}\right)/\eta^2 && \text{(Lemma 5.4 for } t_i, \beta_i) \\
&\quad = (1 \pm o(1)) \cdot \Delta(\xi/\eta)^2 \\
&\quad = \Omega(\epsilon^2\Delta/\eta^4) && \text{(Definition of } \xi) \\
&\quad = \omega(\log \Delta). && \text{(Definition of } \epsilon)
\end{aligned}
$$

We will later see in Section 5.3.2 that this implies that any LLL algorithm with parameters

$d = \text{poly}(\Delta)$ and $p = \exp(-\Omega(\Delta\epsilon^2/\eta^4))$ suffices to maintain the invariant in each iteration. Notice that if we select $\eta = \log^{1+o(1)}\Delta$, then $p = \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta)$, as desired.

5.3.2 Maintenance of the Invariant

In this section we show how to apply a distributed LLL algorithm, with parameters $d = \text{poly}(\Delta)$ and $p = \exp(-\Omega(\delta_i^2 \cdot \min\{d_i, t_i, p_i\})$, to achieve the following task: given a graph G_i meeting the property $\mathcal{H}_i$, color a subset of edges of G_i so that the graph induced by the remaining uncolored edges satisfies the property $\mathcal{H}_{i+1}$. We write $\Psi(e) = \Psi_i(e)$ for notational simplicity. Consider the following modification to the underlying graph G_i:

- Each edge e discards colors from its palette to achieve uniform palette size p_i.
- Each vertex v locally simulates some imaginary subtrees attached to v and obeying $\mathcal{H}_i$ to achieve uniform color degree t_i. That is, if a color c appears in the palette of some edge incident to a vertex v, then c must appear in the palette of *exactly* t_i edges incident to v.

These (imaginary) modifications to the underlying graph are introduced to enforce broadly *uniform* progress in every part of the graph. Observe that if $\mathcal{H}_i$ applies to the imaginary graph it also applies to the true graph as well, since we are concerned with *lower* bounds on palette sizes and *upper* bounds on (c-)degrees. Our analysis largely focusses on how the following $O(1)$-round procedure affects the imaginary graph.

OneShotColoring.

(1) Each edge e selects a color $\text{Color}^\star(e) \in \Psi(e)$ uniformly at random.

(2) An edge e successfully colors itself $\text{Color}^\star(e)$ if no neighboring edge also selects $\text{Color}^\star(e)$.

We write $S(v)$ to denote the set of real edges incident to v, and we write $N_c(v)$ to denote the set of real and imaginary edges incident to v that have c in their palettes. Let $S^\diamond(v)$ (resp., $N_c^\diamond(v)$) be the subset of $S(v)$ (resp., $N_c^\diamond(v)$) that are still uncolored after OneShotColoring. Let $\Psi^\diamond(e)$ be the result of removing all colors c from $\Psi(e)$ such that some edge incident to e successfully colors itself by c.

The following concentration bound implies that $\mathcal{H}_{i+1}$ holds with high probability in the graph induced by the real uncolored edges after OneShotColoring, and thus we can apply a

distributed LLL algorithm to obtain G_{i+1} that meets the invariant $\mathcal{H}_{i+1}$. See Section 5.4 for proof.

Lemma 5.5. *Suppose that $\mathcal{H}_i$ holds. The following concentration bounds hold for any $\delta > 0$.*

$$\Pr\left[|S^\diamond(v)| > (1+\delta)d_i^\diamond\right] = \exp\left(-\Omega(\delta^2 d_i)\right)$$
$$\Pr\left[|N_c^\diamond(v)| > (1+\delta)t_i^\diamond \mid N_c^\diamond(v) \neq \emptyset\right] = \exp\left(-\Omega(\delta^2 t_i)\right)$$
$$\Pr\left[|\Psi^\diamond(e)| < (1-\delta)p_i^\diamond \mid e \textit{ remains uncolored }\right] = \exp\left(-\Omega(\delta^2 p_i)\right)$$

We write $N^k(v)$ to denote the set of all vertices within distance k of v. It is straightforward to see that (i) $S^\diamond(v)$ depends only on the colors selected by the edges whose endpoints are both in $N^2(v)$, (ii) $N_c^\diamond(v)$ depends only on the colors selected by the edges whose endpoints are both in $N^3(v)$, and (iii) $\Psi^\diamond(e)$ depends only on the colors selected by the edges whose endpoints are both in $N^2(u) \cup N^2(v)$, where $e = \{u, v\}$. Thus, the parameters for the distributed LLL are $d = \text{poly}(\Delta)$ and $p = \exp\left(-\Omega\left(\delta_i^2 \cdot \min\{d_i, t_i, p_i\}\right)\right)$, as desired.

5.3.3 Proof of Lemma 5.3

In this section, we prove Lemma 5.3. We assume $\min\{d_{i-1}, t_{i-1}, p_{i-1}\} = \omega(\log\Delta)$, $\beta_{i-1} = o(1/\log\Delta)$, and $\delta_{i-1} = o(\beta_{i-1}/\log\Delta)$. The two terms $(1-1/p_{i-1})^{2t_{i-1}}$ and $\frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}$ show up in the definition of $d_{i-1}^\diamond$, $t_{i-1}^\diamond$, and $p_{i-1}^\diamond$. We begin by showing that these two terms are both $e^{-2}(1+o(1/\log\Delta))$. We use the fact that $\frac{t_{i-1}}{p_{i-1}} = \frac{1}{\beta_{i-1}+1}$ in the following calculation.

$$\begin{aligned}
&(1-1/p_{i-1})^{2t_{i-1}} \\
&\quad= e^{-2t_{i-1}/p_{i-1}}(1-O(t_{i-1}/p_{i-1}^2)) && \text{(Taylor expansion of } e^x\text{)} \\
&\quad= e^{-2}\cdot e^{2(1-t_{i-1}/p_{i-1})}(1-O(t_{i-1}/p_{i-1}^2)) \\
&\quad= e^{-2}\cdot e^{2(1-t_{i-1}/p_{i-1})}\left(1-O\left(\frac{1}{(1+\beta_{i-1})p_{i-1}}\right)\right) && \text{(Defn. } \beta_{i-1}\text{)} \\
&\quad= e^{-2}\cdot e^{2(1-t_{i-1}/p_{i-1})}(1-o(1/\log\Delta)) && (p_{i-1} = \omega(\log\Delta)) \\
&\quad= e^{-2}\cdot e^{2\beta_{i-1}/(\beta_{i-1}+1)}(1-o(1/\log\Delta)) \\
&\quad= e^{-2}\cdot(1+O(2\beta_{i-1}/(\beta_{i-1}+1)))(1-o(1/\log\Delta))
\end{aligned}$$

$$
\begin{aligned}
&= e^{-2} \cdot (1 + o(1/\log\Delta))(1 - o(1/\log\Delta)) \\
&= e^{-2}(1 + o(1/\log\Delta)). && (*)
\end{aligned}
$$

$$
\begin{aligned}
&\frac{t_{i-1}}{p_{i-1}}(1 - 1/p_{i-1})^{2t_{i-1}} \\
&\quad = e^{-2} \cdot \frac{t_{i-1}}{p_{i-1}} \cdot (1 + o(1/\log\Delta)) && \text{by } (*) \\
&\quad = e^{-2}(1 + o(1/\log\Delta))/(1 + \beta_{i-1}) \\
&\quad = e^{-2}(1 + o(1/\log\Delta))/(1 + o(1/\log\Delta)) \\
&\quad = e^{-2}(1 \pm o(1/\log\Delta)). && (**)
\end{aligned}
$$

We are in a position to derive the first three equations in Lemma 5.3 (i.e., estimates of d_i, t_i, and p_i). Recall that $\delta_{i-1} = o(1/\log^2\Delta)$ and $1/p_{i-1} = o(1/\log\Delta)$.

$$
\begin{aligned}
d_i &= d_{i-1} \cdot (1 + \delta_{i-1}) \left(1 - (1 - 1/p_{i-1})^{2(t_{i-1}-1)}\right) \\
&= d_{i-1} \cdot (1 + o(1/\log^2\Delta)) \left(1 - e^{-2}(1 + o(1/\log\Delta))/(1 - 1/p_{i-1})^2\right) && \text{By } (*) \\
&= d_{i-1} \cdot (1 + o(1/\log^2\Delta)) \left(1 - e^{-2}(1 + o(1/\log\Delta))\right) \\
&= d_{i-1} \cdot (1 \pm o(1/\log\Delta))(1 - e^{-2}).
\end{aligned}
$$

$$
\begin{aligned}
t_i &= t_{i-1} \cdot (1 + \delta_{i-1}) \left(1 - \frac{t_{i-1}}{p_{i-1}}(1 - 1/p_{i-1})^{2t_{i-1}}\right) \left(1 - (1 - 1/p_{i-1})^{2t_{i-1}}\right) \\
&= t_{i-1} \cdot (1 + o(1/\log^2\Delta)) \left(1 - e^{-2}(1 \pm o(1/\log\Delta))\right)^2 && \text{By } (**) \\
&= t_{i-1} \cdot (1 \pm o(1/\log\Delta))(1 - e^{-2})^2.
\end{aligned}
$$

$$
\begin{aligned}
p_i &= p_{i-1} \cdot (1 - \delta_{i-1}) \left(1 - \frac{t_{i-1}}{p_{i-1}}(1 - 1/p_{i-1})^{2t_{i-1}}\right)^2 \\
&= p_{i-1} \cdot (1 - o(1/\log^2\Delta)) \left(1 - e^{-2}(1 \pm o(1/\log\Delta))\right)^2 && \text{By } (**) \\
&= p_{i-1} \cdot (1 \pm o(1/\log\Delta))(1 - e^{-2})^2.
\end{aligned}
$$

Lastly, we derive the last equation in Lemma 5.3: an estimate of the second-order error β_i.

$$\begin{aligned}
\beta_i &= \frac{p_i}{t_i} - 1 \\
&= \frac{(1-\delta_{i-1})p^\diamond_{i-1}}{(1+\delta_{i-1})t^\diamond_{i-1}} - 1 \\
&= (1 - O(\delta_{i-1})) \cdot \frac{p_{i-1}}{t_{i-1}} \cdot \frac{1 - \frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}}{1-(1-1/p_{i-1})^{2t_{i-1}}} - 1 && \text{Definition of } p^\diamond_{i-1} \text{ and } t^\diamond_{i-1} \\
&= (1 - O(\delta_{i-1})) \cdot \frac{\frac{p_{i-1}}{t_{i-1}} - (1-1/p_{i-1})^{2t_{i-1}}}{1-(1-1/p_{i-1})^{2t_{i-1}}} - 1 \\
&= \frac{\left(\frac{p_{i-1}}{t_{i-1}} - 1\right) + O(\delta_{i-1})\left(-\frac{p_{i-1}}{t_{i-1}} + (1-1/p_{i-1})^{2t_{i-1}}\right)}{1-(1-1/p_{i-1})^{2t_{i-1}}} \\
&= \frac{\left(\frac{p_{i-1}}{t_{i-1}} - 1\right) + O(\delta_{i-1})\left(-\frac{p_{i-1}}{t_{i-1}} + (1-1/p_{i-1})^{2t_{i-1}}\right)}{1-e^{-2}(1+o(1/\log\Delta))} && \text{by (*)} \\
&= \frac{\beta_{i-1} - O(\delta_{i-1})}{(1-e^{-2})(1-o(1/\log\Delta))} && -\frac{p_{i-1}}{t_{i-1}} + \left(1 - \frac{1}{p_{i-1}}\right)^{2t_{i-1}} = -\Theta(1) \\
&= \frac{\beta_{i-1}(1-o(1/\log^2\Delta))}{(1-e^{-2})(1-o(1/\log\Delta))} && \delta_{i-1} = o(1/\log^2\Delta) \\
&= \beta_{i-1} \cdot (1 \pm o(1/\log\Delta))/(1-e^{-2}).
\end{aligned}$$

5.4 Proof of Lemma 5.5

In this section we prove the concentration bounds of Lemma 5.5. For notational simplicity, we ignore all subscripts i, i.e., p, d, t are the palette size, degree, and c-degree before the ith round of coloring, all of which satisfy invariant $\mathcal{H}_i$. Recall that we introduce imaginary edges, if necessary, to ensure that the entire graph has uniform c-degree t and uniform palette size p. $S(v)$ is the set of real edges incident to v, $|S(v)| \le d$, and $N_c(v)$ the set of real and imaginary edges incident to v with c in their palettes. The arguments of this section do not differentiate between real and imaginary edges. From Lemma 5.3 we use the fact that $t = \Theta(p)$, i.e., t and p are interchangeable in those parts of the proof that are not sensitive to the leading constant.

We make extensive use of Theorem 5.4 and Lemma 5.6 to prove Lemma 5.5. The-

orem 5.4 is from Dubhashi and Panconesi's book [53] on the concentration of measure, where it is called the *method of bounded variances*. Ignoring the leading constant in the exponent, Theorem 5.4 is strictly more powerful than Chernoff-Hoeffding and Azuma-type inequalities, and is best suited in applications that have the following two features:

- We are interested in deviations of $f(\mathbf{X}_n)$ from its expectation (up to $\pm s$) that are significantly smaller than the number of underlying random variables (n) times the Lipschitz bound satisfied by the martingale (M). This feature renders Azuma's inequality too weak to be of any use.[24]
- The Lipschitz bound is pessimistic: although $D_i = \mathrm{E}[f|\mathbf{X}_i] - \mathrm{E}[f|\mathbf{X}_{i-1}]$ can be as large as M, its variance (σ_i^2) conditioned on any $\mathbf{X}_{i-1}$ is substantially smaller.

For example, in the first round of coloring, the c-degree of a vertex v depends on $\Theta(\Delta^3)$ random variables (colors chosen by edges in the 3-neighborhood) but we are interested in deviations from the expected c-degree that are $s = O(\Delta)$. Any single edge could have a significant effect on v's c-degree ($M = \Theta(1)$), but the *variances* of these effects are substantially smaller. In particular, the sum of variances $\sum_i \sigma_i^2$ will be $O(\Delta)$.

Theorem 5.4 ([53, Equation (8.5)])**.** *Let $X_1, \ldots, X_n$ be an arbitrary set of random variables. Let $f(X_1, \ldots, X_n)$ be such that $\mathrm{E}[f]$ is finite. We write $D_i \stackrel{\text{def}}{=} \mathrm{E}[f|\mathbf{X}_i] - \mathrm{E}[f|\mathbf{X}_{i-1}]$. Suppose that there exist M and values $\{\sigma_i^2\}_{1 \le i \le n}$ meeting the following conditions.*

- *For any assignment to the random variables $\mathbf{X}_{i-1}$, $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] \le \sigma_i^2$.*
- *For any assignment to the random variables $\mathbf{X}_i$, $|D_i| \le M$.*

Then $\Pr[f > \mathrm{E}[f] + s] \le \exp\left(-\frac{s^2}{2\left(\sum_{i=1}^n \sigma_i^2 + Ms/3\right)}\right)$.

Lemma 5.6 follows from straightforward calculation.

Lemma 5.6. *Let X be a random variable such that (i) $\mathrm{E}[X] = 0$, (ii) $\Pr[X = a] = \alpha$ and $\Pr[X = b] = 1 - \alpha$, and (iii) $|a - b| \le k$. Then we have the following.*

- $\mathrm{Var}[X] \le \alpha(1-\alpha)k^2 \le \alpha k^2$.
- $|b| \le \alpha k$.
- $|a| \le (1-\alpha)k \le k$.

[24] A vector $(X_1, \ldots, X_i)$ of random variables is written $\mathbf{X}_i$.

Throughout this section, we use the following notation. For each edge e and each color c, define $z_{e,c}$ as the indicator random variable that e *successfully* colors itself c, thus $z_{e,c} = 0$ if $c \notin \Psi(e)$.

5.4.1 Concentration of Vertex Degree

Let $v^\bullet$ be a vertex. We claim that $\mathrm{E}[|S^\diamond(v^\bullet)|] \leq d^\diamond$. An edge e successfully colors itself with probability $(1-1/p)^{2(t-1)}$, since there are $2(t-1)$ edges competing with e for $\mathrm{Color}^\star(e)$, and each of these $2(t-1)$ edges selects $\mathrm{Color}^\star(e)$ with probability $1/p$. Thus, by linearity of expectation,

$$\mathrm{E}[|S^\diamond(v^\bullet)|] = (1-(1-1/p)^{2(t-1)})|S(v^\bullet)| \leq (1-(1-1/p)^{2(t-1)})d = d^\diamond.$$

For brevity, we write $S \stackrel{\text{def}}{=} S(v^\bullet)$, $S^\diamond \stackrel{\text{def}}{=} S^\diamond(v^\bullet)$, and $z \stackrel{\text{def}}{=} |S| - |S^\diamond|$. The goal of this section is to show that $\Pr[z < \mathrm{E}[z] - s] = \exp\left(-\Omega(s^2/|S|)\right)$, which implies the desired concentration bound $\Pr\left[|S^\diamond(v^\bullet)| > (1+\delta)d^\diamond\right] = \exp\left(-\Omega(\delta^2 d)\right)$, by setting $s = \delta d^\diamond$.

Notations. We write $z_e \stackrel{\text{def}}{=} \sum_{c\in\Psi(e)} z_{e,c}$ and $z_c \stackrel{\text{def}}{=} \sum_{e\in S} z_{e,c}$. In other words, z_e is the indicator random variable that e successfully colors itself; z_c is the indicator random variable that some edge in S successfully colors itself by c. We can express z as $z = \sum_{e\in S} z_e$ or $z = \sum_c z_c$, where the summation is over all colors $c \in \bigcup_{e\in S}\Psi(e)$.

Let S' denote the set of edges such that $e' \in S'$ if there exists $e = \{v^\bullet, u\} \in S$ such that (i) $\Psi(e) \cap \Psi(e') \neq \emptyset$, and (ii) e' is incident to e. For each edge $e' \in S'$ and for each color $c \in \Psi(e')$, we define $R(e', c)$ as the subset of S such that $e \in R(e', c)$ if (i) e is incident to e', and (ii) $c \in \Psi(e)$. We write $w(e', c) = |R(e', c)|$ and $w(e') = \sum_{c\in\Psi(e')} w(e', c)$. Notice that the value $w(e', c)$ may exceed 2 when $e' \notin S$ is an imaginary edge incident to $v^\bullet$. Intuitively, $w(e')$ measures the influence of $\mathrm{Color}^\star(e')$ on z. Notice that $\sum_{e'\in S'} w(e') \leq 2|S|pt$.

We consider the sequence of random variables $(X_1, \ldots, X_{|S|+|S'|})$, where the initial $|S'|$ variables are the colors selected by the edges in S', in arbitrary order, and the remaining $|S|$ variables are the colors selected by the edges in S, in arbitrary order. We let $z = f(X_1, \ldots, X_{|S|+|S'|})$ in Theorem 5.4. To prove the desired concentration bound, it suffices to show that we can set $M = O(1)$ and σ_i^2 to achieve $\sum_{i=1}^{|S|+|S'|} \sigma_i^2 = O(|S|)$. In what follows, we analyze the effect of exposing the value of the random variable X_i, given that all variables in $\mathbf{X}_{i-1}$ have been fixed.

Exposing an Edge in S'. Consider the case where $X_i = \text{Color}^\star(e^\star)$ is the color selected by the edge $e^\star \in S'$. Recall $D_i = \mathrm{E}[z|\mathbf{X}_i] - \mathrm{E}[z|\mathbf{X}_{i-1}]$. Our goal is to show that $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(w(e)/(pt))$ and $|D_i| = O(1)$. Hence we set $\sigma_i^2 = O(w(e)/(pt))$, which implies $\sum_{1 \le i \le |S'|} \sigma_i^2 = O(|S|)$, as desired.

By linearity of expectation, $D_i = \sum_c (\mathrm{E}[z_c|\mathbf{X}_i] - \mathrm{E}[z_c|\mathbf{X}_{i-1}])$, where the summation ranges over all colors c that appear in $\bigcup_{e \in S} \Psi(e)$. We write $D_{i,c} = \mathrm{E}[z_c|\mathbf{X}_i] - \mathrm{E}[z_c|\mathbf{X}_{i-1}]$, and make the following observations:

- $D_{i,c} \neq 0$ only if $c \in \Psi(e^\star)$. For each $c \in \Psi(e^\star)$, $D_{i,c}$ depends only on whether $e^\star$ selects the color c, which occurs with probability $1/p$. In particular, $D_{i,c} < 0$ only if $e^\star$ selects c, and $D_{i,c} > 0$ only if $e^\star$ does not select c. Thus, $\mathrm{Cov}[D_{i,c}, D_{i,c'}|\mathbf{X}_{i-1}] \le 0$ for all color pairs $\{c, c'\}$.
- For each $e \in S$, both $\mathrm{E}[z_{e,c}|\mathbf{X}_i]$ and $\mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]$ are within $[0, 1/p]$, since $z_{e,c} = 1$ only if $c \in \Psi(e)$ and e selects c, which occurs with probability $1/p$. Thus, $\max_{X_i} D_{i,c} - \min_{X_i} D_{i,c} \le w(e^\star, c)/p$.

By Lemma 5.6 (with $k \le w(e^\star, c)/p$ and $\alpha = 1/p$), we have $\mathrm{Var}[D_{i,c}|\mathbf{X}_{i-1}] \le (1/p)(w(e^\star, c)/p)^2$. We bound the variance $\mathrm{Var}[D_i|\mathbf{X}_{i-1}]$ as follows.

$$\begin{aligned}
\mathrm{Var}[D_i|\mathbf{X}_{i-1}] &= \sum_c \mathrm{Var}[D_{i,c}|\mathbf{X}_{i-1}] + \sum_{c,c'} \mathrm{Cov}[D_{i,c}, D_{i,c'}|\mathbf{X}_{i-1}] \\
&= \sum_c O((w(e^\star, c)/p)^2/p) && \mathrm{Cov}[D_{i,c}, D_{i,c'}|\mathbf{X}_{i-1}] \le 0 \\
&= \sum_c O(w(e^\star, c)/p^2) && w(e^\star, c) < t = \Theta(p) \\
&= O(w(e^\star)/p^2) \\
&= O(w(e^\star)/(pt)).
\end{aligned}$$

We bound $|D_i|$ as follows. Consider $c \in \Psi(e^\star)$. Recall that we already have the bound $|D_{i,c}| \le w(e^\star, c)/p \le (t-1)/p$. If c is not selected by $e^\star$, which occurs with probability $1 - 1/p$, we have a tighter bound $|D_{i,c}| \le w(e^\star, c)/p^2 \le (t-1)/p^2$ by Lemma 5.6 with $k \le w(e^\star, c)/p$ and $\alpha = 1/p$. Therefore,

$$|D_i| \le \sum_c |D_{i,c}| \le 1 \cdot \frac{t-1}{p} + (p-1) \cdot \frac{t-1}{p^2} = O(1).$$

Exposing an Edge in S. Consider the case where $X_i = \mathrm{Color}^\star(e^\star)$ is the color selected by the edge $e^\star \in S$. Suppose that $X_i = c^\star$. Recall $D_i = \sum_c D_{i,c}$. It is straightforward to see that (i) $|D_{i,c}| \leq 1$ if $c = c^\star$, (ii) $|D_{i,c}| \leq 1/p$ if $c \in \Psi(e^\star) - \{c^\star\}$, and (iii) $|D_{i,c}| = 0$ otherwise. Thus, $|D_i| = O(1)$, and $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(1)$. We set $\sigma_i^2 = O(1)$, and so $\sum_{|S'|<i\leq|S|+|S'|} \sigma_i^2 = O(|S|)$.

5.4.2 Concentration of Palette Size

Let $e^\bullet = \{u, v\}$ be an edge, and let $c^\bullet = \mathrm{Color}^\star(e^\bullet)$ be the color selected by $e^\bullet$. We do not consider $c^\bullet$ as a random variable in the analysis (i.e., we expose the color selected by $e^\bullet$ first). Let $\mathcal{E}$ be the event that $e^\bullet$ does not successfully color itself. Since $e^\bullet$ remains uncolored with at least a constant probability, we are allowed to ignore the condition "$e^\bullet$ remains uncolored" in Lemma 5.5 in the subsequent calculation. To prove the desired concentration bound regarding palette size $\Pr\left[|\Psi^\diamond(e)| < (1-\delta)p^\diamond \mid e \text{ remains uncolored }\right] = \exp\left(-\Omega(\delta^2 p)\right)$, it suffices to show that (i) $|\operatorname{E}[|\Psi^\diamond(e^\bullet)|] - p^\diamond| = O(1)$, and (ii) $\Pr[|\Psi^\diamond(e^\bullet)| < (1-\delta)\operatorname{E}[|\Psi^\diamond(e^\bullet)|]] = \exp(-\Omega(\delta^2 \operatorname{E}[|\Psi^\diamond(e^\bullet)|]))$.

Notations. We write S_u (resp., S_v) to denote the set of edges e incident to $e^\bullet$ on u (resp., v) such that $\Psi(e) \cap \Psi(e^\bullet) - \{c^\bullet\} \neq \emptyset$. We write S' to denote the set of edges such that $e' \in S'$ if there exists $e \in S_u \cup S_v$ meeting the following conditions: (i) e' is incident to e, (ii) $e' \notin S_u \cup S_v \cup \{e^\bullet\}$, and (iii) $\Psi(e) \cap \Psi(e') \cap \Psi(e^\bullet) - \{c^\bullet\} \neq \emptyset$. Notice that $\Psi^\diamond(e^\bullet)$ is determined by the colors selected by the edges in $S_u \cup S_v \cup S'$. We have $|S_u| \leq (p-1)(t-1) < pt$, $|S_v| \leq (p-1)(t-1) < pt$, and $|S'| \leq 2(p-1)(t-1)^2 < 2pt^2$.

Expected Value. In what follows, consider a color $c \in \Psi(e^\bullet) - \{c^\bullet\}$.

- Let $e \in S_u \cup S_v$ such that $c \in \Psi(e)$. We have $\operatorname{E}[z_{e,c}] = \frac{1}{p}(1-\frac{1}{p})^{2t-3}$. Notice that $e^\bullet$ selects $c^\bullet \neq c$, so there are $2t-3$ (rather than $2t-2$) edges competing with e for the color c.
- Let $e' = \{u, x\} \in S_u$ and $e'' = \{v, y\} \in S_v$ such that $c \in \Psi(e') \cap \Psi(e'')$. We define $z_{e',e'',c} \stackrel{\text{def}}{=} z_{e',c} \cdot z_{e'',c}$. If $x = y$, then $z_{e',e'',c} = 0$. Otherwise, $x \neq y$ and $\operatorname{E}[z_{e',e'',c}] = \frac{1}{p^2}(1-\frac{1}{p})^{4t-6-b(e',e'')}$, where $b(e', e'') \leq 3$ is the number of edges e such that (i) $e \neq e^\bullet$, and (ii) e is incident to both e' and e''.

Let z_c be the indicator random variable that some edge incident to $e^\bullet$ successfully colors itself by c, that is,

$$z_c \stackrel{\text{def}}{=} \sum_{e \,:\, e \in S_u \cup S_v,\ c \in \Psi(e)} z_{e,c} - \sum_{e', e'' \,:\, e' \in S_u,\ e'' \in S_v,\ c \in \Psi(e') \cap \Psi(e'')} z_{e', e'', c}.$$

The number of edges $e \in S_u \cup S_v$ such that $c \in \Psi(e)$ is exactly $2t - 2$. The number of pairs $(e' = \{u, x\} \in S_u, e'' = \{v, y\} \in S_v)$ such that $c \in \Psi(e') \cap \Psi(e'')$ and $x \neq y$ is at least $(t-1)^2 - (t-1)$ and at most $(t-1)^2$. By linearity of expectation (recall $t = \Theta(p)$),

$$\mathrm{E}[z_c] = \frac{2t}{p}(1 - 1/p)^{2t} - \frac{t^2}{p^2}(1 - 1/p)^{4t} \pm O(1/p).$$

Define $z \stackrel{\text{def}}{=} \sum_{c \in \Psi(e^\bullet) - \{c^\bullet\}} z_c$. Then, we have:

$$\begin{aligned}
\mathrm{E}[|\Psi^\diamond(e^\bullet)|] &= |\Psi(e^\bullet)| - \mathrm{E}[z] && |\Psi^\diamond(e^\bullet)| = |\Psi(e^\bullet)| - z \\
&= p \cdot \left(1 - \frac{2t}{p}(1 - 1/p)^{2t} + \frac{t^2}{p^2}(1 - 1/p)^{4t} \pm O(1/p)\right) \\
&= p \cdot \left(1 - \frac{2t}{p}(1 - 1/p)^{2t} + \frac{t^2}{p^2}(1 - 1/p)^{4t}\right) \pm O(1) \\
&= p^\diamond \pm O(1). && \text{Definition of } p^\diamond
\end{aligned}$$

Hence $|\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|] - p^\diamond| = O(1)$.

Concentration Bound. Consider the sequence of random variables $(X_1, \ldots, X_{|S_u|+|S_v|+|S'|})$, where the initial $|S'|$ variables are the colors selected by the edges in S', in arbitrary order, and the remaining $|S_u| + |S_v|$ variables are the colors selected by the edges in $S_u \cup S_v$, in arbitrary order. Let $z = f(X_1, \ldots, X_{|S_u|+|S_v|+|S'|})$ in Theorem 5.4. To prove the desired concentration bound $\Pr[|\Psi^\diamond(e^\bullet)| < (1-\delta)\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|]] = \exp(-\Omega(\delta^2\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|]))$, it suffices to show that $\Pr[z > \mathrm{E}[z] + s] = \exp(-\Omega(s^2/p))$, by setting $s = \delta\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|]$, and recall that $\mathrm{E}[|\Psi^\diamond(e^\bullet)|] = p^\diamond \pm O(1) = \Theta(p)$. In view of Theorem 5.4, we only need to show that we can set $M = O(1)$ and σ_i^2 such that $\sum_{i=1}^{|S_u|+|S_v|+|S'|} \sigma_i^2 = O(p)$.

Exposing an Edge in S'. Consider the case where $X_i = \mathrm{Color}^\star(e^\star)$ is the color selected by the edge $e^\star \in S'$. Our goal is to show that $|D_i| = O(1/t)$. This implies $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] =$

$O(1/t^2)$, and so we may set $\sigma_i^2 = O(1/t^2)$. Since $|S'| = O(pt^2)$, we have $\sum_{i=1}^{|S'|} \sigma_i^2 = O(p)$.

Let R denote the set of edges in $S_u \cup S_v$ that are incident to $e^\star$. Notice that $1 \leq |R| \leq 2$. We define:

$$z_c^{(i)} \stackrel{\text{def}}{=} \sum_{e' \,:\, e' \in R,\ c \in \Psi(e')} z_{e',c} - \sum_{e', e'' \,:\, e' \in S_u,\ e'' \in S_v,\ c \in \Psi(e') \cap \Psi(e''),\ \{e, e''\} \cap R \neq \emptyset} z_{e',e'',c}.$$

Intuitively, $z_c^{(i)}$ is the result of subtracting all terms from the definition of z_c not involving edges in R. We now argue that $\mathrm{E}[z_c|\mathbf{X}_i] - \mathrm{E}[z_c|\mathbf{X}_{i-1}] = \mathrm{E}[z_c^{(i)}|\mathbf{X}_i] - \mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]$. This is due to the two observations: (i) If $e \notin R$, then $\mathrm{E}[z_{e,c}|\mathbf{X}_i] = \mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]$. (ii) If $\{e', e''\} \cap R = \emptyset$, then $\mathrm{E}[z_{e',e'',c}|\mathbf{X}_i] = \mathrm{E}[z_{e',e'',c}|\mathbf{X}_{i-1}]$.

Consider a color $c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}$. The probability that some edge in R selects c is at most $|R|/p \leq 2/p$. Thus, the conditional expectations $\mathrm{E}[z_c^{(i)}|\mathbf{X}_i]$ and $\mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]$ must be within $[0, 2/p]$, and so $|\,\mathrm{E}[z_c^{(i)}|\mathbf{X}_i] - \mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]| \leq 2/p$. For the case of $c \neq X_i$, which occurs with probability $1 - 1/p$, we have a tighter bound $|\,\mathrm{E}[z_c^{(i)}|\mathbf{X}_i] - \mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]| \leq 2/p^2$ by Lemma 5.6 with $k \leq 2/p$ and $\alpha = 1/p$. We bound $|D_i|$ as follows.

$$\begin{aligned}
|D_i| &\leq \sum_{c \in \Psi(e^\bullet) - \{c^\bullet\}} |\,\mathrm{E}[z_c|\mathbf{X}_i] - \mathrm{E}[z_c|\mathbf{X}_{i-1}]| \\
&= \sum_{c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}} |\,\mathrm{E}[z_c^{(i)}|\mathbf{X}_i] - \mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]| \\
&\leq (2/p) + (2/p^2)(|\Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}| - 1) \\
&= O(1/p) = O(1/t).
\end{aligned}$$

Exposing an Edge in $S_u \cup S_v$. Consider the case where $X_i = \mathrm{Color}^\star(e^\star)$ is the color selected by the edge $e^\star \in S_u \cup S_v$. We define $w(e^\star) \stackrel{\text{def}}{=} |\Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}|$. The goal is to show that (i) $|D_i| = O(1)$ and (ii) $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(w(e^\star)/p)$. By setting $\sigma_i^2 = O(w(e^\star)/p)$, we achieve

$$\sum_{i=|S'|+1}^{|S'|+|S_u|+|S_v|} \sigma_i^2 = \sum_{e \in S_u \cup S_v} O(w(e)/p) = O(pt/p) = O(t) = O(p).$$

By the linearity of expectation, $D_i = \sum_{c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}} D_{i,c}$, where $D_{i,c} = \mathrm{E}[z_c|\mathbf{X}_i] - \mathrm{E}[z_c|\mathbf{X}_{i-1}]$. Since both $\mathrm{E}[z_c|\mathbf{X}_i]$ and $\mathrm{E}[z_c|\mathbf{X}_{i-1}]$ are within $[0, 1]$, we have $|D_{i,c}| \leq 1$. We have a tighter bound $|D_{i,c}| \leq 1/p$ in the event that $\mathrm{Color}^\star(e^\star) \neq c$ (by Lemma 5.6 with

$k \leq 1$ and $\alpha = 1/p$). Thus, $|D_i| \leq 1 + (w(e^\star) - 1)/p = O(1)$.

In order to prove that $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(w(e^\star)/p)$, we need the following two observations.

- Consider a color $c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}$. Recall that $|D_{i,c}| \leq 1/p$ for the case c is not selected by $e^\star$, which occurs with probability $1 - 1/p$. Thus, $\mathrm{E}[D_{i,c} \cdot D_{i,c}|\mathbf{X}_{i-1}] \leq (1/p) \cdot 1 + (1 - 1/p) \cdot 1/p^2 = O(1/p)$.
- Consider two distinct colors c and c' in $\Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}$. If $e^\star$ selects c or c' (which occurs with probability $2/p$), $D_{i,c} \cdot D_{i,c'} \leq 1 \cdot (1/p)$. Otherwise $D_{i,c} \cdot D_{i,c'} \leq (1/p) \cdot (1/p)$. Therefore, $\mathrm{E}[D_{i,c} \cdot D_{i,c'}|\mathbf{X}_{i-1}] \leq (2/p) \cdot 1/p + (1 - 2/p) \cdot 1/p^2 = O(1/p^2)$.

We now bound $\mathrm{Var}[D_i|\mathbf{X}_{i-1}]$ as follows.

$$\begin{aligned}
\mathrm{Var}[D_i|\mathbf{X}_{i-1}] &\leq \sum_{c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}} \; \sum_{c' \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}} \mathrm{E}[D_{i,c} \cdot D_{i,c'}|\mathbf{X}_{i-1}] \\
&\leq w(e^\star) \cdot O(1/p) + w(e^\star)(w(e^\star) - 1) \cdot O(1/p^2) \\
&= O(w(e^\star)/p).
\end{aligned}$$

5.4.3 Concentration of Color Degree

For the remainder of this section, fix a vertex $v^\bullet$ and a color $c^\bullet$ in the palette $\Psi(e)$ for some e incident to $v^\bullet$. For convenience, we write $R \stackrel{\text{def}}{=} N_{c^\bullet}(v^\bullet)$. Define $R^\diamond$ as the subset of R such that $e = \{v^\bullet, u\} \in R^\diamond$ if (i) e is not successfully colored by a color in $\Psi(e) - \{c^\bullet\}$, and (ii) no edge incident to e on u successfully colors itself $c^\bullet$. We write $z \stackrel{\text{def}}{=} |R \setminus R^\diamond|$. Let $\mathcal{E}'$ be the event that $N^\diamond_{c^\bullet}(v^\bullet) \neq \emptyset$. Observe that if $\mathcal{E}'$ occurs, then no edge incident to $v^\bullet$ successfully colors itself $c^\bullet$. Thus, conditioning on $\mathcal{E}'$ happening, $R \setminus R^\diamond$ equals $N^\diamond_{c^\bullet}(v^\bullet)$.

Our goal is to show that (i) $\Pr[z < \mathrm{E}[z] - s] = \exp(-\Omega(s^2/t))$, and (ii) $\mathrm{E}[|R^\diamond|] = |R| - \mathrm{E}[z] = t^\diamond \pm O(1)$. Since $\mathcal{E}'$ occurs with constant probability, the above (i) and (ii) together imply the desired concentration bound $\Pr[|N^\diamond_{c^\bullet}(v^\bullet)| > (1 + \delta)t^\diamond \mid \mathcal{E}'] = \exp(-\Omega(\delta^2 t))$, by setting $s = \delta t^\diamond \pm O(1)$. Recall that $t^\diamond = \Theta(t)$.

Expected Value. With respect to an edge $e = \{v^\bullet, u\} \in R$, we define the following notations based on parts (i) and (ii) of the definition of $R^\diamond$.

- Define z_e^a as the indicator random variable that some edge incident to e on u successfully colors itself $c^\bullet$. We have $\mathrm{E}[z_e^a] = (t - 1) \cdot \frac{1}{p}(1 - \frac{1}{p})^{2t-2} = \frac{t}{p}(1 - \frac{1}{p})^{2t} \pm O(1/p)$.

- Define z_e^b as the indicator random variable that e is successfully colored by a color in $\Psi(e) - \{c^\bullet\}$. We have $\mathrm{E}[z_e^b] = (p-1) \cdot \frac{1}{p}(1-\frac{1}{p})^{2t-2} = (1-\frac{1}{p})^{2t} \pm O(1/p)$.

Let $z_e^{a,b} \stackrel{\text{def}}{=} z_e^a \cdot z_e^b$. Notice that z_e^a and z_e^b are nearly independent but not independent. Let $z_e \stackrel{\text{def}}{=} z_e^a + z_e^b - z_e^{a,b}$, and so we have $z = |R \setminus R^\diamond| = \sum_{e \in R} z_e$. We calculate $\mathrm{E}[z_e^{a,b}]$ as follows. Let e' be any edge incident to e such that $c^\bullet \in \Psi(e')$, and let c be any color in $\Psi(e) - \{c^\bullet\}$. With respect to (e, e', c), we define the following two sets:

- S_a is the set of all edges e'' such that (i) $e'' \neq e, e'$, (ii) e'' is incident to e', and (iii) $c^\bullet \in \Psi(e'')$. Intuitively, S_a is the set of all edges other than e that contend with e' for the color $c^\bullet$. Notice that $|S_a| = 2t-3$, since $\Psi(e)$ must contain $c^\bullet$.
- S_b is the set of all edges e'' such that $e'' \in S_b$ if (i) $e'' \neq e, e'$, (ii) e'' is incident to e, and (iii) $c \in \Psi(e'')$. Intuitively, S_b is the set of all edges other than e' that contend with e for the color c. Notice that $2t-3 \leq |S_b| \leq 2t-2$, since $\Psi(e')$ may or may not contain c. The extent to which S_a and S_b intersect is unknown.

Fixing the edge e incident to $v^\bullet$, let $x(c, e')$ denote the probability that (i) e' successfully colors itself $c^\bullet$ and (ii) e successfully colors itself c. In view of the definition of S_a and S_b, we have:

$$\begin{aligned}
x(c,e') &= \frac{1}{p^2} \prod_{e'' \in S_a \setminus S_b} (1-1/p) \prod_{e'' \in S_b \setminus S_a} (1-1/p) \prod_{e'' \in S_a \cap S_b} (1-2/p) \\
&= \frac{1}{p^2}(1-1/p)^{|S_a \setminus S_b|}(1-1/p)^{|S_b \setminus S_a|}(1-2/p)^{|S_a \cap S_b|} \\
&= \frac{1}{p^2}(1-1/p)^{|S_a \setminus S_b|}(1-1/p)^{|S_b \setminus S_a|}(1-1/p)^{2|S_a \cap S_b|}\left(1 - O\left(\frac{|S_a \cap S_b|}{p^2}\right)\right) \\
&= \frac{1}{p^2}(1-1/p)^{|S_a|+|S_b|}(1-O(1/p)) \qquad \text{(Notice that } |S_a \cap S_b| < t = \Theta(p).\text{)} \\
&= \frac{1}{p^2}(1-1/p)^{4t-O(1)}(1-O(1/p)) \\
&= \frac{1}{p^2}(1-1/p)^{4t} \pm O(1/p^3).
\end{aligned}$$

We now calculate $\mathrm{E}[z_e^{a,b}]$ and show that $\mathrm{E}[|R^\diamond|] = |R| - \mathrm{E}[z] = t^\diamond \pm O(1)$.

$$\begin{aligned}
\mathrm{E}[z_e^{a,b}] &= \sum_{\substack{(c,e') \,:\, e' \text{ incident to } e,\\ c^\bullet \in \Psi(e'),\, c \in \Psi(e) - \{c^\bullet\}}} x(c,e') \qquad \text{(union of disj. events)} \\
&= (t-1)(p-1) \cdot \left(\frac{1}{p^2}(1-1/p)^{4t} \pm O(1/p^3) \right) \\
&= \frac{t}{p}(1-1/p)^{4t} \pm O(1/p).
\end{aligned}$$

$$\begin{aligned}
\mathrm{E}[|R^\diamond|] &= |R| - \mathrm{E}[z] \\
&= t - \sum_{e \in R} \left(\mathrm{E}[z_e^a] + \mathrm{E}[z_e^b] - \mathrm{E}[z_e^{a,b}] \right) \\
&= t \cdot \left(1 - \frac{t}{p}(1-1/p)^{2t} - (1-1/p)^{2t} + \frac{t}{p}(1-1/p)^{4t} \pm O(1/p) \right) \\
&= t \cdot \left(1 - \frac{t}{p}(1-1/p)^{2t} - (1-1/p)^{2t} + \frac{t}{p}(1-1/p)^{4t} \right) \pm O(1) \\
&= t^\diamond \pm O(1). \qquad \text{(Definition of } t^\diamond\text{)}
\end{aligned}$$

Concentration Bound. We have established that $|R^\diamond|$ has the correct expectation and now need to prove that it has sufficiently good concentration around that expectation. The analysis here becomes more complicated because we have to consider the colors selected in some 3-neighborhood. The *palette size* and *degree* analyses focussed only on 2-neighborhoods.

Based on the definition of z_e^a and z_e^b, we define the following sets.

- Recall that $R = N_{c^\bullet}(v^\bullet)$. Let R_1 be the set of all edges e such that (i) $e \notin R$, (ii) $c^\bullet \in \Psi(e)$, and (iii) e is incident to some edge in R. Similarly, let R_2 be the set of all edges e such that (i) $e \notin R \cup R_1$, (ii) $c^\bullet \in \Psi(e)$, and (iii) e is incident to some edge in R_1. Notice that the value z_e^a, for any $e \in R$, is determined by the information about which edges in $R \cup R_1 \cup R_2$ select $c^\bullet$. We write $\alpha = |R \cup R_1 \cup R_2|$.
- Let R' be the set of all edges e' such that (i) $e' \notin R$ and (ii) there exists $e \in R$ such that $\Psi(e) \cap \Psi(e') - \{c^\bullet\} \neq \emptyset$. Notice that the the value z_e^b, for any $e \in R$, is determined by the colors selected by the edges in $R \cup R'$. We write $\beta = |R \cup R'|$.

For each $e \in R$, z_e^a is simply the summation of $z_{e',c^\bullet}$ over all edges $e' \in R_1$ incident to

e. For each $e'' \in R_2$, we write $w(e'')$ to denote the number of edges in R_1 incident to e''. Intuitively, $w(e'')$ measures the influence of $\text{Color}^\star(e'')$ on $\sum_{e \in R} z_e^a$.

We consider the sequence of random variables $(X_1, \ldots, X_{\alpha+\beta})$, where the initial α random variables reveal which edges in $R \cup R_1 \cup R_2$ select the color $c^\bullet$ according to the ordering R_2, R_1, R, and the remaining β random variables reveal the colors selected by the edges in $R \cup R'$ according to the ordering R', R. We let $z = f(X_1, \ldots, X_{\alpha+\beta})$ in Theorem 5.4. To prove the desired concentration bound $\Pr[z < \mathrm{E}[z] - s] = \exp(-\Omega(s^2/t))$, it suffices to show that we can set $M = O(1)$ and σ_i^2 such that $\sum_{i=1}^{\alpha+\beta} \sigma_i^2 = O(t)$. In what follows, we analyze the effect of exposing the value of X_i, given that all variables in $\mathbf{X}_{i-1}$ have been fixed.

Revealing whether $c^\bullet$ is Selected by an Edge in $R \cup R_1 \cup R_2$. Consider the case where X_i reveals whether $c^\bullet$ is selected by the edge $e^\star \in R \cup R_1 \cup R_2$. Notice that X_i is binary, and recall that $D_i = \mathrm{E}[z|\mathbf{X}_i] - \mathrm{E}[z|\mathbf{X}_{i-1}]$. There are at most two distinct outcomes of $D_i|\mathbf{X}_{i-1}$, in which one occurs with probability $1/p$. Thus, by Lemma 5.6 we have:

$$\mathrm{Var}[D_i|\mathbf{X}_{i-1}] \leq \left(\max_{X_i} D_i|\mathbf{X}_{i-1} - \min_{X_i} D_i|\mathbf{X}_{i-1}\right)^2 /p = O(\max_{\mathbf{X}_i} |D_i|^2/p).$$

Thus, to achieve $\sum_{i=1}^{\alpha} \sigma_i^2 = O(t)$ and $M = O(1)$ it suffices to show the following.

- For the case $e^\star \in R_2$, we must prove $|D_i| = O(w(e^\star)/p)$.[25] Since $w(e^\star) < t = \Theta(p)$, $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O((w(e^\star)/p)^2/p) = O(w(e^\star)/p^2)$, so we can set $\sigma_i^2 = O(w(e^\star)/p^2)$.
- For the case $e^\star \in R \cup R_1$, we must prove $|D_i| = O(1)$. Hence we may set $\sigma_i^2 = \mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(1/p)$.

Notice that $\sum_{e^\star \in R_2} w(e^\star) < t^3$, $|R_1| < t^2$, and $|R| = t$. Thus, $\sum_{i=1}^{\alpha} \sigma_i^2 = O(t)$. With respect to the edge $e^\star \in R \cup R_1 \cup R_2$, we make the following definitions.

$$Y^a \stackrel{\text{def}}{=} \{e' \in R_1 \ : \ e' = e^\star \text{ or } e' \text{ is incident to } e^\star\} \quad D_i^a \stackrel{\text{def}}{=} \sum_{e' \in Y^a} (\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] + \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}])$$

$$Y^b \stackrel{\text{def}}{=} \{e \in R \ : \ e = e^\star \text{ or } e \text{ is incident to } e^\star\} \qquad D_i^b \stackrel{\text{def}}{=} \sum_{e \in Y^b} |\mathrm{E}[z_e^b|\mathbf{X}_i] - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}]|$$

[25] Intuitively, if $e^\star$ chooses color $c^\bullet$, it prevents $w(e^\star)$ edges in R_1 from successfully coloring themselves $c^\bullet$, but the prior probability of these edges coloring themselves $c^\bullet$ was only $O(1/p)$, hence the total influence on the expectation of z should be $O(w(e^\star)/p)$.

Intuitively, Y^a and Y^b are the subsets of R_1 and R that are "relevant" to D_i in the following sense:

$$\begin{aligned} \mathrm{E}[z_{e'',c^\bullet}|\mathbf{X}_i] &= \mathrm{E}[z_{e'',c^\bullet}|\mathbf{X}_{i-1}] && \text{for all } e'' \in R_1 \setminus Y^a, \\ \mathrm{E}[z^b_{e'}|\mathbf{X}_i] &= \mathrm{E}[z^b_{e'}|\mathbf{X}_{i-1}] && \text{for all } e' \in R \setminus Y^b. \end{aligned}$$

Our plan of bounding $|D_i|$ is as follows. First we show that $|D_i| \le 4D^a_i + D^b_i$ in Claim 1, and then we bound D^a_i and D^b_i separately in Claims 2 and 3. The three claims together establish a desired bound on $|D_i|$.

Claim 1. $|D_i| \le 4D^a_i + D^b_i$.

Proof. We define the following notations.

$$\begin{aligned} P_1 &\stackrel{\text{def}}{=} \{(e,e') \ : e \in R \setminus Y^b, e' \in Y^a, \ e \text{ is incident to } e'\} \\ P_2 &\stackrel{\text{def}}{=} \{(e,e') \ : e \in Y^b, e' \in R_1 \setminus Y^a, \ e \text{ is incident to } e'\} \\ P_3 &\stackrel{\text{def}}{=} \{(e,e') \ : e \in Y^b, e' \in Y^a, \ e \text{ is incident to } e'\} \end{aligned}$$

$$\begin{aligned} Q_j &\stackrel{\text{def}}{=} -\textstyle\sum_{(e,e')\in P_j} \big(\mathrm{E}[z_{e',c^\bullet} \cdot z^b_e|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} \cdot z^b_e|\mathbf{X}_{i-1}]\big) && (\text{for each } j = 1,2,3) \\ F_j &\stackrel{\text{def}}{=} \textstyle\sum_{e\in R} \big(\mathrm{E}[z^j_e|\mathbf{X}_i] - \mathrm{E}[z^j_e|\mathbf{X}_{i-1}]\big) && (\text{for each } j = a,b) \end{aligned}$$

The definitions of P_1, P_2, and P_3 depend on Y^a and Y^b, which depend on the edge $e^\star$. For instance, if $e^\star \in R$, then $Y^b = R$, which implies that $P_1 = \emptyset$. Recall that the edge $e^\star$ can be any edge in $R \cup R_1 \cup R_2$, and the proof of this claim applies to all choices of $e^\star \in R \cup R_1 \cup R_2$.

Notice that for any pair $(e \in R, e' \in R_1)$ such that e is incident to e' but $(e,e') \notin P_1 \cup P_2 \cup P_3$, we must have $\mathrm{E}[z_{e',c^\bullet} \cdot z^b_e|\mathbf{X}_i] = \mathrm{E}[z_{e',c^\bullet} \cdot z^b_e|\mathbf{X}_{i-1}]$ due to the definition of Y^a and Y^b. We rewrite the term D_i as follows.

$$\begin{aligned} D_i &= \mathrm{E}[z|\mathbf{X}_i] - \mathrm{E}[z|\mathbf{X}_{i-1}] \\ &= \sum_{e\in R} \big(\mathrm{E}[z_e|\mathbf{X}_i] - \mathrm{E}[z_e|\mathbf{X}_{i-1}]\big) \\ &= \sum_{e\in R} \Big(\big(\mathrm{E}[z^a_e|\mathbf{X}_i] - \mathrm{E}[z^a_e|\mathbf{X}_{i-1}]\big) + \big(\mathrm{E}[z^b_e|\mathbf{X}_i] - \mathrm{E}[z^b_e|\mathbf{X}_{i-1}]\big) \\ &\qquad - \big(\mathrm{E}[z^a_e \cdot z^b_e|\mathbf{X}_i] - \mathrm{E}[z^a_e \cdot z^b_e|\mathbf{X}_{i-1}]\big) \Big) \end{aligned}$$

(Recall that z_e^a is the summation of $z_{e',c^\bullet}$ over all edges $e' \in R_1$ incident to e.)

$$= F_a + F_b - \sum_{(e,e') \,:\, e \in R,\ e' \in R_1,\ e' \text{ incident to } e} \left(\mathrm{E}[z_{e',c^\bullet} \cdot z_e^b | \mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} \cdot z_e^b | \mathbf{X}_{i-1}] \right)$$

(Any pair $(e, e') \notin P_1 \cup P_2 \cup P_3$ contributes zero to this summation.)

$$= F_a + F_b + Q_1 + Q_2 + Q_3.$$

To prove this claim it suffices to show that (i) $|F_a + Q_1| \leq 2D_i^a$, (ii) $|F_b + Q_2| \leq D_i^b$, and (iii) $|Q_3| \leq 2D_i^a$. We expand F_a using the fact that z_e^a is the summation of $z_{e',c^\bullet}$ over all edges $e' \in R_1$ incident to e.

$$|F_a + Q_1| \leq \left| Q_1 + \sum_{(e,e') \,:\, e \in R,\ e' \in R_1,\ e' \text{ incident to } e} \left(\mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_{i-1}] \right) \right|$$

Since any pair $(e, e') \notin P_1 \cup P_3$ contributes 0 in the summation,

$$\leq \left| Q_1 + \sum_{(e,e') \in P_1 \cup P_3} \left(\mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_{i-1}] \right) \right|$$

and by definition of Q_1,

$$\leq \sum_{(e,e') \in P_1} \left| \mathrm{E}[z_{e',c^\bullet}(1 - z_e^b) | \mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}(1 - z_e^b) | \mathbf{X}_{i-1}] \right| + \sum_{(e,e') \in P_3} \left| \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_{i-1}] \right|$$

When $e \notin R \setminus Y^b$, $\mathrm{E}[z_e^b | \mathbf{X}_{i-1}] = \mathrm{E}[z_e^b | \mathbf{X}_i]$, so

$$\leq \sum_{(e,e') \in P_1} \left(1 - \mathrm{E}[z_e^b | \mathbf{X}_{i-1}]\right) \left| \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_{i-1}] \right| + \sum_{(e,e') \in P_3} \left| \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_{i-1}] \right|$$

and since $0 \leq 1 - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}] \leq 1$,

$$\leq \sum_{(e,e')\in P_1\cup P_3} |\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}]|$$

Finally, any edge $e' \in R_1$ is incident to at most 2 edges in R, so

$$\leq 2 \sum_{e'\in Y^a} |\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}]|$$
$$\leq 2D_i^a.$$

For each $e \in Y^b$, we write $B(e)$ to denote the set of all edges $e' \in R_1 \setminus Y^a$ that are incident to e, i.e., $\{e\} \times B(e) \subseteq P_2$. Notice that $0 \leq \mathrm{E}[\sum_{e'\in B(e)} z_{e',c^\bullet}|\mathbf{X}_{i-1}] = \mathrm{E}[\sum_{e'\in B(e')} z_{e',c^\bullet}|\mathbf{X}_i] \leq 1$, since $e = \{v^\bullet, u\}$ and all edges in $B(e)$ share the vertex u, and so at most one could be successfully colored $c^\bullet$. By definition, none are incident to $e^\star$. We can now bound $|F_b + Q_2|$ as follows.

$$|F_b + Q_2| \leq \left| Q_2 + \sum_{e\in Y^b} \mathrm{E}[z_e^b|\mathbf{X}_i] - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}] \right|$$

According to the definition of $B(e)$ and Q_2,

$$\leq \sum_{e\in Y^b} \left| \mathrm{E}\left[z_e^b \left(1 - \sum_{e'\in B(e)} z_{e',c^\bullet} \right) \middle| \mathbf{X}_i \right] - \mathrm{E}\left[z_e^b \left(1 - \sum_{e'\in B(e)} z_{e',c^\bullet} \right) \middle| \mathbf{X}_{i-1} \right] \right|$$

For every $e' \in R_1 \setminus Y^a$, we have $\mathrm{E}\left[z_{e',c^\bullet}|\mathbf{X}_i\right] = \mathrm{E}\left[z_{e',c^\bullet}|\mathbf{X}_{i-1}\right]$, which implies

$$\leq \sum_{e\in Y^b} \left(1 - \mathrm{E}\left[\sum_{e'\in B(e)} z_{e',c^\bullet} \middle| \mathbf{X}_{i-1} \right] \right) \cdot \left| \mathrm{E}[z_e^b|\mathbf{X}_i] - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}] \right|$$
$$\leq \sum_{e\in Y^b} \left| \mathrm{E}[z_e^b|\mathbf{X}_i] - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}] \right|$$
$$= D_i^b.$$

Our last task is to bound the absolute value of Q_3.

$$\begin{aligned}|Q_3| &\leq \sum_{(e,e')\in P_3} \left(\mathrm{E}[z_{e',c^\bullet} \cdot z_e^b | \mathbf{X}_i] + \mathrm{E}[z_{e',c^\bullet} \cdot z_e^b | \mathbf{X}_{i-1}]\right) \\ &\leq \sum_{(e,e')\in P_3} \left(\mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_i] + \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_{i-1}]\right)\end{aligned}$$

Since any edge $e' \in R_1$ is incident to at most 2 edges in R,

$$\begin{aligned}&\leq 2 \sum_{e'\in Y^a} \left(\mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_i] + \mathrm{E}[z_{e',c^\bullet} | \mathbf{X}_{i-1}]\right) \\ &\leq 2D_i^a.\end{aligned}$$

□

Claim 2. *If $e^\star \in R_2$, then $D_i^a = O(w(e^\star)/p)$. If $e^\star \in R \cup R_1$, then $D_i^a = O(1)$.*

Proof. We first consider the case that $e^\star \in R_2$. In this case $|Y^a| = w(e^\star)$. Recall that $Y^a \subseteq R_1$, and so all $e \in Y^a$ have not yet decided whether to select $c^\bullet$ when X_i is revealed. Therefore, both $\mathrm{E}[z_{e,c^\bullet} | \mathbf{X}_i]$ and $\mathrm{E}[z_{e,c^\bullet} | \mathbf{X}_{i-1}]$ are within the range $[0, 1/p]$, and so $D_i^a = O(w(e^\star)/p)$. Next, consider the case that $e^\star \in R \cup R_1$. All edges in Y^a must share a vertex with $e^\star$, and so at most two edges in Y^a can successfully color themselves by $c^\bullet$. Hence

$$D_i^a \leq \sum_{e\in Y^a} \left(\mathrm{E}[z_{e,c^\bullet} | \mathbf{X}_i] + \mathrm{E}[z_{e,c^\bullet} | \mathbf{X}_{i-1}]\right) \leq 2 + 2 = 4 = O(1).$$

□

Claim 3. *If $e^\star \in R_1 \cup R_2$, then $D_i^b = O(1/p)$. If $e^\star \in R$, then $D_i^b = O(1)$.*

Proof. Recall that $z_e^b = \sum_{c\in\Psi(e)-\{c^\bullet\}} z_{e,c}$ for any edge $e \in Y^b$, and so

$$D_i^b \leq \sum_{e\in Y^b} \sum_{c\in\Psi(e)-\{c^\bullet\}} |\,\mathrm{E}[z_{e,c} | \mathbf{X}_i] - \mathrm{E}[z_{e,c} | \mathbf{X}_{i-1}]|.$$

We first show that $|\,\mathrm{E}[z_{e,c} | \mathbf{X}_i] - \mathrm{E}[z_{e,c} | \mathbf{X}_{i-1}]| = O(1/p^2)$ if $e^\star \neq e$. We write k_1 (resp., k_2) to denote the number of edges incident to e that have decided to select $c^\bullet$ (resp., have decided

to not select $c^\bullet$) by the time X_i is revealed.

$$\mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}] = \begin{cases} 0 & (e \text{ has decided to select } c^\bullet) \\ \frac{1}{p-1} \cdot (1-1/p)^{2t-1-k_1-k_2} \cdot (1-1/(p-1))^{k_2} & (e \text{ has decided to not select } c^\bullet) \\ \frac{1}{p} \cdot (1-1/p)^{2t-1-k_1-k_2} \cdot (1-1/(p-1))^{k_2} & (e \text{ has not made any decision}) \end{cases}$$

In any case, $\mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}] = O(1/p)$. There are two possibilities of $\mathrm{E}[z_{e,c}|\mathbf{X}_i]$ based on X_i, i.e., whether $e^\star$ selects $c^\bullet$.

$$\mathrm{E}[z_{e,c}|\mathbf{X}_i] = \begin{cases} \mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]/(1-1/p) & (e^\star \text{ selects } c^\bullet) \\ \mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}] \cdot (1-1/(p-1))/(1-1/p) & (e^\star \text{ does not select } c^\bullet) \end{cases}$$

In any case, $|\mathrm{E}[z_{e,c}|\mathbf{X}_i] - \mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]| = O(1/p^2)$. We are now in a position to bound D_i^b. For the case that $e^\star \in R_1 \cup R_2$, we have $|Y^b| \leq 2$ and $e^\star \notin Y^b$, and so $D_i^b \leq 2 \cdot (p-1) \cdot O(1/p^2) = O(1/p)$. For the case that $e^\star \in R$, we have $|Y^b| = |R| = t$ and $e^\star \in Y^b$, and so $D_i^b \leq 1 + (t-1) \cdot (p-1) \cdot O(1/p^2) = O(1)$. □

Revealing the Color Selected by an Edge in $R \cup R'$. Next, we analyze the effect of exposing the value of X_i, where $\alpha < i \leq \alpha + \beta$, given that all variables in $\mathbf{X}_{i-1}$ have been fixed.

Observe that z_e^a, for all $e \in R$, are already determined by $\{X_j : j \in [\alpha]\}$. If $z_e^a = 1$, then $z_e = 1$ regardless of the value of z_e^b; if $z_e^a = 0$, then $z_e = z_e^b$. For those edges $e \in R$ such that z_e is not determined by $\{X_j : j \in [\alpha]\}$, the random variable $z_e = z_e^b$ behaves the same as z_e in the analysis of concentration of vertex degree, so the analysis in Section 5.4.1 can be applied here (think of $S = R$ and $S' = R'$).

In more detail, for each edge $e' \in R'$, we define $w'(e')$ as $\sum_{e \in R,\ e' \text{ incident to } e} |\Psi(e') \cap \Psi(e) - \{c^\bullet\}|$. We have $\sum_{e' \in R'} w'(e') \leq |R|(p-1)(t-1) < pt^2$. Now consider the color $X_i = \mathrm{Color}^\star(e^\star)$ selected by the edge $e^\star \in R \cup R'$. From the analysis in Section 5.4.1, we infer the following.

- If $e^\star \in R'$, then $|D_i| = O(1)$ and $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(w'(e^\star)/(pt))$. Hence we can set $\sigma_i^2 = O(w'(e^\star)/(pt))$.
- If $e^\star \in R$, then $|D_i| = O(1)$ and $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(1)$. Hence we can set $\sigma_i^2 = O(1)$.

Thus, $\sum_{j=\alpha+1}^{\alpha+\beta} \sigma_i^2 = O(t)$, as desired.

5.5 Distributed Lovász Local Lemma on Trees

In this section, we study the distributed LLL on *tree-structured dependency graphs*, which we define as follows. Let T be a tree. Each vertex v holds some variables $\mathcal{V}(v)$ and is associated with a bad event $E(v)$ that depends only on variables within distance $r/2$ of v; that is, $\mathrm{vbl}(E(v)) = \bigcup_{u \in N^{r/2}(v)} \mathcal{V}(u)$. If S is a subset of the vertices, we use $\mathrm{vbl}(S)$ to be short for $\bigcup_{v \in S} \mathrm{vbl}(E(v)) = \bigcup_{v \in S} \bigcup_{u \in N^{r/2}(v)} \mathcal{V}(u)$.

The dependency graph for the set of bad events $\mathcal{E}$ is exactly T^r, which is the graph obtained by adding edges to all pairs of vertices of distance at most r in T. Thus, the maximum degree of the dependency graph is Δ^r, where Δ is the maximum degree of T. We fix the parameter $d = \Delta^r$. Notice that the tree-structured dependency graphs (with parameter r) arise naturally from any $r/2$-time RandLOCAL experiment that is run on a tree T. Throughout this section we assume $r/2 \geq 1$ is an integer and that $\Delta \geq 3$.

5.5.1 Deterministic LLL Algorithm

A (λ, γ)-*network decomposition* is a partition of the vertex set into $V_1, \ldots, V_\lambda$ such that each connected component induced by each V_i has diameter at most γ. Fischer and Ghaffari [61] showed that given a (λ, γ)-decomposition of $G_{\mathcal{E}}^2$, an LLL instance satisfying $p(ed)^\lambda < 1$ is solvable in $O(\lambda(\gamma + 1))$ time. We use a slight generalization of standard network decompositions. A $(\lambda_1, \gamma_1, \lambda_2, \gamma_2)$-network decomposition is a partition of the vertices into $V_1, \ldots, V_{\lambda_1}, U_1, \ldots, U_{\lambda_2}$ such that connected components induced by V_i have diameter at most γ_1 and those induced by U_i have diameter at most γ_2.

Lemma 5.7 (Fischer and Ghaffari [61])**.** *Suppose that a $(\lambda_1, \gamma_1, \lambda_2, \gamma_2)$-network decomposition of $G_{\mathcal{E}}^2$ is given. Any LLL instance on $G_{\mathcal{E}}$ satisfying $p(ed)^{\lambda_1 + \lambda_2} < 1$ can be solved in* DetLOCAL *in $O(\lambda_1(\gamma_1 + 1) + \lambda_2(\gamma_2 + 1))$ time.*

The proof of Theorem 5.5 is based on the network decompositions for trees found in Section 5.6. A *distance-d dominating set* of a graph G is a vertex set S such that for each vertex v in the graph G, there exists $u \in S$ such that $\mathrm{dist}(u, v) \leq d$.

Theorem 5.5. *Any tree-structured LLL satisfying $p(ed)^\lambda < 1$ with $\lambda \geq 2$ can be solved in* DetLOCAL *in $O(\max\{\log_\lambda s, \frac{\log s}{\log\log s}\} + \log^* n)$ time, where $s \leq n$ is the size of any distance-$O(1)$ dominating set of the tree T.*

Proof. Recall that the dependency graph is T^r for some tree T and constant r. In Section 5.6 we show that a standard $(2, O(\log s))$-decomposition for $(T^r)^2 = T^{2r}$ is computable in $O(\log s + \log^* n)$ time, and if $\lambda = \Omega(1)$ is sufficiently large, a $(1, O(\log_\lambda s), O(\lambda^2), 0)$-decomposition for T^{2r} is computable in $O(\log_\lambda s + \log^* n)$ time.

When $\lambda = O(1)$ is sufficiently small, we apply Lemma 5.7 with the first network decomposition. Because the decomposition has two parts, this works with LLL criterion $p(ed)^2 < 1$. When λ is sufficiently large we compute a $(1, O(\log_{\hat{\lambda}} s), O(\hat{\lambda}^2), 0)$-decomposition in $O(\log_{\hat{\lambda}} s + \log^* n)$ time, where $\hat{\lambda} = \min\{\lambda, \sqrt{\frac{\log s}{\log\log s}}\}$. We solve the LLL by applying Lemma 5.7, which takes time $O(\hat{\lambda}^2 + \log_{\hat{\lambda}} s + \log^* n) = O(\max\{\log_\lambda s, \frac{\log s}{\log\log s}\} + \log^* n)$. Observe that because of the $\hat{\lambda}^2$ term, we cannot benefit from LLL instances with $\lambda \gg \sqrt{\frac{\log s}{\log\log s}}$. □

Note that the time bound for Theorem 5.5 is in terms of s rather than n. We will apply Theorem 5.5 after performing a graph shattering step, the output of which creates many disjoint tree-structured instances with size $\mathrm{poly}(\Delta)\log n$, but which contain only $\log n$-size distance-$O(1)$ dominating sets. We want the time bound to be in terms of $s = \log n$ and independent of Δ. If we combine Theorem 5.5 with the $O(d^2 + \log^* n)$ Fischer and Ghaffari's [61] shattering routine, we obtain a $O(d^2 + \max\{\log_\lambda \log n, \frac{\log\log n}{\log\log\log n}\})$-time RandLOCAL LLL algorithm for criterion $p(ed)^\lambda < 1$, $\lambda \geq 4$, which is efficient only when d is small. In Section 5.5.2 we present a new method for computing a partial assignment to the variables that effectively shatters a large dependency graph into many independent subproblems, each satisfying a polynomial LLL criterion w.r.t. the unassigned variables.

5.5.2 Randomized LLL Algorithm

Consider a tree-structured LLL instance T^r with LLL criterion $p(ed)^\lambda < 1$. In subsequent discussion, unless otherwise stated, the underlying graph is, by default, assumed to be T. Our shattering routine will work towards finding a *good* partial assignment.

Definition 5.1. *A partial assignment ϕ to the variables in the LLL system is* good *if it satisfies the following three properties.*

1. *If all variables in* $\mathrm{vbl}(E(v)) = \bigcup_{u \in N^{r/2}(v)} \mathcal{V}(u)$ *are assigned, then the bad event $E(v)$ does not occur under the assignment ϕ.*
2. *Let V' be the set of all vertices v such that* $\mathrm{vbl}(E(v))$ *contains some unassigned*

variables. Each connected component C induced by V' has size at most $\mathrm{poly}(\Delta) \cdot O(\log n)$*, and C contains a distance-$2r$ dominating set with size at most $O(\log n)$.*

3. *Conditioned on the partial assignment ϕ, the probability of any bad event $E(v)$ is at most $p' = \sqrt{p}$. (In particular, (3) implies (1) as a special case.)*

Due to Definition 5.1(3), conditioned on a good partial assignment ϕ, the bad events in each connected component C induced by V' form an LLL system with the LLL criterion $p'(ed)^{\lambda/2} < 1$. Thus, the good partial assignment ϕ effectively shatters the tree T into small components, each of which is an independent LLL system. In Sections 5.5.3–5.5.5 we prove the following efficient "shattering lemma."

Lemma 5.8. *Suppose we are given a tree-structured LLL instance T^r satisfying LLL criterion $p(ed)^\lambda < 1$, where $\lambda \geq 2(4^r + 8r)$. There is a* RandLOCAL *algorithm that computes a good partial assignment ϕ in $O(\log_\lambda \log n)$ time.*

By applying Lemma 5.8 and then Theorem 5.5 to the LLL instance of each component, we are now able to efficiently solve tree-structured LLL instances in $O(\log \log n)$ time or faster, independent of the maximum degree d of the dependency graph. We have the following theorem.

Theorem 5.6. *Let T^r be a tree-structured LLL instance satisfying criterion $p(ed)^\lambda < 1$ with $\lambda \geq 2(4^r + 8r)$. This LLL can be solved in* RandLOCAL *in $O(\max\{\log_\lambda \log n, \frac{\log \log n}{\log \log \log n}\})$ time.*

The statement of Lemma 5.8 actually suggests an algorithm to compute a good partial assignment ϕ. First, draw a total assignment ϕ to $\mathcal{V}$ according to the distribution of the variables. If any bad event $E(v)$ occurs under ϕ, update ϕ by *unsetting* all variables in $\mathrm{vbl}(E(v))$. More generally, whenever $\Pr[E(v)|\phi]$ exceeds $\sqrt{p}$, update ϕ by unsetting all variables in $\mathrm{vbl}(E(v))$. This can be viewed as a *contagion dynamic* played out on the dependency graph. Bad events that occur under the initial total assignment are *infected*, and infected vertices can cause nearby neighbors to become infected.

If this contagion process were actually simulated, it would take $\Omega(\log n)$ parallel steps to reach a stable state, which is too slow. We will provide a different method to achieve a stable state that is exponentially faster, by avoiding a direct simulation.

5.5.3 Criterion for Infection

Let u be a vertex in the undirected tree T. Then $T - \{u\}$ consists of $\deg(u)$ subtrees $T_1, \ldots, T_{\deg(u)}$; we call T_k the *kth subtree* of u. Define $C_u(k, [i, j])$ to be the set of vertices in the kth subtree of u whose distance to u lies in the interval $[i, j]$. For example, $C_u(k, [1, 1])$ only contains the kth neighbor of u. For any vertex set S, define $\widehat{\deg}_S(u)$ as follows,

$$\widehat{\deg}_S(u) = \left|\{k \ : \ C_u(k, [1, r]) \cap S \neq \emptyset\}\right|.$$

In other words, it is the number of *distinct* subtrees of u containing at least one S-vertex within distance r.

Let $\mu \geq 4$ and $\lambda' \geq 1$ be two integers such that $\lambda \geq 2(\mu^r + \lambda')$. The following bad events $B(S, v)$ and $B(v)$ are defined w.r.t. the following process. First, we fix a total assignment ϕ to the variables, then progressively add vertices to the set S. All variables in $\mathrm{vbl}(S)$ are considered *unset*; for example, conditioning on "$\mathrm{vbl}(E(v)) \setminus \mathrm{vbl}(S)$" means keeping ϕ's assignment to $\mathrm{vbl}(E(v)) \setminus \mathrm{vbl}(S)$ and *resampling* $\mathrm{vbl}(S)$ according to their distribution.

$$B(S, v) : \left[\Pr\left[E(v) \mid \mathrm{vbl}(E(v)) \setminus \mathrm{vbl}(S)\right] \geq (ed)^{-\lambda/2}\right],$$
$$B(v) : \left[\bigcup_{S \subset N^r(v),\, |S| \leq \mu^r} B(S, v)\right].$$

In other words, $B(S, v)$ is the event that, if we *were* to resample $\mathrm{vbl}(S)$, the probability that $E(v)$ occurs is at least $(ed)^{-\lambda/2}$. The event $B(v)$ occurs if it is *possible* to find a subset S of cardinality at most μ^r such that $B(S, v)$ occurs.

We can now consider the probability that these events occur, over a *randomly* selected initial total assignment ϕ.

$$\Pr_\phi[B(S, v)] \leq \frac{\Pr_\phi[E(v)]}{\Pr_\phi[E(v) \mid B(S, v)]} \ \leq \ \frac{(ed)^{-\lambda}}{(ed)^{-\lambda/2}} \ = \ (ed)^{-\lambda/2} \ \leq \ (ed)^{-(\mu^r + \lambda')}.$$

By a union bound over the $|N^r(v)|^{\mu^r} \leq d^{\mu^r}$ choices of S,

$$\Pr_\phi[B(v)] \leq \sum_S \Pr_\phi[B(S, v)] < (ed)^{-\lambda'}.$$

Intuitively, $B(v)$ is the event that $E(v)$ is *too close* to happening. That is, relatively few variables need to be resampled to give $E(v)$ a likely probability of happening. Lemma 5.9 shows that the criterion for infection "$\widehat{\deg}_S(v) > \mu$" is a good proxy for the harder-to-analyze criterion "$E(v)$ is too close to happening".

Lemma 5.9. *Fix a total variable assignment ϕ. Let S be any vertex set such that, for each vertex v, if $B(v)$ occurs under ϕ or $\widehat{\deg}_S(v) > \mu$, then v must be in S. Then $\Pr[E(v) \mid \text{vbl}(E(v)) \setminus \text{vbl}(S)] < (ed)^{-\lambda/2}$ for each vertex v.*

Proof. If $v \in S$, then the probability of seeing $E(v)$ after resampling $\text{vbl}(S)$ is, according to the original LLL criterion, at most $p < (ed)^{-\lambda}$. In what follows we assume $v \notin S$.

To prove the lemma, it suffices to show that there exists a vertex set S' such that (i) $S' \subset N^r(v)$, (ii) $|S'| \leq \mu^r$, and (iii) $\text{vbl}(S') \cap \text{vbl}(E(v)) = \text{vbl}(S) \cap \text{vbl}(E(v))$. Notice that (iii) implies that resampling $\text{vbl}(S')$ is equivalent to resampling $\text{vbl}(S)$ from v's point of view. Since $v \notin S$, by assumption, event $B(v)$ does not occur. Since $|S'| \leq \mu^r$, event $B(S', v)$ does not occur. Hence $\Pr[E(v) \mid \text{vbl}(E(v)) \setminus \text{vbl}(S')] < (ed)^{-\lambda/2}$, as desired.

Root the tree at v. We call a vertex $u \in S$ "highest" if u is in $N^r(v)$ and no ancestor of u is in S. If H is the set of highest vertices, then $\text{vbl}(S) \cap \text{vbl}(E(v)) = \text{vbl}(H) \cap \text{vbl}(E(v))$, so we only need to bound $|H|$ by μ^r. Suppose, for the sake of contradiction, that $|H| \geq \mu^r + 1$. Define the path $(v = v_0, v_1, \ldots, v_r)$ by selecting v_i as the child of v_{i-1} that maximizes the number of vertices in H contained in the subtree rooted at v_i. We prove by induction that the subtree rooted at v_i contains at least $\mu^{r-i} + 1$ H-vertices. The base case $i = 0$ holds by assumption. If there are $\mu + 1$ subtrees of v_i containing H-vertices, then v_i would be infected. Thus, by the pigeonhole principle, the number of H-vertices in the subtree rooted at v_{i+1} must be at least $\lceil (\mu^{r-i} + 1)/\mu \rceil = \mu^{r-(i+1)} + 1$. Hence the subtree rooted at v_r contains $\mu^0 + 1 = 2$ H-vertices; this is a contradiction since the only vertex in this subtree eligible to be in H is v_r itself. □

5.5.4 Contagion Process

A (q_0, r, μ)-*contagion process* on an n-vertex tree T is played out as follows. Initially, each vertex is infected with probability q_0, and these events are independent for vertices at distance greater than r. If S is the set of infected vertices at some time and $\widehat{\deg}_S(v) > \mu$, then v becomes infected. In this section our goal is, given the initially infected vertices, to compute a superset of those vertices that is *stable* and *small*.

Definition 5.2. *Let S_0 be the initially infected vertices and $S \supset S_0$.*

- *S is called* stable *if it causes no more infection.*
- *S is called* small *if each connected component induced by $\bigcup_{v \in S} N^r(v)$ contains a distance-$2r$ dominating set of size at most $O(\log n)$.*

In Lemma 5.10, we show that one can efficiently compute a set S that is both *stable* and *small*.

Lemma 5.10. *Consider a (q_0, r, μ)-contagion process played on an n-vertex tree T with maximum degree Δ. There is a* RandLOCAL *algorithm that computes a small stable set S in $O(\log_\mu \log n)$ time, where r is constant, $q_0 \leq (ed)^{-8r}$, $d = \Delta^r$, and $\mu \geq 4$.*

The proof of Lemma 5.10 is deferred to Section 5.5.5. Lemma 5.11 connects the contagion problem to finding a good partial assignment.

Lemma 5.11. *Suppose there is a τ-round* RandLOCAL *algorithm for finding a small stable set S for a $((ed)^{-\lambda'}, r, \mu)$-contagion process. Then there exists a $(\tau + O(1))$-round* RandLOCAL *algorithm for finding a good partial assignment ϕ to a tree-structured LLL instance with criterion $p(ed)^\lambda < 1$, where $\lambda \geq 2(\mu^r + \lambda')$.*

Proof. Let $q_0 = (ed)^{-\lambda'}$. Consider the (q_0, r, μ)-contagion process defined by choosing a random assignment ϕ' to the variables in the LLL system and initially infecting all vertices v such that $B(v)$ occurs. The lower bound on λ implies $\Pr[B(v)] \leq q_0 = (ed)^{-\lambda'}$. Given the small stable set S, we let ϕ be the result of unassigning all variables in $\mathrm{vbl}(S) = \bigcup_{v \in S} \mathrm{vbl}(E(v)) = \bigcup_{v \in S} \bigcup_{u \in N^{r/2}(v)} \mathcal{V}(u)$.

We now verify that ϕ is a good partial assignment. Since S is stable, for each vertex v, if $B(v)$ occurs under ϕ or $\widehat{\deg}_S(v) > \mu$, then v must be in S. By Lemma 5.9, $\Pr[E(v) \mid \mathrm{vbl}(E(v)) \setminus \mathrm{vbl}(S)] < (ed)^{-\lambda/2} < \sqrt{p}$ for each vertex v, and so Definition 5.1(1,3) are satisfied. Let $V' = \bigcup_{v \in S} N^r(v)$ be the set of all vertices v such that $\mathrm{vbl}(E(v))$ contains some unassigned variables. Since S is small, each connected component C induced by V' contains a distance-$2r$ dominating set with size at most $O(\log n)$. Since $2r = O(1)$, the cardinality of C is at most $\mathrm{poly}(\Delta) \cdot O(\log n)$. Hence Definition 5.1(2) is also satisfied. □

We are now in a position to prove Lemma 5.8. Recall that the LLL criterion of in Lemma 5.8 is $\lambda \geq 2(4^r + 8r)$. We pick the largest *even* integer μ such that $\lambda \geq 2(\mu^r + 8r)$, and we set $\lambda' = 8r$. Notice that $\mu \geq 4$ and $\log \mu = \Theta(\log \lambda)$. By Lemma 5.10, a small

stable set S for the $((ed)^{-8r}, r, \mu)$-contagion process can be computed in $O(\log_\mu \log n) = O(\log_\lambda \log n)$ time. By Lemma 5.11, this implies a $O(\log_\lambda \log n)$-time RandLOCAL algorithm to finding a good partial assignment ϕ under the LLL criterion $p(ed)^\lambda < 1$.

5.5.5 Finding a Small Stable Set

We prove Lemma 5.10 in this section. The algorithm for Lemma 5.10 simulates a more virulent contagion process for τ steps using threshold $\mu/2$ rather than μ, then simulates a reverse-contagion for τ steps, where vertices become *uninfected* if they were not initially infected and they have nearby infected vertices in at most μ subtrees. We prove that when $\tau = \Theta(\log_\mu \log n)$, the final infected set $S = L_\tau$ is both stable and small. This process is called Find-Small-Stable-Set. The sets generated by this process satisfy that $U_0 \subseteq \cdots \subseteq U_\tau = L_0 \supseteq \cdots \supseteq L_\tau$.

Find-Small-Stable-Set.

(1) $U_0 \leftarrow \{u \in V \mid u \text{ is initially infected}\}$. That is, $u \in U_0$ if $B(u)$ occurs initially.

(2) For $1 \le i \le \tau$, do $U_i \leftarrow U_{i-1} \cup \{u \in V \mid \widehat{\deg}_{U_{i-1}}(u) > \mu/2\}$.

(3) $L_0 \leftarrow U_\tau$.

(4) For $1 \le i \le \tau$, do $L_i \leftarrow L_{i-1} \setminus \{u \in L_{i-1} \setminus U_0 \mid \widehat{\deg}_{L_{i-1}}(u) \le \mu\}$.

(5) Return L_τ.

We show that $S = L_\tau$ is stable in Lemma 5.16. Let $L_{\tau+1}$ be the set of all vertices u such that $\widehat{\deg}_{L_\tau}(u) > \mu$. Our goal is to show that if $u \notin L_\tau$, then $\widehat{\deg}_{L_\tau}(u) \le \mu$ (i.e., $u \notin L_{\tau+1}$) with high probability.

Root T at an arbitrary vertex, and let T' refer to the rooted version. Define T'_u to be the subtree of T' rooted at u, and define $C'_u(k, [i, j])$ as $C_u(k, [i, j]) \cap T'_u$. Given a vertex set W, define $\deg'_W(u)$ as the number of different k such that $C'_u(k, [1, r]) \cap W \neq \emptyset$. Although the original contagion process is played on T, it is easier to analyze a similar process played on T', where only descendants can cause a vertex to become infected.

In general, if $\{X(u)\}_{u \in V}$ is an ensemble of events associated with vertices and W a subset of vertices, we write $X(W)$ to denote the event $\bigcup_{u \in W} X(u)$, i.e., there exists $u \in W$ such that $X(u)$ occurs. We write X to denote the set of vertices $\{u \in V \mid X(u) \text{ occurs}\}$. For any two events A and B, we write $A \Rightarrow B$ to denote $A \subseteq B$, i.e., A implies B. With

respect to a vertex u, consider the following three sequences of events.

$(F_i(u))$: for each $0 \le i \le \tau$, let $F_i(u)$ be $(u \notin U_i) \wedge (u \in L_{i+1})$.

$(H_i(u))$: let $H_0(u)$ be $(u \in U_0)$;
for each $0 \le i < \tau$, let $H_{i+1}(u)$ be $H_0(u) \vee (\deg'_{H_i}(u) \ge \mu/2)$.

$(\tilde{F}_i(u))$: let $\tilde{F}_0(u)$ be $H_\tau(u)$; for each $0 \le i < \tau$, let $\tilde{F}_{i+1}(u)$ be $\deg'_{\tilde{F}_i}(u) \ge \mu/2$.

Lemma 5.12. *No vertex can belong to both $U_\tau \setminus L_\tau$ and $L_{\tau+1}$.*

Proof. Suppose there were such a vertex u. If $u \in L_{\tau+1}$ then it must have more than μ neighbors in L_τ, which were also in $L_{\tau-1} \subseteq \cdots \subseteq L_0 = U_\tau$. But if $u \in U_\tau$ then it would also remain in $L_0, \ldots, L_\tau$, contradicting the assumption that $u \in U_\tau \setminus L_\tau$. □

By Lemma 5.12, to prove that $S = L_\tau$ is stable, it suffices to prove that

$$\Pr[F_\tau(u)] = \Pr[(u \notin L_\tau) \wedge (u \in L_{\tau+1})] = 1/\mathrm{poly}(n).$$

Lemma 5.13 connects the true contagion process on T to an imagined one played on T'.

Lemma 5.13. *For each vertex u in T, and for each $0 \le i \le \tau$, we have $F_i(u) \Rightarrow \tilde{F}_i(u)$.*

Proof. We first show that $(u \in U_i) \Rightarrow H_i(u)$, for each $0 \le i \le \tau$. The base case ($i = 0$) follows from the definition of $H_0(u)$. Assume by inductive hypothesis that $(u \in U_{i-1}) \Rightarrow H_{i-1}(u)$. We have:

$$(u \in U_i \setminus U_0) \Rightarrow \left(\widehat{\deg}_{U_{i-1}}(u) > \mu/2\right) \Rightarrow \left(\deg'_{U_{i-1}}(u) \ge \mu/2\right) \Rightarrow \left(\deg'_{H_{i-1}}(u) \ge \mu/2\right).$$

This implies $(u \in U_i) \Rightarrow H_i(u)$, since $(u \in U_0) \Rightarrow H_0(u) \Rightarrow H_i(u)$.

Next, we prove by induction that $F_i(u) \Rightarrow \tilde{F}_i(u)$, for each $0 \le i \le \tau$. The base case $i = 0$ follows from the above result:

$$F_0(u) \Rightarrow (u \in L_1) \Rightarrow (u \in L_0 = U_\tau) \Rightarrow H_\tau(u) \Rightarrow \tilde{F}_0(u).$$

Assume inductively that $F_{i-1}(u) \Rightarrow \tilde{F}_{i-1}(u)$. Let u be any vertex in $L_{i+1} \setminus U_i$, i.e., the event $F_i(u)$ occurs. Since $u \notin U_i \supseteq U_0$, the only way Find-Small-Stable-Set could put

$u \in L_{i+1} \setminus U_i$ is if

$$\begin{aligned} \widehat{\deg}_{L_i}(u) &> \mu \\ \text{and } \widehat{\deg}_{U_{i-1}}(u) &\leq \mu/2, \end{aligned}$$

which implies

$$\widehat{\deg}_{F_{i-1}}(u) = \widehat{\deg}_{L_i}(u) - \widehat{\deg}_{U_{i-1}}(u) > \mu/2.$$

and hence

$$\deg'_{F_{i-1}}(u) \geq \mu/2.$$

By inductive hypothesis, we have

$$\left(\deg'_{F_{i-1}}(u) \geq \mu/2\right) \Rightarrow \left(\deg'_{\tilde{F}_{i-1}}(u) \geq \mu/2\right) \Rightarrow \tilde{F}_i(u),$$

which completes the induction. □

For brevity, define $p_i = \max_u \Pr[\tilde{F}_i(u)]$ and $q_i = \max_u \Pr[H_i(u)]$. We prove two auxiliary lemmas.

Lemma 5.14. $p_\tau \leq (\Delta^{2((r^2/2)+1)} p_0)^{(\frac{\mu}{2})^{\tau/(r/2)}}$.

Proof. Suppose that u is a vertex such that $\tilde{F}_i(u)$ occurs. Then, by definition of $\tilde{F}_i(u)$, there exist $\mu/2$ different indices k such that $\tilde{F}_{i-1}(C'_u(k,[1,r]))$ occurs. A consequence of this observation is that

$$\begin{aligned} \tilde{F}_{i-1}(C'_u(k,[1,r])) &\Rightarrow \tilde{F}_{i-2}(C'_u(k,[2,2r])) \\ &\Rightarrow \tilde{F}_{i-3}(C'_u(k,[3,3r])) \cdots \\ &\Rightarrow \tilde{F}_{i-(r/2)}(C'_u(k,[r/2,r^2/2])). \end{aligned}$$

Therefore, if $\tilde{F}_i(u)$ occurs, there must exist $\mu/2$ indices k such that $\tilde{F}_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))$ occurs. The $\mu/2$ events $\{\tilde{F}_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))\}$ are independent, since $\tilde{F}_i(v)$ depends only on $\mathrm{vbl}(T'_v) = \bigcup_{w \in N^{r/2}(v) \cup T'_v} \mathcal{V}(w)$. This indepen-

dence property is one reason why it is easier to analyze a contagion on T' rather than T.

By a union bound over all vertices in $C'_u(k, [r/2, r^2/2])$, we have

$$\Pr[\tilde{F}_{i-(r/2)}(C'_u(k, [r/2, r^2/2]))] \leq \Delta^{r^2/2-1} p_{i-(r/2)}.$$

Taking a union bound over at most $\binom{\Delta}{\mu/2}$ choices of $\mu/2$ distinct indices k, we infer that

$$p_i \leq \Delta^{\mu/2} (\Delta^{r^2/2-1} p_{i-(r/2)})^{\mu/2} \leq (\Delta^{(r^2/2)} p_{i-(r/2)})^{\mu/2}$$

for each $r/2 \leq i \leq \tau$. Assume τ is a multiple of $r/2$, and recall $\mu/2 \geq 2$. We can bound p_τ as follows.

$$p_\tau \leq p_0^{(\frac{\mu}{2})^{\tau/(r/2)}} \cdot \prod_{j=1}^{\tau/(r/2)} \left(\Delta^{(r^2/2)}\right)^{(\frac{\mu}{2})^j} \leq (\Delta^{r^2} p_0)^{(\frac{\mu}{2})^{\tau/(r/2)}}. \qquad \square$$

Lemma 5.15. $p_0 = q_\tau \leq \Delta^{r/2} q_0$.

Proof. Recall that $H_i(u)$ is $(u \in H_0) \vee (\deg'_{H_{i-1}}(u) \geq \mu/2)$. This implies that

$$H_{i-1}(C'_u(k, [1, r])) \Rightarrow H_0(C'_u(k, [1, r])) \vee H_{i-2}(C'_u(k, [2, 2r])).$$

Repeating this $(r/2) - 1$ times, $H_{i-1}(C'_u(k, [1, r]))$ implies that

$$H_0(C'_u(k, [1, r(r/2 - 1)]) \vee H_{i-(r/2)}(C'_u(k, [r/2, r^2/2])).$$

Since $H_0(C'_u(k, [1, r(r/2 - 1)]) \Rightarrow H_{i-(r/2)}(C'_u(k, [r/2, r^2/2]))$, we conclude that

$$H_{i-1}(C'_u(k, [1, r])) \Rightarrow H_0(C'_u(k, [1, r/2 - 1]) \vee H_{i-(r/2)}(C'_u(k, [r/2, r^2/2])).$$

Thus, if $H_i(u)$ occurs, then either (i) $H_0(N^{r/2-1}(u))$ occurs, or (ii) there exist $\mu/2$ different indices k such that $H_{i-(r/2)}(C'_u(k, [r/2, r^2/2]))$ occurs. The events $H_{i-(r/2)}(C'_u(k, [r/2, r^2/2]))$ for all k are independent, since $H_i(v)$ depends only on $\mathrm{vbl}(T'_v) = \bigcup_{w \in N^{r/2}(v) \cup T'_v} \mathcal{V}(w)$.

By a union bound, $\Pr[H_{i-(r/2)}(C'_u(k, [r/2, r^2/2]))] \leq \Delta^{r^2/2-1} q_{i-r/2}$. Suppose that τ is a multiple of $r/2$. Taking a union bound over at most $\binom{\Delta}{\mu/2}$ choices of $\mu/2$ distinct indices

k, we have

$$\begin{aligned}
q_\tau &\le \Pr[H_0(N^{r/2-1}(u))] + \binom{\Delta}{\mu/2} \cdot \Delta^{r^2/2-1} q_{\tau-(r/2)} \\
&\le \Delta^{r/2-1} q_0 + \Delta^{\mu/2} (\Delta^{r^2/2-1} q_{\tau-(r/2)})^{\mu/2} \\
&\le \Delta^{r/2-1} q_0 + (\Delta^{r^2/2} q_{\tau-(r/2)})^{\mu/2} \\
&\le \Delta^{r/2-1} q_0 + q_0^{(\frac{\mu}{2})^{\tau/(r/2)}} \cdot \prod_{j=1}^{\tau/(r/2)} \left(\Delta^{r^2/2}\right)^{(\frac{\mu}{2})^j} \\
&\le \Delta^{r/2-1} q_0 + \left(\Delta^{2(r^2/2)} q_0\right)^{(\frac{\mu}{2})^{\tau/(r/2)}} && (\mu/2 \ge 2) \\
&\le \Delta^{r/2-1} q_0 + \left(\Delta^{2(r^2/2)} q_0\right)^{2} && ((\mu/2)^{\tau/(r/2)} \ge 2) \\
&\le \Delta^{r/2-1} q_0 + \Delta^{4(r^2/2)-8r^2} q_0 && (q_0 \le (ed)^{-8r} \text{ and } d = \Delta^r) \\
&\le \Delta^{r/2} q_0. && \square
\end{aligned}$$

We are now ready to prove that $S = L_\tau$ is stable.

Lemma 5.16. *For each vertex $u \notin L_\tau$, $\widehat{\deg}_{L_\tau}(u) \le \mu$ with high probability, and so L_τ is stable.*

Proof. It suffices to show that $\Pr[F_\tau(u)] = 1/\mathrm{poly}(n)$. By Lemma 5.13, $\Pr[F_\tau(u)] \le \Pr[\tilde{F}_\tau(u)] = p_\tau$. We show that $p_\tau = 1/\mathrm{poly}(n)$.

$$\begin{aligned}
p_\tau &\le (\Delta^{r^2} p_0)^{(\frac{\mu}{2})^{\tau/(r/2)}} && \text{(Lemma 5.14)} \\
&\le (\Delta^{r^2+r/2} q_0)^{(\frac{\mu}{2})^{\tau/(r/2)}} && \text{(Lemma 5.15)} \\
&\le (\Delta^{r^2+r/2-8r^2})^{(\frac{\mu}{2})^{\tau/(r/2)}} && (q_0 \le (ed)^{-8r} \text{ and } d = \Delta^r) \\
&\le (\Delta^{-27})^{(\frac{\mu}{2})^{\tau/(r/2)}} && (r \ge 2) \\
&\le (\Delta^{-27})^{\Theta(\log n)} && (\tau = \Theta(\log_\mu \log n) \text{ and } r = O(1)) \\
&\le 1/\mathrm{poly}(n). && \square
\end{aligned}$$

In Lemma 5.18 we prove that U_τ is small, which implies that $S = L_\tau \subseteq U_\tau$ is also small. We write $T^{[a,b]}$ to denote the graph defined by the vertex set $V(T)$ and the edge set $\{\{u, v\} \mid \mathrm{dist}_T(u, v) \in [a, b]\}$. We first prove an auxiliary lemma.

Lemma 5.17. *Fix a $c \geq 1$. With probability $1 - n^{-\Omega(c)}$, the graph $H = T^{[r+1,4r]}$ has no connected subgraph D such that (i) $|D| \geq c \log n$, and (ii) there is a subset $D' \subseteq D \cap U_0$ containing at least half of the vertices in D, and $\mathrm{dist}_T(u, v) > r$ for distinct $u, v \in D'$.*

Proof. The proof is similar to that of [23, Lemma 3.3]. Suppose that such D exists, and consider a tree $\hat{T}$ in H spanning D. There are at most $4^{c \log n}$ different rooted unlabeled $c \log n$-node trees; and each of them can be embedded into H in less that $n \cdot \Delta^{4r(c \log n - 1)}$ ways. Moreover, there are at most $2^{c \log n}$ ways of selecting a subset $D' \subseteq D$. Since $|D'| \geq c \log n/2$ and $\mathrm{dist}_T(u, v) > r$ for distinct $u, v \in D'$, the probability that such $\hat{T}$ exists is at most $q_0^{c \log n/2}$.

Recall that $q_0 \leq (ed)^{-8r}$, $d = \Delta^r$, $r \geq 2$, and $\Delta \geq 3$. A union bound over all possibilities of $\hat{T}$ implies that such D exists with probability at most

$$\begin{aligned} p' &= 4^{c \log n} \cdot n \cdot \Delta^{4r(c \log n - 1)} \cdot 2^{c \log n} \cdot q_0^{c \log n/2} \\ &\leq n^{3c+1} \Delta^{-4c(r^2 - r) \log n} e^{-4cr \log n} \\ &\leq n^{(4 - 4(r^2 - r) \log \Delta - 4 \log e)c} \\ &\leq n^{-14c}. \end{aligned}$$

□

Lemma 5.18. *With high probability, each connected component in the subgraph of T induced by $\bigcup_{v \in U_\tau} N^r(v)$ contains a distance-$2r$ dominating set of size at most $O(\log n)$, and so U_τ is small.*

Proof. Let C be any connected component induced by $\bigcup_{v \in U_\tau} N^r(v)$ We pick a distance-$2r$ dominating set D of C greedily, preferring vertices in U_0 over U_1, and U_1 over U_2, etc. Each time a vertex v is picked we remove from consideration all vertices in $N^r(v)$. Recall that $U_0 \subseteq \cdots \subseteq U_\tau$. The set D is obviously a distance-r dominating set of $U_\tau \cap C$. Since $U_\tau \cap C$ is itself a distance-r dominating set of C, the set D is a distance-$2r$ dominating set of C.

We write u_i to denote the ith vertex added to D, and define $D_i = \{u_1, \ldots, u_i\}$. Let m_i denote the number of connected components induced by D_i in the graph $T^{[r+1,2r]}$ (rather than T). We claim that if $u_i \notin U_0$, then $m_i < m_{i-1}$. This implies that at least half of the vertices in D belong to U_0. Observe that the set D is connected in $H = T^{[r+1,4r]}$ (since D is a distance-$2r$ dominating set of C), and so by Lemma 5.17, $|D| = O(\log n)$ with high probability.

We prove the above claim in the remainder of the proof. Consider the moment some $u_i \notin U_0$ is added to D. We will show that the connected component of D_i in the graph

$T^{[r+1,2r]}$ that contains u_i is formed by merging u_i with at least two connected components of D_{i-1} in the graph $T^{[r+1,2r]}$.

The algorithm Find-Small-Stable-Set added u_i to U_j because u_i had at least $\mu/2 \geq 2$ subtrees containing U_{j-1}-vertices that are within $N^r(u_i)$. Let T_1 and T_2 be any two such subtrees. For each $k = 1, 2$, let v_k be a U_{j-1}-vertex contained in both T_k and $N^r(u_i)$. Then there must be a vertex $w_k \in N^r(v_k)$ such that w_k has been already added to D, since otherwise the greedy algorithm should prefer v_k over u_i. Observe that w_1 and w_2 belong to separate connected components of D_{i-1} in the graph $T^{[r+1,2r]}$, since $u_i \notin N^r(w_1) \cup N^r(w_2)$; but w_1, w_2, and u_i are in the same component of D_i in the graph $T^{[r+1,2r]}$, since $w_k \in N^r(v_k) \subseteq N^{2r}(u_i)$, for both $k = 1, 2$. □

We have proven (Lemmas 5.16 and 5.18) that the algorithm Find-Small-Stable-Set computes a set $S = L_\tau$ that is *stable* and *small*, in $O(\log_\mu \log n)$ time. Lemma 5.11 shows that any such algorithm can be used to find a *good* partial assignment to the variables in any tree-structured LLL instance with $p(ed)^\lambda < 1$ and $\lambda \geq 2(4^r + 8r)$.[26] The *stability* criterion is used to show that the derived LLL instances satisfy $p'(ed)^{\lambda/2} < 1$ and $p' = \sqrt{p}$. The *smallness* criterion implies that the instances have size $\text{poly}(\Delta) \log n$ and $\log n$-size, distance-$O(1)$ dominating sets. Because $\log \mu = \Theta(\log \lambda)$, the time to find the good partial assignment is $O(\log_\lambda \log n)$.

5.6 Network Decomposition of Trees

Our interest in network decompositions stems from Lemma 5.7 due to [61], which shows that they imply non-trivial *deterministic* LLL algorithms. Most work on network decompositions [120] has focussed on arbitrary graphs.

Recall that a (λ, γ)-network decomposition is a partition of the vertices into λ parts $V_1, \ldots, V_\lambda$ such that each V_i induces connected components with diameter at most γ. We define a $(\lambda_1, \gamma_1, \lambda_2, \gamma_2)$-network decomposition to be a partition of the vertices into $\lambda_1 + \lambda_2$ parts $V_1, \ldots, V_{\lambda_1}, U_1, \ldots, U_{\lambda_2}$ such that each V_i (resp. U_i) induces connected components with diameter γ_1 (resp. γ_2).

[26] It is possible to replace $2(4^r + 8r)$ with $2(4^r + cr)$ for some smaller c, but not too small. We do not attempt to optimize this coefficient.

In this section we give two network decomposition algorithms for T^k where $T = (V, E)$ is an n-vertex tree that contains a distance-d dominating set S of size s. In our application d and k are constants. We assume all vertices agree on the numbers (d, k, s). We *do not* need a specific dominating set S be given as an input.

5.6.1 A Simple Network Decomposition

We first design a simple decomposition that partitions any tree-structured graph T^k into 2 parts.

Theorem 5.7. *Let T be a tree containing a distance-d dominating set of size s. There is a* DetLOCAL *algorithm $\mathcal{A}$ that computes a $(2, O(\log s + d/k))$-network decomposition of T^k in $O(k \log s + d + k \log^* n)$ time, i.e., $O(\log s + \log^* n)$ time when $d, k = O(1)$.*

In what follows we prove Theorem 5.7. We assume the underlying communications network is T rather than T^k. Consider the following two tree operations. They are similar to the ones described in Chapter 3, which are inspired by Miller and Reif [109]. The second operation is parameterized by an integer $\ell \geq 2$. In our application we set $\ell = \Theta(k)$.

Rake: Remove all leaves and isolated vertices.

Compress: Remove all vertices that belong to some path P such that (i) all vertices in P have degree at most 2, and (ii) the number of vertices in P is at least ℓ.

Let $\mathcal{A}'$ be the algorithm on the tree T defined as follows. (1) Do $3d+1$ Rake operations; (2) repeat the following sequence $\log s$ times: perform one Compress and $\ell - 1$ Rake operations.

Lemma 5.19. *Algorithm $\mathcal{A}'$ removes all vertices in T.*

Proof. Let S be any size-s distance-d dominating set of T. Root T at an arbitrary vertex and let size(v) be the number of vertices in the subtree rooted at v that belong to S. For any vertex $v \in V$, we prove by induction that (i) if size$(v) \leq 1$, then v is removed in Step (1) of $\mathcal{A}'$, and (ii) if $1 <$ size$(v) \leq 2^i$, then v is removed on or before the ith iteration of Step (2) of $\mathcal{A}'$.

For the case size$(v) \leq 1$, the height of the subtree rooted at v is at most $3d$, and so the entire subtree (including v) must be removed after $3d + 1$ Rake operations. Consider the case $2^{i-1} <$ size$(v) \leq 2^i$. By the inductive hypothesis, all vertices u with size$(u) \leq 2^{i-1}$ have been removed before the ith iteration of Step (2). With respect to the vertex v,

define V' to be the set of all vertices u such that (i) $\text{size}(u) > 2^{i-1}$, and (ii) u is in the subtree rooted at v. The set V' induces a path with one endpoint at v, since otherwise $\text{size}(v) > 2 \cdot 2^{i-1} = 2^i$. Let C be a connected component induced by vertices in V' that are not removed yet. If $|C| \geq \ell$, then all vertices in C are removed after 1 Compress. Otherwise, all vertices in C are removed after $\ell - 1$ Rake operations. □

To compute a $(2, O(\log s + d/k))$-network decomposition of T^k, it suffices to compute a partition $V = V_1 \cup V_2$ meeting the following two conditions.

(C1) For both labels $c \in \{1, 2\}$, any two vertices u and v in two distinct connected components of V_c must have $\text{dist}_T(u, v) > k$. This guarantees that the set of connected components of V_c remains unaltered if we change the underlying graph from T to T^k.

(C2) For both labels $c \in \{1, 2\}$, each connected component of V_c has diameter at most $O(k \log s + d)$. This implies the diameter upper bound of $O(\log s + d/k)$ when the underlying graph is T^k.

Recall that $\mathcal{A}'$ performs $L_r = (3d + 1) + (\ell - 1) \log s$ Rake and $L_c = \log s$ Compress operations; let $L = L_r + L_c = (3d + 1) + \ell \log s$. We write U_i to denote the set of all vertices that are removed during the ith operation. We are now in a position to present the algorithm $\mathcal{A}$. The algorithm $\mathcal{A}$ begins by computing the decomposition $V = \bigcup_{i=1}^{L} U_i$ using $\mathcal{A}'$. Then, for $i = L$ down to 1, label all vertices $v \in U_i$ by $\{1, 2\}$ as follows.

Case 1. If the ith operation is Rake, then label U_i as follows. Let $v \in U_i$. For the case that v is of degree-1 in the subgraph induced by $\bigcup_{j=i}^{L} U_j$, let u be the unique neighbor of v in $\bigcup_{j=i}^{L} U_j$. If $u \notin U_i$, then v adopts the same label as u. Otherwise, $u \in U_i$ must also be of degree-1 in $\bigcup_{j=i}^{L} U_j$, and we label both u and v the same by any $c \in \{1, 2\}$. For the case that v is an isolated vertex of $\bigcup_{j=i}^{L} U_j$, we label v by any $c \in \{1, 2\}$.

Case 2. If the ith operation is Compress, then label U_i as follows. Let P be a path that is a connected component of U_i. The number of vertices in P is at least $\ell = \Theta(k)$. Compute a labeling of the vertices in P meeting the following conditions: (i) each connected component induced by vertices of the same label has size within $[k, 7k]$, (ii) if v is an endpoint of P that is adjacent to a vertex $u \in \bigcup_{j=i+1}^{L} U_j$, then the label of v is the same as the label of u.

Such a labeling of P can be computed in $O(k)$ time if we are given an independent set I of P such that each connected component of $P \setminus I$ has size within $[3k, 6k]$, i.e., I is a

$(3k+1, 3k)$-ruling set. Suppose that we already have such a set I. For each $v \in I$, we find an arbitrary subpath $P_v \subseteq P$ that contains v and has exactly k vertices. All vertices in $\bigcup_{v \in I} P_v$ are labeled 1, and the remaining vertices in P are labeled 2. At this moment, each connected component induced by vertices of label 1 has size k, and each connected component induced by vertices of label 2 has size within $[k, 6k]$. If there is a component C violating Condition (ii) of the previous paragraph, we flip the label of all vertices in C (i.e., from 1 to 2 or from 2 to 1). If $\ell \geq ck$ for some large enough universal constant c, then we obtain a labeling satisfying both Condition (i) and Condition (ii).

The computation of the independent set I can be done in $O(k \log^* n)$ time, as we explain below. Suppose that we have an independent set I' of P such that each connected component of $P \setminus I$ has size within $[\alpha, 2\alpha]$. We show that in $O(\alpha \log^* n)$ time we can compute an independent set I'' of P such that each connected component of $P \setminus I$ has size within $[\beta, 2\beta]$, for any prescribed number $\beta \leq 2\alpha + 1$. Let $\tilde{P}$ be the "imaginary path" formed by contracting all vertices in $P \setminus I$. A maximal independent set $\tilde{I}$ of $\tilde{P}$ can be computed in $O(\alpha \log^* n)$ time. At this point, each connected component C of $P \setminus \tilde{I}$ has size within $[2\alpha + 1, 4\alpha + 2]$. The component size constraint $[\beta, 2\beta]$ can be met by adding new vertices to $\tilde{I}$ to subdivide the oversized components. The desired independent set I can be computed by $\log k$ iterated applications of the above procedure, and the runtime is $\sum_{i=1}^{\log k} O(2^i \log^* n) = O(k \log^* n)$.

Time Complexity. The total running time of $\mathcal{A}$ is $O(L_r + kL_c) + O(k \log^* n) = O(k \log s + d + k \log^* n)$, since the independent set computation of paths removed by Compress operation can be computed in $O(k \log^* n)$ time *in parallel*.

Validity of Labeling. We now verify that the labeling resulting from $\mathcal{A}$ satisfies the two conditions (C1) and (C2). Consider two distinct connected components C and C' induced by V_1. In view of Case 2 of algorithm $\mathcal{A}$, any path P' connecting a vertex in C and a vertex in C' in T must contain a subpath P'' consisting of k vertices in V_2. The same is true if we swap V_1 and V_2, and so (C1) holds. Consider a connected component C by V_1 or V_2. Let $i^\star$ be the largest index i such that $U_i \cap C \neq \emptyset$, and let $v^\star$ be any vertex in $C \cap U_{i^\star}$. We show that for any vertex $u \in C$, the unique path P connecting u and $v^\star$ in T contains $O(L_r + kL_c) = O(k \log s + d)$ vertices, and so (C2) holds. Consider any index $i \in [1, i^\star]$. If the ith operation is Rake, then we have $|P \cap U_i| \leq 2$ (in view of Case 1). If the ith operation is Compress, then we have $|P \cap U_i| \leq 7k$ (in view of Case 2). Thus, indeed

$|P| = O(L_r + kL_c)$.

5.6.2 A Mixed-diameter Network Decomposition

In this section we show how to compute a network decomposition where, for any parameter λ, one part has diameter roughly $\log_\lambda s$ and the remaining graph is properly $O(\lambda^2)$-colored, i.e., they form $O(\lambda^2)$ parts with diameter zero.

Theorem 5.8. *Let T be a tree containing a distance-d dominating set of size s. There is a* DetLOCAL *algorithm $\mathcal{A}$ that computes a $(1, O(\log_{\lambda/k} s + (d/k)), O(\lambda^2), 0)$-network decomposition of T^k in $O(k \log_{\lambda/k} s + d + k \log^* n)$ time, where $\lambda = \Omega(k)$ is sufficiently large, i.e., $\lambda \geq ck$ for some universal constant c. When $k, d = O(1)$ the time bound is $O(\log_\lambda s + \log^* n)$.*

In what follows we prove Theorem 5.8. We write T_i to denote the set of vertices that are not removed during the first $i-1$ tree operations. Consider the following two tree operations.

Rake: Remove all leaves and isolated vertices.

Compress: Remove all vertices v such that $|N^{2.5k}(v) \cap T_i| \leq \lambda$.

We set $m = \frac{\lambda}{2.5k} - 1$. Let $\mathcal{A}^*$ be the algorithm on the tree T defined as follows. (1) Do $3d+1$ Rake operations; (2) repeat the following sequence $\log_m s$ times: do one Compress followed by $2.5k$ Rake operations.

Lemma 5.20. *Algorithm $\mathcal{A}^*$ removes all vertices in T.*

Proof. Let S be any size-s distance-d dominating set of T. Root T at an arbitrary vertex, and let size(v) be the number of vertices in the subtree rooted at v that belong to S. We prove by induction that (i) if size$(v) \leq 1$, then v is removed in Step (1) of $\mathcal{A}^*$, and (ii) if $1 < \text{size}(v) \leq m^i$, v is removed within the first i iterations in Step (2) of $\mathcal{A}^*$.

For the case of size$(v) \leq 1$, the height of the subtree rooted at v is at most $3d$, and so the entire subtree (including v) must be removed after $3d+1$ Rake operations. For the case of $m^{i-1} < \text{size}(v) \leq m^i$, we assume by induction that all vertices u with size$(u) \leq m^{i-1}$ have been removed within the first $i-1$ iterations of Step (2). Let v be any vertex with size$(v) \in (m^{i-1}, m^i]$, and define V' as the set of all vertices u such that (i) size$(u) > m^{i-1}$, and (ii) u is in the subtree rooted at v. Notice that all descendants of v other than those

in V' have been removed within the first $i-1$ iterations of Step (2). Therefore, the set V' induces a subtree rooted at v having at most $m-1$ leaves. For those vertices $u \in V'$ with $\mathrm{dist}_T(u,v) \geq 2.5k$, we have $|N^{2.5k}(u) \cap T_i| \leq m(2.5k)+1 \leq \lambda$, so they will be removed after one Compress. The rest of the vertices in V' will be removed during the next $2.5k$ Rake operations. □

Now, we present our network decomposition algorithm $\mathcal{A}$. First, we run $\mathcal{A}^*$ on T. Then, for any vertex v removed by Compress, we mark all vertices in $N^{k/2}(v)$, i.e.,

$$\mathcal{M} = \{u \mid \exists v \text{ removed by Compress}, \ u \in N^{k/2}(v)\}$$

is the set of all marked vertices. We let $\tilde{T}$ be the graph defined as $V(\tilde{T}) = \mathcal{M}$, and $\{u,v\} \in E(\tilde{T})$ if $\mathrm{dist}_T(u,v) \leq k$.

The $(1, O(k \log_{\lambda/k} s + d), O(\lambda^2), 0)$ network decomposition of T^k is computed by assigning color 0 to all unmarked vertices, and coloring the remaining vertices in $\tilde{T}$ with $\{1, \ldots, O(\lambda^2)\}$. We next show that (i) $\Delta(\tilde{T}) \leq \lambda$, and so the $O(\lambda^2)$-coloring can be computed using Linial's algorithm [107] in $O(k \log^* n)$ time, and (ii) each connected component induced by unmarked vertices (in T^k) has diameter $O(\log_{\lambda/k} s + (d/k))$. Thus, $\mathcal{A}$ indeed computes a $(1, O(\log_{\lambda/k} s + (d/k)), O(\lambda^2), 0)$-network decomposition of T^k in $O(k \log_{\lambda/k} s + d + k \log^* n)$ time.

Proof of (i). For any marked vertex v, we claim that $|N^k(v) \cap \mathcal{M}| \leq \lambda$ (in T), and so $\Delta(\tilde{T}) \leq \lambda$. Let u be the first vertex marked in $N^k(v)$. The vertex u is added to $\mathcal{M}$ due to the removal of a vertex $w \in N^{k/2}(u)$ in a Compress operation (it is possible that $u = w$). Suppose that w was removed in $i^\star$th tree operation. Then we have $|N^{2.5k}(w) \cap T_{i^\star}| \leq \lambda$. We claim that $N^k(v) \cap \mathcal{M} \subseteq N^k(v) \cap T_{i^\star} \subseteq N^{2.5k}(w) \cap T_{i^\star}$, and this implies $|N^k(v) \cap \mathcal{M}| \leq \lambda$, and so $\Delta(\tilde{T}) \leq \lambda$. Since the $i^\star$th tree operation is the first iteration such that a vertex in $N^k(v)$ is marked due to the removal of another vertex during the $i^\star$th tree operation, $N^k(v) \cap T_{i^\star}$ contains all marked vertices within distance-k of v. Since $\mathrm{dist}(v,w) \leq \mathrm{dist}(v,u) + \mathrm{dist}(u,w) \leq 1.5k$, we have $N^k(v) \cap T_{i^\star} \subseteq N^{2.5k}(w) \cap T_{i^\star}$.

Proof of (ii). The diameter of each connected component (in T) induced by the unmarked vertices is $O(k \log_{\lambda/k} s + d)$, since the total number of Rake is $O(k \log_\lambda s) + 3d + 1$, and all vertices removed by Compress are marked. We show that the set of connected components induced by the unmarked vertices remains the same if we change the underlying

graph from T to T^k. This implies the diameter upper bound $O(\log_{\lambda/k} s + (d/k))$ when the underlying graph is T^k.

Consider any pair of unmarked vertices u and v. Notice that u and v must be removed by Rake. Suppose that u and v are not connected in T after deleting those vertices removed by Compress from T. Assume the first time they become disconnected in T is iteration i, which is due to the removal of a vertex w in Compress. Since all vertices in $N^{k/2}(w)$ are marked, the unique shortest path in T connecting u and v must has a subpath consisting of at least $2(k/2)+1 > k$ marked vertices. Thus, u and v are also disconnected in T^k after deleting all marked vertices.

5.7 Deterministic Algorithms for Edge Coloring Trees

Let $T = (V, E)$ be a tree with n vertices and $N^+(v) = N(v) \cup \{v\}$ be the inclusive neighborhood of v. We decompose T using two operations inspired by Miller and Reif [109], the second of which is parameterized by an integer $k \geq 2$.

Rake: Remove all leaves and isolated vertices from T.

Compress: Remove the set $\{v \in V \mid \text{for every } u \in N^+(v), \deg_T(u) \leq k\}$ from T.

Theorem 5.9. *Alternately applying Compress and Rake* $\log_k n+1$ *times removes all vertices from any* n*-vertex tree* T*.*

Proof. Root T at an arbitrary vertex and let $\text{size}(v)$ be the number of vertices in the subtree rooted at v. We prove by induction that if $\text{size}(v) \leq k^i$, v will be removed after the first $i+1$ rounds of Compress and Rake. The claim is trivially true when $i = 0$. Assume the claim is true for $i-1$. Let v be any vertex with $\text{size}(v) \in (k^{i-1}, k^i]$, and define V' to be the set of all vertices u such that (i) $\text{size}(u) \in (k^{i-1}, k^i]$ and (ii) u is in the subtree rooted at v. Notice that each vertex $u \in V'$ has $\deg_{V'}(u) \leq k$, since otherwise $\text{size}(u) > k^i$. By the inductive hypothesis, all descendants of v that are not in V' have been removed after i rounds of Compress and Rake. The $(i+1)$th Compress will remove any remaining vertices in $V' - \{v\}$. However, the degree of the parent of v is unbounded, so v may not be removed. If v still remains, the $(i+1)$th Rake will remove it. □

Theorem 5.10. *There is an* $O(\log_\Delta n)$*-time* DetLOCAL *algorithm for* Δ*-edge coloring a tree* T *with maximum degree* $\Delta \geq 3$*.*

Proof. Let β be the constant such that Linial's algorithm [107] finds a $\beta\Delta^2$-edge coloring in $O(\log^* n - \log^* \Delta + 1)$ time. We begin by decomposing T with Compress and Rake steps, using parameter $k = \max\{2, \lfloor(\Delta/\beta)^{1/3}\rfloor\}$. Define $T_i = (V_i, E_i)$ to be the forest before the ith round of Compress and Rake, and let V_i^c and V_i^r be those vertices removed by the ith Compress and Rake, respectively.

We edge color the trees $T_{\log_k n+1}, \ldots, T_1 = T$ in this order. Given a coloring of T_{i+1}, we need to color the remaining uncolored edges in T_i. Let $u \in T_{i+1}$ be a vertex, and let $v_1, \ldots, v_x \in V_i^r$ be the vertices adjacent to u removed by the ith Rake. At this point u is incident to at most $\Delta - x$ colored edges. We assign to $\{u, v_1\}, \ldots, \{u, v_x\}$ any distinct available colors from their palettes.

We now turn to the vertices removed by the ith Compress. First, suppose that Δ is large enough such that $k = \lfloor(\Delta/\beta)^{1/3}\rfloor$. Let ϕ be a βk^2-edge coloring of the (as yet uncolored) subgraph of T_i (i.e., the edges that are incident to some vertices in V_i^c). Partition the palette $\{1, \ldots, \Delta\}$ into βk^2 parts $P_1, \ldots, P_{\beta k^2}$. Each part has size $\Delta/(\beta k^2) \geq k$. Each $v \in V_i^c$ colors each edge $\{v, u\}$ any available color in $P_{\phi(\{v,u\})}$. Since $\deg_{T_i}(u) \leq k$, at most $k - 1$ of its incident edges may already be colored, and so there must be at least one available color in $P_{\phi(\{v,u\})}$ for $\{v, u\}$ to use. All calls to Linial's βk^2-edge coloring algorithm can be executed in parallel, so the overall time is $O(\log_k n + \log^* n - \log^* k) = O(\log_\Delta n)$.

When $k = 2$, the subgraph induced by $V_1^c \cup \cdots \cup V_{\log_k n+1}^c$ consists of a set of paths. In $O(\log^* n)$ time, we find an *initial* 3-edge coloring of these paths. We now color $T_{\log_k n+1}, \ldots, T_1$ in this order. Coloring the edges removed during a Rake is done as before. The set V_i^c removed in one Compress induces some paths, each end-edge of which may be adjacent to one (previously colored) edge in T_{i+1}. If the initial color of an end-edge conflicts with the coloring of T_{i+1}, we recolor it any available color. When $k = 2$ this procedure takes $O(\log^* n + \log_k n) = O(\log_\Delta n)$ time. □

An *oriented* tree is a rooted tree where each vertex that is not the root knows its parent. We show that a $(\Delta + 1)$-edge coloring of an oriented tree can be found in $O(\log^* n)$ time, but Δ-edge coloring takes $\Omega(\log_\Delta n)$ time.

Theorem 5.11. *Any oriented tree T can be $(\Delta + 1)$-edge colored in $O(\log^* n)$ time.*

Proof. Initially pick color $\phi_0(\{u, \text{parent}(u)\}) = i$ if $\text{ID}(u)$ is the ith largest ID among its siblings. Observe that for any i, $\phi_0^{-1}(i)$ is a subgraph consisting of oriented paths, and that $\phi_0^{-1}(\Delta)$ is at most one edge, attached to the root. For each $i \in \{1, \ldots, \Delta - 1\}$, in parallel,

recolor $\phi_0^{-1}(i)$ using the color set $\{i, \Delta, \Delta+1\}$ in such a way that the most ancestral edge in each path remains colored i. The result is a legal $(\Delta+1)$-edge coloring. This takes $O(\log^* n)$ time [45, 107]. □

Theorem 5.12. *Any Δ-edge coloring algorithm for oriented trees takes $\Omega(\log_\Delta n)$ time in* RandLOCAL.

Proof. Let T be an oriented Δ-regular tree with height $h = \Theta(\log_\Delta n)$ and $\mathcal{A}$ be an edge coloring algorithm running in $h/3$ time. The color of $\{u, \text{parent}(u)\}$ is uniquely determined by the colors of the edges incident to leaf-descendants of u. Let V' denote the set of leaf-descendants of u. In general, $N^{h/3}(u)$ and $\bigcup_{v \in V'} N^{h/3}(v)$ do not intersect. In this case, u only has a $1/\Delta$ chance of guessing the correct edge color; if it guesses incorrectly, there must be a violation somewhere in the subtree rooted at u. □

5.8 Lower Bounds for Augmenting Path-Type Algorithms

In this section, we show that for $c \in [1, \frac{\Delta}{3}]$, any algorithm for $(\Delta+c)$-edge coloring based on "extending partial colorings by recoloring subgraphs" needs $\Omega(\frac{\Delta}{c} \log \frac{cn}{\Delta})$ rounds.

Theorem 5.13. *Let Δ be the maximum degree and $c \in [1, \frac{\Delta}{3}]$. For any n, there exists an n-vertex graph $G = (V, E)$ and a partial edge coloring $\phi : E \to \{1, \ldots, \Delta+c, \perp\}$, with exactly one uncolored edge e_0 ($\phi(e_0) = \perp$) satisfying the following property. For any total edge coloring $\phi' : E \to \{1, \ldots, \Delta+c\}$ of G, ϕ and ϕ' differ on a subgraph of diameter $\Omega(\frac{\Delta}{c} \log(\frac{cn}{\Delta}))$.*

Suppose that G is a partially $(\Delta+c)$-edge colored graph, where an edge e_0 in uncolored. A natural approach to color e_0 is to find an "augmenting path" $e_0 e_1 \cdots e_\ell$, and then recolor the path. That is, for $0 \le i \le \ell - 1$, let the new color of e_i be the old color of e_{i+1}, and then color the last edge e_ℓ by choosing any available color (if possible). This type of approach has successfully led to a distributed algorithm for Brooks' theorem [122].[27] However, Theorem 5.13 implies the existence of a graph where any augmenting subgraph has diameter $\Omega(\frac{\Delta}{c} \log \frac{cn}{\Delta})$, which is expensive for large Δ. The remainder of this section is a proof of Theorem 5.13.

[27] Specifically, given a $(\Delta+1)$-vertex coloring, a Δ-coloring can be computed in $O(\log^3 n / \log \Delta)$ time, i.e., $\text{poly}(\log n)$ time, independent of Δ.

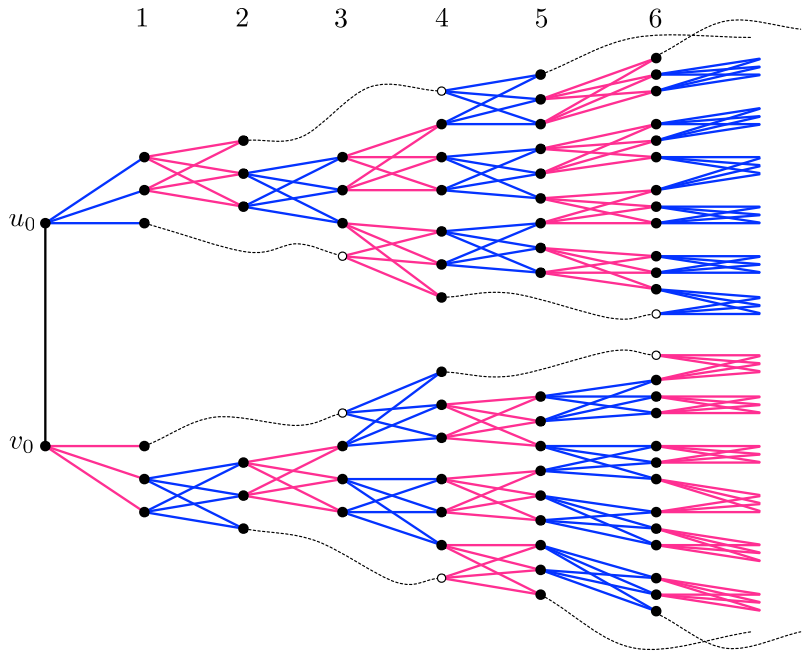

Figure 5.1: An example of the lower bound graph construction.

Construction. Without loss of generality, assume that $\Delta + c$ is even, and let $k = \frac{\Delta+c}{2}$. We divide the color palette $\{1, \ldots, \Delta + c\}$ into two equal-size sets $S_0 = \{1, \ldots, k\}$ and $S_1 = \{k + 1, \ldots, \Delta + c\}$. Let $k' = \Delta - k$.

The graph $G^*(\ell, \Delta, c)$ consists of one uncolored edge $e_0 = \{u_0, v_0\}$; all other vertices are arranged in layers $1, \ldots, \ell$ and all other edges connect two vertices in adjacent layers or layers i and $i + 3$, for some i. In $G^*(\ell, \Delta, c)$, e_0 is a bridge and the subgraphs attached to u_0 and v_0 are structurally isomorphic, but colored differently. Thus, we focus on the half of G^* attached to u_0.

One may refer to Figure 5.1 for an example of the construction when $\Delta = 5$, $c = 1$, $k = 3$, $k' = 2$, and $\ell \geq 7$. Edges colored by palette $S_0 = \{1, 2, 3\}$ are blue, and edges colored by palette $S_1 = \{4, 5, 6\}$ are pink. Leftover vertices in layer $i - 2$ are also depicted (hollow) in layer i, and joined by a dashed curve. They represent the same vertex, not two different vertices.

Base Case. Layer 1 consists of k vertices attached to u_0. They are initially colored with distinct colors from S_0.

Inductive Step. The $(i+1)$th layer is constructed as follows. We take all the vertices at layer i and the *leftover* vertices at layer $i-2$ and partition them into groups of size k'; any ungrouped vertices are called *leftovers* at level i. (In Figure 5.1 a leftover vertex in layer $i-2$ is drawn twice, solid in layer $i-2$ and hollow when it is *promoted* to layer i; they are connected by a dashed line.) The grouping is arbitrary, so long as all vertices promoted from layer $i-2$ are grouped. Each group forms the lefthand side of a complete bipartite graph $K_{k',k}$. Layer $i+1$ consists of the righthand side of all the (disjoint) copies of $K_{k',k}$. All the edges in these graphs are properly colored with S_b where $b = i \bmod 2$. (The subgraph attached to v_0 is constructed in the same way, except that we flip the parity: the complete bipartite graphs are colored with S_b, $b = (i+1) \bmod 2$.)

Define n_i and l_i as the number of layer-i vertices and layer-i leftover vertices.[28] According to the construction, (n_i) and (l_i) satisfy the following recurrences.

$$\begin{aligned} n_1 &= k \\ l_{-1} = l_0 &= 0 \\ n_{i+1} &= k\lfloor \frac{n_i + l_{i-2}}{k'} \rfloor && \text{for } i+1 \geq 2 \\ l_i &= (n_i + l_{i-2}) \bmod k' && \text{for } i \geq 1 \end{aligned}$$

Clearly $n_i = \Theta((k/k')^i)$. Since $k/k' = \frac{\Delta+c}{\Delta-c} = 1 + \frac{2c}{\Delta-c} = 1+\epsilon$, the total number of vertices in $G^*(\ell, \Delta, c)$ is $n = \Theta(\epsilon^{-1} n_\ell) = \Theta(\epsilon^{-1}(1+\epsilon)^\ell)$ and $\ell = \Theta(\log_{1+\epsilon}(\epsilon n)) = \Theta(\frac{\Delta}{c} \log \frac{cn}{\Delta})$. In particular, when c is constant and $\Delta < n^{1-\Omega(1)}$, $\ell = \Omega(\Delta \log n)$. The diameter of the graph is at least $\ell/3$ since, by construction, no edge crosses more than 3 layers.

Let ϕ be the initial partial edge-coloring of $G^*(\ell, \Delta, c)$, with e_0 left uncolored, and ϕ' be any total edge-coloring. We claim that ϕ' recolors at least one edge in the subgraph induced by layers $\ell-5, \ldots, \ell$. Suppose otherwise. Fix any vertex v in layer $\ell-6$. It has exactly k neighbors in a higher layer, either $\ell-5$ (if v is not a leftover vertex) or $\ell-3$ (if v is a leftover vertex); each such neighbor u is adjacent to k edges to a higher layer, all of which are colored from the palette S_1 (without loss of generality, assume ℓ is even). That means that all edges connecting v to a higher layer must be colored from S_0. By a reverse induction from $\ell-6$ down to 0, it follows that all edges from u_0 to layer 1 must be colored with S_0. A symmetric argument on v_0's side shows that all edges from v_0 to layer 1 must

[28] The leftover vertices at layer $i-2$ are still considered as layer i vertices, even though they have been promoted to layer i.

be colored with S_1, hence e_0 cannot be properly colored by ϕ'.

Chapter 6

The Complexity of Distributed Vertex Coloring

6.1 Overview

Much of what we know about the LOCAL model has emerged from studying the complexity of four canonical symmetry breaking problems and their variants: maximal independent set (MIS), $(\Delta + 1)$-vertex coloring, maximal matching, and $(2\Delta - 1)$-edge coloring. The palette sizes "$\Delta + 1$" and "$2\Delta - 1$" are minimal to still admit a greedy sequential solution; here Δ is the maximum degree of any vertex.

Early work [3, 12, 107, 108, 115, 120] showed that all the problems are reducible to MIS, all four problems require $\Omega(\log^* n)$ time, even with randomization; all can be solved in $O(\text{poly}(\Delta) + \log^* n)$ time (optimal when Δ is constant), or in $2^{O(\sqrt{\log n})}$ time for any Δ. Until recently, it was actually consistent with known results that all four problems had the same complexity.

Kuhn, Moscibroda, and Wattenhofer (KWM) [102] proved that the "independent set" problems (MIS and maximal matching) require $\Omega\Big(\min\Big\{\frac{\log\Delta}{\log\log\Delta}, \sqrt{\frac{\log n}{\log\log n}}\Big\}\Big)$ time, with or without randomization, via a reduction from $O(1)$-approximate minimum vertex cover. This lower bound provably separated MIS/maximal matching from simpler symmetry-breaking problems like $O(\Delta^2)$-coloring, which can be solved in $O(\log^* n)$ time [107].

We now know the KMW lower bounds cannot be extended to the canonical coloring problems, nor to variants of MIS like $(2, t)$-ruling sets, for $t \geq 2$ [23, 27, 68]. Elkin, Pettie, and Su [55] proved that $(2\Delta - 1)$-list edge coloring can be solved by a randomized

algorithm in $O(\log\log n + \mathsf{Det}(\operatorname{poly}\log n))$ time, which shows that neither the $\Omega\Big(\frac{\log\Delta}{\log\log\Delta}\Big)$ nor $\Omega\Big(\sqrt{\frac{\log n}{\log\log n}}\Big)$ KMW lower bound applied to this problem. Here $\mathsf{Det}(n')$ represents the *deterministic* complexity of the problem in question on n'-vertex graphs. Improving on [23, 135], Harris, Schneider, and Su [86] proved a similar separation for $(\Delta+1)$-vertex coloring. Their randomized algorithm solves the problem in $O(\sqrt{\log\Delta} + \log\log n + \mathsf{Det}_d(\operatorname{poly}\log n))$ time, where Det_d is the complexity of $(\deg+1)$-list coloring.

The "$\mathsf{Det}(\operatorname{poly}\log n)$" terms in the running times of [55, 86] are a consequence of the *graph shattering* technique applied to distributed symmetry breaking. Barenboim, Elkin, Pettie, and Schneider [23] showed that all the classic symmetry breaking problems could be reduced in $O(\log\Delta)$ or $O(\log^2\Delta)$ time, w.h.p., to a situation where we have independent subproblems of size $\operatorname{poly}\log(n)$, which can then be solved with the best available deterministic algorithm.[29]

In Theorem 2.1 of this thesis, we give a simple proof illustrating *why* graph shattering is inherent to the LOCAL model: the randomized complexity of any LCL problems is at least its deterministic complexity on $\sqrt{\log n}$-size instances. This lower bound explains why the state-of-the-art randomized symmetry breaking algorithms have such strange stated running times: they all depend on a randomized graph shattering routine (Rand.) and a deterministic (Det.) algorithm.

- $O(\log\Delta + 2^{O(\sqrt{\log\log n})})$ for MIS (Rand. [68] and Det. [120]),
- $O(\sqrt{\log\Delta} + 2^{O(\sqrt{\log\log n})})$ for $(\Delta+1)$-vertex coloring (Rand. [86] and Det. [120]),
- $O(\log\Delta + (\log\log n)^3)$ for maximal matching (Rand. [23] and Det. [61]),
- $O((\log\log n)^6)$ for $(2\Delta-1)$-edge coloring (Rand. [55] and Det. [62, 69]).

In each, the term that depends on n is the complexity of the best deterministic algorithm, scaled down to $\operatorname{poly}\log(n)$-size instances. In general, improvements in the deterministic complexities of these problems imply improvements to their randomized complexities, but only if the running times are improved in terms of "n" rather than "Δ." For example, a recent line of research has improved the complexity of $(\Delta+1)$-coloring in terms of Δ, from $O(\Delta+\log^* n)$ [21], to $\tilde{O}(\Delta^{3/4}) + O(\log^* n)$ [17], to the state-of-the-art bound of $O(\sqrt{\Delta\log\Delta}\log^*\Delta + \log^* n)$ due to Fraigniaud, Heinrich, and Kosowski [65], as improved by Barenboim, Elkin, and Goldenberg [20]. These improvements do not have consequences for

[29] In the case of MIS, the subproblems actually have size $\operatorname{poly}(\Delta)\log n$, but satisfy the additional property that they contain distance-5 dominating sets of size $O(\log n)$, which is often just as good as having $\operatorname{poly}\log(n)$ size. See [23, §3] or [68, §4] for more discussion of this.

	Randomized		Deterministic	
Upper Bounds	$O(\mathsf{Det}_d(\text{poly}\log n))$	**new**	$O(\sqrt{\Delta \log \Delta} \log^* \Delta + \log^* n)$	[20, 65]
	$O(\sqrt{\log \Delta} + \log\log n + \mathsf{Det}_d(\text{poly}\log n))$	[86]	$O(\sqrt{\Delta} \log^{5/2} \Delta + \log^* n)$	[65]
	$O(\log \Delta + \mathsf{Det}_d(\text{poly}\log n))$	[23]	$O(\Delta^{3/4} \log \Delta + \log^* n)$	[17]
	$O(\log \Delta + \sqrt{\log n})$	[135]	$O(\Delta + \log^* n)$	[21]
	$O(\Delta \log\log n)$	[103]	$O(\Delta \log \Delta + \log^* n)$	[103]
	$O(\log n)$	[3, 92, 108]	$O(\Delta \log n)$	[12]
			$O(\Delta^2 + \log^* n)$	[80, 107]
			$O(\Delta^{O(\Delta)} + \log^* n)$	[79]
			$2^{O(\sqrt{\log n})}$	[120]
			$2^{O(\sqrt{\log n \log\log n})}$	[12]
Lower Bounds	$\Omega(\log^* n)$	[115]	$\Omega(\log^* n)$	[107]
	$\Omega(\mathsf{Det}(\sqrt{\log n}))$	Theorem 2.1		

Table 6.1: Development of lower and upper bounds for distributed $(\Delta+1)$-list coloring in the LOCAL model. The terms $\mathsf{Det}(n')$ and $\mathsf{Det}_d(n')$ are the deterministic complexities of $(\Delta+1)$-list coloring and $(\deg+1)$-list coloring on n'-vertex graphs. All algorithms listed, except for [86] and ours, also solve the $(\deg+1)$-list coloring problem.

randomized coloring algorithms using graph shattering [23, 86] since we can only assume $\Delta = (\log n)^{\Omega(1)}$ in the shattered instances. See Table 6.1 for a summary of lower and upper bounds for distributed $(\Delta+1)$-list coloring in the LOCAL model.

In this chapter we prove that $(\Delta+1)$-list coloring can be solved in $O(\mathsf{Det}_d(\text{poly}\log n))$ time w.h.p. Our algorithm's performance is best contrasted with the $\Omega(\mathsf{Det}(\text{poly}\log n))$ randomized lower bound of Theorem 2.1, where Det is the deterministic complexity of $(\Delta+1)$-list coloring. Despite the syntactic similarity between the $(\deg+1)$- and $(\Delta+1)$-list coloring problems, there is no hard evidence showing their complexities are the same, asymptotically. On the other hand, every deterministic algorithmic technique developed for $(\Delta+1)$-list coloring applies equally well to $(\deg+1)$-list coloring [12, 17, 20, 65, 120]. In particular, there is only *one* tool that yields upper bounds in terms of n (independent of Δ), and that is network decompositions [12, 120].

Intellectually, our algorithm builds on a succession of breakthroughs by Schneider and Wattenhofer [135], Barenboim, Elkin, Pettie, and Schneider [23], Elkin, Pettie, and Su, [55], and Harris, Schneider, and Su [86], which we shall now review.

6.1.1 Fast Coloring using Excess Colors

Schneider and Wattenhofer [135] gave the first evidence that the canonical coloring problems may not be subject to the KMW lower bounds. They showed that for any constants $\epsilon > 0$ and $\gamma > 0$, when $\Delta \geq \log^{1+\gamma} n$ and the palette size is $(1+\epsilon)\Delta$, vertex coloring can be solved w.h.p. in just $O(\log^* n)$ time [135, Corollary 14]. The emergence of this log-star behavior in [135] is quite natural. Consider the case where the palette size of each vertex is at least $k\Delta$, where $k \geq 2$. Suppose each vertex v selects $k/2$ colors at random from its palette. A vertex v can successfully color itself if one of its selected colors is not selected by any neighbor in $N(v)$. The total number of colors selected by vertices in $N(v)$ is at most $k\Delta/2$. Therefore, the probability that a color selected by v is also selected by someone in $N(v)$ is at most $1/2$, so v successfully colors itself with probability at least $1 - 2^{-k/2}$. In expectation, the degree of any vertex after this coloring procedure is at most $\Delta' = \Delta/2^{k/2}$. In contrast, the number of *excess colors*, i.e., the current available palette size minus the number of uncolored neighbors, is non-decreasing over time. It is still at least $(k-1)\Delta = (k-1)2^{k/2}\Delta'$ Intuitively, repeating the above procedure for $O(\log^* n)$ rounds suffices to color all vertices.

Similar ideas have also been applied in other papers [55, 135]. However, for technical reasons, we cannot directly apply the results in these papers. The main difficulty in our setting is that we need to deal with *oriented* graphs with widely varying out-degrees, palette sizes, and excess colors; the guaranteed number of excess colors at a vertex depends on its out-degree, *not* the global parameter Δ.

Lemma 6.1 summarizes the properties of our ultrafast coloring algorithm when each vertex has many excess colors; its proof appears in Section 6.4. Recall that $\Psi(v)$ denotes the palette of v, so $|\Psi(v)| - \deg(v)$ is the number of excess colors at v.

Lemma 6.1. *Consider a directed acyclic graph, where vertex v is associated with a parameter $p_v \leq |\Psi(v)| - \deg(v)$ We write $p^\star = \min_{v \in V} p_v$. Suppose that there is a number $C = \Omega(1)$ such that all vertices v satisfy $\sum_{u \in N_{\text{out}}(v)} 1/p_u \leq 1/C$. Let $d^\star$ be the maximum out-degree of the graph. There is an algorithm that takes $O\left(1 + \log^* p^\star - \log^* C\right)$ time and achieves the following. Each vertex v remains uncolored with probability at most $\exp(-\Omega(\sqrt{p^\star})) + d^\star \exp(-\Omega(p^\star))$. This is true even if the random bits generated outside a constant radius around v are determined adversarially.*

We briefly explain the intuition underlying Lemma 6.1. Consider the following coloring procedure. Each vertex selects $C/2$ colors from its available colors randomly. Vertex v

successfully colors itself if at least one of its selected colors is not in conflict with any color selected by vertices in $N_{\text{out}}(v)$. For each color c selected by v, the probability that c is also selected by some vertex in $N_{\text{out}}(v)$ is $(C/2)\sum_{u \in N_{\text{out}}(v)} 1/p_u \leq 1/2$. Therefore, the probability that v still remains uncolored after this procedure is $\exp(-\Omega(C))$, improving the gap between the number of excess colors and the out-degree (i.e., the parameter C) exponentially. We are done after repeating this procedure for $O(1+\log^* p^\star - \log^* C)$ rounds. Lemma 6.2 is a more user-friendly version of Lemma 6.1 for simpler situations.

Lemma 6.2. *Suppose $|\Psi(v)| \geq (1+\rho)\Delta$ for each vertex v, and $\rho = \Omega(1)$. There is an algorithm that takes $O(1 + \log^* \Delta - \log^* \rho)$ time and achieves the following. Each vertex v remains uncolored with probability at most $\exp(-\Omega(\sqrt{\rho\Delta}))$. This is true even if the random bits generated outside a constant radius around v are determined adversarially.*

Proof. We apply Lemma 6.1. Orient the graph arbitrarily, and then set $p_v = \rho\Delta$ for each v. Use the parameters $C = \rho$, $p^\star = \rho\Delta$, and $d^\star = \Delta$. The time complexity is $O(1 + \log^* p^\star - \log^* C) = O(1 + \log^* \Delta - \log^* \rho\})$. The failure probability is $\exp(-\Omega(\sqrt{p^\star})) + d^\star \exp(-\Omega(p^\star)) = \exp(-\Omega(\sqrt{\rho\Delta}))$. □

6.1.2 Gaining Excess Colors

Schneider and Wattenhofer [135] illustrated that vertex coloring can be performed very quickly, given enough excess colors. However, in the $(\Delta+1)$-list coloring problem there *is just one excess color* initially, so the problem is how to *create* them. Elkin, Pettie, and Su [55] observed that if the graph induced by $N(v)$ is not too dense, then v can obtain a significant number of excess colors after *one* iteration of the following simple random coloring routine. Each vertex v, with probability 1/5, selects a color c from its palette $\Psi(v)$ uniformly at random; then vertex v successfully colors itself by c if c is not chosen by any vertex in $N(v)$. Intuitively, if $N(v)$ is not too close to a clique, then a significant number of *pairs* of vertices in the neighborhood $N(v)$ get assigned the same color. Each such pair effectively reduces v's palette size by 1 but its degree by 2, thereby increasing the number of excess colors at v by 1.

There are many *global* measures of sparsity, such as arboricity and degeneracy. We are aware of two locality-sensitive ways to measure it: the $(1-\epsilon)$-*local sparsity* of [4, 55, 110, 143], and the ϵ-*friends* from [86], defined formally as follows.

Definition 6.1 ([55]). *A vertex v is $(1-\epsilon)$-locally sparse if the subgraph induced by $N(v)$ has at most $(1-\epsilon)\binom{\Delta}{2}$ edges; otherwise v is $(1-\epsilon)$-locally dense.*

Definition 6.2 ([86]). *An edge $e = \{u, v\}$ is an ϵ-friend edge if $|N(u) \cap N(v)| \geq (1-\epsilon)\Delta$. We call u an ϵ-friend of v if $\{u, v\}$ is an ϵ-friend edge. A vertex v is ϵ-dense if v has at least $(1-\epsilon)\Delta$ ϵ-friends, otherwise it is ϵ-sparse.*

Throughout this chapter, we only use Definition 6.2. Lemma 6.3 shows that in $O(1)$ time we can create excess colors at all locally sparse vertices.

Lemma 6.3. *Consider the $(\Delta+1)$-list coloring problem. There is an $O(1)$-time algorithm that colors a subset of vertices such that the following is true for each $v \in V$ with $\deg(v) \geq (5/6)\Delta$.*

(i) With probability $1 - \exp(-\Omega(\Delta))$, the number of uncolored neighbors of v is at least $\Delta/2$.

(ii) With probability $1 - \exp(-\Omega(\epsilon^2\Delta))$, v has at least $\Omega(\epsilon^2\Delta)$ excess colors, where ϵ is the highest value such that v is ϵ-sparse.

The algorithm behind Lemma 6.3 is the random coloring routine described above. If a vertex v is ϵ-sparse, then there must be $\Omega(\epsilon^2\Delta^2)$ pairs of vertices $\{u, w\} \subseteq N(v)$ such that $\{u, w\}$ is <u>not</u> an edge. If $|\Psi(u) \cap \Psi(w)| = \Omega(\Delta)$,[30] then the probability that both u and v are colored by the same color is $\Omega(1/\Delta)$, so the expected number of excess colors created at v is at least $\Omega\left(\frac{\epsilon^2\Delta^2}{\Delta}\right) = \Omega(\epsilon^2\Delta)$.

Similar but slightly weaker lemmas were proved in [55, 86]. The corresponding lemma from [55] does not apply to *list* coloring, and the corresponding lemma from [86] obtains a high probability bound only if $\epsilon^4\Delta = \Omega(\log n)$. Optimizing this requirement is of importance, since this is the threshold about how locally sparse a vertex needs to be in order to obtain excess colors. The proof of Lemma 6.3, which is not the main contribution of this work, appears in Section 6.6.

The notion of local sparsity is especially useful for addressing the $(2\Delta-1)$-edge coloring problem [55], since it can be phrased as $(\Delta'+1)$-vertex coloring the *line graph* ($\Delta' = 2\Delta-2$), which is everywhere $(\frac{1}{2} + o(1))$-locally sparse and is also everywhere $(\frac{1}{2} - o(1))$-sparse.

[30] If the condition is not met, then we have $|(\Psi(u) \cup \Psi(w)) \setminus \Psi(v)| = \Omega(\Delta)$, and so with constant probability one of u and v successfully colors itself with a color not in $\Psi(v)$, and this also increases the number of excess colors at v by 1.

6.1.3 Coloring Locally Dense Vertices

In the vertex coloring problem we cannot count on any kind of local sparsity, so the next challenge is to make local *density* also work to our advantage. Harris, Schneider, and Su [86] developed a remarkable new graph decomposition that can be computed in $O(1)$ rounds of communication. The decomposition takes a parameter ϵ, and partitions the vertices into an ϵ-sparse set, and several vertex-disjoint ϵ-*dense* components induced by the ϵ-friend edges, each with weak diameter at most 2.

Based on this decomposition, they designed a $(\Delta+1)$-list coloring algorithm that takes $O(\sqrt{\log\Delta} + \log\log n + \mathsf{Det}_d(\operatorname{poly}\log n)) = O(\sqrt{\log\Delta}) + 2^{O(\sqrt{\log\log n})}$ time. We briefly overview their algorithm, as follows.

Coloring ϵ-Sparse Vertices. By utilizing the excess colors, Harris et al. [86] showed that the ϵ-sparse set can be colored in $O(\log\epsilon^{-1} + \log\log n + \mathsf{Det}_d(\operatorname{poly}\log n))$ time using techniques in [55] and [23]. More specifically, they applied the algorithm of [55, Corollary 4.1] using the $\epsilon'\Delta = \Omega(\epsilon^2\Delta)$ excess colors, i.e., $\epsilon' = \Theta(\epsilon^2)$. This takes $O\left(\log(\epsilon^{-1})\right) + T\left(n, O\left(\frac{\log^2 n}{\epsilon'}\right)\right)$ time, where $T(n',\Delta') = O(\log\Delta' + \log\log n' + \mathsf{Det}_d(\operatorname{poly}\log n'))$ is the time complexity of the $(\deg+1)$-list coloring algorithm of [23, Theorem 5.1] on n'-vertex graphs of maximum degree Δ'.

Coloring ϵ-Dense Vertices. For ϵ-dense vertices, Harris et al. [86] proved that by coordinating the coloring decisions within each dense component, it takes only $O(\log_{1/\epsilon}\Delta + \log\log n + \mathsf{Det}_d(\operatorname{poly}\log n))$ time to color the dense sets, i.e., the bound *improves* as $\epsilon \to 0$. The time for the overall algorithm is minimized by choosing $\epsilon = \exp(-\Theta(\sqrt{\log\Delta}))$.

The algorithm for coloring ϵ-dense vertices first applies $O(\log_{1/\epsilon}\Delta)$ iterations of *dense coloring steps* to reduce the maximum degree to $\Delta' = O(\log n)\cdot 2^{O\left(\log_{1/\epsilon}\Delta\right)}$, and then apply the $(\deg+1)$-list coloring algorithm of [23, Theorem 5.1] to color the remaining vertices in $O(\log\Delta' + \log\log n + \mathsf{Det}_d(\operatorname{poly}\log n)) = O(\log_{1/\epsilon}\Delta + \log\log n + \mathsf{Det}_d(\operatorname{poly}\log n))$ time.

In what follows, we informally sketch the idea behind the dense coloring steps. To finish in $O(\log_{1/\epsilon}\Delta)$ iterations, it suffices that the maximum degree is reduced by a factor of $\epsilon^{-\Omega(1)}$ in each iteration. Consider an ϵ-dense vertex v in a component S induced by the ϵ-friend edges. Harris et al. [86] proved that the number of ϵ-dense neighbors of v that are not in S is at most $\epsilon\Delta$. Intuitively, if we let each dense component output a random coloring that has no conflict within the component, then the probability that the color choice of a

vertex $v \in S$ is in conflict with an *external neighbor* of v is $O(\epsilon)$. Harris et al. [86] showed that this intuition can be nearly realized, and they developed a coloring procedure that is able to reduce the maximum degree by a factor of $O(\sqrt{\epsilon^{-1}})$ in each iteration.

6.1.4 New Results

In this chapter we give a fast randomized algorithm for $(\Delta + 1)$-vertex coloring. It is based on a hierarchical version of the Harris-Schneider-Su decomposition with $\log\log\Delta - O(1)$ levels determined by an increasing sequence of sparsity thresholds $(\epsilon_1, \ldots, \epsilon_\ell)$, with $\epsilon_i = \sqrt{\epsilon_{i+1}}$. Following [86], we begin with a single iteration of the *initial coloring step* (Lemma 6.3), in which a constant fraction of the vertices are colored. The guarantee of this procedure is that any vertex v at the ith layer (which is ϵ_i-dense but ϵ_{i-1}-sparse), has $\Omega(\epsilon_{i-1}^2\Delta)$ *pairs* of vertices in its neighborhood $N(v)$ assigned the same color, thereby creating that many excess colors in the palette of v.

At this point, the most natural way to proceed is to apply a Harris-Schneider-Su style *dense coloring step* to each layer, with the hope that each will take roughly constant time. Recall that (i) any vertex v at the ith layer already has $\Omega(\epsilon_{i-1}^2\Delta)$ excess colors, and (ii) the dense coloring step reduces the maximum degree by a factor of $\epsilon^{-\Omega(1)}$ in each iteration. Thus, in $O\left(\log_{1/\epsilon_i} \frac{\Delta}{\epsilon_{i-1}^{2.5}\Delta}\right) = O(1)$ time we should be able to create a situation where any uncolored vertices have $O(\epsilon_{i-1}^{2.5}\Delta)$ uncolored neighbors but $\Omega(\epsilon_{i-1}^2\Delta)$ excess colors in their palette. With such a large gap, a Schneider-Wattenhofer style coloring algorithm (Lemma 6.2) should complete in very few additional steps.

It turns out that in order to color ϵ_i-dense components efficiently, we need to maintain relatively large *lower bounds* on the available palette and relatively small *upper bounds* on the number of external neighbors (i.e., the neighbors outside the ϵ_i-dense component). Thus, it is important that when we first consider a vertex, we have not already colored too many of its neighbors. Roughly speaking, our algorithm classifies the dense blocks at layer i into *small*, *medium*, and *large* based chiefly on the block size, and partitions the set of all blocks of all layers into $O(1)$ *groups*. We apply the dense coloring steps *in parallel* for all blocks in the same group. Whenever we process a block B, we need to make sure that all its vertices have a large enough palette. For large blocks, the palette size guarantee comes from the lower bound on the block size. For small and medium blocks, the palette size guarantee comes from the ordering of the blocks being processed; we will show that whenever a small or medium block B is considered, each vertex $v \in B$ has a sufficiently

large number of neighbors that have yet to be colored.

All of the coloring steps outlined above finish in $O(\log^* \Delta)$ time. The bottleneck procedure is the algorithm of Lemma 6.2, and the rest takes only $O(1)$ time. Each of these coloring steps may not color all vertices it considers. The vertices left uncolored are put in $O(1)$ classes, each of which either induces a bounded degree graph or is composed of $O(\operatorname{poly}\log n)$-size components, w.h.p. The former type can be colored deterministically in $O(\log^* n)$ time and the latter in $\mathsf{Det}_d(\operatorname{poly}\log n)$ time. In view of Linial's lower bound [107] we have $\mathsf{Det}_d(\operatorname{poly}\log n) = \Omega(\log^* n)$ and the running time of our $(\Delta+1)$-list coloring algorithm is

$$O(\log^* \Delta) + O(\log^* n) + O(\mathsf{Det}_d(\operatorname{poly}\log n)) = O(\mathsf{Det}_d(\operatorname{poly}\log n)).$$

Recent Developments. After the initial publication of this work, our algorithm was adapted to solve $(\Delta+1)$-coloring in several other models of computation, namely the *congested clique*, the *MPC*[31] model, and the *centralized local computation* model [11, 37, 124, 125]. Chang, Fischer, Ghaffari, Uitto, and Zheng [37], improving [124, 125], showed that $(\Delta+1)$-coloring can be solved in the congested clique in $O(1)$ rounds, w.h.p. In the MPC model, Assadi, Chen, and Khanna [11] solve $(\Delta+1)$-coloring in $O(1)$ rounds using $\tilde{O}(n)$ memory per machine, whereas Chang et al. [37] solve it in $O(\sqrt{\log\log n})$ time with just $O(n^\epsilon)$ memory per machine. In the centralized local computation model, Chang et al. [37] proved that $(\Delta+1)$-coloring queries can be answered with just *polynomial* probe complexity $\Delta^{O(1)}\log n$.

Organization. In Section 6.2 we define a hierarchical decomposition based on [86]. Section 6.3 gives a high-level description of the algorithm, which uses a variety of coloring routines whose guarantees are specified by the following lemmas.

- Lemma 6.1 analyzes the procedure ColorBidding, which is a generalization of the Schneider-Wattenhofer coloring routing; it is proved in Section 6.4.
- Lemma 6.3 shows that the procedure OneShotColoring creates many excess colors; it is proved in Section 6.6.
- Lemmas 6.7–6.10 analyze two versions of an algorithm DenseColoringStep, which is a

[31] massively parallel computation

generalization of the Harris-Schneider-Su routine [86] for coloring locally dense vertices; they are proved in Section 6.5.

6.2 Hierarchical Decomposition

In this section, we extend the work of Harris, Schneider, and Su [86] to define a hierarchical decomposition of the vertices based on local sparsity. Let $G = (V, E)$ be the input graph, Δ be the maximum degree, and $\epsilon \in (0, 1)$ be a parameter. An edge $e = \{u, v\}$ is an *ϵ-friend edge* if $|N(u) \cap N(v)| \geq (1 - \epsilon)\Delta$. We call u an *ϵ-friend* of v if $\{u, v\}$ is an *ϵ-friend edge*. A vertex v is called *ϵ-dense* if v has at least $(1 - \epsilon)\Delta$ ϵ-friends, otherwise it is *ϵ-sparse*. Observe that it takes one round of communication to tell whether each edge is an ϵ-friend, and hence one round for each vertex to decide if it is ϵ-sparse or ϵ-dense.

We write V_ϵ^{s} (and V_ϵ^{d}) to be the set of ϵ-sparse (and ϵ-dense) vertices. Let v be a vertex in a set $S \subseteq V$ and $V' \subseteq V$. Define $\bar{d}_{S,V'}(v) = |(N(v) \cap V') \setminus S|$ to be the *external degree* of v with respect to S and V', and $a_S(v) = |S \setminus (N(v) \cup \{v\})|$ to be the *anti-degree* of v with respect to S. A connected component C of the subgraph formed by the ϵ-dense vertices and the ϵ-friend edges is called an *ϵ-almost clique*. This term makes sense in the context of Lemma 6.4 from [86], which summarizes key properties of almost cliques.

Lemma 6.4 ([86])**.** *Fix any $\epsilon < 1/5$. The following conditions are met for each ϵ-almost clique C, and each vertex $v \in C$.*

(i) $\bar{d}_{C,V_\epsilon^{\mathsf{d}}}(v) \leq \epsilon\Delta$. *(Small external degree w.r.t. ϵ-dense vertices.)*

(ii) $a_C(v) < 3\epsilon\Delta$. *(Small anti-degree.)*

(iii) $|C| \leq (1 + 3\epsilon)\Delta$. *(Small size, a consequence of (ii).)*

(iv) $\mathrm{dist}_G(u, v) \leq 2$ *for each $u, v \in C$. (Small weak diameter.)*

Lemma 6.4(iv) implies that *any* sequential algorithm operating solely on C can be simulated in $O(1)$ rounds in the LOCAL model. The node in C with minimum ID can gather all the relevant information from C in 2 rounds of communication, compute the output of the algorithm locally, and disseminate these results in another 2 rounds of communication. For example, the DenseColoringStep algorithm (versions 1 and 2) presented in Section 6.5 are nominally sequential algorithms but can be implemented in $O(1)$ distributed rounds.

6.2.1 A Hierarchy of Almost Cliques

Throughout this section, we fix some increasing sequence of sparsity parameters $(\epsilon_1, \ldots, \epsilon_\ell)$ and a subset of vertices $V^\star \subseteq V$, which, roughly speaking, are those left uncolored by the initial coloring procedure of Lemma 6.3 and also satisfy the two conclusions of Lemma 6.3(i,ii). The sequence $(\epsilon_1, \ldots, \epsilon_\ell)$ always adheres to Definition 6.3.

Definition 6.3. *A sequence $(\epsilon_1, \ldots, \epsilon_\ell)$ is a valid* sparsity sequence *if the following conditions are met: (i) $\epsilon_i = \sqrt{\epsilon_{i-1}} = \epsilon_1^{2^{-(i-1)}}$, and (ii) $\epsilon_\ell \leq 1/K$ for some sufficiently large K.*

Layers. Define $V_1 = V^\star \cap V^{\mathsf{d}}_{\epsilon_1}$ and $V_i = V^\star \cap (V^{\mathsf{d}}_{\epsilon_i} \setminus V^{\mathsf{d}}_{\epsilon_{i-1}})$, for $i > 1$. Define $V_{\mathsf{sp}} = V^\star \cap V^{\mathsf{s}}_{\epsilon_\ell} = V^\star \setminus (V_1 \cup \cdots \cup V_\ell)$. It is clear that $(V_1, \ldots, V_\ell, V_{\mathsf{sp}})$ is a partition of $V^\star$. We call V_i the *layer-i* vertices, and call V_{sp} the *sparse vertices*. In other words, V_i is the subset of $V^\star$ that are ϵ_i-dense but ϵ_{i-1}-sparse. Remember that the definition of sparsity is with respect to the entire graph $G = (V, E)$ not the subgraph induced by $V^\star$.

Blocks. The layer-i vertices V_i are partitioned into *blocks* as follows. Let $\{C_1, C_2, \ldots\}$ be the set of ϵ_i-almost cliques, and let $B_j = C_j \cap V_i$. Then $(B_1, B_2, \ldots)$ is a partition of V_i. Each $B_j \neq \emptyset$ is called a *layer-i block*. See Figure 6.1 for an illustration; the shaded region indicates a layer-i block B and the hollow regions are those ϵ_{i-1}-almost cliques.

A layer-i block B is a *descendant* of a layer-i' block B', $i < i'$, if B and B' are both subsets of the same $\epsilon_{i'}$-almost clique. Therefore, the set of all blocks in all layers naturally forms a rooted tree $\mathcal{T}$, where the root represents V_{sp}, and every other node represents a block in some layer. For example, in Figure 6.1, the blocks contained in $C_1, \ldots, C_k$ are at layers $1, \ldots, i-1$, and are all descendants of B.

6.2.2 Block Sizes and Excess Colors

We classify the blocks into three types: *small*, *medium*, and *large*. A block B at layer i is called *large-eligible* if

$$|B| \geq \frac{\Delta}{\log(1/\epsilon_i)}.$$

Large blocks. The set of large blocks is a maximal set of unrelated, large-eligible blocks, which prioritizes blocks by size, breaking ties by layer. More formally, a large-eligible

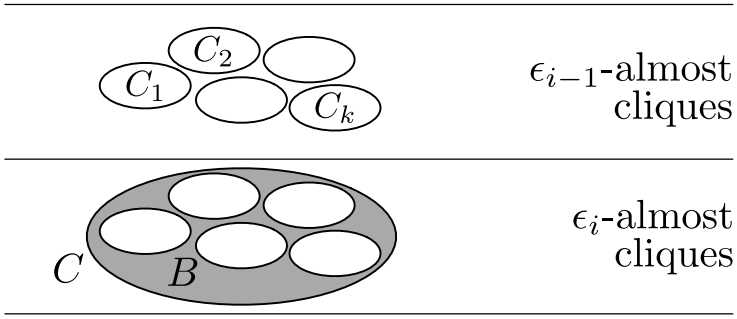

Figure 6.1: Almost-cliques and blocks.

layer-i block B is large if and only if, for every large-eligible B' at layer j that is an ancestor or descendant of B, either $|B'| < |B|$ or $|B'| = |B|$ and $j < i$.

Medium blocks. Every large-eligible block that is not large is a medium block.

Small blocks. All other blocks are small.

Define V_i^{S}, V_i^{M}, and V_i^{L}, to be, respectively, the sets of all vertices in layer-i small blocks, layer-i medium blocks, and layer-i large blocks. For each $X \in \{\mathsf{S}, \mathsf{M}, \mathsf{L}\}$, we write $V_{2+}^X = \bigcup_{i=2}^{\ell} V_i^X$ to be the set of all vertices of type X, excluding those in layer 1.

Overview of Our Algorithm. The decomposition and $\mathcal{T}$ are trivially computed in $O(1)$ rounds of communication. The first step of our algorithm is to execute an $O(1)$-round coloring procedure (OneShotColoring) which colors a small constant fraction of the vertices in G; the relevant guarantees of this algorithm were stated in Lemma 6.3. Let $V^\star$ be the subset of uncolored vertices that, in addition, satisfy the conclusions of Lemma 6.3(i,ii). Once $V^\star$ is known, it can be partitioned into the following sets

$$\left(V_1^{\mathsf{S}}, \ldots, V_\ell^{\mathsf{S}}, V_1^{\mathsf{M}}, \ldots, V_\ell^{\mathsf{M}}, V_1^{\mathsf{L}}, \ldots, V_\ell^{\mathsf{L}}, V_{\mathsf{sp}}\right)$$

These are determined by the hierarchical decomposition with respect to a particular sparsity sequence $(\epsilon_1, \ldots, \epsilon_\ell)$.[32] We color the vertices of $V^\star \setminus V_{\mathsf{sp}}$ in six stages according to the

[32]Note that the classification of vertices into small, medium, and large blocks can only be done *after* OneShotColoring is complete. Recall that if C is an ϵ_i-almost clique, $B = C \cap V_i$ is the subset of C that is both ϵ_{i-1}-sparse and uncolored by OneShotColoring. Thus, whether the layer-i block in C is large-eligible depends on how many vertices are successfully colored.

ordering

$$\left(V_{2+}^{\mathsf{S}}, V_1^{\mathsf{S}}, V_{2+}^{\mathsf{M}}, V_1^{\mathsf{M}}, V_{2+}^{\mathsf{L}}, V_1^{\mathsf{L}}\right).$$

As we argue below, coloring vertices in the order small, medium, large ensures that when a vertex is considered, it has *sufficiently many* remaining colors in its palette, as formalized by Lemma 6.5 below. The reason for dealing with layer-1 vertices separately stems from the fact that a vertex at layer $i > 1$ is known to be ϵ_i-dense but ϵ_{i-1}-sparse, but layer-1 vertices are not known to have any non-trivial sparsity. At the end of this process a small portion of vertices $U \subseteq V^\star \setminus V_{\mathsf{sp}}$ may remain uncolored. However, they all have sufficiently large palettes such that $U \cup V_{\mathsf{sp}}$ can be colored efficiently in $O(\log^* n)$ time.

Lemma 6.5. *For each layer $i \in [1, \ell]$, the following is true.*

- *For each $v \in V_i^{\mathsf{S}}$ with $|N(v) \cap V^\star| \geq \Delta/3$, we have $|N(v) \cap (V_{2+}^{\mathsf{M}} \cup V_1^{\mathsf{M}} \cup V_{2+}^{\mathsf{L}} \cup V_1^{\mathsf{L}} \cup V_{\mathsf{sp}})| \geq \frac{\Delta}{4}$.*
- *For each $v \in V_i^{\mathsf{M}}$, we have $|N(v) \cap (V_{2+}^{\mathsf{L}} \cup V_1^{\mathsf{L}} \cup V_{\mathsf{sp}})| \geq \frac{\Delta}{2\log(1/\epsilon_i)}$.*

In other words, regardless of how we proceed to *partially* color the vertices in small blocks, each $v \in V_i^{\mathsf{S}}$ always has at least $\frac{\Delta}{4}$ available colors in its palette, due to the number of its (still uncolored) neighbors in medium and large blocks, and V_{sp}. Similarly, regardless of how we partially color the vertices in small and medium blocks, each $v \in V_i^{\mathsf{M}}$ always has at least $\frac{\Delta}{2\log(1/\epsilon_i)}$ available colors in its palette.

Before proving Lemma 6.5 we first establish a useful property that constrains the structure of the block hierarchy $\mathcal{T}$. Intuitively, Lemma 6.6 shows that a node (block) in $\mathcal{T}$ can have exactly one child of essentially any size, but if it has two or more children then the union of all strict descendants must be very small.

Lemma 6.6. *Let C be an ϵ_i-almost clique and $C_1, \ldots, C_l$ be the ϵ_{i-1}-almost cliques contained in C. Either $l = 1$ or $\sum_{j=1}^{l} |C_j| \leq 2(3\epsilon_i + \epsilon_{i-1})\Delta$. In particular, if B is the layer-i block contained in C, either B has one child in $\mathcal{T}$ or the number of vertices in all strict descendants of B is at most $2(3\epsilon_i + \epsilon_{i-1})\Delta < 7\epsilon_i\Delta$.*

Proof. Suppose, for the purpose of obtaining a contradiction, that $l \geq 2$ and $\sum_{j=1}^{l} |C_j| > 2(3\epsilon_i + \epsilon_{i-1})\Delta$. Without loss of generality, suppose C_1 is the smallest, so $\sum_{j=2}^{l} |C_j| > (3\epsilon_i + \epsilon_{i-1})\Delta$. Any $v \in C_1$ is ϵ_{i-1}-dense and therefore has at least $(1 - \epsilon_{i-1})\Delta$ neighbors that are ϵ_{i-1}-friends. By the anti-degree property of Lemma 6.4, v is adjacent to all but at most $3\epsilon_i\Delta$ vertices in C. Thus, by the pigeonhole principle v is joined by edges to more

than $\epsilon_{i-1}\Delta$ members of $C_2 \cup \cdots \cup C_l$. By the pigeonhole principle again, at least one of these edges is one of the ϵ_{i-1}-friend edges incident to v. This means that C_1 *cannot* be a connected component in the graph formed by ϵ_{i-1}-dense vertices and ϵ_{i-1}-friend edges. □

Proof of Lemma 6.5. First consider the case of $v \in V_i^{\mathsf{M}}$. Let B be the layer-i medium block containing v. Every medium block is large-eligible but not large, meaning it must have a large ancestor or descendant B' with at least as many vertices. If B' is a layer-j block, then

$$|B'| \;=\; \max\{|B'|\,,\,|B|\} \;\geq\; \frac{\Delta}{\log(1/\epsilon_k)}, \quad \text{where } k = \max\{i,j\}.$$

Let C be the layer-k almost clique containing both B and B'. By Lemma 6.4, v has at most $3\epsilon_k\Delta$ non-neighbors in C, which, since $B' \subseteq C$, means that the number of neighbors of v in B' is at least

$$\begin{aligned}
|B'| - 3\epsilon_k\Delta &\geq \frac{\Delta}{\log(1/\epsilon_k)} - 3\epsilon_k\Delta \\
&\geq \frac{\Delta}{2\log(1/\epsilon_k)} && \{\epsilon_k \leq \epsilon_\ell \text{ sufficiently small}\} \\
&\geq \frac{\Delta}{2\log(1/\epsilon_i)} && \{\log(1/\epsilon_k) \leq \log(1/\epsilon_i)\}
\end{aligned}$$

Therefore, $|N(v) \cap (V_{2+}^{\mathsf{L}} \cup V_1^{\mathsf{L}} \cup V_{\mathsf{sp}})| \geq \frac{\Delta}{2\log(1/\epsilon_i)}$.

Now consider any vertex $v \in V_i^{\mathsf{S}}$ with $|N(v) \cap V^\star| \geq \Delta/3$. Let B be the layer-i small block containing v. We partition the set $N(v) \cap V^\star$ into three groups $A_1 \cup A_2 \cup A_3$.

$A_1 = N(v) \cap \left(V_{2+}^{\mathsf{M}} \cup V_1^{\mathsf{M}} \cup V_{2+}^{\mathsf{L}} \cup V_1^{\mathsf{L}} \cup V_{\mathsf{sp}}\right)$.

$A_2 =$ the neighbors in all ancestor and descendant small blocks of B, including B.

$A_3 =$ the remaining neighbors.

To prove the lemma, it suffices to show that $|A_1| \geq \frac{\Delta}{4}$. Since $|A_1 \cup A_2 \cup A_3| \geq \frac{\Delta}{3}$, we need to prove $|A_2 \cup A_3| \leq \frac{\Delta}{12}$. We first bound $|A_3|$, then $|A_2|$.

Note that v is ϵ_j-dense for $j \in [i,\ell]$, so, according to Lemma 6.4, v must have at least $(1-\epsilon_j)\Delta$ ϵ_j-friends. Let u be any neighbor of v not in an ancestor/descendant of B, which means that either (i) $u \in V_{\mathsf{sp}}$ or (ii) for some $j \in [i,\ell]$, v and u are in distinct ϵ_j-almost cliques. In case (i) u is counted in A_1. In case (ii) it follows that u *cannot* be an ϵ_j-friend

of v. Since, by Lemma 6.4, v has at most $\epsilon_j\Delta$ ϵ_j-non-friends,

$$|A_3| \leq \sum_{j=i}^{\ell} \epsilon_j \Delta < 2\epsilon_\ell \Delta.$$

We now turn to A_2. Define $i^\star \in [1, i-1]$ to be the largest index such that B has at least two descendants at layer $i^\star$, or let $i^\star = 0$ if no such index exists. Let $A_{2,\text{low}}$ be the set of vertices in A_2 residing in blocks at layers $1, \ldots, i^\star$, and let $A_{2,\text{high}} = A_2 \setminus A_{2,\text{low}}$. By the definition of small blocks,

$$\begin{aligned} |A_{2,\text{high}}| &< \sum_{j=i^\star+1}^{\ell} \frac{\Delta}{\log(1/\epsilon_j)} \\ &< \frac{2\Delta}{\log(1/\epsilon_\ell)}. && \{\text{geometric sum}\} \end{aligned}$$

If $i^\star = 0$ then $A_{2,\text{low}} = \emptyset$. Otherwise, by Lemma 6.6, the number of vertices in $A_{2,\text{low}}$ is at most $7\epsilon_{i^\star+1}\Delta \leq 7\epsilon_i\Delta \leq 7\epsilon_\ell\Delta$. Since ϵ_ℓ is a sufficiently small constant,

$$|A_2 \cup A_3| < 2\epsilon_\ell\Delta + \frac{2\Delta}{\log(1/\epsilon_\ell)} + 7\epsilon_\ell\Delta < \Delta/12,$$

which completes the proof. □

Remark 6.1. *The reader might wonder why the definition of medium blocks is needed, as all layer-i medium blocks already have the block size lower bound $\frac{\Delta}{\log(1/\epsilon_j)}$, which guarantees a sufficiently large palette size lower bound for the vertices therein. It might be possible to consider all the medium blocks as large blocks, but this will destroy the property that for any two blocks B and B' in different layers, if B is a descendant of B', then B and B' cannot both be large; without this property, the coloring algorithm for large blocks will likely be more complicated.*

6.3 Main Algorithm

Our algorithm follows the *graph shattering* framework [23]. In each step of the algorithm, we specify an invariant that all vertices must satisfy in order to continue to participate. Those *bad vertices* that violate the invariant are removed from consideration; they form connected

components of size $O(\text{poly}\log n)$ w.h.p., so we can color them later in $\mathsf{Det}_d(\text{poly}\log n)$ time. More precisely, the emergence of the small components is due to the shattering lemma [23, 61] (Lemma 1.2).

As we will see, some parts of our randomized algorithm consist of $t = O(\log^* \Delta)$ steps, and so whether a vertex v is bad actually depends on the random bits in its radius-t neighborhood. Nonetheless, we are still able to apply Lemma 1.2. The reason is that we are able to show that, for any specified constant k, each vertex v becomes bad in one particular step with probability at most Δ^{-k}, and this is true regardless of the outcomes in all previous steps and the choices of random bits outside of a constant radius of v.

Sparsity Sequence. The sparsity sequence for our algorithm is defined by $\epsilon_1 = \Delta^{-1/10}$, $\epsilon_i = \sqrt{\epsilon_{i-1}}$ for $i > 1$, and $\ell = \log\log\Delta - O(1)$ is the largest index such that $\frac{1}{\epsilon_\ell} \geq K$ for some sufficiently large constant K.

6.3.1 Initial Coloring Step

At any point in time, the number of *excess colors* at v is the size of v's remaining palette minus the number of v's uncolored neighbors. This quantity is obviously non-decreasing over time. In the first step of our coloring algorithm, we execute the algorithm of Lemma 6.3, which in $O(1)$ time colors a portion of the vertices. This algorithm has the property that each remaining uncolored vertex gains a certain number of excess colors, which depends on its local sparsity. In order to proceed a vertex must satisfy both conditions.

- If v is ϵ_ℓ-dense, the number of uncolored neighbors of v is at least $\Delta/2$.
- if v is ϵ_i-sparse, v must have $\Omega(\epsilon_i^2\Delta)$ excess colors.

If either condition fails to hold, v is put in the set V_{bad}. We invoke the conditions of Lemma 6.3 only with $\epsilon \geq \epsilon_1 = \Delta^{-1/10}$. Thus, if $\Delta = \Omega(\log^2 n)$, then with high probability (i.e., $1 - 1/\text{poly}(n)$), $V_{\mathsf{bad}} = \emptyset$. Otherwise, each component of V_{bad} must, by Lemma 1.2, have size $O(\text{poly}(\Delta) \cdot \log n) = O(\text{poly}\log n)$, w.h.p. We do not invoke a deterministic algorithm to color V_{bad} just yet. In subsequent steps of the algorithm, we will continue to add bad vertices to V_{bad}. These vertices will be colored at the end of the algorithm.

6.3.2 Coloring Vertices by Layer

By definition, $V^\star$ is the set of all vertices that remain uncolored after the initial coloring step *and* are not put in V_{bad}. The partition $V^\star = V_{2+}^{\mathsf{S}} \cup V_1^{\mathsf{S}} \cup V_{2+}^{\mathsf{M}} \cup V_1^{\mathsf{M}} \cup V_{2+}^{\mathsf{L}} \cup V_1^{\mathsf{L}} \cup V_{\mathsf{sp}}$ is computed in $O(1)$ time. In this section, we show how we can color *most* of the vertices in $V_{2+}^{\mathsf{S}} \cup V_1^{\mathsf{S}} \cup V_{2+}^{\mathsf{M}} \cup V_1^{\mathsf{M}} \cup V_{2+}^{\mathsf{L}} \cup V_1^{\mathsf{L}}$, in that order, leaving a small portion of uncolored vertices.

Consider the moment we begin to color V_{2+}^{S}. We claim that each layer-i vertex $v \in V_{2+}^{\mathsf{S}}$ must have at least $\Delta/6 > \frac{\Delta}{2\log(1/\epsilon_i)}$ *excess colors w.r.t.* V_{2+}^{S}. That is, its palette size minus the number of its neighbors in V_{2+}^{S} is large. There are two relevant cases to consider.

- If the condition $|N(v) \cap V^\star| \geq \Delta/3$ in Lemma 6.5 is already met, then v has at least $\Delta/4 > \Delta/6$ excess colors w.r.t. V_{2+}^{S}.
- Suppose $|N(v) \cap V^\star| < \Delta/3$. One criterion for adding v to V_{bad} is that v is ϵ_ℓ-dense but has less than $\Delta/2$ uncolored neighbors after the initial coloring step. We know v is ϵ_ℓ-dense and not in V_{bad} (because it is in V_{2+}^{S}), so it must have had at least $\Delta/2$ uncolored neighbors after initial coloring. If $|N(v) \cap V^\star| < \Delta/3$ then at least $(\Delta/2 - \Delta/3) = \Delta/6$ of v's uncolored neighbors must have joined V_{bad}, which provide v with $\Delta/6$ excess colors w.r.t. V_{2+}^{S}.

Similarly, for the sets V_1^{S}, V_{2+}^{M}, and V_1^{M}, we have the same excess colors guarantee $\frac{\Delta}{2\log(1/\epsilon_i)}$ for each layer-i vertex therein.

We apply the following lemmas to color the locally dense vertices $V^\star \setminus V_{\mathsf{sp}}$; refer to Section 6.5 for their proofs. For small and medium blocks, we use Lemma 6.7 to color V_{2+}^{S} and V_{2+}^{M}, and use Lemma 6.8 to color V_1^{S} and V_1^{M}.

The reason that the layer-1 blocks need to be treated differently is that layer-1 vertices do not obtain excess colors from the initial coloring step (Lemma 6.3). For comparison, for $i > 1$, each layer-i vertex v is ϵ_{i-1}-sparse, and so v must have $\Omega(\epsilon_{i-1}^2\Delta) = \Omega(\epsilon_i^4\Delta)$ excess colors. If we reduce the degree of v to $\epsilon_i^5\Delta$, then we obtain a sufficiently big gap between the excess colors and degree at v.

Lemma 6.7 (Small and medium blocks; layers other than 1)**.** *Let $S = V_{2+}^{\mathsf{S}}$ or $S = V_{2+}^{\mathsf{M}}$. Suppose that each layer-i vertex $v \in S$ has at least $\frac{\Delta}{2\log(1/\epsilon_i)}$ excess colors w.r.t. S. There is an $O(1)$-time algorithm that colors a subset of S meeting the following condition. For each vertex $v \in V^\star$, and for each $i \in [2, \ell]$, with probability at least $1 - \exp(-\Omega(\text{poly}(\Delta)))$, the number of uncolored layer-i neighbors of v in S is at most $\epsilon_i^5\Delta$. Vertices that violate this property join the set V_{bad}.*

Lemma 6.8 (Small and medium blocks; layer 1)**.** *Let $S = V_1^{\mathsf{S}}$ or $S = V_1^{\mathsf{M}}$. Suppose that each vertex $v \in S$ has at least $\frac{\Delta}{2\log(1/\epsilon_1)}$ excess colors w.r.t. S. There is an $O(1)$-time algorithm that colors a subset of S meeting the following condition. Each $v \in S$ is colored with probability at least $1 - \exp(-\Omega(\mathrm{poly}(\Delta)))$; all uncolored vertices in S join V_{bad}.*

The following lemmas consider large blocks. Lemma 6.9 colors V_{2+}^{L} and has guarantees similar to Lemma 6.7, whereas Lemma 6.10 colors nearly all of V_1^{L} and partitions the remaining uncolored vertices among three sets, R, X, and V_{bad}, with certain guarantees.

Lemma 6.9 (Large blocks; layer other than 1)**.** *There is an $O(1)$-time algorithm that colors a subset of V_{2+}^{L} meeting the following condition. For each $v \in V^\star$ and each layer number $i \in [2, \ell]$, with probability at least $1 - \exp(-\Omega(\mathrm{poly}(\Delta)))$, the number of uncolored layer-i neighbors of v in V_{2+}^{L} is at most $\epsilon_i^5 \Delta$. Vertices that violate this property join the set V_{bad}.*

Remember that our goal is to show that the bad vertices V_{bad} induce connected components of size $O(\mathrm{poly}\log n)$. However, if in a randomized procedure each vertex is added to V_{bad} with probability $1 - 1/\mathrm{poly}(\Delta)$, then the shattering lemma only guarantees that the size of each connected component of V_{bad} is $O(\mathrm{poly}(\Delta)\log n)$, which is not necessarily $\mathrm{poly}\log n$. This explains why Lemma 6.10 has two types of guarantees.

Lemma 6.10 (Large blocks; layer 1)**.** *Let c be any sufficiently large constant. Then there is a constant time (independent of c) algorithm that colors a subset of V_1^{L} while satisfying one of the following cases.*

- *The uncolored vertices of V_1^{L} are partitioned among R or V_{bad}. The graph induced by R has degree $O(c^2)$; each vertex joins V_{bad} with probability $\Delta^{-\Omega(c)}$.*
- *If $\Delta > \log^{\alpha c} n$, where $\alpha > 0$ is some universal constant, then the uncolored vertices of V_1^{L} are partitioned among R and X, where the graph induced by R has degree $O(c^2)$ and the components induced by X have size $\log^{O(c)} n$, w.h.p.*

In our $(\Delta+1)$-list coloring algorithm, we apply Lemmas 6.7, 6.8, 6.9, and 6.10 to color the vertices in $V^\star \setminus V_{\mathsf{sp}}$, and they are processed in this order: $(V_{2+}^{\mathsf{S}}, V_1^{\mathsf{S}}, V_{2+}^{\mathsf{M}}, V_1^{\mathsf{M}}, V_{2+}^{\mathsf{L}}, V_1^{\mathsf{L}})$.

Coloring the Leftover Vertices X and R. Notice that the algorithm for Lemma 6.10 generates a leftover uncolored subset R which induces a constant-degree subgraph, and (in case $\Delta > \log^{\Theta(c)} n$) a leftover uncolored subset X where each connected component

has size at most $O(\text{poly}\log n)$. Remember that the vertices in R and X do not join V_{bad}. All vertices in X are colored deterministically in $\mathsf{Det}_d(\text{poly}\log n)$ time; the vertices in R are colored deterministically in $O(\text{poly}(\Delta') + \log^* n) = O(\log^* n)$ time [20, 65, 107], with $\Delta' = O(c^2) = O(1)$.

The Remaining Vertices. Any vertex in $V^\star$ that violates at least one condition specified in the lemmas is added to the set V_{bad}. All remaining uncolored vertices join the set U. In other words, U is the set of all vertices in $V^\star \setminus (V_{\mathsf{sp}} \cup V_{\mathsf{bad}} \cup R \cup X)$ that remain uncolored after applying the lemmas.

6.3.3 Coloring the Remaining Vertices

At this point all uncolored vertices are in $U \cup V_{\mathsf{sp}} \cup V_{\mathsf{bad}}$. We show that $U \cup V_{\mathsf{sp}}$ can be colored efficiently in $O(\log^* \Delta)$ time using Lemma 6.1, then consider V_{bad}.

Coloring the Vertices in U. Let G' be the directed acyclic graph induced by U, where all edges are oriented from the sparser to the denser endpoint. In particular, an edge $e = \{u, u'\}$ is oriented as (u, u') if u is at layer i, u' at layer i', and $i > i'$, or if $i = i'$ and $\text{ID}(u) > \text{ID}(u')$. We write $N_{\text{out}}(v)$ to denote the set of out-neighbors of v in G'.

For each layer-i vertex v in G', and each layer j, the number of layer-j neighbors of v in G' is at most $O(\epsilon_j^5 \Delta)$, due to Lemmas 6.7 and 6.9. The out-degree of v is therefore at most $\sum_{j=1}^{i} \epsilon_j^5 \Delta = O(\epsilon_i^5 \Delta) = O(\epsilon_{i-1}^{5/2} \Delta)$.

We write $\Psi(v)$ to denote the set of available colors of v. The number of excess colors at v is $|\Psi(v)| - \deg(v) = \Omega(\epsilon_{i-1}^2 \Delta)$. Thus, there is an $\Omega(1/\sqrt{\epsilon_{i-1}})$-factor gap between the palette size of v and the out-degree of v.

Lemma 6.1 is applied to color nearly all vertices in U in $O(\log^* \Delta)$ time, with any remaining uncolored vertices added to V_{bad}. We use the following parameters of Lemma 6.1. In view of the above, there exists a constant $\eta > 0$ such that, for each $i \in [2, \ell]$ and each layer-i vertex v in G', we set $p_v = \eta \epsilon_{i-1}^2 \Delta \leq |\Psi(v)| - \deg(v)$. There is a constant $C > 0$ such that for each $i \in [2, \ell]$ and each layer-i vertex $v \in U$, we have:

$$\sum_{u \in N_{\text{out}}(v)} 1/p_u \leq \sum_{j=2}^{i} O\left(\frac{\epsilon_{j-1}^{5/2} \Delta}{\epsilon_{j-1}^2 \Delta}\right) = \sum_{j=2}^{i} O(\epsilon_{j-1}^{1/2}) < 1/C.$$

The remaining parameters to Lemma 6.1 are

$$p^\star = \eta\epsilon_1^2\Delta = \Omega(\Delta^{8/10}), \quad d^\star = \Delta, \quad C = \Omega(1).$$

Thus, by Lemma 6.1 the probability that a vertex still remains uncolored (and is added to V_{bad}) after the algorithm is

$$\exp(-\Omega(\sqrt{p^\star})) + d^\star \exp(-\Omega(p^\star)) = \exp(-\Omega(\Delta^{2/5})).$$

Coloring the Vertices in V_{sp}. The set V_{sp} can be colored in a similar way using Lemma 6.1. We let G'' be any acyclic orientation of the graph induced by V_{sp}, e.g., orienting each edge $\{u, v\}$ towards the vertex v such that $\mathrm{ID}(u) > \mathrm{ID}(v)$. The number of available colors of each $v \in V_{\mathsf{sp}}$ minus its out-degree is at least $\Omega(\epsilon_\ell^2\Delta)$, which is at least $\gamma\Delta$, for some constant $\gamma > 0$, according to the way we select the sparsity sequence. We define $p_v = \gamma\Delta < |\Psi(v)| - \deg(v)$. We have $\sum_{u \in N_{\mathrm{out}}(v)}(1/p_u) \leq \mathrm{outdeg}(v)/(\gamma\Delta) \leq 1/\gamma$. Thus, we can apply Lemma 6.1 with $C = \gamma$. Notice that both $p^\star$ and $d^\star$ are $\Theta(\Delta)$, and so the probability that a vertex still remains uncolored after the algorithm (and is added to V_{bad}) is $\exp(-\Omega(\sqrt{\Delta}))$.

Coloring the Vertices in V_{bad}. At this point, all remaining uncolored vertices are in V_{bad}. If $\Delta \gg \operatorname{poly}\log n$, then $V_{\mathsf{bad}} = \emptyset$, w.h.p., in view of the failure probabilities $\exp(-\Omega(\operatorname{poly}(\Delta)))$ specified in the lemmas used in the previous coloring steps. Otherwise, $\Delta = O(\operatorname{poly}\log n)$, and by Lemma 1.2, each connected component of V_{bad} has size at most $\operatorname{poly}(\Delta)\log n = O(\operatorname{poly}\log n)$. In any case, it takes $\mathsf{Det}_d(\operatorname{poly}\log n)$ to color all vertices in V_{bad} deterministically.

See Figure 6.2 for a synopsis of every step of the $(\Delta+1)$-list coloring algorithm. Steps 1, 2, and 3(a–f) take constant time. Steps 4, 6, and 7 take $O(\log^* n) = O(\mathsf{Det}_d(\operatorname{poly}\log n))$ time [107, 115]. The bottleneck in the algorithm are Steps 5 and 8, which take $O(\mathsf{Det}_d(\operatorname{poly}\log n))$ time. The algorithm succeeds in the prescribed time, so long as the input to Steps 4, 5, and 8 are as they should be, i.e., inducing subgraphs with constant degree, or $\operatorname{poly}\log n$-size components, respectively. (These are instances of $(\deg+1)$-list coloring.) When $\Delta \gg \operatorname{poly}\log n$ is sufficiently large, the set V_{bad} is empty, w.h.p., but X may be non-empty, and induce components with size $\operatorname{poly}\log n$.

$(\Delta+1)$-List Coloring Algorithm

1. Determine the ϵ-almost cliques, for $\epsilon \in \{\epsilon_1, \ldots, \epsilon_\ell\}$. (Lemma 6.4.)
2. Perform the initial coloring step using algorithm OneShotColoring (Lemma 6.3) and partition the remaining uncolored vertices into $V^\star$ and V_{bad}. Further partition $V^\star$ into a sparse set V_{sp} and a hierarchy $\mathcal{T}$ of small, medium, and large blocks. Partition $V^\star \backslash V_{\mathsf{sp}}$ into 6 sets: $V_{2+}^{\mathsf{S}}, V_1^{\mathsf{S}}, V_{2+}^{\mathsf{M}}, V_1^{\mathsf{M}}, V_{2+}^{\mathsf{L}}, V_1^{\mathsf{L}}$.
3. Color most of $V_{2+}^{\mathsf{S}}, V_1^{\mathsf{S}}, V_{2+}^{\mathsf{M}}, V_1^{\mathsf{M}}, V_{2+}^{\mathsf{L}}, V_1^{\mathsf{L}}$ in six steps.
   a) Color a subset of V_{2+}^{S} using algorithm DenseColoringStep (version 1). Any vertices that violate the conclusion of Lemma 6.7 are added to V_{bad}.
   b) Color V_1^{S} using algorithm DenseColoringStep (version 1). Any remaining uncolored vertices are added to V_{bad} (Lemma 6.8).
   c) Color a subset of V_{2+}^{M} using algorithm DenseColoringStep (version 1). Any vertices that violate the conclusion of Lemma 6.7 are added to V_{bad}.
   d) Color V_1^{M} using algorithm DenseColoringStep (version 1). Any remaining uncolored vertices are added to V_{bad} (Lemma 6.8).
   e) Color a subset of V_{2+}^{L} using algorithm DenseColoringStep (version 2). Any vertices that violate the conclusion of Lemma 6.9 are added to V_{bad}.
   f) Color V_1^{L} using algorithm DenseColoringStep (version 2). Each remaining uncolored vertex is added to one of X, R, or V_{bad}. (See Lemma 6.10.)
4. W.h.p. R induces a graph with constant maximum degree. Color R in $O(\log^* n)$ time deterministically using a standard algorithm [20, 65, 107].
5. W.h.p. X induces a graph whose components have size $\operatorname{poly}\log n$. Color X in $O(\mathsf{Det}_d(\operatorname{poly}\log n))$ time deterministically; see [120].
6. Color those uncolored vertices U in $\left(V_{2+}^{\mathsf{S}} \cup V_{2+}^{\mathsf{M}} \cup V_{2+}^{\mathsf{L}}\right) \backslash V_{\mathsf{bad}}$ in $O(\log^* \Delta)$ time using algorithm ColorBidding (Lemma 6.1). Any vertices in U that remain uncolored are added to V_{bad}.
7. Color V_{sp} in $O(\log^* \Delta)$ time using algorithm ColorBidding (Lemma 6.1). Any vertices that remain uncolored are added to V_{bad}.
8. W.h.p. V_{bad} induces components of size $\operatorname{poly}\log n$. Color V_{bad} in $O(\mathsf{Det}_d(\operatorname{poly}\log n))$ time deterministically; see [120].

Figure 6.2: The $(\Delta+1)$-list coloring algorithm.

6.3.4 Time Complexity

The time for OneShotColoring (Fig. 6.2, Step 2) is $O(1)$. The time for processing each of V_{2+}^{S}, V_1^{S}, V_{2+}^{M}, V_1^{M}, V_{2+}^{L}, V_1^{L} (Steps 3(a–f)) is $O(1)$. Observe that each of Steps 2 and 3(a–f) may put vertices in V_{bad}, that Steps 3(a,c,e) leave some vertices uncolored, and that Step 3(f) also puts vertices in special sets X and R. With high probability, R induces components with constant degree, which can be colored deterministically in $O(\log^* n)$ time (Step 4). The uncolored vertices (U) from Steps 3(a,c,e) have a large gap between their palette size and degree, and can be colored in $O(\log^* \Delta)$ time using the ColorBidding algorithm (Lemma 6.1) in Step 6. The same type of palette size-degree gap exists for V_{sp} as well so ColorBidding colors it in $O(\log^* \Delta)$ time; for Step 7 we are applying Lemma 6.1 again, but with different parameters.

Finally, Steps 5 and 8 solve a $(\deg+1)$-list coloring problem on a graph whose components have size $\operatorname{poly}\log n$. Observe that V_{bad} is guaranteed to induce components with size $\operatorname{poly}(\Delta)\log n$, which happens to be $\operatorname{poly}\log n$ since no vertices are added to V_{bad}, w.h.p., if $\Delta \gg \operatorname{poly}\log n$ is sufficiently large. In contrast, in Step 5 X can be non-empty even when Δ is large, but it still induces components with size $\operatorname{poly}\log n$.

Since $\log^* \Delta \leq \log^* n = O(\mathsf{Det}_d(\operatorname{poly}\log n))$ [107], the bottleneck in the algorithm is solving $(\deg+1)$-list coloring, in Steps 5 and 8.

Theorem 6.1. *In the* LOCAL *model, the* $(\Delta+1)$*-list coloring problem can be solved, w.h.p., in* $O(\mathsf{Det}_d(\operatorname{poly}\log n))$ *time, where* $\mathsf{Det}_d(n')$ *is the deterministic complexity of* $(\deg+1)$*-list coloring on* n'*-vertex graphs.*

Next, we argue that if the palettes have $\operatorname{poly}\log n$ extra colors initially, we can list color the graph in $O(\log^* \Delta)$ time.

Theorem 6.2. *There is a universal constant* $\gamma > 0$ *such that the* $(\Delta + \log^\gamma n)$*-list coloring problem can be solved in the* LOCAL *model, w.h.p., in* $O(\log^* \Delta)$ *time.*

Proof. For all parts of our $(\Delta + 1)$-list coloring algorithm, except the first case of Lemma 6.10, the probability that a vertex v joins V_{bad} is $\exp(-\Omega(\operatorname{poly}(\Delta)))$. Let α and c be the constants in Lemma 6.10 and $k_1 = \Theta(c) \geq \alpha c$ be such that if $\Delta > \log^{k_1} n$, then the probability that a vertex v joins V_{bad} in our $(\Delta + 1)$-list coloring algorithm is $\exp(-\Omega(\operatorname{poly}(\Delta))) = 1/\operatorname{poly}(n)$. Note that when $\Delta > \log^{k_1} n$, no vertex is added to V_{bad} in Lemma 6.10.

Let $R' = R \cup X$ be the leftover vertices in Lemma 6.10 for the case $\Delta > \log^{k_1} n$. There exists a constant $k_2 > 0$ such that the subgraph induced by R' has maximum degree $\log^{k_2} n$. We set $\gamma = \max\{k_1, k_2\} + 1$. Now we show how to solve the $(\Delta + \log^\gamma n)$-list coloring problem in $O(\log^* \Delta)$ time.

If $\Delta \leq \log^{\gamma-1} n$, then we apply the algorithm of Lemma 6.2 directly, with $\rho = \frac{\log^\gamma n}{\Delta} - 1 = \Omega(\log n)$. The algorithm takes $O(1 + \log^* \Delta - \log^* \rho) = O(1)$ time, and the probability that a vertex v is not colored is $\exp(-\Omega(\sqrt{\rho\Delta})) = \exp(-\Omega(\log^{\gamma/2} n)) \ll 1/\text{poly}(n)$.

If $\Delta > \log^{\gamma-1} n$, then we apply Steps 1,2,3,6, and 7 of our $(\Delta+1)$-list coloring algorithm. Due to the lower bound on Δ, we have $V_{\mathsf{bad}} = \emptyset$, w.h.p., which obviates the need to implement Step 8.

This algorithm takes $O(\log^* \Delta)$ time, and produces an uncolored subgraph $R' = R \cup X$ that has maximum degree $\Delta' \leq \log^{k_2} n$. In lieu of Steps 4 and 5, we apply the algorithm of Lemma 6.2 to color R' in $O(1 + \log^* \Delta' - \log^* \rho) = O(1)$ time, where $\rho = \frac{\log^\gamma n}{\Delta'} - 1 = \Omega(\log n)$. □

If every vertex is ϵ-sparse, with $\epsilon^2\Delta$ sufficiently large, then the algorithm of Lemma 6.3 gives *every* vertex $\Omega(\epsilon^2\Delta)$ excess colors, w.h.p. Combining this observation with Theorem 6.2, we have the following result, which shows that the $(\Delta + 1)$-list coloring problem can be solved very efficiently when all vertices are sufficiently locally sparse.

Theorem 6.3. *There is a universal constant $\gamma > 0$ such that the following holds. Suppose G is a graph with maximum degree Δ in which each vertex is ϵ-sparse, where $\epsilon^2\Delta > \log^\gamma n$. A $(\Delta + 1)$-list coloring of G can be computed in the* LOCAL *model, w.h.p., in $O(\log^* \Delta)$ time.*

Remark 6.2. *Theorem 6.3 insists on every vertex being* ϵ-sparse *according to Definition 6.2. It is straightforward to show connections between this definition of sparsity and others standard measures from the literature. For example, such a graph is* $(1-\epsilon')$-locally sparse*, where $\epsilon' = \Omega(\epsilon^2)$, according to Definition 6.1. Similarly, any $(1-\epsilon')$-locally sparse graph is $\Omega(\epsilon')$-sparse. Graphs of* degeneracy $d \leq (1-\epsilon')\Delta$ *or* arboricity $\lambda \leq (1/2 - \epsilon')\Delta$ *are trivially $(1 - \Omega(\epsilon'))$-locally sparse.*

Remark 6.3. *We have made no effort to minimize the constant γ in Theorems 6.2 and 6.3, and it is impractically large. It would be useful to know if these theorems remain true when γ is small, say 1, i.e., is $(\Delta + \log n)$-coloring solvable in $O(\log^* \Delta)$ time, w.h.p.?*

6.4 Fast Coloring using Excess Colors

In this section, we prove Lemma 6.1. Consider a directed acyclic graph $G = (V, E)$, where each vertex v has a palette $\Psi(v)$. Each vertex v is associated with a parameter $p_v \leq |\Psi(v)| - \deg(v)$, i.e., p_v is a lower bound on the number of excess colors at v. All vertices agree on values $p^\star \leq \min_{v \in V} p_v$, $d^\star \geq \max_{v \in V} \mathrm{outdeg}(v)$, and $C = \Omega(1)$, such that the following is satisfied for all v.

$$\sum_{u \in N_{\mathrm{out}}(v)} 1/p_u \leq 1/C. \tag{6.1}$$

Intuitively, the sum $\sum_{u \in N_{\mathrm{out}}(v)} 1/p_u$ measures the amount of "contention" at a vertex v. In the ColorBidding algorithm each vertex v selects each color $c \in \Psi(v)$ with probability $\frac{C}{2|\Psi(v)|} < \frac{C}{2p_v}$ and permanently colors itself if it selects a color not selected by any out-neighbor.

Procedure ColorBidding.

1. Each color $c \in \Psi(v)$ is added to S_v independently with probability $\frac{C}{2|\Psi(v)|}$.
2. If there exists a color $c^\star \in S_v \setminus \left(\bigcup_{u \in N_{\mathrm{out}}(v)} S_u \right)$, v permanently colors itself $c^\star$.

In Lemma 6.11 we present an analysis of ColorBidding. We show that after an iteration of ColorBidding, the amount of "contention" at a vertex v decreases by (roughly) an $\exp(C/6)$-factor, with very high probability.

Lemma 6.11. *Consider an execution of ColorBidding. Let v be any vertex. Let D be the summation of $1/p_u$ over all vertices u in $N_{\mathrm{out}}(v)$ that remain uncolored after ColorBidding. Then the following holds.*

$$\Pr[\ v \text{ remains uncolored }] \leq \exp(-C/6) + \exp(-\Omega(p^\star)).$$
$$\Pr[D \geq (1+\lambda)\exp(-C/6)/C] \leq \exp\left(-2\lambda^2 p^\star \exp(-C/3)/C\right) + d^\star \exp(-\Omega(p^\star)).$$

Proof. For each vertex v, we define the following two events.

E_v^{good} : v selects a color that is not selected by any vertex in $N_{\mathrm{out}}(v)$.

E_v^{bad} : the number of colors in $\Psi(v)$ that are selected by some vertices in $N_{\text{out}}(v)$ is at least $\frac{2}{3} \cdot |\Psi(v)|$.

Notice that E_v^{good} is the event where v successfully colors itself. We first show that $\Pr[E_v^{\text{bad}}] = \exp(-\Omega(p^\star))$. Fix any color $c \in \Psi(v)$. The probability that c is selected by some vertex in $N_{\text{out}}(v)$ is

$$1 - \prod_{u \in N_{\text{out}}(v)} \left(1 - \tfrac{C}{2|\Psi(u)|}\right) \leq 1 - \prod_{u \in N_{\text{out}}(v)} \left(1 - \tfrac{C}{2p_u}\right) \leq \sum_{u \in N_{\text{out}}(v)} \tfrac{C}{2p_u} \leq \tfrac{1}{2},$$

where the last inequality follows from (6.1). Since these events are independent for different colors, $\Pr[E_v^{\text{bad}}] \leq \Pr[\text{Binomial}(n', p') \geq \frac{2n'}{3}]$ with $n' = |\Psi(v)| \geq p_v$ and $p' = \frac{1}{2}$. By a Chernoff bound, we have:

$$\Pr\left[E_u^{\text{bad}}\right] \leq \exp(-\Omega(n'p')) = \exp(-\Omega(p^\star)).$$

Conditioned on $\overline{E_v^{\text{bad}}}$, v will color itself unless it fails to choose *any* of $|\Psi(v)|/3$ specific colors from its palette. Thus,

$$\Pr\left[\overline{E_v^{\text{good}}} \;\middle|\; \overline{E_v^{\text{bad}}}\right] \leq \left(1 - \tfrac{C}{2|\Psi(v)|}\right)^{|\Psi(v)|/3} \leq \exp\left(\tfrac{-C}{6}\right). \tag{6.2}$$

We are now in a position to prove the first inequality of the lemma. The probability that v remains uncolored is at most $\Pr\left[\overline{E_v^{\text{good}}} \;\middle|\; \overline{E_v^{\text{bad}}}\right] + \Pr\left[E_v^{\text{bad}}\right]$, which is at most $\exp(-C/6) + \exp(-\Omega(p^\star))$.

Next, we prove the second inequality, on the upper tail of the random variable D. Let $N_{\text{out}}(v) = (u_1, \ldots, u_k)$. Let E_i^{bad} and E_i^{good} be short for $E_{u_i}^{\text{bad}}$ and $E_{u_i}^{\text{good}}$, and let $\mathcal{E}$ be the event $\bigcup_i E_i^{\text{bad}}$. By a union bound,

$$\Pr[\mathcal{E}] \leq \text{outdeg}(v) \cdot \exp(-\Omega(p^\star)) \leq d^\star \cdot \exp(-\Omega(p^\star)).$$

Let $X = \sum_{i=1}^{k} X_i$, where $X_i = 1/p_{u_i}$ if *either* $\overline{E_i^{\text{good}}}$ or E_i^{bad} occurs, and $X_i = 0$ otherwise. Observe that if we condition on $\overline{\mathcal{E}}$, then X is exactly D, the random variable we want to bound.

By linearity of expectation,

$$\begin{aligned}
\mu = \mathrm{E}[X \mid \overline{\mathcal{E}}] &= \sum_i \mathrm{E}[X_i \mid \overline{\mathcal{E}}] \\
&\le \sum_i \frac{1}{p_{u_i}} \cdot \Pr\left[\overline{E_i^{\text{good}}} \;\middle|\; \overline{E_i^{\text{bad}}}\right] \\
&\le \sum_i \exp(-C/6)/p_{u_i} && \text{Equation (6.2)} \\
&\le \exp(-C/6)/C && \text{Equation (6.1)}
\end{aligned}$$

Each variable X_i is within the range $[a_i, b_i]$, where $a_i = 0$ and $b_i = 1/p_{u_i}$. We have $\sum_{i=1}^{k}(b_i - a_i)^2 \le \sum_{u \in N_{\text{out}}(v)} 1/(p_u \cdot p^\star) \le 1/(Cp^\star)$. By Hoeffding's inequality,[33] we have

$$\begin{aligned}
\Pr[X \ge (1+\lambda)\exp(-C/6)/C \mid \overline{\mathcal{E}}] &\le \Pr[X \ge (1+\lambda)\mu \mid \overline{\mathcal{E}}] \\
&\le \exp\left(\frac{-2(\lambda\mu)^2}{\sum_{i=1}^{k}(b_i - a_i)^2}\right) \\
&\le \exp\left(-2(\lambda \exp(-C/6)/C)^2 (p^\star C)\right) \\
&= \exp\left(-2\lambda^2 p^\star \exp(-C/3)/C\right).
\end{aligned}$$

Thus,

$$\begin{aligned}
\Pr[D \ge (1+\lambda)\exp(-C/6)/C] &\le \Pr[X \ge (1+\lambda)\exp(-C/6)/C \mid \overline{\mathcal{E}}] + \Pr[\mathcal{E}] \\
&\le \exp\left(-2\lambda^2 p^\star \exp(-C/3)/C\right) + d^\star \exp(-\Omega(p^\star)). \qquad \square
\end{aligned}$$

Proof of Lemma 6.1. In what follows, we show how Lemma 6.11 can be used to derive Lemma 6.1. Our plan is to apply ColorBidding for $k^\star = \log^* p^\star - \log^* C + O(1)$ iterations.

[33] The variables $\{X_1, \ldots, X_k\}$ are not independent, but we are still able to apply Hoeffding's inequality. The reason is as follows. Assume that $N_{\text{out}}(v) = (u_1, \ldots, u_k)$ is sorted in reverse topological order, and so for each $1 \le j \le k$, we have $N_{\text{out}}(u_j) \cap \{u_j, \ldots, u_k\} = \emptyset$. Thus, conditioning on (i) $\overline{E_i^{\text{bad}}}$ and (ii) *any* colors selected by vertices in $\bigcup_{1 \le j < i} N_{\text{out}}(u_j) \cup \{u_j\}$, the probability that $\overline{E_i^{\text{good}}}$ occurs is still at most $\exp(\frac{-C}{6})$.

For the kth iteration we use the parameter C_k, which is defined as follows:

$$\begin{aligned} C_1 &= \min\{\sqrt{p^\star},\ C\}, \\ C_k &= \min\left\{\sqrt{p^\star},\ \frac{C_{k-1}}{(1+\lambda)\exp(-C_{k-1}/6)}\right\} \\ k^\star &= \min\{k \mid C_k = \sqrt{p^\star}\} \qquad \text{(the last iteration)} \end{aligned}$$

Here $\lambda > 0$ must be selected to be a sufficiently small constant so that $(1+\lambda)\exp(-C_{k-1}/6) < 1$. This guarantees that the sequence (C_k) is strictly increasing. For example, if $C \geq 6$ initially, we can fix $\lambda = 1$ throughout.

We analyze each iteration of ColorBidding using the same (initial) vector of (p_v) values, i.e., we do not count on the number of excess colors at any vertex increasing over time.

At the *end* of the kth iteration, $k \in [1, k^\star]$, we have the following invariant $\mathcal{H}_k$ that we expect all vertices to satisfy:

- If $k \in [1, k^\star)$, $\mathcal{H}_k$ stipulates that for each uncolored vertex v after the kth iteration, the summation of $1/p_u$ over all uncolored $u \in N_{\text{out}}(v)$ is less than $1/C_{k+1}$.
- $\mathcal{H}_{k^\star}$ stipulates that all vertices are colored at the end of the $k^\star$th iteration.

The purpose of $\mathcal{H}_k$, $k \in [1, k^\star)$, is to guarantee that C_{k+1} is a valid parameter for the $(k+1)$th iteration of ColorBidding. For each $k \in [1, k^\star]$, at the end of the kth iteration we remove from consideration all vertices violating $\mathcal{H}_k$, and add them to the set V_{bad}. Thus, by definition of $\mathcal{H}_{k^\star}$, after the last iteration, all vertices other than the ones in V_{bad} have been colored.

To prove the lemma, it suffices to show that the probability of v joining V_{bad} is at most $\exp(-\Omega(\sqrt{p^\star})) + d^\star \exp(-\Omega(p^\star))$, and this is true even if the randomness outside a constant radius around v is determined adversarially. By Lemma 6.11, the probability that a vertex is removed at the end of the kth iteration, where $k \in [1, k^\star)$, is at most

$$\exp(\Omega(p^\star/C_{k+1})) + d^\star \exp(-\Omega(p^\star)) \leq \exp(-\Omega(\sqrt{p^\star})) + d^\star \exp(-\Omega(p^\star)).$$

The probability that a vertex is removed at the end of the $k^\star$th iteration is at most $\exp(-C_{k^\star}/6) + \exp(-\Omega(p^\star)) \leq \exp(-\Omega(\sqrt{p^\star}))$. By a union bound over all $k^\star = \log^* p^\star - \log^* C + O(1)$ iterations, the probability that a vertex joins V_{bad} is $\exp(-\Omega(\sqrt{p^\star})) + d^\star \exp(-\Omega(p^\star))$. □

6.5 Coloring Locally Dense Vertices

Throughout this section, we consider the following setting. We are given a graph $G = (V, E)$, where some vertices are already colored. We are also given a *subset* S of the uncolored vertices, which is partitioned into g disjoint *clusters* $S = S_1 \cup S_2 \cup \cdots \cup S_g$, each with weak diameter 2. (In particular, this implies that otherwise sequential algorithms can be executed on each cluster in $O(1)$ rounds in the LOCAL model.) Our goal is to color a large fraction of the vertices in S in only constant time.

We assume that the edges within S are oriented from the sparser to the denser endpoint, breaking ties by comparing IDs. In particular, an edge $e = \{u, u'\}$ is oriented as (u, u') if u is at layer i, u' at layer i', and $i > i'$, or if $i = i'$ and $\mathrm{ID}(u) > \mathrm{ID}(u')$. Notice that this orientation is *acyclic*. We write $N_{\mathrm{out}}(v) \subseteq S$ to denote the set of out-neighbors of v in S.

In Section 6.5.1 we describe a procedure DenseColoringStep (version 1) that is efficient when each vertex has many excess colors w.r.t. S. It is analyzed in Lemma 6.12, which is then used to prove Lemmas 6.7 and 6.8. In Section 6.5.2 we describe a procedure DenseColoringStep (version 2), which is a generalization of Harris, Schneider, and Su's [86] procedure. It is analyzed in Lemma 6.13, which is then used to prove Lemmas 6.9 and 6.10.

6.5.1 Case 1: Many Excess Colors are Available

In this section we focus on the case where each vertex $v \in S$ has many excess colors w.r.t. S. We make the following assumptions about the vertex set S.

Excess colors. Each $v \in S$ is associated with a parameter Z_v, which indicates a lower bound on the number of excess colors of v w.r.t. S. That is, the palette size of v minus $|N(v) \cap S|$ is at least Z_v.

External degree. For each cluster S_j, each vertex $v \in S_j$ is associated with a parameter D_v such that $|N_{\mathrm{out}}(v) \cap (S \setminus S_j)| \leq D_v$.

The ratio of these two quantities plays an important role in the analysis. Define δ_v as

$$\delta_v = D_v / Z_v.$$

We briefly explain how we choose the clustering $S = S_1 \cup S_2 \cup \cdots \cup S_g$ and set these parameters in the settings of Lemma 6.7 and Lemma 6.8. For Lemma 6.8, S is either V_1^{S}

or V_1^{M}, and each cluster of S is the intersection of S and an ϵ_1-almost clique (a layer-1 block). For Lemma 6.7, S is either V_{2+}^{S} or V_{2+}^{M}, and each cluster of S is the intersection of S and an ϵ_ℓ-almost clique. In all cases, clusters have weak diameter 2. All vertices in the same layer adopt the same D- and Z-values. A layer-i vertex v takes

$$Z_v = \frac{\Delta}{2\log(1/\epsilon_i)},$$
$$\text{and } D_v = \epsilon_i \Delta.$$

The choices of these parameters are valid in view of the excess colors implied by Lemma 6.5 and the external degree upper bound of Lemma 6.4.

Procedure DenseColoringStep (version 1).

1. Let $\pi : \{1, \ldots, |S_j|\} \to S_j$ be the unique permutation that lists S_j in increasing order by layer number, breaking ties (within the same layer) by ID. For q from 1 to $|S_j|$, the vertex $\pi(q)$ selects a color $c(\pi(q))$ uniformly at random from

$$\Psi(\pi(q)) \setminus \{c(\pi(q')) \mid q' < q \text{ and } \{\pi(q), \pi(q')\} \in E(G)\}.$$

2. Each $v \in S_j$ permanently colors itself $c(v)$ if $c(v)$ is not selected by any vertices in $N_{\text{out}}(v)$.

Notice that π is a reverse topological ordering of S_j, i.e., if $\pi(q')$ precedes $\pi(q)$, then $\pi(q) \notin N_{\text{out}}(\pi(q'))$. Because each S_j has weak diameter 2, Step 1 of DenseColoringStep can be simulated with only $O(1)$ rounds of communication. Intuitively, the probability that a vertex $v \in S$ remains uncolored after DenseColoringStep (version 1) is at most δ_v, since it is *guaranteed* not to have any conflicts with neighbors in the same cluster. The following lemma gives us the probabilistic guarantee of the DenseColoringStep (version 1).

Lemma 6.12. *Consider an execution of DenseColoringStep (version 1). Let T be any subset of S, and let $\delta = \max_{v \in T} \delta_v$. For any $t \geq 1$, the number of uncolored vertices in T is at least t with probability at most* $\Pr[\text{Binomial}(|T|, \delta) \geq t]$.

Proof. Let $T = \{v_1, \ldots, v_{|T|}\}$ be listed in increasing order by layer number, breaking ties by vertex ID. Remember that vertices in T can be spread across multiple clusters in S.

Imagine exposing the color choices of all vertices in S, one by one, in this order $v_1, \ldots, v_{|T|}$. The vertex v_k in cluster S_j will successfully color itself if it chooses any color not already selected by a vertex in $N_{\text{out}}(v_k) \cap (S \setminus S_j)$. Since $|N_{\text{out}}(v_k) \cap (S \setminus S_j)| \leq D_{v_k}$ and v_k has at least Z_{v_k} colors to choose from at this moment, the probability that it fails to be colored is at most $D_{v_k}/Z_{v_k} = \delta_{v_k} \leq \delta$, *independent of the choices made by higher priority vertices* $v_1, \ldots, v_{k-1}$. Thus, for any t, the number of uncolored vertices in T is stochastically dominated by the binomial variable Binomial$(|T|, \delta)$. □

Proof of Lemma 6.7. We execute DenseColoringStep (version 1) for 6 iterations, where each participating vertex $x \in S$ uses the same (initial) values of Z_x and D_x, namely $Z_x = \frac{\Delta}{2\log(1/\epsilon_i)}$ and $D_x = \epsilon_i \Delta$ if x is at layer i.

Consider any vertex $v \in V^\star$, and any layer number $i \in [2, \ell]$. Let T be the set of layer-i neighbors of v in S. To prove Lemma 6.7, it suffices to show that after 6 iterations of DenseColoringStep (version 1), with probability $1 - \exp(-\Omega(\text{poly}(\Delta)))$, the number of uncolored vertices in T is at most $\epsilon_i^5 \Delta$.

We define the following parameters.

$$\begin{aligned} \delta = \max_{u \in T}\{\delta_u\} \;&=\; 2\epsilon_i \log(1/\epsilon_i), \\ t_1 = |T|, \\ \text{and } t_k = \max\left\{(2\delta)t_{k-1}, \epsilon_i^5 \Delta\right\}&. \end{aligned}$$

Since $(2\delta)^6 |T| \leq \epsilon_i^5 \Delta$, we have $t_7 = \epsilon_i^5 \Delta$.

Assume that at the beginning of the kth iteration, the number of uncolored vertices in T is at most t_k. Indeed for $k = 1$, we initially have $t_1 = |T|$. By Lemma 6.12, after the kth iteration, the expected number of uncolored vertices in T is at most $\delta t_k \leq t_{k+1}/2$. By a Chernoff bound, with probability at most $\exp(-\Omega(t_{k+1})) \leq \exp(-\Omega(\epsilon_i^5 \Delta)) = \exp(-\Omega(\text{poly}(\Delta)))$, the number of uncolored vertices in T is more than t_{k+1}.

Therefore, after 6 iterations of DenseColoringStep (version 1), with probability $1 - \exp(-\Omega(\text{poly}(\Delta)))$, the number of uncolored vertices in T is at most $t_7 = \epsilon_i^5 \Delta$, as required. □

Proof of Lemma 6.8. In the setting of Lemma 6.8 we only consider layer-1 vertices, but have the higher burden of coloring *each* vertex with high enough probability. Since $\epsilon_1 = \Delta^{-1/10}$, we have $Z_v = \frac{\Delta}{2\log(1/\epsilon_1)}$, $D_v = \epsilon_1 \Delta$, and $\delta_v = D_v/Z_v = 2\epsilon_1 \log(1/\epsilon_1)$, for all vertices $v \in S$.

We begin with one iteration of DenseColoringStep (version 1). By Lemma 6.12 and a Chernoff bound, for each $v \in S$, the number of uncolored vertices of $N(v) \cap S$ is at most $2\delta_v \Delta = \Delta' < O(\Delta^{9/10} \log \Delta)$ with probability $1 - \exp(-\Omega(\text{poly}(\Delta)))$. Any uncolored vertex $v \in S$ that violates this property, i.e., for which $|N(v) \cap S| > \Delta'$, is added to V_{bad} and removed from further consideration.

Consider the graph G' induced by the remaining uncolored vertices in S. The maximum degree of G' is at most Δ'. Each vertex v in G' satisfies $|\Psi(v)| \geq Z_v = \frac{\Delta}{2\log(1/\epsilon_1)} = (1+\rho)\Delta'$, where ρ is $\Delta^{\Omega(1)}$. We run the algorithm of Lemma 6.2 on G', and then put all vertices that still remain uncolored to the set V_{bad}. By Lemma 6.2, the time for this procedure is $O(\log^* \Delta - \log^* \rho) = O(1)$, and the probability that a vertex v remains uncolored and is added to V_{bad} is at most $\exp(-\Omega(\sqrt{\rho\Delta})) = \exp(-\Omega(\text{poly}(\Delta)))$. □

6.5.2 Case 2: No Excess Colors are Available

In this section we focus on the case where there is no guarantee on the number of excess colors. The palette size lower bound of each vertex $v \in S_j$ comes from the assumption that $|S_j|$ is large, and v is adjacent to all but a very small portion of vertices in S_j. For the case $S = V_{2+}^{\mathsf{L}}$ (Lemma 6.9), each cluster S_j is a large block in some layer $i \in [2, \ell]$. For the case $S = V_1^{\mathsf{L}}$ (Lemma 6.10), each S_j is a layer-1 large block. For each $v \in S$, we define $N^\star(v)$ to be the set of all vertices $u \in N(v) \cap S$ such that the layer number of u is smaller than or equal to the layer number of v. Observe that $N_{\text{out}}(v) \subseteq N^\star(v)$ since $N_{\text{out}}(v)$ excludes some vertices at v's layer, depending on the ordering of IDs. For the case of $S = V_1^{\mathsf{L}}$, all clusters $S_1, \ldots, S_g$ are layer-1 blocks, and so $N^\star(v) = N(v) \cap S$. We make the following assumptions.

Identifiers. List the clusters $S_1, \ldots, S_g$ in non-decreasing order by layer number. We assume each cluster and each vertex within a cluster has an ID that is consistent with this order, in particular:

$$\begin{aligned} &\text{ID}(S_1) < \cdots < \text{ID}(S_g) \\ &\max_{v \in S_j} \text{ID}(v) < \min_{u \in S_{j+1}} \text{ID}(u), \text{ for all } j \in [1, g) \end{aligned}$$

Given arbitrary IDs, it is straightforward to compute new IDs satisfying these properties in $O(1)$ time. (It is not required that each cluster S_j to know the index j.)

Degree upper bounds. Each cluster S_j is associated with a parameter D_j such that all

$v \in S_j$ satisfy the following two conditions:

(i) $|S_j \setminus (N(v) \cup \{v\})| = |S_j \setminus (N^\star(v) \cup \{v\})| \leq D_j$ (anti-degree upper bound),

(ii) $|N^\star(v) \setminus S_j| \leq D_j$ (external degree upper bound).

Shrinking rate. Each cluster S_j is associated with a parameter δ_j such that

$$1/K \geq \delta_j \geq \frac{D_j \log(|S_j|/D_j)}{|S_j|},$$

for some sufficiently large constant K.

The procedure DenseColoringStep (version 2) aims to successfully color a large fraction of the vertices in each cluster S_j. In Step 1, each cluster selects a $(1-\delta_j)$-fraction of its vertices uniformly at random, permutes them randomly, and marches through this permutation one vertex at a time. As in DenseColoringStep (version 1), when a vertex v is processed it picks a random color $c(v)$ from its available palette that were not selected by previously processed vertices in S_j. Step 2 is the same: if $c(v)$ has not been selected by any neighbors of $N_{\text{out}}(v)$ it permanently commits to $c(v)$. There are only two reasons a vertex in S_j may be left uncolored by DenseColoringStep (version 2): it is not among the $(1-\delta_j)$-fraction of vertices participating in Step 1, or it has a color conflict with an external neighbor in Step 2. The first cause occurs with probability δ_j and, intuitively, the second cause occurs with probability about δ_j because vertices typically have *many* options for colors when they are processed but *few* external neighbors that can generate conflicts. Lemma 6.13 captures this formally; it is the culmination and corollary of Lemmas 6.14–6.16, which are proved later in this section. Lemma 6.13 is used to prove Lemmas 6.9 and 6.10.

Procedure DenseColoringStep (version 2).

1. Each cluster S_j selects $(1-\delta_j)|S_j|$ vertices u.a.r. and generates a permutation π of those vertices u.a.r. The vertex $\pi(q)$ selects a color $c(\pi(q))$ u.a.r. from

$$\Psi(\pi(q)) - \{c(\pi(q')) \mid q' < q \text{ and } \{\pi(q), \pi(q')\} \in E(G)\}.$$

2. Each $v \in S_j$ that has selected a color $c(v)$ permanently colors itself $c(v)$ if $c(v)$ is not selected by any vertices $u \in N_{\text{out}}(v)$.

Lemma 6.13. *Consider an execution of* DenseColoringStep *(version 2). Let T be any subset of S, and let $\delta = \max_{j:S_j \cap T \neq \emptyset} \delta_j$. For any number t, the probability that the number of uncolored vertices in T is at least t is at most $\binom{|T|}{t} \cdot (O(\delta))^t$.*

Our assumption about the identifiers of clusters and vertices guarantees that for each $v \in S_j$, we have $N_{\text{out}}(v) \subseteq \bigcup_{i=1}^{j} S_i$. Therefore, in the proof of Lemma 6.13, we expose the random bits of the clusters in the order $(S_1, \ldots, S_g)$. Once the random bits of $S_1, \ldots, S_j$ are revealed, we can determine whether any particular $v \in S_j$ successfully colors itself.

Our proofs of Lemmas 6.9 and 6.10 are based on a constant number of iterations of DenseColoringStep (version 2). In each iteration, the parameters D_j and δ_j might be different. In subsequent discussion, the term *anti-degree* of $v \in S_j$ refers to the number of uncolored vertices in $S_j \setminus (N(v) \cup \{v\})$, and the term *external degree* of $v \in S_j$ refers to the number of uncolored vertices in $N^\star(v) \setminus S_j$. Suppose S_j is a layer-i large block. The parameters for S_j in each iteration are as follows. Let $\beta > 0$ be a sufficiently large constant to be determined.

Degree upper bounds. By Lemma 6.4, $D_j^{(1)} = 3\epsilon_i \Delta$ upper bounds the initial anti-degree and external degree. For $k > 1$, the parameter $D_j^{(k)}$ is chosen such that $D_j^{(k)} \geq \beta \delta_j^{(k-1)} \cdot D_j^{(k-1)}$. We write $\mathcal{D}_j^{(k)}$ to denote the invariant that at the *beginning* of the kth iteration, $D_j^{(k)}$ is an upper bound on the anti-degree and external degree of all uncolored vertices in $S_j \setminus V_{\mathsf{bad}}$.

Cluster size upper bounds. By Lemma 6.4, $U_j^{(1)} = (1 + 3\epsilon_i)\Delta$ is an upper bound on the initial cluster size. For $k > 1$, the parameter $U_j^{(k)}$ is chosen such that $U_j^{(k)} \leq \beta \delta_j^{(k-1)} \cdot U_j^{(k-1)}$. We write $\mathcal{U}_j^{(k)}$ to denote the invariant that at the beginning of the kth iteration, the number of uncolored vertices in $S_j \setminus V_{\mathsf{bad}}$ is at most $U_j^{(k)}$.

Cluster size lower bounds. $L_j^{(1)} = \frac{\Delta}{\log(1/\epsilon_i)}$. For $k > 1$, the parameter $L_j^{(k)}$ is chosen such that $L_j^{(k)} \geq \delta_j^{(k-1)} \cdot L_j^{(k-1)}$. We write $\mathcal{L}_j^{(k)}$ to denote the invariant that at the beginning of the kth iteration, the number of uncolored vertices in $S_j \setminus V_{\mathsf{bad}}$ is at least $L_j^{(k)}$. By the definition of *large* blocks, $\mathcal{L}_j^{(1)}$ holds initially.

Shrinking rates. For each k, the shrinking rate $\delta_j^{(k)}$ of cluster S_j for the kth iteration is chosen such that

$$1/K \geq \delta_j^{(k)} \geq \frac{D_j^{(k)} \log\left(L_j^{(k)}/D_j^{(k)}\right)}{L_j^{(k)}}.$$

Additionally, we require that $\delta_1^{(k)} \leq \cdots \leq \delta_g^{(k)}$, with $\delta_j^{(k)} = \delta_{j+1}^{(k)}$ if S_j and S_{j+1} are in

the same layer.

Although the initial values of $D_j^{(1)}, U_j^{(1)}, L_j^{(1)}$ are determined, there is considerable freedom in choosing the remaining values to satisfy the four rules above. We refer to the following equations involving $D_j^{(k)}, U_j^{(k)}, L_j^{(k)}$, and $\delta_j^{(k)}$ as the *default settings* of these parameters. Unless stated otherwise, the proofs of Lemmas 6.9 and 6.10 use the default settings.

$$D_j^{(k)} = \beta\delta_j^{(k-1)} \cdot D_j^{(k-1)}, \qquad U_j^{(k)} = \beta\delta_j^{(k-1)} \cdot U_j^{(k-1)},$$

$$L_j^{(k)} = \delta_j^{(k-1)} \cdot L_j^{(k-1)}, \qquad \delta_j^{(k)} = \frac{D_j^{(k)} \log\left(L_j^{(k)}/D_j^{(k)}\right)}{L_j^{(k)}}.$$

Validity of Parameters. Before the first iteration the invariants $\mathcal{D}_j^{(1)}$, $\mathcal{U}_j^{(1)}$, and $\mathcal{L}_j^{(1)}$ are met initially, for each cluster S_j. Suppose S_j is a layer-i large block. Lemma 6.4 shows that the initial value of $D_j^{(1)}$ is a valid upper bound on the external degree (at most $\epsilon_i\Delta$) and anti-degree (at most $3\epsilon_i\Delta$). We also have

$$U_j^{(1)} = (1 + 3\epsilon_i)\Delta \geq |S_j| \geq \frac{\Delta}{\log(1/\epsilon_i)} = L_j^{(1)},$$

where the lower bound is from the definition of *large* and the upper bound is from Lemma 6.4.

For $k > 1$, the invariants $\mathcal{D}_j^{(k)}$ and $\mathcal{U}_j^{(k)}$ might not hold naturally. Before the kth iteration begins we *forcibly* restore them by removing from consideration all vertices in the clusters that violate either invariant, putting these vertices in V_{bad}. Notice that DenseColoringStep (version 2) *always* satisfies invariant $\mathcal{L}_j^{(k)}$.

Maintenance of Invariants. We calculate the probability for the invariants $\mathcal{D}_j^{(k+1)}$ and $\mathcal{U}_j^{(k+1)}$ to naturally hold at a cluster S_j. In what follows, we analyze the kth iteration of the algorithm, and assume that $\mathcal{D}_j^{(k)}$ and $\mathcal{U}_j^{(k)}$ hold initially. Let $T \subseteq S$ be a set of vertices that are uncolored at the beginning of the kth iteration, and suppose $\delta_j^{(k)} = \max_{j':S_{j'}\cap T\neq\emptyset} \delta_{j'}^{(k)}$. By Lemma 6.13, after the kth iteration, the probability that the number of uncolored vertices in T is at least t is at most $\binom{|T|}{t} \cdot \left(O(\delta_j^{(k)})\right)^t$. Using this result, we derive the

following bounds:

$$\Pr\left[\mathcal{U}_j^{(k+1)}\right] \geq 1 - \exp\left(-\Omega(U_j^{(k+1)})\right),$$
$$\Pr\left[\mathcal{D}_j^{(k+1)}\right] \geq 1 - O\left(U_j^{(k)}\right)\exp\left(-\Omega(D_j^{(k+1)})\right).$$

We first consider $\Pr\left[\mathcal{U}_j^{(k+1)}\right]$. We choose T as the set of uncolored vertices in $S_j \setminus V_{\mathsf{bad}}$ at the beginning of the kth iteration, and set $t = U_j^{(k+1)}$. We have $t = U_j^{(k+1)} = \beta\delta_j^{(k)} \cdot U_j^{(k)} \geq \beta\delta_j^{(k)}|T|$, and this implies $\delta_j^{(k)}|T|/t \leq 1/\beta$. If we select β to be a large enough constant, then

$$\begin{aligned}
1 - \Pr\left[\mathcal{U}_j^{(k+1)}\right] &\leq \binom{|T|}{t} \cdot \left(O(\delta_j^{(k)})\right)^t \\
&\leq \left(O\left(\delta_j^{(k)}\right) \cdot e|T|/t\right)^t \\
&\leq (O(1/\beta))^t \\
&= \exp\left(-\Omega\left(U_j^{(k+1)}\right)\right).
\end{aligned}$$

Next, consider $\Pr\left[\mathcal{D}_j^{(k+1)}\right]$. For each vertex $v \in S_j \setminus V_{\mathsf{bad}}$ that is uncolored at the beginning of the kth iteration, define $\mathcal{E}_v^a$ (resp., $\mathcal{E}_v^e$) as the event that the anti-degree (resp., external degree) of v at the end of the kth iteration is higher than $D_j^{(k+1)}$. If we can show that both $\Pr[\mathcal{E}_v^a]$ and $\Pr[\mathcal{E}_v^e]$ are at most $\exp\left(-\Omega\left(D_j^{(k+1)}\right)\right)$, then we conclude $\Pr[\mathcal{D}_j^{(k+1)}] \geq 1 - O\left(U_j^{(k)}\right)\exp\left(-\Omega(D_j^{(k+1)})\right)$ by a union bound over at most $U_j^{(k)}$ vertices $v \in S_j \setminus V_{\mathsf{bad}}$ that are uncolored at the beginning of the kth iteration.

We show that $\Pr[\mathcal{E}_v^e] \leq \exp\left(-\Omega\left(D_j^{(k+1)}\right)\right)$. We choose T as the set of uncolored vertices in $N^\star(v) \setminus (S_j \cup V_{\mathsf{bad}})$ at the beginning of the kth iteration, and set $t = D_j^{(k+1)}$. Since the layer number of each vertex in $N^\star(v)\setminus(S_j \cup V_{\mathsf{bad}})$ is smaller than or equal to the layer number of S_j, our requirement about the shrinking rates implies that $\delta_j^{(k)} \geq \max_{j':S_{j'}\cap T\neq\emptyset} \delta_{j'}^{(k)}$.

We have $t = D_j^{(k+1)} = \beta\delta_j^{(k)} \cdot D_j^{(k)} \geq \beta\delta_j^{(k)}|T|$, and this implies $\delta_j^{(k)}|T|/t \leq 1/\beta$. If we select β to be a large enough constant, then

$$\Pr[\mathcal{E}_v^e] \leq \binom{|T|}{t} \cdot \left(O\left(\delta_j^{(k)}\right)\right)^t \leq \left(O\left(\delta_j^{(k)}\right) \cdot e|T|/t\right)^t \leq (O(1/\beta))^t = \exp\left(-\Omega\left(D_j^{(k+1)}\right)\right).$$

The bound $\Pr[\mathcal{E}_v^a] \leq \exp\left(-\Omega\left(D_j^{(k+1)}\right)\right)$ is proved in the same way. Based on the proba-

bility calculations above, we are now prepared to prove Lemmas 6.9 and 6.10.

Proof of Lemma 6.9. We perform 6 iterations of DenseColoringStep (version 2) using the default settings of all parameters. Recall that the shrinking rate for the kth iteration is $\delta_j^{(k)} = \frac{D_j^{(k)} \log\left(L_j^{(k)}/D_j^{(k)}\right)}{L_j^{(k)}}$ for each cluster S_j. If S_j is a layer-i block, we have $\delta_j^{(k)} = O\big(\epsilon_i \log^2(1/\epsilon_i)\big)$ for each $k \in [1,6]$ since $D_j^{(\cdot)}$ and $L_j^{(\cdot)}$ decay at the same rate, asymptotically.

Consider any vertex $v \in V^\star$, and a layer number $i \in [2,\ell]$. Let T be the set of layer-i neighbors of v in S. To prove Lemma 6.9, it suffices to show that after 6 iterations of DenseColoringStep (version 2), with probability $1 - \exp(-\Omega(\text{poly}(\Delta)))$, the number of uncolored vertices in T is at most $\epsilon_i^5 \Delta$.

Define (t_k) as in the proof of Lemma 6.7.

$$\begin{aligned} t_1 &= |T|, \\ \text{and } t_k &= \max\left\{\beta \delta_j^{(k-1)} t_{k-1},\ \epsilon_i^5 \Delta\right\}. \end{aligned}$$

Here $\delta_j^{(k)}$ is the common shrinking rate of any layer-i cluster S_j. We have $t_7 = \epsilon_i^5 \Delta$ since $\epsilon_i \le \epsilon_\ell$ is sufficiently small.

Assume that at the beginning of the kth iteration, the number of uncolored vertices in $T \setminus V_{\mathsf{bad}}$ is at most t_k, and the invariants $\mathcal{D}_j^{(k)}$, $\mathcal{L}_j^{(k)}$, and $\mathcal{U}_j^{(k)}$ are met for each cluster S_j such that $S_j \cap T \neq \emptyset$. By Lemma 6.13, after the kth iteration, the probability that the number of uncolored vertices in $T \setminus V_{\mathsf{bad}}$ is more than t_{k+1} is

$$\binom{t_k}{t_{k+1}} \cdot \left(O\left(\delta_j^{(k)}\right)\right)^{t_{k+1}} \le \left(O\left(\delta_j^{(k)}\right) \cdot e t_k / t_{k+1}\right)^{t_{k+1}} \le (O(1/\beta))^{t_{k+1}} = \exp(-\Omega(t_{k+1})).$$

Notice that $\exp(-\Omega(t_{k+1})) \le \exp(-\Omega(\epsilon_i^5 \Delta)) = \exp(-\Omega(\text{poly}(\Delta)))$. For the maintenance of the invariants, $\mathcal{L}_j^{(k+1)}$ holds with probability 1; the probability that the invariants $\mathcal{D}_j^{(k+1)}$ and $\mathcal{U}_j^{(k+1)}$ are met for all clusters S_j such that $S_j \cap T \neq \emptyset$ is at least $1 - O(|T|)\exp(-\Omega(\text{poly}(\Delta))) = 1 - \exp(-\Omega(\text{poly}(\Delta)))$. By a union bound over all six iterations, with probability $1 - \exp(-\Omega(\text{poly}(\Delta)))$, the number of uncolored layer-i neighbors of v in $S \setminus V_{\mathsf{bad}}$ is at most $t_7 = \epsilon_i^5 \Delta$. □

Proof of Lemma 6.10. In the setting of Lemma 6.10, we deal with only layer-1 large blocks, and so $D_1^{(k)} = \cdots = D_g^{(k)}$, $U_1^{(k)} = \cdots = U_g^{(k)}$, $L_1^{(k)} = \cdots = L_g^{(k)}$, $\delta_1^{(k)} = \cdots = \delta_g^{(k)}$, for each

iteration k. For this reason we drop the subscripts. Our algorithm consists of three phases, as follows. Recall that c is a large enough constant related to the failure probability specified in the statement of Lemma 6.10.

The Low Degree Case. The following algorithm and analysis apply to all values of Δ. The conclusion is that we can color most of V_1^{L} such that the probability that any vertex joins V_{bad} is $\Delta^{-\Omega(c)}$ and all remaining uncolored vertices (i.e., R) induce a graph with maximum degree $O(c^2)$. Since the guarantee on V_{bad} is that it induces components with size $\mathrm{poly}(\Delta)\log n$, this analysis is only appropriate when Δ is, itself, $\mathrm{poly}\log n$. We deal with larger Δ in *The High Degree Case* and prove that the uncolored vertices can be partitioned into R and X with the same guarantee on R, and the stronger guarantee that X induces $\mathrm{poly}\log n$-size components, regardless of Δ.

Phase 1. The first phase consists of 9 iterations of DenseColoringStep (version 2), using the default settings of all parameters. Due to the fact that $\epsilon_1 = \Delta^{-1/10}$, we have $\delta^{(k)} = O(\Delta^{-1/10}\log^2\Delta)$ for each $k \in [1,9]$. Therefore, at the end of the 9th iteration, we have the parameters

$$\begin{aligned} D^{(10)} &= \Theta(\log^{18}\Delta), \\ L^{(10)} &= \Theta(\Delta^{1/10}\log^{17}\Delta), \\ \text{and } U^{(10)} &= \Theta(\Delta^{1/10}\log^{18}\Delta). \end{aligned}$$

In view of the previous calculations, the probability that all invariants hold for a specific cluster S_j and all $k \in [1,10]$ is at least $1 - \exp(-\Omega(\log^{18}\Delta))$. If a cluster S_j does not satisfy an invariant for some k, then *all* vertices in S_j halt and join V_{bad}. They do not participate in the kth iteration or subsequent steps.

Phase 2. For the 10th iteration, we switch to a *non*-default shrinking rate

$$\delta^{(10)} = \Delta^{-1/20}.$$

However, we still define

$$\begin{aligned} U^{(11)} &= \beta\delta^{(10)} \cdot U^{(10)} = \Theta(\Delta^{1/20}\log^{18}\Delta) \\ \text{and } L^{(11)} &= \delta^{(10)} \cdot L^{(10)} = \Theta(\Delta^{1/20}\log^{17}\Delta) \end{aligned}$$

according to their default setting. Since $\beta\delta^{(10)} \cdot D^{(10)} = o(1)$, we should not adopt the default definition of $D^{(11)}$. Instead, we fix it to be the sufficiently large constant c.

$$D^{(11)} = c.$$

Using the previous probability calculations, for each cluster S_j the invariant $\mathcal{U}^{(11)}$ holds with probability at least $1 - \exp(-\Omega(\Delta^{1/20}\text{poly}\log\Delta))$, and the invariant $\mathcal{L}^{(11)}$ holds with certainty. We will show that for a given cluster S_j, the probability that $D^{(11)}$ is a valid degree bound (i.e., $\mathcal{D}^{(11)}$ holds) is at least $1 - \Delta^{-\Omega(c)}$. If a cluster S_j does not meet at least one of $\mathcal{U}^{(11)}$, $\mathcal{L}^{(11)}$, or $\mathcal{D}^{(11)}$, then all vertices in S_j halt and join V_{bad}.

Phase 3. For the 11th iteration, we use the default shrinking rate

$$\delta^{(11)} = \frac{D^{(11)}\log(L^{(11)}/D^{(11)})}{L^{(11)}} = \Theta\left(\frac{1}{\Delta^{1/20}\log^{16}\Delta}\right).$$

We will show that after the 11th iteration, for each cluster S_j, with probability at least $1 - \Delta^{-\Omega(c)}$, there are at most c^2 uncolored vertices $v \in S_j$ such that there is at least one uncolored vertex in $N_{\text{out}}(v) \setminus S_j$. If S_j does not satisfy this property, we put all remaining uncolored vertices in S_j to V_{bad}. For each cluster S_j satisfying this property, in $O(1)$ additional rounds we color all vertices in S_j but c^2 of them since at most c^2 have potential conflicts outside of S_j. At this point, the remaining uncolored vertices R induce a subgraph of maximum degree at most $c^2 + D^{(10)} = c^2 + c = O(c^2)$.

The choice of parameters are summarized as follows (Table 6.2). Note that we use the default shrinking rate $\delta^{(i)} = \frac{D^{(i)}\log(L^{(i)}/D^{(i)})}{L^{(i)}}$ for all i except $i = 10$.

	$D^{(i)}$	$L^{(i)}$	$U^{(i)}$	$\delta^{(i)}$
$i \in [9]$	$\Theta\left(\Delta^{\frac{10-i}{10}}\log^{2i-2}\Delta\right)$	$\Theta\left(\Delta^{\frac{11-i}{10}}\log^{2i-3}\Delta\right)$	$\Theta\left(\Delta^{\frac{11-i}{10}}\log^{2i-2}\Delta\right)$	$\Theta\left(\Delta^{-\frac{1}{10}}\log^{2}\Delta\right)$
$i = 10$	$\Theta\left(\log^{18}\Delta\right)$	$\Theta\left(\Delta^{\frac{1}{10}}\log^{17}\Delta\right)$	$\Theta\left(\Delta^{\frac{1}{10}}\log^{18}\Delta\right)$	$\Delta^{-\frac{1}{20}}$
$i = 11$	c	$\Theta\left(\Delta^{\frac{1}{20}}\log^{17}\Delta\right)$	$\Theta\left(\Delta^{\frac{1}{20}}\log^{18}\Delta\right)$	$\Theta\left(\Delta^{-\frac{1}{20}}\log^{-16}\Delta\right)$

Table 6.2: Parameter setting for the low-degree case.

Analysis of Phase 2. Recall $\delta^{(10)} = \Delta^{-1/20}$ and $D^{(10)} = \Theta(\log^{18}\Delta)$. By Lemma 6.13, the probability that the external degree or anti-degree of $v \in S_j$ is at most c is:

$$1 - \binom{D^{(10)}}{c}\left(O\left(\delta^{(10)}\right)\right)^c \geq 1 - \binom{O\left(\log^{18}\Delta\right)}{c}\left(O\left(\Delta^{-1/20}\right)\right)^c \geq 1 - \Delta^{-\Omega(c)}.$$

By a union bound over at most $U^{(10)} = \Theta(\Delta^{1/10}\log^{18}\Delta)$ vertices $v \in S_j$ that are uncolored at the beginning of the 10th iteration, the parameter setting $D^{(11)} = c$ is a valid upper bound of external degree and anti-degree for S_j after the 10th iteration with probability at least $1 - \Delta^{-\Omega(c)}$.

Analysis of Phase 3. Consider a vertex $v \in S_j$ that is uncolored at the beginning of the 11th iteration. Define the event $\mathcal{E}_v$ as follows. The event $\mathcal{E}_v$ occurs if, after the 11th iteration, v is still uncolored, and there is at least one uncolored vertex in $N_{\text{out}}(v) \setminus (S_j \cup V_{\mathsf{bad}})$. Our goal is to show that the number of vertices $v \in S_j$ such that $\mathcal{E}_v$ occurs is at most c^2 with probability at least $1 - \Delta^{-\Omega(c)}$.

Consider any size-c^2 subset Y of S_j. As a consequence of Lemma 6.13, we argue that the probability that $\mathcal{E}_v$ occurs for all $v \in Y$ is at most

$$\left(D^{(11)}\right)^{c^2} \cdot \left(O\left(\delta^{(11)}\right)\right)^{c^2\left(1+1/D^{(11)}\right)}.$$

The reason is as follows. Pick some $v \in Y$. If $\mathcal{E}_v$ occurs, then there must exist a neighbor $v' \in N_{\text{out}}(v) \setminus (S_j \cup V_{\mathsf{bad}})$ that is uncolored. The number of uncolored vertices in $N_{\text{out}}(v) \setminus (S_j \cup V_{\mathsf{bad}})$ at the beginning of the 11th iteration is at most $D^{(11)}$, so there are at most $(D^{(11)})^{c^2}$ ways of mapping each $v \in Y$ to a vertex $v' \in N_{\text{out}}(v) \setminus (S_j \cup V_{\mathsf{bad}})$ of v. Let $T = \bigcup_{v \in Y}\{v, v'\}$. A vertex outside of S_j can be adjacent to at most $D^{(11)}$ vertices in S_j, and so $|T| \geq c^2(1 + 1/D^{(11)})$. The probability that all vertices in T are uncolored is $\left(O(\delta^{(11)})\right)^{c^2\left(1+1/D^{(11)}\right)}$ by Lemma 6.13. By a union bound over at most $\left(D^{(11)}\right)^{c^2}$ choices of T, we obtain the desired probabilistic bound.

Recall that $U^{(11)} = \Theta(\Delta^{1/20}\log^{18}\Delta) = L^{(11)} \cdot \Theta(\log\Delta)$ and $L^{(11)} = \Theta(\Delta^{1/20}\log^{17}\Delta)$ are the cluster size upper bound and lower bound at the beginning of the 11th iteration. By a union bound over at most $\left(U^{(11)}\right)^{c^2}$ choices of a size-c^2 subset of S_j, the probability f that there exists c^2 vertices $v \in S_j$ such that $\mathcal{E}_v$ occurs is

$$f = \left(U^{(11)}\right)^{c^2} \cdot \left(D^{(11)}\right)^{c^2} \cdot \left(O\left(\delta^{(11)}\right)\right)^{c^2\left(1+1/D^{(11)}\right)}.$$

Recall that $D^{(11)} = c$ is sufficiently large. We have

$$\left(U^{(11)}\right)^{c^2} = \left(O\left(L^{(11)} \log \Delta\right)\right)^{c^2}, \tag{6.3}$$

$$\left(D^{(11)}\right)^{c^2} = O(1), \tag{6.4}$$

$$\left(O\left(\delta^{(11)}\right)\right)^{c^2\left(1+1/D^{(11)}\right)} = \left(O\left(\tfrac{\log(L^{(11)})}{L^{(11)}}\right)\right)^{c^2+c}. \tag{6.5}$$

where $L^{(11)} = \Theta(\Delta^{1/20} \log^8 \Delta)$. Taking the product of (6.3), (6.4), and (6.5), we have:

$$f = \Theta(\log \Delta)^{O(c^2)} \cdot \Theta\left(\Delta^{-1/20}\right)^c = \Delta^{-\Omega(c)},$$

as required.

Remark 6.4. *The analysis of Phase 2 would proceed in the same way if we had chosen $\delta^{(10)}$ according to its default setting of $\Theta(\Delta^{-1/10} \log^2 \Delta)$. We choose a larger value of $\delta^{(10)}$ in order to keep $L^{(11)}$ artificially large ($\Delta^{\Omega(1)}$), and thereby allow Phase 3 to fail with smaller probability $\Delta^{-\Omega(c)}$.*

The High Degree Case. The Low Degree Case handles all Δ that are $\operatorname{poly}\log n$. We now assume Δ is sufficiently large, i.e., $\Delta > \log^{\alpha c} n$, where α is some large universal constant, and we want to design an algorithm such that no vertex joins V_{bad}, and all uncolored vertices are partitioned into R and X, with R having the same $O(c^2)$-degree guarantee as before, and the components induced by X have size $\log^{O(c)} n = \operatorname{poly}\log n$, regardless of Δ. Intuitively, the proof follows the same lines as the Low Degree Case, but in Phase 1 we first reduce the maximum degree to $\Delta' = \log^{O(c)} n$ then put any bad vertices that fail to satisfy an invariant into X (rather than V_{bad}). According to the shattering lemma (Lemma 1.2), the components induced by X have size $\operatorname{poly}(\Delta') \log n = \log^{O(c)} n$. The High Degree Case consists of 13 iterations of DenseColoringStep (version 2) with the following parameter settings (Table 6.3).

We use the default shrinking rate $\delta^{(i)} = \frac{D^{(i)} \log(L^{(i)}/D^{(i)})}{L^{(i)}}$ for all i except $i \in \{10, 11, 12\}$. Phase 1 consists of all iterations $i \in [11]$; Phase 2 consists of iteration $i = 12$; Phase 3 consists of iteration $i = 13$. The algorithm and the analysis are similar to the small degree case, so in subsequent discussion we only point out the differences. In order to have all $\delta^{(i)} \ll 1$, we need to have $\Delta^{1/20} \gg \log^{5c} n$. We proceed under the assumption that $\Delta > \log^{\alpha c} n$ (α is some large universal constant), so this condition is met.

	$D^{(i)}$	$L^{(i)}$	$U^{(i)}$	$\delta^{(i)}$
$i \in [9]$	$\Theta\Big(\Delta^{\frac{10-i}{10}} \log^{2i-2} \Delta\Big)$	$\Theta\Big(\Delta^{\frac{11-i}{10}} \log^{2i-3} \Delta\Big)$	$\Theta\Big(\Delta^{\frac{11-i}{10}} \log^{2i-2} \Delta\Big)$	$\Theta\Big(\Delta^{-\frac{1}{10}} \log^{2} \Delta\Big)$
$i = 10$	$\Theta\big(\max\{\log^{18} \Delta, \log n\}\big)$	$\Theta\Big(\Delta^{\frac{1}{10}} \log^{17} \Delta\Big)$	$\Theta\Big(\Delta^{\frac{1}{10}} \log^{18} \Delta\Big)$	$\Delta^{-\frac{1}{20}} \log^{-18} \Delta$
$i = 11$	$\Theta(\log n)$	$\Theta\Big(\Delta^{\frac{1}{20}} / \log \Delta\Big)$	$\Theta\Big(\Delta^{\frac{1}{20}}\Big)$	$\Delta^{-\frac{1}{20}} \log^{5c} n$
$i = 12$	$\Theta(\log n)$	$\Theta\Big(\dfrac{\log^{5c} n}{\log \Delta}\Big)$	$\Theta\big(\log^{5c} n\big)$	$\log^{-3c} n$
$i = 13$	c	$\Theta\Big(\dfrac{\log^{2c} n}{\log \Delta}\Big)$	$\Theta\big(\log^{2c} n\big)$	$\Theta\Big(\dfrac{\log \Delta \log\log n}{\log^{2c} n}\Big)$

Table 6.3: Parameter setting for the high-degree case.

Phase 1. In view of previous calculations, all invariants hold for a cluster S_j ($\mathcal{U}^{(i)}$, $\mathcal{L}^{(i)}$, and $\mathcal{D}^{(i)}$, for $i \in [1, 12]$) with probability at least $1 - \exp(-\Omega(\log n)) = 1 - 1/\mathrm{poly}(n)$, since all parameters $D^{(i)}$, $L^{(i)}$, and $U^{(i)}$ are chosen to be $\Omega(\log n)$. Therefore, no cluster S_j is put in V_{bad} due to an invariant violation, w.h.p.

Phase 2. Consider iteration $i = 12$. It is straightforward that the invariants $\mathcal{U}^{(13)}$ and $\mathcal{L}^{(13)}$ hold w.h.p., since $L^{(13)} = \Omega(\log n)$ and $U^{(13)} = \Omega(\log n)$. Now we consider the invariant $\mathcal{D}^{(13)}$. By Lemma 6.13, the probability that the external degree or anti-degree of $v \in S_j$ is at most c is:

$$1 - \binom{D^{(12)}}{c} \left(O\big(\delta^{(12)}\big)\right)^c \geq 1 - \binom{O(\log n)}{c} \left(O\big(\log^{-3c} n\big)\right)^c \geq 1 - (\log n)^{-\Omega(c^2)}.$$

This failure probability is *not* small enough to guarantee that $\mathcal{D}^{(13)}$ holds everywhere w.h.p. In the high degree case, if a vertex v belongs to a cluster S_j such that $\mathcal{D}^{(13)}$ does not hold, we add the remaining uncolored vertices in S_j (at most $U^{(12)} = O(\log^{5c} n)$ of them) to X.

Phase 3. Similarly, we will show that after the 13th iteration, for each cluster S_j, with probability at least $1 - (\log n)^{-\Omega(c^2)}$, there are at most c^2 uncolored vertices $v \in S_j$ such that there is at least one uncolored vertex in $N_{\mathrm{out}}(v) \setminus (S_j \cup X)$. If S_j does not satisfy this property, we put all remaining uncolored vertices in S_j to X. For each cluster S_j satisfying this property, in one additional round we can color all vertices in S_j but c^2 of them. At this point, the remaining uncolored vertices induce a subgraph R of maximum degree at most $c^2 + D^{(13)} = c^2 + c = O(c^2)$. Following the analysis in the small degree case, the

probability that a vertex v is added to X in the 13th iteration is

$$\begin{aligned} f &= \left(U^{(13)}\right)^{c^2} \cdot \left(D^{(13)}\right)^{c^2} \cdot \left(O\left(\delta^{(13)}\right)\right)^{c^2\left(1+1/D^{(13)}\right)} \\ &= O\left(\log^{2c} n\right)^{c^2} \cdot O(1) \cdot O\left(\frac{\log \Delta \log\log n}{\log^{2c} n}\right)^{c^2+c} \\ &= O\left((\log n)^{-2c^2} \cdot (\log \Delta \log\log n)^{c^2+c}\right) \\ &= (\log n)^{-\Omega(c^2)}. \end{aligned}$$

Size of Components in X. To bound the size of each connected component of X, we use the shattering lemma (Lemma 1.2). Define $G' = (V', E')$ as follows. The vertex set V' consists of all vertices in S that remains uncolored at the beginning of iteration 12. Two vertices u and v are linked by an edge in E' if (i) u and v belong to the same cluster, or (ii) u and v are adjacent in the original graph G. It is clear that the maximum degree Δ' of G' is $U^{(12)} + D^{(12)} = O(\log^{5c} n)$. In view of the above analysis, the probability of $v \in X$ is $1 - (\log n)^{-\Omega(c^2)} = 1 - (\Delta')^{-\Omega(c)}$, and this is true even if the random bits outside of a constant-radius neighborhood of v in G' are determined adversarially. Applying Lemma 1.2 to the graph G', the size of each connected component of X is $O(\text{poly}(\Delta') \log n) = \log^{O(c)} n$, w.h.p., both in G' and in the original graph G, since G' is the result of adding some additional edges to the subgraph of G induced by V'. □

The reader may recall that the proofs of Lemmas 6.9 and 6.10 were based on the veracity of Lemma 6.13. The remainder of this section is devoted to proving Lemma 6.13, which bounds the probability that a certain number of vertices remain uncolored by DenseColoringStep (version 2). By inspection of the DenseColoringStep (version 2) pseudocode, a vertex in S_j can remain uncolored for two different reasons:

- it never selects a color, because it is not among the $(1 - \delta_j)|S_j|$ participating vertices in Step 1, or
- it selects a color in Step 1, but is later decolored in Step 2 because of a conflict with some vertex in $S_{j'}$ with $j' < j$.

Lemmas 6.14–6.16 analyze different properties of DenseColoringStep (version 2), which are then applied to prove Lemma 6.13. Throughout we make use of the property that every $\delta_j < 1/K$ for some sufficiently large K.

Lemma 6.14. *Let $T = \{v_1, \ldots, v_k\}$ be any subset of S_j and $c_1, \ldots, c_k$ be any sequence of colors. The probability that v_i selects c_i in* DenseColoringStep *(version 2), for all $i \in [1, k]$, is $\left(O\left(\frac{\log(|S_j|/D_j)}{|S_j|}\right)\right)^{|T|}$.*

Proof. Let $p^\star$ be the probability that, for all $i \in [1, k]$, v_i selects c_i. Let $M = (1 - \delta_j)|S_j|$ be the number of participating vertices in Step 1. Notice that if v_i is not among the participating vertices, then v_i will not select any color, and thus cannot select c_i. Since we are upper bounding $p^\star$, it is harmless to condition on the event that v_i is a participating vertex. We write p_i to denote the rank of $v_i \in T$ in the random permutation of S_j.

Suppose that the ranks $p_1, \ldots, p_k$ were fixed. Recall that each vertex $v_i \in S_j$ is adjacent to all but at most D_j vertices in S_j. Thus, at the time v_i is considered it must have at least

$$\begin{aligned}
&M - p_i + \delta_j|S_j| - D_j \\
&\geq M - p_i + D_j \log(|S_j|/D_j) - D_j && \text{(constraint on } \delta_j\text{)} \\
&= (M - p_i) + D_j(\log(|S_j|/D_j) - 1)
\end{aligned}$$

available colors to choose from, at most one of which is c_i. Thus,

$$p^\star \leq \mathop{\mathrm{E}}_{p_1,\ldots,p_k}\left[\prod_{i=1}^{k} \frac{1}{(M - p_i) + D_j(\log(|S_j|/D_j) - 1)}\right].$$

We divide the analysis into two cases: (i) $k \geq M/2$ and (ii) $k < M/2$. For the case $k \geq M/2$, regardless of the choices of $p_1, \ldots, p_k$, we always have

$$\prod_{i=1}^{k} \frac{1}{(M - p_i) + D_j(\log(|S_j|/D_j) - 1)} \leq \frac{1}{k!} = (O(1/k))^k \leq (O(1/|S_j|))^{|T|}.$$

We now turn to the case $k < M/2$. We imagine choosing the rank vector $(p_1, \ldots, p_k)$ one element at a time. Regardless of the values of $(p_1, \ldots, p_{i-1})$, we always have

$$\begin{aligned}
&\mathrm{E}\left[\frac{1}{((M - p_i) + D_j(\log(|S_j|/D_j) - 1)} \;\middle|\; p_1, \ldots, p_{i-1}\right] \\
&\leq \frac{1}{M - (i-1)} \sum_{x=0}^{M-i} \frac{1}{x + D_j(\log(|S_j|/D_j) - 1)},
\end{aligned}$$

since there are $M-(i-1)$ choices for p_i and the worst case is when $\{p_1,\ldots,p_{i-1}\} = \{1,\ldots,i-1\}$. Observe that the terms in the sum are strictly decreasing, which means the average is maximized when $i = k < M/2$ is maximized. Continuing,

$$\leq \frac{1}{M/2}\sum_{x=0}^{M/2}\frac{1}{x + D_j(\log(|S_j|/D_j)-1)}$$

The sum is the difference between two harmonic sums, hence

$$\begin{aligned}
&= O\Big(\frac{1}{M}\cdot\big(\log M - \log(D_j(\log(|S_j|/D_j)-1))\big)\Big)\\
&= O\Big(\frac{\log(|S_j|/D_j)}{|S_j|}\Big), && \text{since } M = \Theta(|S_j|).
\end{aligned}$$

Therefore, regardless of k, $p^\star \leq \Big(O\Big(\frac{\log(|S_j|/D_j)}{|S_j|}\Big)\Big)^{|T|}$, as claimed. □

Lemma 6.15. *Let T be any subset of S_j. The probability that all vertices in T are decolored in* DenseColoringStep *(version 2) is $\Big(O\Big(\frac{D_j\log(|S_j|/D_j)}{|S_j|}\Big)\Big)^{|T|}$, even allowing the colors selected in $S_1,\ldots,S_{j-1}$ to be determined adversarially.*

Proof. There are in total at most $D_j^{|T|}$ different color assignments to T that can result in decoloring all vertices in T, since each vertex $v \in T \subseteq S_j$ satisfies $|N_{\text{out}}(v)\setminus S_j| \leq |N^\star(v)\setminus S_j| \leq D_j$. By Lemma 6.14 (and a union bound over $D_j^{|T|}$ color assignments to T) the probability that all vertices in T are decolored is $D_j^{|T|}\cdot\Big(O\Big(\frac{\log(|S_j|/D_j)}{|S_j|}\Big)\Big)^{|T|} = \Big(O\Big(\frac{D_j\log(|S_j|/D_j)}{|S_j|}\Big)\Big)^{|T|}$. Recall that for each $v \in T \subseteq S_j$, we have $N_{\text{out}}(v)\setminus S_j \subseteq \bigcup_{k=1}^{j-1} S_k$, and so whether v is decolored is independent of the random bits in $S_{j+1},\ldots,S_g$. The above analysis (which is based on Lemma 6.14) holds even allowing the colors selected in $S_1,\ldots,S_{j-1}$ to be determined adversarially. □

Lemma 6.16. *Let T be any subset of S_j. The probability that all vertices in T do not select a color in Step 1 of* DenseColoringStep *(version 2) is $(O(\delta_j))^{|T|}$. The probability only depends on the random bits within S_j.*

Proof. The lemma follows from the fact that in DenseColoringStep (version 2) a vertex $v \in S_j$ does not participate in Step 1 with probability δ_j, and the events for two vertices $u, v \in S_j$ to not participate in Step 1 are negatively correlated. □

Proof of Lemma 6.13. Recall that we assume the clusters $S = \{S_1, \ldots, S_g\}$ are ordered in such a way that for any $u \in S_j$, we have $N_{\text{out}}(u) \subseteq N^\star(u) \subseteq \bigcup_{k=1}^{j} S_k$. In the proof we expose the random bits of the clusters in the order $(S_1, \ldots, S_g)$.

Consider any subset $T \subseteq S$. Let $U = U_1 \cup U_2$ be a size-t subset $U \subseteq T$. We calculate the probability that all vertices in U_1 do not participate in Step 1, and all vertices in U_2 are decolored in Step 2. Notice that there are at most 2^t ways of partitioning U into $U_1 \cup U_2$.

We write $U_1^{(j)} = U_1 \cap S_j$. Whether a vertex $v \in U_1^{(j)}$ fails to select a color only depends on the random bits in S_j. Thus, by Lemma 6.16, the probability that all vertices in U_1 fail to select a color is at most $\prod_{j=1}^{k} (O(\delta_j))^{\left|U_1^{(j)}\right|} \leq (O(\delta))^{|U_1|}$. Recall $\delta = \max_{j:S_j \cap T \neq \emptyset} \delta_j$.

We write $U_2^{(j)} = U_2 \cap S_j$. Whether a vertex $v \in U_2^{(j)}$ is decolored only depends the random bits in $S_1, \ldots, S_j$. However, regardless of the random bits in $S_1, \ldots, S_{j-1}$, the probability that all vertices in $U_2^{(j)}$ are decolored is $(O(\delta_j))^{\left|U_1^{(j)}\right|}$ by Lemma 6.15. Recall $\delta \geq \delta_j \geq \frac{D_j \log(|S_j|/D_j)}{|S_j|}$. Thus, the probability that all vertices in U_2 are decolored is at most $\prod_{j=1}^{k} (O(\delta_j))^{\left|U_2^{(j)}\right|} \leq (O(\delta))^{|U_2|}$.

Therefore, by a union bound over at most $\binom{|T|}{t}$ choices of U and at most 2^t ways of partitioning U into $U_1 \cup U_2$, the probability that the number of uncolored vertices in T is at least t is at most $2^t \cdot \binom{|T|}{t} \cdot (O(\delta))^t = \binom{|T|}{t} \cdot (O(\delta))^t$. This concludes the analysis of DenseColoringStep (version 2). □

6.6 Proof of Lemma 6.3

In this section, we prove Lemma 6.3. Fix a constant parameter $p \in (0, 1/4)$. The procedure OneShotColoring is a simple $O(1)$-round coloring procedure that breaks ties by ID. We orient each edge $\{u, v\}$ towards the endpoint with lower ID, that is, $N_{\text{out}}(v) = \{u \in N(v) \mid \text{ID}(u) < \text{ID}(v)\}$. We assume that each vertex v is associated with a palette $\Psi(v)$ of size $\Delta + 1$, and this is used implicitly in the proofs of the lemmas in this section.

Procedure OneShotColoring.

1. Each uncolored vertex v decides to participate independently with probability p.
2. Each participating vertex v selects a color $c(v)$ from its palette $\Psi(v)$ uniformly at random.

3. A participating vertex v successfully colors itself if $c(v)$ is not chosen by any vertex in $N_{\text{out}}(v)$.

After OneShotColoring, each vertex v removes all colors from $\Psi(v)$ that are taken by some neighbor $u \in N(v)$. The number of *excess colors* at v is the size of v's remaining palette minus the number of uncolored neighbors of v. We prove one part of Lemma 6.3 by showing that after a call to OneShotColoring, the number of excess colors at any ϵ-sparse v is $\Omega(\epsilon^2\Delta)$, with probability $1 - \exp(-\Omega(\epsilon^2\Delta))$. The rest of this section constitutes a proof of Lemma 6.3.

Consider an execution of OneShotColoring with any constant $p \in (0, 1/4)$. Recall that we assume $1/\epsilon \geq K$, for some large enough constant K. Let v be an ϵ-sparse vertex. Define the following two numbers.

$f_1(v)$: the number of vertices $u \in N(v)$ that successfully color themselves by some $c \notin \Psi(v)$.

$f_2(v)$: the number of colors $c \in \Psi(v)$ such that at least two vertices in $N(v)$ successfully color themselves c.

It is clear that $f_1(v) + f_2(v)$ is a lower bound on the number of excess colors at v after OneShotColoring. Our first goal is to show that $f_1(v) + f_2(v) = \Omega(\epsilon^2\Delta)$ with probability at least $1 - \exp(-\Omega(\epsilon^2\Delta))$. We divide the analysis into two cases (Lemma 6.19 and Lemma 6.20), depending on whether $f_1(v)$ or $f_2(v)$ is likely to be the dominant term. For any v, the preconditions of either Lemma 6.19 or Lemma 6.20 are satisfied. Our second goal is to show that for each vertex v of degree at least $(5/6)\Delta$, with high probability, at least $(1-1.5p)|N(v)| > (1-(1.5)/4)\cdot(5/6)\Delta > \Delta/2$ neighbors of v remain uncolored after after OneShotColoring. This is done in Lemma 6.21.

Lemmas 6.17 and 6.18 establish some generally useful facts about OneShotColoring, which are used in the proofs of Lemma 6.19 and 6.20.

Lemma 6.17. *Let Q be any set of colors, and let S be any set of vertices with size at most 2Δ. The number of colors in Q that are selected in Step 2 of OneShotColoring by some vertices in S is less than $|Q|/2$ with probability at least $1 - \exp(-\Omega(|Q|))$.*

Proof. Let E_c denote the event that color c is selected by at least one vertex in S. Then $\Pr[E_c] \leq \frac{p|S|}{\Delta+1} < 2p < 1/2$, since $p < 1/4$ and $|S| \leq 2\Delta$. Moreover, the collection of events $\{E_c\}$ are negatively correlated [54].

Let X denote the number of colors in Q that are selected by some vertices in S. By linearity of expectation, $\mathrm{E}[X] < 2p \cdot |Q|$. We apply a Chernoff bound with $\delta = \frac{(1/2)-2p}{2p}$ and $\mu = 2p \cdot |Q|$. Recall that $0 < p < 1/4$, and so $\delta > 0$. For the case of $\delta \in [0, 1]$, we have:

$$\Pr[X \geq (1+\delta)\mu = |Q|/2] \leq \exp(-\delta^2\mu/3) = \exp(-\Omega(|Q|)).$$

Similarly, if $\delta > 1$, we still have:

$$\Pr[X \geq (1+\delta)\mu = |Q|/2] \leq \exp(-\delta\mu/3) = \exp(-\Omega(|Q|)). \qquad \square$$

Lemma 6.18. *Fix a sufficiently small $\epsilon > 0$. Consider a set of vertices $S = \{u_1, \ldots, u_k\}$ with cardinality $\epsilon\Delta/2$. Let Q be a set of colors such that each $u_i \in S$ satisfies $|\Psi(u_i) \cap Q| \geq (1 - \epsilon/2)(\Delta + 1)$. Moreover, each $u_i \in S$ is associated with a vertex set R_i such that (i) $S \cap R_i = \emptyset$, and (ii) $|R_i| \leq 2\Delta$. Then, with probability at least $1 - \exp(-\Omega(\epsilon^2\Delta))$, there are at least $p\epsilon(\Delta+1)/8$ vertices $u_i \in S$ such that the color c selected by u_i satisfies (i) $c \in Q$, and (ii) c is not selected by any vertex in $R_i \cup S \setminus \{u_i\}$.*

Proof. Define $Q_i = \Psi(u_i) \cap Q$. We call a vertex u_i *happy* if u_i selects some color $c \in Q$ and c is not selected by any vertex in $R_i \cup S \setminus \{u_i\}$. Define the following events.

E_i^{good}: u_i selects a color $c \in Q_i$ such that c is not selected by any vertices in R_i.

E_i^{bad}: the number of colors in Q_i that are selected by some vertices in R_i is at least $|Q_i|/2$.

E_i^{repeat}: the color selected by u_i is also selected by some vertices in $\{u_1, \ldots, u_{i-1}\}$.

Let X_i be the indicator random variable that *either* E_i^{good} or E_i^{bad} occurs, and let $X = \sum_{i=1}^{k} X_i$. Let Y_i be the indicator random variable that E_i^{repeat} occurs, and let $Y = \sum_{i=1}^{k} Y_i$. *Assuming that E_i^{bad} does not occur* for each $i \in [1, k]$, it follows that $X - 2Y$ is a lower bound on the number of happy vertices. Notice that by Lemma 6.17, $\Pr[E_i^{\text{bad}}] = \exp(-\Omega(|Q_i|)) = \exp(-\Omega(\Delta))$. Thus, assuming that no E_i^{bad} occurs merely distorts our probability estimates by a negligible $\exp(-\Omega(\Delta))$. We prove concentration bounds on X and Y, which together imply the lemma.

We show that $X \geq p\epsilon\Delta/7$ with probability $1 - \exp(-\Omega(\epsilon\Delta))$. It is clear that

$$\Pr[X_i = 1] \geq \Pr\left[E_i^{\text{good}} \mid \overline{E_i^{\text{bad}}}\right] \geq \frac{p \cdot |Q_i|/2}{\Delta + 1} \geq \frac{p(1 - \epsilon/2)}{2} > \frac{p}{3}.$$

Moreover, since $\Pr[X_i = 1 \mid E_i^{\text{bad}}] = 1$, the above inequality also holds, when condi-

tioned on *any* colors selected by vertices in R_i. Thus, $\Pr[X \le t]$ is upper bounded by $\Pr[\text{Binomial}(n', p') \le t]$ with $n' = |S| = \epsilon\Delta/2$ and $p' = \frac{p}{3}$. We set $t = p\epsilon\Delta/7$. Notice that $n'p' = p\epsilon\Delta/6 > t$. Thus, according to a Chernoff bound on the binomial distribution, $\Pr[X \le t] \le \exp(\frac{-(n'p'-t)^2}{2n'p'}) = \exp(-\Omega(\epsilon\Delta))$.

We show that $Y \le p\epsilon^2\Delta/2$ with probability $1 - \exp(-\Omega(\epsilon^2\Delta))$. It is clear that $\Pr[Y_i = 1] \le \frac{p(i-1)}{\Delta+1} \le \frac{p\epsilon}{2}$, even if we condition on arbitrary colors selected by vertices in $\{u_1, \ldots, u_{i-1}\}$. We have $\mu = \mathrm{E}[Y] \le \frac{p\epsilon}{2} \cdot |S| = \frac{p\epsilon^2\Delta}{4}$. Thus, by a Chernoff bound (with $\delta = 1$), $\Pr[Y \ge p\epsilon^2\Delta/2] \le \Pr[Y \ge (1+\delta)\mu] \le \exp(-\delta^2\mu/3) = \exp(-\Omega(\epsilon^2\Delta))$.

To summarize, with probability at least $1 - \exp(-\Omega(\epsilon^2\Delta))$, we have $X - 2Y \ge p\epsilon\Delta/7 - 2p\epsilon^2\Delta/2 > p\epsilon(\Delta+1)/8$. □

Lemma 6.19 considers the case when a large fraction of v's neighbors are likely to color themselves with colors outside the palette of v, and therefore be counted by $f_1(v)$. This lemma holds regardless of whether v is ϵ-sparse or not.

Lemma 6.19. *Suppose that there is a subset $S \subseteq N(v)$ such that $|S| = \epsilon\Delta/5$, and for each $u \in S$, $|\Psi(u) \setminus \Psi(v)| \ge \epsilon(\Delta+1)/5$. Then $f_1(v) \ge \frac{p\epsilon^2\Delta}{100}$ with probability at least $1 - \exp(-\Omega(\epsilon^2\Delta))$.*

Proof. Let $S = (u_1, \ldots, u_k)$ be sorted in increasing order by ID. Define $R_i = N_{\text{out}}(u_i)$, and $Q_i = \Psi(u_i) \setminus \Psi(v)$. Notice that $|Q_i| \ge \epsilon\Delta/5$. Define the following events.

E_i^{good}: u_i selects a color $c \in Q_i$ and c is not selected by any vertex in R_i.

E_i^{bad}: the number of colors in Q_i that are selected by vertices in R_i is more than $|Q_i|/2$.

Let X_i be the indicator random variable that either E_i^{good} or E_i^{bad} occurs, and let $X = \sum_{i=1}^{k} X_i$. Given that the events E_i^{bad} for all $i \in [1,k]$ do not occur, we have $X \le f_1(v)$,[34] since if E_i^{good} occurs, then u_i successfully colors itself by some color $c \notin \Psi(v)$. By Lemma 6.17, $\Pr[E_i^{\text{bad}}] = \exp(-\Omega(|Q_i|)) = \exp(-\Omega(\epsilon\Delta))$. Thus, up to this negligible error, we can assume that E_i^{bad} does not occur, for each $i \in [1,k]$.

We show that $X \ge \epsilon^2\Delta/100$ with probability $1 - \exp(-\Omega(\epsilon^2\Delta))$. It is clear that $\Pr[X_i = 1] \ge \Pr[E_i^{\text{good}} \mid \overline{E_i^{\text{bad}}}] \ge \frac{p|Q_i|/2}{\Delta+1} \ge \frac{p\epsilon}{10}$, and this inequality holds even when conditioning on *any* colors selected by vertices in R_i and $\bigcup_{1 \le j < i} R_j \cup \{u_j\}$. Since $S = (u_1, \ldots, u_k)$ is sorted in increasing order by ID, $u_i \notin R_j = N_{\text{out}}(u_j)$ for any $j \in [1,i)$. Thus, $\Pr[X \le t]$

[34] In general, X does not necessarily equal $f_1(v)$, since in the calculation of X we only consider the vertices in S, which is a subset of $N(v)$.

is upper bounded by $\Pr[\text{Binomial}(n', p') \leq t]$ with $n' = |S| = \epsilon\Delta/5$ and $p' = \frac{p\epsilon}{10}$. We set $t = \frac{n'p'}{2} = \frac{p\epsilon^2\Delta}{100}$. Thus, according to a lower tail of the binomial distribution, $\Pr[X \leq t] \leq \exp\left(\frac{-(n'p'-t)^2}{2n'p'}\right) = \exp(-\Omega(\epsilon^2\Delta))$. □

Lemma 6.20 considers the case that many pairs of neighbors of v are likely to color themselves the same color, and contribute to $f_2(v)$. Notice that any ϵ-sparse vertex that does not satisfy the preconditions of Lemma 6.19 *does* satisfy the preconditions of Lemma 6.20.

Lemma 6.20. *Let v be an ϵ-sparse vertex. Suppose that there is a subset $S \subseteq N(v)$ such that $|S| \geq (1-\epsilon/5)\Delta$, and for each $u \in S$, $|\Psi(u) \cap \Psi(v)| \geq (1-\epsilon/5)(\Delta+1)$. Then $f_2(v) \geq p^3\epsilon^2\Delta/2000$ with probability at least $1 - \exp(-\Omega(\epsilon^2\Delta))$.*

Proof. Let $S' = \{u_1, \ldots, u_k\}$ be any subset of S such that (i) $|S'| = \frac{p\epsilon\Delta}{100}$, (ii) for each $u_i \in S'$, there exists a set $S_i \subseteq S \backslash (S' \cup N(u_i))$ of size $\frac{\epsilon\Delta}{2}$. The existence of $S', S_1, \ldots, S_k$ is guaranteed by the ϵ-sparseness of v. In particular, S must contain at least $\epsilon\Delta - \epsilon\Delta/5 > p\epsilon\Delta/100 = |S'|$ *non*-ϵ-friends of v, and for each such non-friend $u_i \in S'$, we have $|S \setminus (S' \cup N(u_i))| \geq |S| - |S'| - |N(u_i)| \geq \Delta((1-\epsilon/5) - p\epsilon/100 - (1-\epsilon)) > \epsilon\Delta/2$.

Order the set $S' = \{u_1, \ldots, u_k\}$ in increasing order by vertex ID. Define $Q_i = \Psi(u_i) \cap \Psi(v)$. Define Q_i^{good} as the subset of colors $c \in Q_i$ such that c is selected by some vertex $w \in S_i$, but c is not selected by any vertex in $(N_{\text{out}}(w) \cup N_{\text{out}}(u_i)) \setminus S'$. Define the following events.

E_i^{good}: u_i selects a color $c \in Q_i^{\text{good}}$.

E_i^{bad}: the number of colors in Q_i^{good} is less than $p\epsilon(\Delta+1)/8$.

E_i^{repeat}: the color selected by u_i is also selected by some vertices in $\{u_1, \ldots, u_{i-1}\}$.

Let X_i be the indicator random variable that either E_i^{good} or E_i^{bad} occurs, and let $X = \sum_{i=1}^k X_i$. Let Y_i be the indicator random variable that E_i^{repeat} occurs, and let $Y = \sum_{i=1}^k Y_i$. Suppose that E_i^{good} occurs. Then there must exist a vertex $w \in S_i$ such that both u_i and w successfully color themselves c. Notice that w and u_i are not adjacent. Thus, $X - Y \leq f_2(v)$, given that E_i^{bad} does not occur, for each $i \in [1, k]$. Notice that $\Pr[E_i^{\text{bad}}] = \exp(-\Omega(\epsilon^2\Delta))$ (by Lemma 6.18 and the definition of Q_i^{good}), and up to this negligible error we can assume that E_i^{bad} does not occur. In what follows, we prove concentration bounds on X and Y, which together imply the lemma.

We show that $X \geq \frac{p^3\epsilon^2\Delta}{1000}$ with probability $1 - \exp(-\Omega(\epsilon^2\Delta))$. It is clear that $\Pr[X_i = 1] \geq p \cdot \frac{p\epsilon(\Delta+1)/8}{\Delta+1} = \frac{p^2\epsilon}{8}$.[35] Thus, $\Pr[X \leq t]$ is upper bounded by $\Pr[\text{Binomial}(n', p') \leq t]$ with $n' = |S'| = \frac{p\epsilon\Delta}{100}$ and $p' = \frac{p^2\epsilon}{8}$. We set $t = \frac{p^3\epsilon^2\Delta}{1000} < n'p'$. According to a tail bound of binomial distribution, $\Pr[X \leq t] \leq \exp(\frac{-(n'p'-t)^2}{2n'p'}) = \exp(-\Omega(\epsilon^2\Delta))$.

We show that $Y \leq \frac{p^3\epsilon^2\Delta}{2000}$ with probability $1 - \exp(-\Omega(\epsilon^2\Delta))$. It is clear that $\Pr[Y_i = 1] \leq p \cdot \frac{(i-1)}{\Delta+1} \leq \frac{p^2\epsilon}{100}$ regardless of the colors selected by vertices in $\{u_1, \ldots, u_{i-1}\}$. We have $\mu = \mathrm{E}[Y] \leq \frac{p^2\epsilon}{100} \cdot |S'| = \frac{p^3\epsilon^2\Delta}{10{,}000}$. Thus, by a Chernoff bound (with $\delta = 4$), $\Pr[Y \geq \frac{p^3\epsilon^2\Delta}{2000}] \leq \Pr[Y \geq (1+\delta)\mu] \leq \exp(-\delta\mu/3) = \exp(-\Omega(\epsilon^2\Delta))$.

To summarize, with probability at least $1 - \exp(-\Omega(\epsilon^2\Delta))$, we have $X - Y \geq p^3\epsilon^2\Delta/1000 - p^3\epsilon^2\Delta/2000 = p^3\epsilon^2\Delta/2000$. □

Lemma 6.21. *The number of vertices in $N(v)$ that remain uncolored after* OneShotColoring *is at least $(1 - 1.5p)|N(v)|$, with probability at least $1 - \exp(-\Omega(|N(v)|))$.*

Proof. Let X be the number of vertices in $N(v)$ participating in OneShotColoring. It suffices to show that $X \leq 1.5p|N(v)|$ with probability $1 - \exp(-\Omega(|N(v)|))$. Since a vertex participates with probability p,

$$\Pr[X \geq (1 + 1/2)p|N(v)|] \leq \exp\left(-\frac{(1/2)^2 p|N(v)|}{3}\right) = \exp(-\Omega(|N(v)|))$$

by Chernoff bound with $\delta = 1/2$. □

[35] In the calculation of X, we first reveal all colors selected by vertices in $V \setminus S'$, and then we reveal the colors selected by $u_1, \ldots, u_k$ in this order. The value of X_i is determined when the color selected by u_i is revealed. Regardless of the colors selected by vertices in $V \setminus S'$ and $\{u_1, \ldots, u_{i-1}\}$, we have $\Pr[X_i = 1] \geq \frac{p^2\epsilon}{8}$.

Chapter 7

Distributed Triangle Detection via Expander Decomposition

7.1 Overview

We consider Triangle Detection problems in distributed networks. In the LOCAL model [127], which has no limit on bandwidth, all variants of Triangle Detection can be solved in exactly *one* round of communication: every vertex v simply announces its neighborhood $N(v)$ to all neighbors. However, in models that take bandwidth into account, e.g., CONGEST, Triangle Detection becomes significantly more complicated. Whereas many graph optimization problems studied in the CONGEST model are intrinsically "global" (i.e., require at least diameter time) [2, 56, 57, 73, 74, 89, 101], Triangle Detection is somewhat unusual in that it can, in principle, be solved using only locally available information.

The Congested Clique Model. Remember that the CONGESTED-CLIQUE model is a variant of CONGEST that allows all-to-all communication. One of the main reasons that some problems can be solved efficiently in CONGESTED-CLIQUE is due to Lenzen's routing [105]. As long as each vertex v is the source and the destination of at most $O(n)$ messages, we can deliver all messages in $O(1)$ rounds. Using this routing algorithm [105] as a communication primitive, many parallel algorithms can be transformed to efficient CONGESTED-CLIQUE algorithms [34].

Distributed Routing in Almost Mixing Time. A *uniform lazy random walk* moves a token around an undirected graph by iteratively applying the following process for some

number of steps: with probability 1/2 the token stays at the current vertex and otherwise it moves to a uniformly random neighbor. In a connected graph $G = (V, E)$, the stationary distribution of a lazy random walk is $\pi(u) = \deg(u)/(2|E|)$. Informally, the *mixing time* $\tau_{\text{mix}}(G)$ of a connected graph G is the minimum number of lazy random walk steps needed to get within a negligible distance of the stationary distribution. Formally:

Definition 7.1 (Mixing time [77])**.** *Let $p_t^s(v)$ be the probability that after t steps of a lazy random walk starting at s, the walk lands at v. The mixing time $\tau_{\text{mix}}(G)$ is the minimum t such that for all $s \in V$ and $v \in V$, we have $|p_t^s(v) - \pi(v)| \leq \pi(v)/|V|$.*

Ghaffari, Kuhn, and Su [77] proved that if each vertex v is the source and the destination of at most $O(\deg(v))$ messages, then all messages can be routed to their destinations in $\tau_{\text{mix}}(G) \cdot 2^{O(\sqrt{\log n \log\log n})}$ rounds. The $2^{O(\sqrt{\log n \log\log n})}$ factor has recently been improved [78] to $2^{O(\sqrt{\log n})}$. The implication of this result is that many problems that can be solved efficiently in the CONGESTED-CLIQUE can also be solved efficiently in CONGEST, *but only if $\tau_{\text{mix}}(G)$ is small*. In particular, MST can be solved in $\tau_{\text{mix}}(G) \cdot 2^{O(\sqrt{\log n})}$ rounds in CONGEST [78]. This shows that the $\tilde{\Omega}(\sqrt{n})$ lower bound [128, 133] can be bypassed in networks with small $\tau_{\text{mix}}(G)$.

At this point, a natural question to ask is whether or not this line of research [77, 78] can be extended to a broader class of graphs (that may have high $\tau_{\text{mix}}(G)$), or even general graphs. The main contribution of this chapter is to show that this is in fact doable, and based on this approach we improve the state-of-the-art algorithms for triangle detection, counting, and enumeration.

Expander Decomposition. It is well known that *any* graph can be decomposed into connected components of conductance $\Omega(\epsilon/\log n)$ (and hence $\text{poly}(\epsilon^{-1}, \log n)$ mixing time) after removing at most an ϵ-fraction of the edges [9, 94, 126, 139, 141]. Moshkovitz and Shapira [113] showed that this bound is essentially tight. In particular, removing any constant fraction ϵ of the edges, the remaining components have conductance at most $O((\log\log n)^2/\log n)$.

The task of finding a good expander decomposition (optimizing the number of removed edges and the conductance of the components) was first studied by Kannan Vempala and Vetta [94] (in the sequential computation model), and was further studied in many other subsequent works [10, 117, 118, 126, 132, 139, 141]. The expander decomposition has a

wide range of applications, and it has been applied to solving linear systems [139], unique games [9, 131, 141], minimum cut [96], and dynamic algorithms [114].

In the expander decomposition algorithm of [139], they used random walks to explore the graph locally to find a sparse cut. If the output cut is S, then the time spent is $\tilde{O}(\mathrm{Vol}(S))$.[36] By iteratively finding a sparse cut and removing it from the graph, in $\tilde{O}(|E|)$ time an expander decomposition is obtained in which all components have $\Omega(1/\mathrm{poly}\log(n))$ conductance.

In this work, we show that a variant of expander decomposition can be constructed efficiently in the CONGEST model. The new twist is to partition the edge set in *three* parts, rather than two (i.e., removed and remaining edges).

Distributed Triangle Detection. Many variants of the triangle detection problem have been studied in the literature [34, 90].

Triangle Detection. Each vertex v reports a bit b_v, and $\bigvee_v b_v = 1$ if and only if the graph contains a triangle.

Triangle Counting. Each vertex v reports a number t_v, and $\sum_v t_v$ is exactly the total number of triangles in the graph.

Triangle Enumeration. Each vertex v reports a list L_v of triangles, and $\bigcup_v L_v$ contains exactly those triangles in the graph.

Local Triangle Enumeration. It may be desirable that every triangle be reported by one of the three participating vertices. It is required that L_v only contain triangles involving v.

Dolev, Lenzen, and Peled [50, Remark 1] showed that Triangle Enumeration can be solved deterministically in $O(n^{1/3}/\log n)$ time in the CONGESTED-CLIQUE. Censor-Hillel et al. [34] presented an algorithm for Triangle Detection and Counting in CONGESTED-CLIQUE that takes $\tilde{O}(n^{1-(2/\omega)+o(1)}) = o(n^{0.158})$ time via a reduction to matrix multiplication. Izumi and Le Gall [90] showed that in CONGEST, the Detection and Enumeration problems can be solved in $\tilde{O}(n^{2/3})$ and $\tilde{O}(n^{3/4})$ time, respectively. They also proved that in both CONGEST and CONGESTED-CLIQUE, the Enumeration problem requires $\Omega(n^{1/3}/\log n)$ time, improving an earlier $\Omega(n^{1/3}/\log^3 n)$ bound of Pandurangan et al. [123]. Izumi and Le Gall [90] proved a large separation between

[36] By definition, $\mathrm{Vol}(S) = \sum_{v \in S} \deg(v)$.

the complexity of the Enumeration and Local Enumeration problems. If triangles must be reported by a participating vertex, $\Omega(n/\log n)$ time is necessary (and sufficient) in CONGEST/CONGESTED-CLIQUE. More generally, the lower bound on Local Enumeration is $\Omega(\Delta/\log n)$ when the maximum degree is Δ.

In this chapter, we show that Triangle Detection, Enumeration, and Counting can be solved in $\tilde{O}(n^{1/2})$ time in CONGEST. This result is achieved by a combination of our new distributed graph partition algorithm, the multi-commodity routing of [77, 78], and a randomized version of the CONGESTED-CLIQUE algorithm for Triangle Enumeration of [34, 50]. We also show that when the input graph has high conductance/low mixing time, that Triangle Enumeration can be solved even faster, in $O(\tau_{\text{mix}}(G)n^{1/3+o(1)})$ time.

7.1.1 Technical Overview

Consider a graph $G = (V, E)$. For a vertex subset S, we write $\text{Vol}(S)$ to denote $\sum_{v\in S}\deg(v)$. Note that by default the degree is with respect to the original graph G. We write $\bar{S} = V \setminus S$, and let $\partial(S) = E(S, \bar{S})$ be the set of edges $e = \{u, v\}$ with $u \in S$ and $v \in \bar{S}$. The *sparsity* (or *conductance*) of a cut $(S, \bar{S})$ is defined as $\Phi(S) = |\partial(S)|/\min\{\text{Vol}(S), \text{Vol}(\bar{S})\}$. The *conductance* Φ_G of a graph G is the minimum value of $\Phi(S)$ over all vertex subsets S.

We have the following relation [91] between the mixing time $\tau_{\text{mix}}(G)$ and conductance Φ_G:

$$\Theta\left(\frac{1}{\Phi_G}\right) \leq \tau_{\text{mix}}(G) \leq \Theta\left(\frac{\log n}{\Phi_G^2}\right).$$

In particular, if the inverse of the conductance is $n^{o(1)}$, then the mixing time is also $n^{o(1)}$.

Our Graph Partition. We introduce a new, efficiently computable graph decomposition that partitions the edge set into *three* parts.

Definition 7.2. *An* n^δ-decomposition *of a graph* $G = (V, E)$ *is a tripartition of the edge set* $E = E_m \cup E_s \cup E_r$ *satisfying the following conditions.*

(*a*) *Each connected component induced by* E_m *has* $O(\text{poly}\log n)$ *mixing time, and each vertex in the component has* $\Omega(n^\delta)$ *incident edges in* E_m*. That is, for each vertex* $v \in V$*, either* $\deg_{E_m}(v) = 0$ *or* $\deg_{E_m}(v) = \Omega(n^\delta)$*.*

(*b*) $E_s = \bigcup_{v\in V} E_{s,v}$*, where* $E_{s,v}$ *is a subset of edges incident to* v *and* $|E_{s,v}| \leq n^\delta$*. We*

view $E_{s,v}$ as oriented *away from v. The overall orientation on E_s is* acyclic*, which certifies that E_s has arboricity*[37] *at most n^δ. Each vertex v knows $E_{s,v}$.*

(c) $|E_r| \leq |E|/6$.

Throughout the chapter we assume $\delta \in (0,1)$ is a constant. The main difference between our graph partition and the ones in other works [139] is that we allow a set E_s that induces a low arboricity subgraph. The purpose of having the set E_s is to allow us to design an efficient CONGEST algorithm to construct the partition. In the sequential computation model, a common approach to find a graph partition is to iteratively find a vertex set S with small $\Phi(S) = O(1/\text{poly}\log(n))$, and then include the boundary edges $\partial(S)$ in the set E_r and remove them from the current graph. The number of iterations can be as high as $\tilde{\Theta}(n)$ since we could have $|S| = \tilde{O}(1)$.

To reduce the number of iterations to at most $O(n^{1-\delta})$, before we start to find S, we do a preprocessing step that removes low degree vertices in such a way that each vertex has degree at least $\Omega(n^\delta)$ in the remaining graph. This guarantees that $|S| = \Omega(n^\delta)$, and so the number of iterations can be upper bounded by $O(n^{1-\delta})$, since the total number of vertices is n.

7.1.2 Additional Related Works

Drucker et al. [51] showed an $\Omega\left(\frac{n}{e^{\sqrt{\log n}}\log n}\right)$ lower bound for triangle detection in the *broadcast* CONGESTED-CLIQUE model, where each vertex can only broadcast one message to all other vertices in each round. In the CONGEST model, lower bounds for finding a triangles and other motifs (subgraphs) has been studied in [1, 51, 81, 98]. The problem of detecting a k-cycle has an $\tilde{\Omega}(\sqrt{n})$ lower bound, for any even number $k \geq 4$ [51, 98]. Detecting a k-clique requires $\tilde{\Omega}(\sqrt{n})$ rounds for every $4 \leq k \leq \sqrt{n}$, and $\tilde{\Omega}(n/k)$ rounds for every $k \geq \sqrt{n}$ [46].

Any *one-round* deterministic algorithm for the *triangle membership problem* (each vertex decides whether it belongs to a triangle) requires messages of size $\Omega(\Delta \log n)$ [1], which meets the trivial upper bound; for the randomized model, there is an $\Omega(\Delta)$ lower bound [64]. The distributed triangle detection problem has also been studied in the property testing setting in the CONGEST model [58].

[37] The arboricity of a graph is the minimum number α such that its edge set can be partitioned into α forests.

Das Sarma et al. [134] first studied the distributed sparsest cut problem. Specifically, given two parameters b and ϕ, if there exists a cut of balance at least b and conductance at most ϕ, their algorithm outputs a cut of conductance at most $\tilde{O}(\sqrt{\phi})$ in $\tilde{O}((n + (1/\phi))/b)$ rounds. This result was later improved by Kuhn and Molla [104] to $\tilde{O}(D+1/(b\phi))$.[38] Their algorithms are built upon techniques in [49].

7.1.3 Organization

In Section 7.2 we present a new distributed algorithm for partitioning a graph into expanding subgraphs and a low-arboricity subgraph. A key subroutine for finding a sparse cut is described in Section 7.3. Section 7.4 presents Triangle Enumeration algorithms for both expanding graphs and general graphs.

7.2 Algorithm for Graph Partitioning

We first introduce some notation. Let $\deg_H(v)$ be the degree of v in the subgraph H, or in the graph induced by edge/vertex set H. Let $V(E^*)$ be the set of vertices induced by the edge set $E^* \subseteq E$. The *strong diameter* of a subgraph H of G is defined as $\max_{u,v \in H} \mathrm{dist}_H(u, v)$ and the *weak diameter* of H is $\max_{u,v \in H} \mathrm{dist}_G(u, v)$.

The goal of this section is to prove the following theorem.

Theorem 7.1. *Given a graph $G = (V, E)$ with $n = |V|$, we can find, w.h.p., an n^δ-decomposition in $\tilde{O}(n^{1-\delta})$ rounds in the* CONGEST *model.*

The algorithm for Theorem 7.1 is based on repeated application of a black box algorithm $\mathcal{A}^*$, which is given a subgraph $G' = (V', E')$ of the original graph $G = (V, E)$, where $V' = V(E')$, $n' = |V'|$, and $m' = |E'|$. In $\mathcal{A}^*$, vertices may halt the algorithm at different times.

Specification of the Black Box. The goal of $\mathcal{A}^*$ is, given $G' = (V', E')$, to partition E' into $E' = E'_m \cup E'_s \cup E'_r$ satisfying some conditions. The edge set E'_m is partitioned into $E'_m = \bigcup_{i=1}^{t} \mathcal{E}_i$ for some t. We write $\mathcal{V}_i = V(\mathcal{E}_i)$ and $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, and define $S = V' \setminus \left(\bigcup_{i=1}^{t} \mathcal{V}_i\right)$.

[38] Kuhn and Molla [104] further claimed that the output cut of their algorithm has balance at least $b/2$, but this claim turns out to be incorrect (Anisur Rahaman Molla, personal communication, 2018).

(C1) The vertex sets $\mathcal{V}_1, \ldots, \mathcal{V}_t, S$ are disjoint and partition V'.

(C2) The edge set E'_s can be decomposed as $E'_s = \bigcup_{v \in S} E'_{s,v}$, where $E'_{s,v}$ is a subset of edges incident to v, viewed as oriented away from v. This orientation is acyclic. For each vertex v such that $E'_{s,v} \neq \emptyset$, we have $|E'_{s,v}| + \deg_{E'_m}(v) \leq n^\delta$. Each vertex v knows the set $E'_{s,v}$.

(C3) Consider a subgraph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$. Vertices in $\mathcal{V}_i$ halt after the same number of rounds, say K. Exactly one of the following subcases will be satisfied.

(C3-1) All vertices in $\mathcal{V}_i$ have degree $\Omega(n^\delta)$ in the subgraph $\mathcal{G}_i$, each connected component of $\mathcal{G}_i$ has $O(\text{poly} \log n)$ mixing time, and $K = O(\text{poly} \log n)$. Furthermore, every vertex in $\mathcal{V}_i$ knows that they are in this sub-case.

(C3-2) $|\mathcal{V}_i| \leq n' - \tilde{\Omega}(Kn^\delta)$, and every vertex in $\mathcal{V}_i$ knows they are in this subcase.

(C4) Each vertex $v \in S$ halts in $\tilde{O}(n'/n^\delta)$ rounds.

(C5) The following inequality is met:

$$E'_r \leq \Big(|E'| \log |E'| - \sum_{i=1}^{t} |\mathcal{E}_i| \log |\mathcal{E}_i|\Big)/(6 \log m).$$

(C6) Each cluster $\mathcal{V}_i$ has a distinct identifier. When a vertex $v \in \mathcal{V}_i$ terminates, v knows the identifier of $\mathcal{V}_i$. If $v \in S$, v knows that it belongs to S.

We briefly explain the intuition behind these conditions. The algorithm $\mathcal{A}^*$ will be applied recursively to all subgraphs $\mathcal{G}_i$ that have yet to satisfy the minimum degree and mixing time requirements specified in Theorem 7.1 and Definition 7.2. Because vertices in different components halt at various times, they also may begin these recursive calls at different times.

The goal of (C2) is to make sure that once a vertex v has $E'_{s,v} \neq \emptyset$, the total number of edges added to $E_{s,v}$ cannot exceed n^δ. The goal of (C3) is to guarantee that the component size drops at a fast rate. The idea of (C5) is that the size of E'_r can be mostly charged to the number of the edges in the small-sized edge sets $\mathcal{E}_i$; this is used to bound the size of E_r in the final output of our graph partitioning algorithm.

Note that in general the strong diameter of a subgraph $\mathcal{G}_i$ can be much higher than the maximum running time of vertices in $\mathcal{G}_i$, and it could be possible that $\mathcal{G}_i$ is not even a connected subgraph of G. However, (C6) guarantees that each vertex $v \in \mathcal{V}_i$ still knows

that it belongs to $\mathcal{V}_i$. This property allows us to recursively execute $\mathcal{A}^*$ on each subgraph $\mathcal{G}_i$.

Lemma 7.1. *There is an algorithm $\mathcal{A}^*$ that finds a partition $E' = E'_m \cup E'_s \cup E'_r$ meeting the above specification in the* CONGEST *model, w.h.p.*

Assuming Lemma 7.1, we are now in a position to prove Theorem 7.1.

Proof of Theorem 7.1. Let $\mathcal{A}^*$ be the algorithm for Lemma 7.1. Initially, we apply $\mathcal{A}^*$ with $G' = G$, and this returns a partition $E' = E'_m \cup E'_s \cup E'_r$.

For each subgraph $\mathcal{G}_i$ in the partition output by an invocation of $\mathcal{A}^*$, do the following. If $\mathcal{G}_i$ satisfies (C3-1), by definition it must have $O(\text{poly}\log n)$ mixing time, and all vertices in $\mathcal{G}_i$ have degree $\Omega(n^\delta)$ in $\mathcal{G}_i$; we add the edge set $\mathcal{E}_i$ to the set E_m and all vertices in $\mathcal{V}_i$ halt. Otherwise we apply the algorithm recursively to $\mathcal{G}_i$, i.e., we begin by applying $\mathcal{A}^*$ to $G' = \mathcal{G}_i$ to further partition its edges. All recursive calls proceed in parallel, but may begin and end at different times. Conditions (C1) and (C6) guarantee that this is possible. (Note that if $\mathcal{G}_i$ is disconnected, then each connected component of $\mathcal{G}_i$ will execute the algorithm in isolation.)

Initially $E_r = \emptyset$ and $E_s = \emptyset$. After each invocation of $\mathcal{A}^*$, we update $E_r \leftarrow E_r \cup E'_r$, $E_s \leftarrow E_s \cup E'_s$, and $E_{s,u} \leftarrow E_{s,u} \cup E'_{s,u}$ for each vertex u.

Analysis. We verify that the three conditions of Definition 7.2 are satisfied. First of all, note that each connected component of E_m terminated in (C3-1) must have $O(\text{poly}\log n)$ mixing time, and all vertices in the component have degree $\Omega(n^\delta)$ within the component. Condition (a) of Definition 7.2 is met. Next, observe that Condition (b) of Definition 7.2 is met due to (C2). If the output of $\mathcal{A}^*$ satisfies that $E'_{s,v} \neq \emptyset$, then $|E_{s,v}|$ together with the number of remaining incident edges (i.e., the ones in E'_m) is less then n^δ. Therefore, $|E_{s,v}|$ cannot exceed n^δ, since only the edges in E'_m that are incident to v can be added to $E_{s,v}$ in future recursive calls. Lastly, we argue that (C5) implies that Condition (c) of Definition 7.2 is satisfied. Assume, inductively, that a recursive call on edge set $\mathcal{E}_i$ eventually contributes at most $|\mathcal{E}_i| \log |\mathcal{E}_i|/(6 \log m)$ edges to E_r. It follows from (C5) that the recursive call on edge set E' contributes $|E'| \log |E'|/(6 \log m)$ edges to E_r. We conclude that $|E_r| \leq |E| \log |E|/(6 \log |E|) = |E|/6$.

Now we analyze the round complexity. In one recursive call of $\mathcal{A}^*$, consider a component $\mathcal{G}_i$ in the output partition, and let K be the running time of vertices in $\mathcal{V}_i$. Due to

(C3), there are two cases. If $\mathcal{G}_i$ satisfied (C3-1), it will halt in $K = O(\text{poly}\log n)$ rounds. Otherwise, (C3-2) is met, and we have $|\mathcal{V}_i| \leq n' - \tilde{\Omega}(Kn^{\delta})$. Let $v \in V$ be any vertex, and let $K_1, \ldots, K_z$ be the running times of all calls to $\mathcal{A}^*$ that involve v. (Whenever v ends up in S or in a component satisfying (C3-1) it halts permanently, so $K_1, \ldots, K_{z-1}$ reflect executions that satisfy (C3-2) upon termination.) Then we must have $\sum_{i=1}^{z} K_i \leq \tilde{O}(n/n^{\delta}) + O(\text{poly}\log n) = \tilde{O}(n^{1-\delta})$. Thus, the whole algorithm stops within $\tilde{O}(n^{1-\delta})$ rounds. □

7.2.1 Subroutines

Before proving Lemma 7.1, we first introduce some helpful subroutines. Lemma 7.3 shows that for subgraphs of sufficiently high strong diameter, we can find a sparse cut of the subgraph, with runtime proportional to the strong diameter. Lemma 7.4 offers a procedure that removes a set of edges in such a way that the vertices in the remaining graph have high degree, and the removed edges form a low arboricity subgraph. Lemma 7.5 shows that if a subgraph already has a low conductance cut, then we can efficiently find a cut of similar quality.

All these subroutines are applied to a connected subgraph $G^* = (V^*, E^*)$ of the underlying network $G = (V, E)$, and the computation does not involve vertices outside of G^*. In subsequent discussion in this section, the parameters n and m are always defined as $n = |V|$ and $m = |E|$, which are independent of the chosen subgraph G^*.

Lemma 7.2. *Let m and D be two numbers. Let $(a_1, \ldots, a_D)$ be a sequence of positive integers such that $D \geq 48 \log^2 m$ and $\sum_{i=1}^{D} a_i \leq m$. Then there exists an index j such that $j \in [D/4, 3D/4]$ and*

$$a_j \leq \frac{1}{12 \log m} \cdot \min\left(\sum_{i=1}^{j-1} a_i,\ \sum_{i=j+1}^{D} a_i\right).$$

Proof. Define $S_k = \sum_{i=1}^{k} a_i$ to be the kth prefix sum. By symmetry, we may assume $S_{\lfloor D/2 \rfloor} \leq S_D - S_{\lfloor D/2 \rfloor}$, since otherwise we can reverse the sequence. Scan each index j from $D/4$ to $D/2$. If an index j does not satisfy $a_j \leq \frac{1}{12 \log m} \cdot S_{j-1}$, then this implies that $S_j > S_{j-1}\left(1 + \frac{1}{12 \log m}\right)$. If no index $j \in [D/4, D/2]$ satisfies this condition then $S_{\lfloor D/2 \rfloor}$ is

larger than

$$\begin{aligned} & S_{\lfloor D/4 \rfloor} \cdot \left(1 + \frac{1}{12 \log m}\right)^{D/4} \\ & \geq S_{\lfloor D/4 \rfloor} \cdot \left(1 + \frac{1}{12 \log m}\right)^{12 \log^2 m} \\ & \geq S_{\lfloor D/4 \rfloor} \cdot m, \end{aligned}$$

which is impossible since $\sum_{i=1}^{D} a_i \leq m$. Therefore, there must exist an index $j \in [D/4, D/2]$ such that $a_j \leq \frac{1}{12 \log m} \cdot S_{j-1} = \frac{1}{12 \log m} \cdot \sum_{i=1}^{j-1} a_i$. By our assumption that $S_{\lfloor D/2 \rfloor} \leq S_D - S_{\lfloor D/2 \rfloor}$, we also have $a_j \leq \frac{1}{12 \log m} \cdot \min\left(\sum_{i=1}^{j-1} a_i,\ \sum_{i=j+1}^{D} a_i\right)$. □

In Lemma 7.3, the requirement that there are no edges linking two vertices in V_{low} implies that the strong diameter of $G^* = (V^*, E^*)$ is $O(n^{1-\delta})$, and so the runtime of Lemma 7.3 is always at most $O(n^{1-\delta})$.

Lemma 7.3 (High Diameter subroutine)**.** *Let $G^* = (V^*, E^*)$ be a connected subgraph and $x \in V^*$ be a vertex for which $\tilde{D} = \max_{v \in V^*} \text{dist}_{G^*}(x, v) \geq 48 \log^2 m$. Define $V_{\text{low}} = \{v \in V^* \mid \deg_{G^*}(v) \leq n^\delta/2\}$. Suppose there are no edges linking two vertices in V_{low}. Then we can find a cut $(C, \bar{C})$ of G^* such that $\min(|C|, |\bar{C}|) \geq \frac{\tilde{D}}{32} n^\delta$ and $\partial(C) \leq \min(\text{Vol}(C), \text{Vol}(\bar{C}))/(12 \log m)$ in $O(\tilde{D})$ rounds deterministically in the* CONGEST *model. Each vertex in V^* knows whether or not it is in C.*

Proof. The algorithm is as follows. First, build a BFS tree of G^* rooted at $x \in V^*$ in $O(\tilde{D})$ rounds. Let L_i be the set of vertices of level i in the BFS tree, and let p_i be the number of edges $e = \{u, v\}$ such that $u \in L_i$ and $v \in L_{i+1}$. We write $L_{a..b} = \bigcup_{i=a}^{b} L_i$. In $O(\tilde{D})$ rounds we can let the root x learn the sequence $(p_1, \ldots, p_{\tilde{D}})$.

Note that in a BFS tree, edges do not connect two vertices in non-adjacent levels. By Lemma 7.2, there exists an index $j \in [\tilde{D}/4, 3\tilde{D}/4]$ such that

$$\begin{aligned} p_j & \leq \frac{1}{12 \log m} \cdot \min\left(\sum_{i=1}^{j-1} p_i,\ \sum_{i=j+1}^{\tilde{D}} p_i\right) \\ & \leq \frac{1}{12 \log m} \cdot \min\left(\text{Vol}(L_{1..j}), \text{Vol}(L_{j+1..\tilde{D}})\right), \end{aligned}$$

and such an index j can be computed locally at the vertex x.

The cut is chosen to be $C = L_{1..j}$, so we have $\partial(C) \leq \min(\mathrm{Vol}(C), \mathrm{Vol}(\bar{C}))/(12 \log m)$. As for the second condition, due to our assumption in the statement of the lemma, for any two adjacent levels L_i, L_{i+1}, there must exist a vertex $v \in L_i \cup L_{i+1}$ such that $v \notin V_{\mathrm{low}}$. By definition of V_{low}, v has more than $n^\delta/2$ neighbors in G^*, and they are all within $L_{i-1..i+2}$. Thus, the number of vertices within any four consecutive levels must be greater than $n^\delta/2$. Since $j \in [\tilde{D}/4, 3\tilde{D}/4]$, we have

$$\min(|C|, |\bar{C}|) \geq \frac{\tilde{D}}{4}/4 \cdot n^\delta/2 \geq \frac{\tilde{D}}{32} n^\delta.$$

To let each vertex in V^* learn whether or not it is in C, the root x broadcasts the index j to all vertices in G^*. After that, each vertex in level smaller than or equal to j knows that it is in C; otherwise it is in $\bar{C}$. □

Intuitively, Lemma 7.4 says that after the removal of a subgraph of small arboricity (i.e., the edge set $E_s^\diamond$), the remaining graph (i.e., the edge set $E^\diamond$) has high minimum degree. The runtime is proportional to the number of removed vertices (i.e., $|V^*| - |V^\diamond|$) divided by the threshold n^δ. Note that the second condition of Lemma 7.4 implies that $E_{s,v}^\diamond = \emptyset$ for all $v \in V^\diamond$.

Lemma 7.4 (Low Degree subroutine)**.** *Let $G^* = (V^*, E^*)$ be a connected subgraph with strong diameter D. We can partition $E^* = E^\diamond \cup E_s^\diamond$ meeting the following two conditions.*

1. *Let $V^\diamond$ be the set of vertices induced by $E^\diamond$. Each $v \in V^\diamond$ has more than $n^\delta/2$ incident edges in $E^\diamond$.*
2. *The edge set $E_s^\diamond$ is further partitioned as $E_s^\diamond = \bigcup_{v \in V^* \setminus V^\diamond} E_{s,v}^\diamond$, where $E_{s,v}^\diamond$ is a subset of incident edges of v, and $|E_{s,v}^\diamond| \leq n^\delta$. Each vertex v knows $E_{s,v}^\diamond$.*

This partition can be found in $O(D + (|V^*| - |V^\diamond|)/n^\delta)$ rounds deterministically in the CONGEST *model.*

Proof. To meet Condition 1, a naive approach is to iteratively "peel off" vertices that have degree at most $n^\delta/2$, i.e., put all their incident edges in E_s, so long as any such vertex exists. On some graphs this process requires $\Omega(n)$ peeling iterations.

We solve this issue by doing a batch deletion. First, build a BFS tree of G^* rooted at an arbitrary vertex $x \in V^*$. We use this BFS tree to let x count the number of vertices that have degree less than n^δ in the remaining subgraph in $O(D)$ rounds.

The algorithm proceeds in iterations. Initially we set $E^\diamond \leftarrow E^*$ and $E_s^\diamond \leftarrow \emptyset$. In each iteration, we identify the subset $Z \subseteq V^*$ whose vertices have at most n^δ incident edges in $E^\diamond$. We orient all the $E^\diamond$-edges touching Z away from Z, if one endpoint is in Z, or away from the endpoint with smaller ID, if both endpoints are in Z. Edges incident to v oriented away from v are added to $E_{s,v}^\diamond$ and removed from $E^\diamond$. The root x then counts the number $z = |Z|$ of such vertices via the BFS tree. If $z > n^\delta/2$, we proceed to the next iteration; otherwise we terminate the algorithm.

The termination condition ensures that each vertex has degree at least $(n^\delta+1)-z > n^\delta/2$, and so Condition 1 is met. It is straightforward to see that the set $E_s^\diamond$ generated by the algorithm meets Condition 2, since for each v, we only add edges to $E_{s,v}^\diamond$ once, and it is guaranteed that $|E_{s,v}^\diamond| \leq n^\delta$. Tie-breaking according to vertex-ID ensures the orientation is acyclic.

Throughout the process, each time one vertex puts any edges into $E_s^\diamond$, it no longer stays in $V^\diamond$. Each iteration can be done in $O(D)$ time. We proceed to the next iteration only if there are more than $n^\delta/2$ vertices being removed from $V^\diamond$. A trivial implementation can lead to an algorithm taking $O(D\left\lceil(|V^*| - |V^\diamond|)/n^\delta\right\rceil))$ rounds. The round complexity can be further improved to $O(D + (|V^*| - |V^\diamond|)/n^\delta)$ by pipelining the iterations. At some point the root x detects that iteration i was the last iteration; in $O(D)$ time it broadcasts a message to all nodes instructing them to roll back iterations $i+1, i+2, \ldots$, which have been executed speculatively. □

The proof of the following lemma is deferred to Section 7.3. It is a consequence of combining Lemmas 7.9 and 7.10.

Lemma 7.5 (Low Conductance subroutine)**.** *Let $G^* = (V^*, E^*)$ be a connected subgraph with strong diameter D. Let $\phi \leq 1/12$ be a number. Suppose that there exists a subset $S \subset V^*$ satisfying*

$$\mathrm{Vol}(S) \leq (2/3)\mathrm{Vol}(V^*) \quad \textit{and} \quad \Phi(S) \leq \frac{\phi^3}{19208 \ln^2(|E^*|e^4)}.$$

Assuming such an S exists, there is a CONGEST *algorithm that finds a cut $C \subset V^*$ such that $\Phi(C) \leq 12\phi$ in $O(D+\mathrm{poly}(\log|E^*|, 1/\phi))$ rounds, with failure probability $1/\mathrm{poly}(|E^*|)$. Each vertex in V^* knows whether or not it belongs to C.*

7.2.2 Proof of Lemma 7.1

We prove Lemma 7.1 by presenting and analyzing a specific distributed algorithm, which makes use of the subroutines specified in Lemmas 7.3, 7.4, and 7.5.

Recall that we are given a subgraph with edge set E' and must ultimately return a partition of it into $E'_m \cup E'_s \cup E'_r$. The algorithm initializes $E'_m \leftarrow E'$, $E'_s \leftarrow \emptyset$, and $E'_r \leftarrow \emptyset$. There are two types of special operations.

Remove. In an Remove operation, some edges are moved from E'_m to either E'_s or E'_r. For the sake of a clearer presentation, each such operation is tagged Remove-i, for some index i.

Split. Throughout the algorithm we maintain a partition of the current set E'_m. In a Split operation, the partition subdivided. Each such operation is tagged as Split-i, for some index i, such that Split-i occurs right after Remove-i.

Throughout the algorithm, we ensure that any part $E^\star$ of the partition of E'_m has an identifier that is known to all members of $V(E^\star)$. It is not required that each part forms a connected subgraph. The partition at the end of the algorithm, $E'_m = \bigcup_{i=1}^{t} \mathcal{E}_i$, is the output partition.

Notations. Since we treat E'_m as the "active" edge set and E'_s and E'_r as repositories of removed edges, $\deg(v)$ refers to the degree of v in the subgraph induced by the *current* E'_m. We write $V_{\text{low}} = \{v \in V' \mid \deg(v) \leq n^\delta\}$.

Algorithm. In the first step of the algorithm, we move each edge $\{u, v\} \in E'_m$ in the subgraph induced by V_{low} to the set $E'_{s,u}$ (Remove-1), assuming $\text{ID}(u) < \text{ID}(v)$. Note that breaking ties by vertex-ID is critical to keep the orientation acyclic. This step only removes all edges in the subgraph induced by V_{low}; edges between V_{low} and $V' \setminus V_{\text{low}}$ are left as is, so the identity of V_{low} is unchanged after this step.

After that, E'_m is divided into connected components. Assume these components are $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2), \ldots$, where $V_i = V(E_i)$. Let D_i be the depth of a BFS tree rooted at an arbitrary vertex in G_i. In $O(D_i)$ rounds, the subgraph G_i is assigned an identifier that is known to all vertices in V_i (Split-1). Note that this step is done in parallel for each G_i, and the time for this step is different for each G_i. From now on there will be no communication between different subgraphs in $\{G_1, G_2, \ldots\}$, and we focus on one specific subgraph G_i in the description of the algorithm.

Depending on how large D_i is, there are two cases. If $D_i \geq 48 \log^2 m$, we go to Case 1, otherwise we go to Case 2.

Case 1: In this case, we have $D_i \geq 48 \log^2 m$. Since there are no edges connecting two vertices in V_{low}, we can apply the High Diameter subroutine, Lemma 7.3, which finds a cut $(C, \bar{C})$ of G_i such that $\min(|C|, |V_i \setminus C|) \geq \frac{D_i}{32} n^\delta$ and $\partial(C) \leq \min(\text{Vol}(C), \text{Vol}(V_i \setminus C))/(12 \log m)$ in $O(D_i)$ rounds. Every vertex in V_i knows whether it is in C or not. All edges of the cut $(C, \bar{C})$ are put into E'_r (Remove-2). Then E_i splits into two parts according to the cut $(C, \bar{C})$ (Split-2). After that, all vertices in V_i terminate. (Observe that the part containing the BFS tree root is connected, but the other part is not necessarily connected.)

Case 2: In this case, we have $D_i \leq 48 \log^2 m$. Since $G_i = (V_i, E_i)$ is a small diameter graph, a vertex $v \in V_i$ is able broadcast a message to all vertices in V_i very fast. We apply the Low Degree subroutine, Lemma 7.4, to obtain a partition $E_i = E^\diamond \cup E^\diamond_s$. We add all edges in $E^\diamond_s$ to E'_s in such a way that $E'_{s,v} \leftarrow E'_{s,v} \cup E^\diamond_{s,v}$ for all $v \in V_i \setminus V^\diamond$, where $V^\diamond = V(E^\diamond)$ (Remove-3).

After removing these edges, the remaining edges of E_i are divided into several connected components, but all remaining vertices have degree larger than $n^\delta/2$. Assume these connected components are $G_{i,1} = (V_{i,1}, E_{i,1})$, $G_{i,2} = (V_{i,2}, E_{i,2})$, ... Let $D_{i,j}$ be the depth of the BFS tree from an arbitrary root vertex in $G_{i,j}$. In $O(D_{i,j})$ rounds we compute such a BFS tree and assign an identifier that is known to all vertices in $V_{i,j}$ (Split-3). That is, the remaining edges in E_i are partitioned into $E_{i,1}$, $E_{1,2}$, ...

In what follows, we focus on one subgraph $G_{i,j}$ and proceed to Case 2-a or Case 2-b.

Case 2-a: In this case, $D_{i,j} \geq 48 \log^2 m$. The input specification of the High Diameter subroutine (Lemma 7.3) is satisfied, since every vertex has degree larger than $n^\delta/2$. We apply the High Diameter subroutine to $G_{i,j}$. This takes $O(D_{i,j})$ rounds. This case is similar to Case 1, and we do the same thing as what we do in Case 1, i.e., remove the edges in the cut found by the subroutine (Remove-4), split the remaining edges (Split-4), and then all vertices in $V_{i,j}$ terminate.

Case 2-b: In this case, $D_{i,j} \leq 48 \log^2 m$. Note that every vertex has degree larger than $n^\delta/2$, and $G_{i,j}$ has small diameter. What we do in this case is to test whether $G_{i,j}$ has any low conductance cut; if yes, we will split $E_{i,j}$ into two components. To do so, we apply the

Low Conductance subroutine, Lemma 7.5, with $\phi = \frac{1}{144 \log m}$. Based on the result, there are two cases.

Case 2-b-i: The subroutine finds a set of vertices C that $\Phi(C) \leq 12\phi = \frac{1}{12 \log m}$, and every vertex knows whether it is in C or not. We move $\partial(C)$ to E'_r (Remove-5), and then split the remaining edges into two edge sets according to the cut $(C, \bar{C})$ (Split-5). After that, all vertices in $V_{i,j}$ terminate.

Case 2-b-ii: Otherwise, the subroutine does not return a subset C, and it means with probability at least $1 - 1/\text{poly}(|E_{i,j}|) = 1 - 1/\text{poly}(n)$, there is no cut $(S, \bar{S})$ with conductance less than $\frac{\phi^3}{19208 \ln^2(|E_{i,j}|e^4)} = \Theta(\log^{-5} m)$. Recall the relation between the mixing time $\tau_{\text{mix}}(G_{i,j})$ and the conductance $\Phi = \Phi_{G_{i,j}}$: $\Theta(\frac{1}{\Phi}) \leq \tau_{\text{mix}}(G_{i,j}) \leq \Theta(\frac{\log |V_{i,j}|}{\Phi^2})$ [91]. Therefore, w.h.p., $G_{i,j}$ has $O(\text{poly} \log n)$ mixing time. All vertices in $V_{i,j}$ terminate without doing anything in this step.

Note that in the above calculation, we use the fact that every vertex in $V_{i,j}$ has degree larger than $n^\delta/2$ in $G_{i,j}$, and this implies that $|V_{i,j}| = \Omega(n^\delta)$ and $|E_{i,j}| = \Omega(n^{2\delta})$, and so $\Theta(\log m) = \Theta(\log n) = \Theta(\log |E_{i,j}|) = \Theta(\log |V_{i,j}|)$.

Analysis. We show that the output of $\mathcal{A}^*$ meets its specifications (C1)–(C6). Recall that $E'_m = \bigcup_{i=1}^t \mathcal{E}_i$ is the final partition of the edge set E'_m when all vertices terminate. Once an edge is moved from E'_m to either E'_r or E'_s, it remains there for the rest of the computation. Condition (C1) follows from the fact that each time we do a split operation, the induced vertex set of each part is disjoint. Condition (C6) follows from the fact that each vertex knows which part of E'_m it belongs to after each split operation. In the rest of this section, we prove that the remaining conditions are met.

Claim 4. *Condition (C2) is met.*

Proof. Note that only Remove-1 and Remove-3 involve E'_s. In Remove-1, any $E'_{s,u}$ that becomes non-empty must have had $u \in V_{\text{low}}$, so $\deg(u) \leq n^\delta$ before Remove-1, and therefore $|E'_{s,u}| + \deg(u) \leq n^\delta$ after Remove-1. In Remove-3, the Low Degree subroutine of Lemma 7.4 computes a partition $E_i = E^\diamond \cup E^\diamond_s$, and then we update $E'_{s,u} \leftarrow E'_{s,u} \cup E^\diamond_{s,u}$ for all $u \in V_i \setminus V^\diamond$. By Lemma 7.4, for any u such that $E^\diamond_{s,u} \neq \emptyset$, we have $|E^\diamond_{s,u}| \leq n^\delta$, and $u \notin V^\diamond$, where $V^\diamond$ is the vertex set induced by the remaining edge set $E^\diamond$. In other words, once u puts at least one edge into $E'_{s,u}$, we have $\deg(u) = 0$ after Remove-3. □

Claim 5. *Conditions (C3) and (C4) are met.*

Proof. We need to verify that in each part of the algorithm, we either spend only $O(\text{poly}\log n)$ rounds, or the size of the current component shrinks by $\tilde{\Omega}(n^\delta)$ vertices *per round*.

After removing all edges in the subgraph induced by V_{low}, the rest of E' is partitioned into connected components $\mathcal{E}_1, \mathcal{E}_2, \ldots$. Consider one such component $\mathcal{E}_i$, and suppose it goes to Case 1. We find a sparse cut $(C, \bar{C})$, and moving $\partial(C)$ to E'_r breaks $\mathcal{E}_i$ into $\mathcal{E}_i^1$ and $\mathcal{E}_i^2$. By Lemma 7.3, we have $\min(|C|, |\bar{C}|) \geq \frac{D_i}{32} n^\delta$, so the size of both $V(\mathcal{E}_i^1) = C$ and $V(\mathcal{E}_i^2) = \bar{C}$ are at most $|V(\mathcal{E}_i)| - \frac{D_i}{32} n^\delta \leq n' - \Omega(D_i) n^\delta$. Since the running time for each vertex in $V(\mathcal{E}_i^1)$ and $V(\mathcal{E}_i^2)$ is $O(D_i)$, the condition (C3-2) is met.

Now suppose that $\mathcal{E}_i$ goes to Case 2. Note that the total time spent before it reaches Case 2 is $O(D_i) = \text{poly}\log n$. In Case 2 we execute the Low Degree subroutine of Lemma 7.4, and let the time spent in this subroutine be τ. By Lemma 7.4, it is either the case that (i) $\tau = O(D_i)$ or (ii) the remaining vertex set $V^\diamond$ satisfies $|V(E_i)| - |V^\diamond| = \Omega(\tau n^\delta)$. In other words, if we spend too much time (i.e., $\omega(D_i)$) on this subroutine, we must lose $\Omega(n^\delta)$ vertices per round.

After that, $\mathcal{E}_i$ is split into $\mathcal{E}_{i,1}$, $\mathcal{E}_{i,2}$, ... We consider the set $\mathcal{E}_{i,j}$. If $\mathcal{E}_{i,j}$ goes to Case 2-a, then the analysis is the same as that in Case 1, and so (C3-2) is met.

Now suppose that $\mathcal{E}_{i,j}$ goes to Case 2-b. Note that the time spent during the Low Conductance subroutine of Lemma 7.5 is $O(\text{poly}\log n)$. Suppose that a low conductance cut $(C, \bar{C})$ is found (Case 2-b-i). Since the cut has conductance less than $\frac{1}{12 \log m}$, by the fact that every vertex has degree higher than $n^\delta/2$, we must have $\min(|C|, |\bar{C}|) = \Omega(n^\delta)$. Assume $\mathcal{E}_{i,j} \setminus \partial(C)$ is split into $\mathcal{E}_{i,j}^1$ and $\mathcal{E}_{i,j}^2$. The size of both $V(\mathcal{E}_{i,j}^1)$ and $V(\mathcal{E}_{i,j}^2)$ must be at most $|V(\mathcal{E}_{i,j})| - \Omega(n^\delta)$. Thus, (C3-2) holds for both parts $\mathcal{E}_{i,j}^1$ and $\mathcal{E}_{i,j}^2$.

Suppose that no cut $(C, \bar{C})$ is found (Case 2-b-ii). If the running time K among vertices in $V_{i,j}$ is $O(\text{poly}\log n)$, then (C3-1) holds. Otherwise, we must have $|V_{i,j}| \leq n' - \tilde{\Omega}(K n^\delta)$ due to the Low Degree subroutine, and so (C3-2) holds.

Condition (C4) follows from the the above proof of (C3), since for each part of the algorithm, it is either the case that (i) this part takes $O(\text{poly}\log n)$ time, or (ii) the number of vertices in the current subgraph is reduced by $\tilde{\Omega}(n^\delta)$ *per round*. □

Claim 6. *Condition (C5) is met.*

Proof. Condition (C5) says that after the algorithm $\mathcal{A}^*$ completes, $|E'_r| \leq f$, where

$$f = \left(|E'| \log |E'| - \sum_{i=1}^{t} |\mathcal{E}_i| \log |\mathcal{E}_i| \right) / (6 \log m).$$

We prove the stronger claim that this inequality holds at all times w.r.t. the current edge partition $\mathcal{E}_1 \cup \cdots \cup \mathcal{E}_t$ of E'_m. In the base case this is clearly true, since $t = 1$ and $E' = E'_m = \mathcal{E}_1$ and $E'_r = \emptyset$. Moving edges from E'_m to E'_s increases f and has no effect on E'_r, so we only have to consider the movement of edges from E'_m to E'_r. Note that this only occurs in Remove-i and Split-i, for $i \in \{2, 4, 5\}$, where in these operations we find a cut $(C, \bar{C})$ and split one of the parts $\mathcal{E}_j$ according to the cut. In all cases we have

$$|\partial(C)| \leq \frac{\min(\mathrm{Vol}(C), \mathrm{Vol}(\bar{C}))}{12 \log m}.$$

Suppose that removing $\partial(C)$ splits $\mathcal{E}_j$ into $\mathcal{E}_j^1$ and $\mathcal{E}_j^2$, with $|\mathcal{E}_j^1| \leq |\mathcal{E}_j^2|$ and $C = V(\mathcal{E}_j^1)$. We bound the change in $|E'_r|$ and f separately. Clearly

$$\Delta |E'_r| = |\partial(C)| \leq \frac{2|\mathcal{E}_j^1| + \partial(C)}{12 \log m} \leq \frac{|\mathcal{E}_j^1|}{6 \log m} + \frac{\partial(C)}{12 \log m}.$$

and

$$\begin{aligned}
\Delta f &= \frac{1}{6 \log m} \cdot \left(|\mathcal{E}_j| \log |\mathcal{E}_j| - \sum_{k \in \{1,2\}} |\mathcal{E}_j^k| \log |\mathcal{E}_j^k| \right) \\
&\geq \frac{1}{6 \log m} \cdot \left(|\mathcal{E}_j^1| \log(|\mathcal{E}_j| / |\mathcal{E}_j^1|) + \partial(C) \log |\mathcal{E}_j| \right) \\
&> \Delta |E'_r|. \qquad \text{(Because } |\mathcal{E}_j^1| < |\mathcal{E}_j|/2\text{)}
\end{aligned}$$

Thus, $|E'_r| \leq f$ also holds after Remove-i and Split-i, for $i \in \{2, 4, 5\}$. □

7.3 Algorithm for Finding a Sparse Cut

Recall that in our decomposition routine, we search for a sparse cut in a *subgraph* $G^* = (V(E^*), E^*)$ of G. In this section, we do not care about anything outside of G^*, and so we slightly abuse the notation to write $G = (V, E)$ to denote the subgraph G^*, and we use

$n = |V(G^*)|$ and $m = |E(G^*)|$ to be the number of vertices and edges in the subgraph. In this section we prove Lemma 7.5, which concerns an efficient distributed analogue of Spielman and Teng's [138, 139] `Nibble` routine.

Many existing works [6, 49, 104, 139] have shown that looking at the distribution of random walks is a good approach to finding a sparse cut. The basic idea is to first sample a source vertex s according to the degree distribution, i.e., the probability that v is sampled is $\deg(v)/(2m)$, and do a lazy random walk from s. Assume there is a sparse cut S with conductance $\Phi(S)$, and $\mathrm{Vol}(S) \leq \mathrm{Vol}(V)/2$. If $s \in S$, then the probability distribution of the random walk will be mostly confined to S within the initial $t_0 = O(\frac{1}{\Phi(S)})$ steps. A common way to utilize this observation is to sort the vertices $(v_1, \ldots, v_n)$ in decreasing order of their random walk probability, and it is guaranteed that for some choice of j, the subset $C = \{v_1, \ldots, v_j\}$ is a sparse cut that is approximately as good as S.

The papers [104, 134] adapted this approach to the CONGEST model. If the cut S satisfies that $b \cdot 2|E| \leq \mathrm{Vol}(S)$ (*i.e., S has balance b*), then a cut C satisfying $\Phi(C) = O(\sqrt{\Phi(S)\log n})$ can be found in $\tilde{O}(D + 1/(b\Phi(S)))$ rounds. The algorithm is inefficient when $1/b = \Theta(|E|/\mathrm{Vol}(S))$ is large. The main source of this inefficiency is that if we sample a vertex s according to the degree distribution, then the probability that $s \in S$ is only $O(b)$. This implies that we have to calculate many random walk distributions before we find a desired sparse cut. If we calculate these random walk distributions *simultaneously*, then *we may suffer from a huge congestion issue.*

Spielman and Teng [139] show that a *random walk* distribution with *truncation* (rounding a probability to zero when it becomes too small) can reveal a sparse cut, provided the starting vertex of the random walk is good. The main contribution of this section is a proof that the Spielman-Teng method for finding cuts of conductance roughly ϕ can be implemented in $\mathrm{poly}(\phi^{-1}, \log n)$ time in the CONGEST model, i.e., with no dependence on the balance parameter b.

Terminology. We first review some definitions and results from Spielman and Teng [139]. Let A be the adjacency matrix of the graph G. We assume a 1-1 correspondence between $V(G)$ and $\{1, \ldots, n\}$. In a *lazy random walk*, the walk stays at the current vertex with probability 1/2 and otherwise moves to a random neighbor of the current vertex. The matrix realizing this walk can be expressed as $T = (AD^{-1} + I)/2$, where D is the diagonal matrix with $(d(1), \ldots, d(n))$ on the diagonal, and $d(i) = \deg(i)$.

Let p_t^v be the probability distribution of the lazy random walk that begins at v and walks

for t steps. In the limit, as $t \to \infty$, $p_t(x)$ approaches $d(x)/2m$, so it is natural to measure $p_t(x)$ *relative* to this baseline.

$$\rho_t(x) = p_t(x)/d(x),$$

Define π_t to be the permutation that sorts $V = \{1, \ldots, n\}$ in decreasing order of ρ_t-values, breaking ties by vertex ID. We never actually compute π_t. To implement our algorithms, it suffices that given $\rho_t(u), \rho_t(v), \mathrm{ID}(u), \mathrm{ID}(v)$, we can determine whether or not u precedes v according to π_t.

$$\rho_t(\pi_t(i)) \geq \rho_t(\pi_t(i+1)), \text{ for all } i.$$

Let p be a distribution on V. The truncation operation $[p]_\epsilon$ rounds $p(x)$ to zero if it falls below a threshold that depends on x.

$$[p]_\epsilon(x) = \begin{cases} p(x) & \text{if } p(x) \geq 2\epsilon d(x), \\ 0 & \text{otherwise.} \end{cases}$$

The truncated random walk starting at vertex v is defined as follows. In subsequent discussion we may omit v if it is known implicitly.

$$\tilde{p}_0^v(x) = \begin{cases} 1 & x = v \text{ and } 1 \geq 2\epsilon d(x), \\ 0 & \text{otherwise.} \end{cases}$$

$$\tilde{p}_t^v = [T\tilde{p}_{t-1}]_\epsilon.$$

The description of the algorithm `Nibble` and Lemma 3.1 in [139] implies the following lemma.[39]

Lemma 7.6 ([139])**.** *For each $\phi \leq 1$, define the parameters*

$$t_0 = \frac{49 \ln(me^4)}{\phi^2}$$

$$\textit{and} \quad \gamma = \frac{5\phi}{392 \ln(me^4)}.$$

[39]There are many versions of the paper [139] available; we refer to `https://arxiv.org/abs/cs/0310051v9`.

For each subset $S \subset V$ satisfying

$$\mathrm{Vol}(S) \leq \frac{2}{3} \cdot \mathrm{Vol}(V)$$
$$\textit{and} \quad \Phi(S) \leq \frac{\phi^3}{19208 \ln^2(me^4)},$$

there exists a subset $S^g \subseteq S$ with the following properties. First, $\mathrm{Vol}(S^g) \geq \mathrm{Vol}(S)/2$. Second, S^g is partitioned into $S^g = \bigcup_{b=1}^{\log m} S_b^g$ such that if a random walk is initiated at any $v \in S_b^g$ with truncation parameter $\epsilon = \frac{\phi}{56 \ln(me^4) t_0 2^b}$, then there exists a number $t \in [1, t_0]$ and an index j such that the following four conditions are met for the cut $C = \{\tilde{\pi}_t^v(1), \ldots, \tilde{\pi}_t^v(j))\}$.

(i) $\Phi(C) \leq \phi$,

(ii) $\tilde{\rho}_t(\tilde{\pi}_t(j)) \geq \gamma/\mathrm{Vol}(C)$,

(iii) $\mathrm{Vol}(C \cap S) \geq (4/7)2^{b-1}$,

(iv) $\mathrm{Vol}(C) \leq (5/6)\mathrm{Vol}(V)$.

In subsequent discussion, with respect to a given parameter $\phi \leq 1$, for any subset $S \subset V$ satisfying the condition of Lemma 7.6, we fix a subset $S^g \subseteq S$ and its decomposition $S^g = \bigcup_{b=1}^{\log m} S_b^g$ to be any choices satisfying Lemma 7.6.

7.3.1 Distributed Algorithm

Now we give our algorithm `Distributed Nibble`. To simplify things, we present it as a sequential algorithm, and prove in Lemma 7.10 that it can be implemented efficiently in the CONGEST model. For any permutation π, we use the notation $\pi(i..j)$ to denote the set $\{\pi(i), \pi(i+1), \ldots, \pi(j)\}$.

Algorithm 1 `Distributed Nibble`

Input: ϕ.

for parameter $b = 1$ to $\lceil \log m \rceil$ **do**

Set parameters $t_0 = 49 \ln(me^4)/\phi^2$, and $\epsilon_b = \frac{\phi}{56 \ln(me^4) t_0 2^b}$, as in Lemma 7.6.

(1) Independently randomly sample $K = c \log m \cdot \frac{\mathrm{Vol}(V)}{2^b}$ vertices $v_1, ..., v_K$ proportional to their degrees, where c is a large enough constant.

Initialize $\tilde{p}_0^{v_i}$.

for $t = 1$ to t_0, for every v_i **do**

(2) Calculate $\tilde{p}_t^{v_i} = [T\tilde{p}_{t-1}^{v_i}]_{\epsilon_b}$.

Denote j_{max} as the largest index such that $\tilde{p}_t^{v_i}(\tilde{\pi}_t^{v_i}(j_{max})) > 0$.

for $x = 0$ to $\log_{1+\phi}(5/6)\mathrm{Vol}(V)$ **do**

(3) Set $j \leq j_{max}$ to be the largest index that $\mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j)) \leq (1+\phi)^x$.

(4) If $\Phi(\tilde{\pi}_t^{v_i}(1..j)) \leq 12\phi$, output the sparse cut $C = \tilde{\pi}_t^{v_i}(1..j)$ and halt.

end for

end for

end for

Return *failed*.

From Lemma 7.6 we know that we can obtain a cut C with some good properties if we start the truncated random walk at a vertex $v \in S_b^g$ with parameter ϵ_b. Therefore, what we do in `Distributed Nibble` is to just sample sufficiently many vertices as the starting points of random walks so that with sufficiently high probability at least one them is in the set S_b^g. The danger here is that calculating all these random walk distributions simultaneously may be infeasible if any part of the graph becomes *too congested*.

In this section we analyze the behavior of `Distributed Nibble` (as a sequential algorithm) and prove that it operates correctly. In Section 7.3.2 we argue that `Distributed Nibble` can be implemented efficiently in the CONGEST model, in $\mathrm{poly}(\log m, 1/\phi)$ time.

Roughly speaking, Lemma 7.7 shows that if the sets $\pi(1..j)$ and $\pi(1..j')$ have similar volume, then the cuts resulting from these two sets have similar sparsity. This justifies lines (3) and (4) of `Distributed Nibble` and allows us to examine a small number of prefixes of the permutation $\tilde{\pi}_t^{v_i}$.

Lemma 7.7. *Let π be any permutation, and let $\phi \leq 1/12$. If, for some index j, $\Phi(\pi(1..j)) \leq \phi$ and $\mathrm{Vol}(\pi(1..j)) \leq (5/6)\mathrm{Vol}(V)$, then $\Phi(\pi(1..j')) \leq 12\phi$ for all indices*

$j' > j$ such that

$$\mathrm{Vol}(\pi(1..j')) \leq (1+\phi)\mathrm{Vol}(\pi(1..j)).$$

Proof. Let $x = \mathrm{Vol}(\pi(1..j))$ and $y = \mathrm{Vol}(\pi(1..j'))$. Recall that $2m = 2|E| = \mathrm{Vol}(V)$, and so $x \leq (5/6)\mathrm{Vol}(V) = (5/6)2m$. We have $x \leq y \leq (1+\phi)x$. Since $x \leq (5/6)2m$ and $\phi \leq 1/12$, we have $\phi x \leq x/12 \leq (2m-x)/2$. Therefore,

$$2m - y \geq 2m - x - \phi x \geq (2m-x)/2.$$

We calculate an upper bound of $\Phi(\pi(1..j'))$ as follows.

$$\begin{aligned}
\Phi(\pi(1..j')) &= \frac{\partial(\pi(1..j'))}{\min(y, 2m-y)} \\
&\leq \frac{\partial(\pi(1..j)) + \sum_{i=j+1}^{j'} d(\pi(i))}{\min(x, (2m-x)/2)} \\
&\leq \frac{\partial(\pi(1..j)) + \phi x}{\min(x, (2m-x))/2} \\
&\leq 12\phi.
\end{aligned}$$

We explain the details of the derivation. The first inequality is due to $x \leq y$ and $(2m - x)/2 \leq 2m - y$, which follow from the above discussion. The second inequality is due to the fact that $\sum_{i=j+1}^{j'} d(\pi(i)) = \mathrm{Vol}(\pi(1..j')) - \mathrm{Vol}(\pi(1..j)) \leq \phi \cdot \mathrm{Vol}(\pi(1..j)) = \phi x$. For the third inequality, note that $\frac{\partial(\pi(1..j))}{\min(x,(2m-x))} \leq \phi$ and $\frac{\phi x}{\min(x,(2m-x))} \leq 5\phi$, since $x \leq (5/6)2m$. □

Lemma 7.8. *Let $S \subset V$ be any subset satisfying*

$$\mathrm{Vol}(S) \leq (2/3)\mathrm{Vol}(V) \quad \textit{and} \quad \Phi(S) \leq \frac{\phi^3}{19208 \ln^2(me^4)}.$$

Then there exists a number b such that $\mathrm{Vol}(S_b^g) \geq 2^b/32$.

Proof. Denote $x = \mathrm{Vol}(S)$. From Condition (iii) of Lemma 7.6 we deduce that if $S_b^g \neq \emptyset$, then there exists a set of vertices C such that $\mathrm{Vol}(S) \geq \mathrm{Vol}(C \cap S) \geq (4/7)2^{b-1}$. Thus, for all b such that $b \geq \lceil \log x \rceil + 2$, we must have $S_b^g = \emptyset$. If the statement of this lemma is false, i.e., $\mathrm{Vol}(S_b^g) < 2^b/32$ for all b, then

$$\mathrm{Vol}(S^g) \leq \sum_{b=1}^{\lceil \log x \rceil + 1} \frac{2^b}{32} < \frac{2^{\lceil \log x \rceil + 2}}{32} < x/4,$$

which contradicts the requirement $\mathrm{Vol}(S^g) \geq \mathrm{Vol}(S)/2$ specified in Lemma 7.6. □

Lemma 7.9 (Correctness)**.** *For any $\phi \leq 1/12$, if there exists a subset $S \subset V$ satisfying*

$$\mathrm{Vol}(S) \leq (2/3)\mathrm{Vol}(V) \quad \textit{and} \quad \Phi(S) \leq \frac{\phi^3}{19208 \ln^2(me^4)},$$

then `Distributed Nibble` *outputs a set of vertices C such that $\Phi(C) \leq 12\phi$ with probability at least $1 - 1/\mathrm{poly}(m)$.*

Proof. From Lemma 7.8 we know there exists a number b such that $\mathrm{Vol}(S_b^g) \geq 2^b/32$. Since we sample v_i proportional to the degree distribution,

$$\Pr[v_i \in S_b^g] = \frac{\mathrm{Vol}(S_b^g)}{\mathrm{Vol}(V)} \geq \frac{2^b}{32 \cdot \mathrm{Vol}(V)}.$$

Since we sample $K = c \log m \cdot \frac{\mathrm{Vol}(V)}{2^b}$ number of vertices,

$$\begin{aligned}\Pr[\exists i \text{ s.t. } v_i \in S_b^g] &\geq 1 - \left(1 - \frac{2^b}{32\mathrm{Vol}(V)}\right)^{c \log m \frac{\mathrm{Vol}(V)}{2^b}} \\ &\geq 1 - m^{-\Omega(c)}.\end{aligned}$$

Now we focus on the truncated random walk starting at this vertex $v_i \in S_b^g$. We fix two numbers $t \in [1, t_0]$ and j such that the four conditions in Lemma 7.6 are satisfied. In particular, Condition (i) and Condition (iv) in Lemma 7.6 say that

$$\begin{aligned}\mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j)) &\leq (5/6)\mathrm{Vol}(V), \\ \Phi(\tilde{\pi}_t^{v_i}(1..j)) &\leq \phi.\end{aligned}$$

Therefore, we are able to apply Lemma 7.7, and so we have $\Phi(\tilde{\pi}_t^{v_i}(1..j')) \leq 12\phi$ for all indices j' such that $\mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j)) \leq \mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j')) \leq (1+\phi)\mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j))$.

In `Distributed Nibble`, we search for a cut with target volume $(1+\phi)^x$, for all possible integers x. Note that Condition (ii) in Lemma 7.6 implies $j \leq j_{max}$. Therefore, in Step (3) of `Distributed Nibble`, at least one index $j^\star$ picked by the algorithm satisfies

$$\mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j)) \leq \mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j^\star)) \leq (1+\phi)\mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j)).$$

By Lemma 7.7, the cut $C = \tilde{\pi}_t^{v_i}(\{1, ..., j^\star\})$ associated with this index $j^\star$ found in Step (4) meets the requirement $\Phi(C) \leq 12\phi$ of the lemma. □

7.3.2 Implementation

We show how to implement `Distributed Nibble` in the CONGEST model. The goal of this section is to prove Lemma 7.10. Note that Lemma 7.5 is a consequence of Lemmas 7.9 and 7.10.

Lemma 7.10. *`Distributed Nibble` can be implemented in the* CONGEST *model using $O(D + \log^9 m/\phi^{10})$ rounds, with success probability $1 - 1/\mathrm{poly}(m)$, where D is the diameter of graph. If `Distributed Nibble` outputs a set C successfully, then each vertex knows whether or not it belongs to C.*

To prove this lemma, we shall analyze `Distributed Nibble` step by step.

Lemma 7.11 (Step (1))**.** *The samples for every level b (from 1 to $\lceil \log m \rceil$) can be generated in $O(D + \log m)$ time.*

Proof. We build a BFS tree rooted at an arbitrary vertex x. For each vertex v, define $s(v)$ as the sum of $d(u)$ for each u in the subtree rooted at v. In $O(D)$ rounds we can let each vertex v learn the number $s(v)$ by a bottom-up traversal of the BFS tree.

In the beginning, for each $b = 1, \ldots, \lceil \log m \rceil$, we generate $K_b = c \log m \frac{\mathrm{Vol}(V)}{2^b}$ number of b-tokens at the root x. Let $L = \Theta(D)$ be the number of layers in the BFS tree. For $i = 1, \ldots, L$, the vertices of layer i do the following. When a b-token arrives at v, the token disappears at v with probability $d(v)/s(v)$ and v includes itself in the bth sample; otherwise, v sends the token to a child u with probability $\frac{s(u)}{s(v)-d(v)}$. Note that v only needs to tell each child u how many b-tokens u gets. Thus, for each b, the process of choosing $K_b = c \log m \frac{\mathrm{Vol}(V)}{2^b}$ vertices from the degree distribution can be done in L rounds. By pipelining, we can do this for all b in $O(D + \log m)$ rounds.

This method has the virtue of selecting exactly K_b vertices in the bth sample. We can also select K_b vertices in expectation, in just $O(D)$ time, simply by computing $\mathrm{Vol}(V)$ with a BFS tree, disseminating it to all vertices, and letting each v join the sample independently with probability $K_b \deg(v)/\mathrm{Vol}(V)$. □

It is not obvious why Step (2) of `Distributed Nibble` should be efficiently implementable in the CONGEST model. Before analyzing it, we give some helpful lemmas about lazy random walks.

Lemma 7.12 ([139])**.** *For all u, v, and t, $\rho_t^v(u) = \rho_t^u(v)$.*

Proof. This lemma was observed in [139] without proof. For the sake of completeness, we provide a short proof here. A sequence of vertices $W = (x_0, x_1, \ldots, x_t)$ is called a *walk* of length t if $x_{i+1} \in N(x_i) \cup \{x_i\}$ for each $i \in [0, t)$. We write $\Pr[W]$ to be the probability that the first t steps of a lazy random walk starting at x_0 tracks W. Let $W^R = (x_t, x_{t-1}, \ldots, x_0)$ be the reversal of W.

Let $\mathcal{W}_t^{u,v}$ be the set of walks of length t starting at u and ending at v. It is clear that $\rho_t^u(v) = \sum_{W \in \mathcal{W}_t^{u,v}} \Pr[W]/d(v)$ and $\rho_t^v(u) = \sum_{W \in \mathcal{W}_t^{v,u}} \Pr[W]/d(u)$. Since $\mathcal{W}_t^{v,u} = \{W^R \mid W \in \mathcal{W}_t^{u,v}\}$, to prove the lemma it suffices to show that $\Pr[W]/d(v) = \Pr[W^R]/d(u)$ for each $W \in \mathcal{W}_t^{u,v}$.

Fix any $W \in \mathcal{W}_t^{u,v}$ and let $W_* = (y_0, \ldots, y_s)$ be the subsequence of W resulting from splicing out immediate repetitions in W. It is clear that $\Pr[W] = 2^{-t} \cdot \prod_{i=0}^{s-1} 1/d(y_i)$, and so

$$\frac{\Pr[W]}{d(v)} = \frac{\Pr[W]}{d(y_s)} = 2^{-t} \cdot \prod_{i=0}^{s} \frac{1}{d(y_i)} = \frac{\Pr[W^R]}{d(y_0)} = \frac{\Pr[W]}{d(u)}. \qquad \square$$

Lemma 7.13. *Fix the parameter b (which influences ϵ_b and hence the truncation operation of the random walk) and define*

$$Z_t(u) = \{v_i \mid v_i \text{ is in the bth sample and } \tilde{p}_{t-1}^{v_i}(u) > 0\}.$$

For every vertex u and every t, with high probability, $|Z_t(u)| \le O(\log^3 m/\phi^3)$.

Proof. Define $S = \{v \in V \mid \tilde{p}_{t-1}^v(u) > 0\}$. By definition $Z_t(u) = S \cap \{v_1, \ldots, v_{K_b}\}$. For each $v \in S$, we have $p_{t-1}^v(u) \ge \tilde{p}_{t-1}^v(u) \ge 2\epsilon_b d(u)$. Recall that p_{t-1} is the probability distribution obtained after $t-1$ steps of the lazy random walk without truncation. By Lemma 7.12,

$$p_{t-1}^u(v) = (p_{t-1}^v(u)/d(u))d(v) \ge 2\epsilon_b \cdot d(v).$$

Therefore, $2\epsilon_b \cdot \mathrm{Vol}(S) \le \sum_{v \in S} p_{t-1}^u(v) \le 1$, and so $\mathrm{Vol}(S) \le \frac{1}{2\epsilon_b}$, which implies

$$\Pr[v_i \in S] \le \frac{1}{2\epsilon_b \cdot \mathrm{Vol}(V)}.$$

Recall that $t_0 = \frac{49 \ln(me^4)}{\phi^2}$ and $\epsilon_b = \frac{\phi}{56 \ln(me^4) t_0 2^b}$. Rewrite the number $K_b = c \log m \frac{\mathrm{Vol}(V)}{2^b}$ as $K_b = \Theta(\epsilon_b \cdot \mathrm{Vol}(V) \cdot \log^3 m/\phi^3)$. Since each of $v_1, \ldots, v_{K_b}$ is chosen independently, using

a Chernoff bound we conclude that there exists a constant $c' > 0$ depending on c such that

$$\Pr[|Z_t(u)| > c' \log^3 m/\phi^3] \leq \exp(-\Omega(\log^3 m/\phi^3)). \qquad \square$$

Lemma 7.14 (Step (2))**.** *Fix the parameter b. Suppose each vertex v knows $\tilde{p}_{t-1}^{v_i}(v)$, for all v_i in the bth sample. Then with high probability, each vertex v can calculate $\tilde{p}_t^{v_i}(v)$, for all v_i, within $O(\log^3 m/\phi^3)$ rounds.*

Proof. The normal way to calculate $[T\tilde{p}_{t-1}(u)]_{\epsilon_b}$ is as follows. For each v_i, each vertex v broadcasts the number $\frac{\tilde{p}_{t-1}^{v_i}(v)}{2d(v)}$ to all its neighbors, and then v collects messages from neighbors. The vertex v can calculate $\tilde{p}_t^{v_i}(v)$ locally by adding $\tilde{p}_{t-1}^{v_i}(v)/2$ and all numbers received from its neighbors, then applying the truncation operation. Note that a straightforward analysis of this protocol leads to a terrible round complexity, since we have to do this for each v_i.

One crucial observation is that a vertex v does not need to care about those v_i with $\tilde{p}_{t-1}^{v_i}(v) = 0$ at time t. We modify this protocol a little bit in such a way that we never send a number if it is 0. Define $Z_t(u) = \{v_i \mid \tilde{p}_{t-1}^{v_i}(u) > 0\}$ as in Lemma 7.13, and so each vertex v only needs to spend $|Z_t(v)|$ rounds to simulate the time step t of the lazy random walk. By Lemma 7.13 and the discussion above, we have proved that Step (2) can be executed in $O(\log^3 m/\phi^3)$ time, for every v_i and any specific t. $\square$

Lemma 7.15 (Steps (3,4))**.** *Fix parameters b, t and x. Steps (3) and (4) can be implemented in $O(\log^6 m/\phi^7)$ rounds for all v_i in the bth sample. For any sparse cut C found in Step (4), every vertex in C knows that it belongs to C.*

Proof. Now we focus on the random walk starting at v_i. Let $U = \{u \mid \exists t' \leq t,\ \tilde{p}_{t'}^{v_i}(u) > 0\}$. We claim that U is a connected vertex set. Suppose U is disconnected. Let W be a connected subset of U such that $v_i \notin W$. Let t' be the minimum number such that there exists a vertex $u' \in W$ with $\tilde{p}_{t'}^{v_i}(u') > 0$. By our choice of u', there is no neighbor v' of u' such that $\tilde{p}_{t'-1}^{v_i}(v') > 0$, and this contradicts the fact that $\tilde{p}_{t'}^{v_i}(u') > 0$. Therefore, U must be a connected vertex set.

Obviously all $\tilde{\pi}_t^{v_i}(j)$ for $j \leq j_{max}$ are in U. We build a BFS tree of U rooted at v_i, which has $t+1$ levels. We will execute Step (3) and Step (4) by sending requests from the root to all vertices in U, collecting information from U to the root, and making a decision locally at the root. Recall that each v_i has its own BFS tree, and in general a vertex u belongs to

multiple BFS trees for different v_i. Luckily, each vertex u only belongs to the BFS tree of those $v_i \in \bigcup_{1\le t\le t_0+1} Z_t(u)$, so with only a $(t_0+1)\cdot \max_{u,t} |Z_t(u)| = O(\log^4 m/\phi^5)$ overhead of running time, we can do Step (3) and Step (4) for all v_i in parallel.

To find each index j specified in Step (3), we can do a "random binary search" on vertices in U. Let $\tilde{\pi} = \tilde{\pi}_t^{v_i}$ and $\tilde{\rho} = \tilde{\rho}_t^{v_i}$ be with respect to $\tilde{p}_t^{v_i}$. Note that by our choice of U we can assume U is a prefix set of $\tilde{\pi}$. We maintain two indices L and R that control the search space. Initially, $L \leftarrow 1$ and $R \leftarrow |U|$. In each iteration, we randomly pick one vertex $\tilde{\pi}(j)$ among $\tilde{\pi}(L..R)$ and calculate $\mathrm{Vol}(\tilde{\pi}(1,\cdots,j))$ by broadcasting $\tilde{\rho}(\tilde{\pi}(j))$ to all vertices in U and propagating information up the BFS tree.[40] If $\mathrm{Vol}(\tilde{\pi}(1,\cdots,j)) \le (1+\phi)^x$, we update $L \leftarrow j$; otherwise we let $R = j-1$. In each iteration, with probability $1/2$ we sample j in the middle half of $[L,R]$ and the size of search space $[L,R]$ shrinks by a factor of at least $3/4$. Therefore, w.h.p., after $O(\log m)$ iterations, we will have isolated $L = R = j$. Each iteration can be done in $O(t) = O(t_0)$ rounds. Due to the congestion overhead, Step (3) can be implemented in $O(\log m \cdot t_0 \cdot \log^4 m/\phi^5) = O(\log^6 m/\phi^7)$ rounds.

Step (4) can be done by simply collecting information about $\partial(\tilde{\pi}_t^{v_i}(1..j))$ and $\mathrm{Vol}(\tilde{\pi}_t^{v_i}(1..j))$; its round complexity is of a lower order than that of Step (3). If the root v_i finds a cut C with $\Phi(C) \le 12\phi$, it broadcasts $\tilde{\rho}(\tilde{\pi}(j))$ to all vertices in U to let the vertices in C know that they are in C. Note that for each vertex u in U, it can infer whether it is in C by comparing $\tilde{\rho}(u)$ and $\tilde{\rho}(\tilde{\pi}(j))$. □

Proof of Lemma 7.10. Combining Lemmas 7.11, 7.14, and 7.15, the running time in Step (1) is $O(D+\log m)$, Step (2) is $O(\log^5 m/\phi^5)$, and Steps (3) and (4) are $O(\log^9 m/\phi^{10})$. The dominating term $O(\log^9 m/\phi^{10})$ comes from enumerating $\log m \cdot t_0 \cdot \log m/\phi = \Theta(\log^3 m/\phi^3)$ combinations of (b,t,x), spending $O(\log^6 m/\phi^7)$ rounds for each combination.

Whenever a vertex v_i finds a sparse cut C, it broadcasts a message to the entire graph saying that it has found a cut, and this takes $O(D)$ rounds. If multiple cuts are found by different vertices, we can select exactly one cut, breaking ties arbitrarily. A more opportunistic version of the algorithm could also take a maximal independent set of compatible cuts. □

[40] Each vertex $u \in U$ does not know the index j such that $u = \tilde{\pi}(j)$, so we cannot do the search deterministically.

7.4 Triangle Enumeration

We use the routing algorithm from [77, 78]. Theorem 7.2 was first stated in [77, Theorem 1.2] with round complexity $\tau_{\text{mix}}(G) \cdot 2^{O(\sqrt{\log n \log\log n})}$; this was recently improved to $\tau_{\text{mix}}(G) \cdot 2^{O(\sqrt{\log n})}$ in [78].

Theorem 7.2 ([77, 78])**.** *Consider a graph $G = (V, E)$ and a set of point-to-point routing requests, each given by the* ID*s of the corresponding source-destination pair. If each vertex v is the source and the destination of at most $\deg(v) \cdot 2^{O(\sqrt{\log n})}$ messages, there is a randomized distributed algorithm that delivers all messages in $\tau_{\text{mix}}(G) \cdot 2^{O(\sqrt{\log n})}$ rounds, w.h.p., in the* CONGEST *model.*

Remark 7.1. *The claim of Theorem 7.2 appears to be unproven for arbitrary* ID*-assignments (Hsin-Hao Su, personal communication, 2018), but is true for well-behaved* ID*-assignments, which we illustrate can be computed efficiently in* CONGEST*. In [77, 78] each vertex $v \in V$ simulates $\deg(v)$ virtual vertices in a random graph G_0 which is negligibly close to one drawn from the Erdős-Rényi distribution $\mathcal{G}(2m, p)$ for some p. Presumably the IDs of v's virtual vertices are $(\text{ID}(v), 1), \ldots, (\text{ID}(v), \deg(v))$. It is proven [77, 78] that effecting a set of routing requests in G_0 takes $2^{O(\sqrt{\log n})}$ time in G_0; however, to translate a routing request $\text{ID}(x) \rightsquigarrow \text{ID}(y)$ in G to G_0, it seems necessary to map it (probabilistically) to $(\text{ID}(x), i) \rightsquigarrow (\text{ID}(y), j)$, where i, j are chosen uniformly at random from $[1, \deg(x)]$ and $[1, \deg(y)]$, respectively. (This is important for the global congestion guarantee that y's virtual nodes receive roughly equal numbers of messages from all sources.) This seems to require that x know how to compute $\deg(y)$ or an approximation thereof based on $\text{ID}(y)$. Arbitrary* ID*-assignments obviously do not betray this information.*

Lemma 7.16. *In $O(D + \log n)$ time we can compute an* ID *assignment $\text{ID} : V \to \{1, \ldots, |V|\}$ and other information such that $\text{ID}(u) < \text{ID}(v)$ implies $\lfloor \log \deg(u) \rfloor \leq \lfloor \log \deg(v) \rfloor$, and any vertex u can locally compute $\lfloor \log \deg(v) \rfloor$ for any v.*

Proof. Build a BFS tree from an arbitrary vertex x in $O(D)$ time. In a bottom-up fashion, each vertex in the BFS tree calculates the *number* of vertices v in its subtree having $\lfloor \log \deg(v) \rfloor = i$, for $i = 0, \ldots, \log n$. This takes $O(D + \log n)$ time by pipelining. At this point the root x has the counts $n_0, \ldots, n_{\log n}$ for each degree class, where $n = \sum_i n_i$. It partitions up the ID-space so that all vertices in class-0 get IDs from $[1, n_0]$, class-1 from $[n_0+1, n_0+n_1]$, and so on. The root broadcasts the numbers $n_0, \ldots, n_{\log n}$, and disseminates

the IDs to all nodes according to their degrees. (In particular, the root gives each child $\log n$ intervals of the ID-space, which they further subdivide, sending $\log n$ intervals to the grandchildren, etc.) With pipelining this takes another $O(D+\log n)$ time. Clearly knowing $n_0, \ldots, n_{\log n}$ and $\mathrm{ID}(v)$ suffice to calculate $\lfloor \log \deg(v) \rfloor$. □

Lemma 7.16 gives us a good ID-assignment to apply Theorem 7.2. It is also useful in our triangle enumeration application. Roughly speaking, vertices with larger degrees also have more bandwidth in the CONGEST model, and therefore should be responsible for learning about larger subgraphs and enumerating more triangles.

Before we present our triangle enumeration algorithm for general graphs, we address the important special case of finding triangles with at least one edge in a component of high conductance.

7.4.1 Triangle Enumeration in High Conductance Graphs

Recall that our graph decomposition routine returns a tripartition $E_m \cup E_s \cup E_r$. Triangles that intersect E_s will be enumerated separately. The purpose of this section is to provide a routine to enumerate triangles that intersect E_m but not E_s, i.e., they are (i) completely contained in E_m or (ii) have at least one edge in E_m and E_r.[41] Whereas each component of E_m has low mixing time, we can say nothing about the mixing time of a component of E_m plus all incident E_r edges.

Definitions. The underlying network is $G = (V, E)$. The input is a subgraph $G_{\text{in}} = (V_{\text{in}}, E_{\text{in}})$ with low mixing time, together with some additional edges E_{out} joining vertices in V_{in} to V. Let $\deg_{\text{in}}(v)$ and $\deg_{\text{out}}(v)$ be the number of E_{in} and E_{out} edges incident to v. In this section we write $n = |V|$ and $m_{\text{in}} = |E_{\text{in}}|$. We assume $V = V(E_{\text{in}} \cup E_{\text{out}})$. Note that Condition (i) of Theorem 7.3 implies that $m_{\text{in}} \leq |E_{\text{in}} \cup E_{\text{out}}| \leq 3m_{\text{in}}$.

Theorem 7.3. *Suppose that G_{in} and E_{out} meet the following conditions:*

(i) For each $v \in V_{\text{in}}$, $\deg_{\text{in}}(v) \geq \deg_{\text{out}}(v)$.

(ii) $\tau_{\text{mix}}(G_{\text{in}}) = n^{o(1)}$.

[41] Our algorithm for constructing the tripartition $E_m \cup E_s \cup E_r$ satisfies the property that there is no triangle with two edges in E_m and one edge in E_r. However, all algorithms in Section 7.4 do not rely on this property.

In the CONGEST *model, all triangles in $E_{\text{in}} \cup E_{\text{out}}$ can be counted and enumerated, w.h.p., in $O(n^{1/3+o(1)})$ rounds.*

Note that Theorem 7.3 applies to the class of graphs with $n^{o(1)}$ mixing time by setting $E_{\text{out}} = \emptyset$. We first describe the algorithm behind Theorem 7.3 and then analyze it in Lemmas 7.17–7.20.

The Easy Case. We first check whether any vertex $v^\star \in V(E_{\text{in}} \cup E_{\text{out}})$ has

$$\deg_{\text{in}}(v^\star) + \deg_{\text{out}}(v^\star) \geq m/(40n^{1/3}\log n) = \zeta.$$

If so, we apply Theorem 7.2 to the subgraph G^+_{in} induced by E_{in} and all edges E_{out} incident to $v^\star$, and have every vertex $u \in V_{\text{in}}$ transmit to $v^\star$ all its incident edges in $E_{\text{in}} \cup E_{\text{out}}$.[42] Condition (i) of Theorem 7.3 implies that $|E_{\text{in}} \cup E_{\text{out}}| \leq 3m_{\text{in}}$, so the total volume of messages entering $v^\star$ is $O(m_{\text{in}})$. Therefore the routing takes $O(\tau_{\text{mix}}(G^+_{\text{in}}) \cdot 2^{O(\sqrt{\log n})} \cdot m/\zeta) = n^{1/3+o(1)}$, and thereafter, $v^\star$ can report all triangles in $E_{\text{in}} \cup E_{\text{out}}$. In the analysis of the following steps, we may assume that the maximum degree in the graph induced by $E_{\text{in}} \cup E_{\text{out}}$ is at most $m_{\text{in}}/(40n^{1/3}\log n)$.

Vertex Classes. Let $\delta = 2^{\lfloor \log(2m_{\text{in}}/n) \rfloor}$ be the average number of incident edges in E_{in} among all n vertices V, rounded down to the nearest power of 2. Write $\deg_{\text{in}}(v) = k_v \cdot \delta$, and call v a *class-0 vertex* if $k_v \in [0, 1/2)$ and a *class-i vertex* if $k_v \in [2^{i-2}, 2^{i-1})$. We use the fact that

$$\sum_{v \in V_{\text{in}} \,:\, k_v \geq 1/2} 2k_v \geq n.$$

By applying Lemma 7.16 to reassign IDs, we may assume that the ID-space of V_{in} is $\{1, \ldots, |V_{\text{in}}|\}$ and that any vertex can compute the class of v, given $\text{ID}(v)$.

Randomized Partition. Our algorithm is a randomized adaptation of the CONGESTED-CLIQUE algorithm of [50]. We partition the vertex set V into $V_1 \cup \cdots \cup V_{n^{1/3}}$

[42] Note that when $v^\star \in V \setminus V_{\text{in}}$, $\tau_{\text{mix}}(G_{\text{in}}) = n^{o(1)}$ implies that $\tau_{\text{mix}}(G^+_{\text{in}}) = n^{o(1)}$ as well. This follows from the observation that the conductance $\Phi_{G^+_{\text{in}}}$ of G^+_{in} is at least 1/3 of the conductance Φ_G of G_{in}. Pick a sparsest cut $(S, \bar{S})$ of G^+_{in}, and let $(S', \bar{S}')$ be the corresponding cut of G_{in}. The sparsity $\Phi(S)$ of $(S, \bar{S})$ must be at least 1/3 of the sparsity $\Phi(S')$ of $(S', \bar{S}')$, because $\text{Vol}(S) \leq 2|S'| + \text{Vol}(S') \leq 3\text{Vol}(S')$. Therefore, $\Phi_{G^+_{\text{in}}} = \Phi(S) \geq \frac{1}{3}\Phi(S') \geq \frac{1}{3}\Phi_{G_{\text{in}}}$.

locally, without communication. Each vertex $v \in V$ selects an integer $r_v \in [1, n^{1/3}]$ uniformly at random, joins V_{r_v}, and transmits 'r_v' to its immediate neighbors in V_{in}. We allocate the (less than) n triads

$$\mathscr{T} = \left\{(j_1, j_2, j_3) \mid 1 \le j_1 \le j_2 \le j_3 \le n^{1/3}\right\}$$

to the vertices in V_{in} in the following way. Enumerate the vertices in increasing order of ID. If v is class-0, then skip v. If v is class-i, $i \ge 1$, then $k_v < 2^{i-1}/\delta$. Allocate to v the next $2^i/\delta \ge 2k_v$ triads from $\mathscr{T}$, and stop whenever all triads are allocated.

We use Lemma 7.16 to generate the IDs of vertices in V_{in}. In view of how vertex class is defined, Lemma 7.16 guarantees that each vertex $v \in V_{\text{in}}$ knows the class of all vertices in V_{in}, and can therefore perform this allocation locally, without communication.

A vertex $v \in V$ that is assigned a triad (j_1, j_2, j_3) is responsible for learning the set of all edges $E(V_{j_1}, V_{j_2}) \cup E(V_{j_2}, V_{j_3}) \cup E(V_{j_1}, V_{j_3})$ and reporting/counting those triangles (x_1, x_2, x_3) with $x_k \in V_{j_k}$.[43]

Transmitting Edges. Every vertex $v \in V_{\text{in}}$ knows the IDs of all its neighbors in V and which part of the vertex partition they are in. For each $v \in V_{\text{in}}$, each incident edge $(v, u) \in E_{\text{in}} \cup E_{\text{out}}$, and each index $r^* \in [1, n^{1/3}]$, v transmits the message "$(v, u), r_v, r_u$" to the unique vertex x handling the triad on $\{r_u, r_v, r^*\}$. Observe that the total message volume is exactly $\Theta(m_{\text{in}} n^{1/3})$.

We analyze the behavior of this algorithm in the CONGEST model, where the last step is implemented by applying Theorem 7.2 to G_{in}. Recall from Condition (i) of Theorem 7.3 that the number of edges in the graph we consider, $\bar{m} = |E_{\text{in}} \cup E_{\text{out}}|$, is in the range $[m_{\text{in}}, 3m_{\text{in}}]$.

Lemma 7.17. *Consider a graph with $\bar{m}$ edges and $\bar{n}$ vertices. We generate a subset S by letting each vertex join S independently with probability p. Suppose that the maximum degree is $\Delta \le \bar{m}p/20 \log \bar{n}$ and $p^2\bar{m} \ge 400 \log^2 \bar{n}$. Then, with probability at least $1 - 10(\log \bar{n})/\bar{n}^5$, the number of edges in the subgraph induced by S is at most $6p^2\bar{m}$.*

Proof. For an edge e_i, define $x_i = 1$ if both two endpoints of edge e_i join S, otherwise $x_i = 0$. Then $X = \sum_{i=1}^{\bar{m}} x_i$ is the number of edges in the subgraph induced by S. We have

[43] In the Triangle Counting application, it is important that v not count every triangle it is aware of. For example, if v is assigned (j, j, j'), v knows about triangles in the subgraph induced by V_j but should not count them; these triangles will be counted only by the vertex u that is assigned (j, j, j).

$\mathrm{E}[X] = p^2\bar{m}$, and by Markov's inequality,

$$\Pr[X \geq 6\,\mathrm{E}[X]] = \Pr[X^c \geq (6\,\mathrm{E}[X])^c] \leq \frac{1}{6^c}\frac{\mathrm{E}[X^c]}{p^{2c}\bar{m}^c},$$

where $c = 5\log\bar{n}$ is a parameter.

$$\begin{aligned}\mathrm{E}[X^c] &= \sum_{i_1,\ldots,i_c \in [1,\bar{m}]} \mathrm{E}\left[\prod_{j=1}^{c} x_{i_j}\right] \\ &= \sum_{k=2}^{2c} f_k \cdot p^k,\end{aligned}$$

where f_k is the number of choices $\{i_1, \ldots, i_c \in [1, \bar{m}]\}$ such that the number of distinct endpoints in the edge set $e_{i_1}, \ldots, e_{i_c}$ is k.

For any choice of $(i_1, \ldots, i_c \in [1, \bar{m}])$, we project it to a vector $\langle k_1, \ldots, k_c\rangle \in \{0, 1, 2\}^c$, where k_j indicates the number of endpoints of e_{i_j} that overlap with the endpoints of the edges $e_{i_1}, \ldots, e_{i_{j-1}}$. Note that $2c - \sum k_j$ is the number of distinct endpoints in the edge set $\{e_{i_1}, \ldots, e_{i_c}\}$. We fix a vector $\langle k_1, \ldots, k_c\rangle$ and count how many choices of $(i_1, \ldots, i_c)$ project to this vector.

Suppose that the edges $e_{i_1}, \ldots, e_{i_{j-1}}$ are fixed. We bound the number of choices of e_{i_j} as follows. If $k_j = 0$, the number of choices is clearly at most m. If $k_j = 1$, the number of choices is at most $(2c)(p\bar{m}/20\log\bar{n})$, since one of its endpoints (which overlaps with the endpoints of the edges $e_{i_1}, \ldots, e_{i_{j-1}}$) has at most $2c$ choices, and the other endpoint (which does not overlap with the endpoints of the edges $e_{i_1}, \ldots, e_{i_{j-1}}$) has at most $\Delta \leq \bar{m}p/20\log\bar{n}$ choices. If $k_j = 2$, the number of choices is at most $(2c)^2$.

Based on the above calculation, we upper bound f_k as follows. In the calculation, x is the number of indices j such that $k_j = 1$, and y is the number of indices j that $k_j = 2$. Note that $\binom{c}{x}\binom{c-x}{y}$ is the number of distinct vectors $\langle k_1, \ldots, k_c\rangle$ realizing the given parameters

c, x, and y. The number f_k is at most

$$\begin{aligned}
&\sum_{\substack{x+y\le c\\ 2c-x-2y=k}} \bar{m}^{c-x-y}\binom{c}{x}\binom{c-x}{y}\left(\frac{2cp\bar{m}}{20\log\bar{n}}\right)^x (4c^2)^y \\
&\le \sum_{\substack{x+y\le c\\ 2c-x-2y=k}} \bar{m}^c 3^c \left(\frac{2cp}{20\log\bar{n}}\right)^x \left(\frac{4c^2}{\bar{m}}\right)^y \\
&\le \sum_{\substack{x+y\le c\\ 2c-x-2y=k}} (3\bar{m})^c \left(\frac{2cp}{20\log\bar{n}}\right)^{x+2y} \\
&\le c(3\bar{m})^c \left(\frac{2cp}{20\log\bar{n}}\right)^{2c-k}.
\end{aligned}$$

The third inequality is due to the fact $p^2\bar{m} \ge 400\log^2\bar{n}$, which implies $(2cp/20\log\bar{n})^2 \ge (4c^2/\bar{m})$. Using the fact that $\frac{2c}{20\log\bar{n}} \le 1/2$, we upper bound $\mathrm{E}[X^c]$ as follows.

$$\begin{aligned}
\mathrm{E}[X^c] &\le \sum_{k=2}^{2c} f_k \cdot p^k \\
&= c(3\bar{m})^c p^{2c} \sum_{k=2}^{2c} \left(\frac{2c}{20\log\bar{n}}\right)^{2c-k} \\
&< 2c(3\bar{m})^c p^{2c}.
\end{aligned}$$

Therefore,

$$\Pr[X \ge 6\,\mathrm{E}[X]] \le \frac{1}{6^c}\frac{\mathrm{E}[X^c]}{p^{2c}\bar{m}^c} \le \frac{2c3^c}{6^c} \le \frac{10\log\bar{n}}{\bar{n}^5}.$$

Note that the probability can be amplified to $\bar{n}^{-t}$ for any constant t by setting $c = t\log\bar{n}$ and using different constants in the statement of the lemma. □

Lemma 7.18. *With probability at least $1-1/n^4$, we have $|E(V_{j_1}, V_{j_2})| \le 6|E_{\text{in}} \cup E_{\text{out}}|/n^{2/3}$ for all $j_1, j_2 \in [1, n^{1/3}]$.*

Proof. Recall that each $v \in V$ joins the set V_i with probability $1/n^{1/3}$. Thus, the probability that a vertex $v \in V$ is in $V_{j_1} \cup V_{j_2}$ is at most $p = 2n^{-1/3}$.[44]

We apply Lemma 7.17 to the subgraph induced by $E_{\text{in}} \cup E_{\text{out}}$ having $\bar{m} = |E_{\text{in}} \cup E_{\text{out}}|$ edges and $\bar{n} = n$ vertices, with sampling probability $p = 2n^{-1/3}$ and $S = V_{j_1} \cup V_{j_2}$. By assumption,

[44] For the case of $j_1 = j_2$, the probability is $n^{-1/3}$.

the maximum degree (of the subgraph induced by $E_{\text{in}} \cup E_{\text{out}}$) is at most $m_{\text{in}}p/(20 \log n) \leq \bar{m}p/(20 \log \bar{n})$, since otherwise we go to the easy case. The maximum degree upper bound implies $\bar{n} \geq (20 \log \bar{n})/p$, and $p^2\bar{m} \geq (p\bar{n})^2 \geq 400 \log^2 \bar{n}$. By Lemma 7.17, we conclude that $\Pr[|E(V_{j_1}, V_{j_2})| > 6\bar{m}/n^{2/3}] \leq \frac{10 \log n}{n^5}$. Note that $|E(V_{j_1} \cup V_{j_2})| \geq |E(V_{j_1}, V_{j_2})|$.

By a union bound over all $n^{2/3}$ choices of j_1 and j_2, the stated upper bound holds everywhere, with probability at least $1 - 1/n^4$. □

Lemma 7.19. *With high probability, each vertex $v \in V_{\text{in}}$ receives $O(\deg_{\text{in}}(v) \cdot n^{1/3})$ edges.*

Proof. Consider any vertex $v \in V_{\text{in}}$. If $k_v < 1/2$, then v receives no message; otherwise v is responsible for between $2k_v$ and $4k_v$ triads, and v collects the edge set $E(V_{j_1}, V_{j_2})$ for at most $12k_v$ pairs of V_{j_1}, V_{j_2}. By Lemma 7.18, w.h.p., $|E(V_{j_1}, V_{j_2})| = O(m_{\text{in}}/n^{2/3})$. Remember that our choice of k_v implies $k_v = \Theta(\deg_{\text{in}}(v) \cdot n/m_{\text{in}})$, and so v receives

$$O(m_{\text{in}}/n^{2/3}) \cdot 12k_v = O(\deg_{\text{in}}(v) \cdot n^{1/3})$$

messages, with high probability. □

Lemma 7.20. *Each vertex $v \in V_{\text{in}}$ sends $O(\deg_{\text{in}}(v) \cdot n^{1/3})$ edges with probability 1.*

Proof. By Condition (i) of Theorem 7.3, $v \in V_{\text{in}}$ is responsible for $\deg_{\text{in}}(v) + \deg_{\text{out}}(v) \leq 2 \deg_{\text{in}}(v)$ incident edges, and each is involved in exactly $n^{1/3}$ triads. □

Lemmas 7.17–7.20 show that the message volume in to/out of every vertex is close to its expectation. By applying Theorem 7.2 and Lemma 7.16, all messages can be routed in $n^{1/3+o(1)}$ time. This concludes the proof of Theorem 7.3. Corollary 7.1 is a simple consequence of Theorem 7.3.

Corollary 7.1. *Let G be a graph with $\tau_{\text{mix}}(G) = n^{o(1)}$. In the* CONGEST *model, Triangle Detection, Enumeration, and Counting can be solved on G, with high probability, in $n^{1/3+o(1)}$ time.*

7.4.2 Triangle Enumeration and Counting in General Graphs

The algorithm for Theorem 7.4 is based on an $n^{1/2}$-decomposition. Since the connected components induced by E_m have low mixing time, we can solve Triangle Enumeration/Counting on them very efficiently using Theorem 7.3, in $n^{1/3+o(1)}$ time, i.e., much less than the time required to compute the $n^{1/2}$-decomposition.

Theorem 7.4. *In the* CONGEST *model, Triangle Detection, Counting, and Enumeration can be solved, w.h.p., in* $\tilde{O}(n^{1/2})$ *rounds.*

Proof. The underlying graph is $G = (V, E)$. We set the parameter $\delta = 1/2$. By Theorem 7.1, we compute an n^δ-decomposition $E = E_m \cup E_s \cup E_r$ using $\tilde{O}(n^{1-\delta})$ rounds. We divide the task of enumerating triangles into three cases. By ensuring that every triangle is output by exactly one vertex, this algorithm also solves Triangle Counting.

The algorithm has three steps. In the first step, we list all triangles intersecting E_s. In the second step, we identify a subset $E_r^{\text{new}} \subseteq E_m$, and in this step we list all triangles intersecting $E_m \setminus E_r^{\text{new}}$. At this point, all remaining triangles that are not yet listed are contained in $E_r^{\text{new}} \cup E_r$, and they will be listed in the third step.

Case 1: All Triangles Intersecting E_s. We handle this case as follows. By Condition (b) of Definition 7.2, $E_s = \bigcup_{v \in V} E_{s,v}$, where $\{E_{s,v}\}$ defines an acyclic n^δ-orientation. We let each v announce $E_{s,v}$ to all its neighbors, in $O(n^\delta)$ time. For the Triangle Counting application it is important that every triangle $\{x, y, z\}$ intersecting E_s be reported by exactly *one* vertex. If (x, y) and (x, z) are oriented and $\text{ID}(y) < \text{ID}(z)$, then y detects and reports the triangle. If (x, z) is oriented, $\{x, y\}$ is unoriented, and $\{y, z\}$ is unoriented or oriented as (z, y), then y detects and reports the triangle. If $(x, z), (y, z)$ are oriented but $\{x, y\}$ is not, and $\text{ID}(y) < \text{ID}(x)$, y reports the triangle.[45]

Case 2: Some Triangles Intersecting E_m. Consider a single connected component $G_{\text{in}} = (V_{\text{in}}, E_{\text{in}})$ induced by E_m, which has mixing time $n^{o(1)}$. We classify vertices in V_{in} as *good* or *bad* depending on whether they naturally satisfy Condition (i) of Theorem 7.3. A vertex is good if $\deg_{\text{in}}(v) \geq \deg_{E_r}(v)$. Let E_{out} be the subset of E_r-edges incident to *good* vertices in V_{in}, and let E_r^{new} be the subset of E_m-edges incident to *bad* vertices in V_{in}. We now apply Theorem 7.3 to enumerate/count all triangles in the edge set $E_{\text{in}} \cup E_{\text{out}}$.

Because triangles completely contained in E_r^{new} will also be found in Case 3, the Triangle *Counting* algorithm should refrain from including these in the tally for Case 2.

Case 3: Triangles Contained in $E_r^{\text{new}} \cup E_r$. Since each edge in E_r^{new} can be charged to an endpoint of an edge in E_r, we have $|E_r^{\text{new}} \cup E_r| \leq 3|E_r| \leq |E|/2$. We apply the

[45] Most of these cases do not occur in the parital orientations produced by our algorithm; nonetheless, they can occur in arbitrary partial acyclic orientations.

algorithm recursively to the graph induced by $E_r^{\text{new}} \cup E_r$. The depth of the recursion is obviously at most $\log m$.

Round Complexity. Computing an n^δ-decomposition $E = E_m \cup E_s \cup E_r$ takes $\tilde{O}(n^{1-\delta})$ rounds. The algorithm for Case 1 takes $O(n^\delta)$ rounds. The algorithm for Case 2 takes $O(n^{1/3+o(1)})$ rounds. The number of recursive calls (Case 3) is $\log m$. Thus, the overall round complexity is

$$\log m \cdot \Big(O(n^\delta) + \tilde{O}(n^{1-\delta}) + O(n^{1/3+o(1)})\Big) = \tilde{O}(n^{1/2}). \qquad \square$$

7.4.3 Subgraph Enumeration

In this section we show that Corollary 7.1 can be extended to enumerating s-vertex subgraphs in $O(n^{(s-2)/s+o(1)})$ rounds. Note that the $\Omega(n^{1/3}/\log n)$ lower bound for triangle enumeration on Erdős-Rényi graphs $\mathcal{G}(n, 1/2)$ [90] can be generalized to an $\Omega(n^{(s-2)/s}/\log n)$ lower bound for enumerating s-vertex cliques; see [64]. This implies that Theorem 7.5 is nearly optimal on $\mathcal{G}(n, 1/2)$ for enumerating certain s-vertex subgraphs.

Theorem 7.5. *Let $s = O(1)$ be any constant. Given a graph G of n vertices with $\tau_{\text{mix}}(G) = n^{o(1)}$, we can list all s-vertex subgraphs of G in $O(n^{(s-2)/s+o(1)})$ rounds, w.h.p., in the* CONGEST *model.*

It has been shown in [50] that listing all s-vertex subgraphs of G can be done in $O(n^{(s-2)/s}/\log n)$ rounds in the deterministic CONGESTED-CLIQUE model. This result, together with the routing algorithm of Lemma 7.2, does not immediately imply Theorem 7.5, since $\deg(v)$ could be much less than n.

Theorem 7.5 is proved using a variant of Theorem 7.3 with $E_{\text{out}} = \emptyset$. The proof of Theorem 7.5 is almost the same as that of Theorem 7.3, and so in what follows we only highlight the difference.

Let $G = (V, E)$ and $m = |E|$. Similarly, we assume the maximum degree is $m/\left(40n^{1/s}\log n\right)$, since otherwise we are in the easy case, where we can apply Theorem 7.2 to have one vertex v learn the entire edge set E in $O(n^{1/s+o(1)}) \leq O(n^{(s-2)/s+o(1)})$ rounds, and we are done after that.

We partition V into $n^{1/s}$ subsets $V_1, \ldots, V_{n^{1/s}}$. Instead of considering triads, here we consider s-tuples: $\big\{(i_1, \ldots, i_s) \mid 1 \leq i_1 \leq \ldots \leq i_s \leq n^{1/s}\big\}$. After a vertex v learns the edge

set $\bigcup_{j_1,j_2\in[1,s]} E(V_{i_{j_1}}, V_{i_{j_2}})$, it has ability to list all s-vertex subgraphs in which the jth vertex is in V_{i_j}. We prove a variant of Lemma 7.18, as follows.

Lemma 7.21. *W.h.p., $|E(V_i, V_j)| = O(m/n^{2/s})$ for all $i, j \in [1, n^{1/s}]$.*

Proof. We set $p = 2n^{-1/s}$. The maximum degree is at most $m/\left(40n^{1/s}\log n\right) \leq mp/20\log n$, and $p^2m \geq 400\log^2 n$. By applying Lemma 7.17 and use the same analysis in Lemma 7.18, we conclude this lemma. □

Proof of Theorem 7.5. Here we only consider the time complexity to deliver all messages. Consider a vertex v. If $k_v < 1/2$, then v receives no message. Otherwise v is responsible for between $2k_v$ and $4k_v$ s-tuples, and v collects $E(V_i, V_j)$ for at most $4s^2k_v$ pairs (V_i, V_j). By Lemma 7.21, w.h.p., $|E(V_i, V_j)| = O(m/n^{2/s})$ for all i, j. Hence the number of edges v received is at most $O(m/n^{2/s}) \cdot 4s^2k_v = O(\deg(v) \cdot n^{(s-2)/s})$.

Note that each vertex v sends at most $O(\deg(v)n^{(s-2)/s})$ messages since for each incident edge e of v, there are at most $O(n^{(s-2)/s})$ s-tuples involving e. By Theorem 7.2, the delivery of all messages can be done in $O(n^{(s-2)/s+o(1)})$ rounds, w.h.p. □

Chapter 8

Conclusion and Future Directions

In this thesis, we have investigated several fundamental questions of the locality of distributed graph problems. In particular, one of the main objectives of our research is to understand the spectrum of natural problem complexities that can exist in the LOCAL model. After a sequence of works [13, 14, 15, 31, 32, 38, 39, 42, 61, 69], we now have a pretty good understanding of the LOCAL complexity landscape for paths/cycles, bounded degree trees, and bounded degree general graphs; but there are still plenty of intriguing open questions.

The Complexity of Distributed LLL. In view of Figure 1.1, one of the most important open problems of the LOCAL complexity landscape is to determine the complexity of distributed LLL.

Conjecture 1. *There exists a sufficiently large constant c such that the complexity of the distributed LLL problem under criterion $pd^c < 1$ is $O(\log\log n)$ in* RandLOCAL *and $O(\log n)$ in* DetLOCAL.

Conjecture 1 is known to be true for tree-structured dependency graphs. In Chapter 5 we proved the following upper bounds. Under criterion $p(ed)^\lambda < 1$, distributed LLL can be solved in $O(\max\{\log_\lambda n, \log n/\log\log n\})$ time in DetLOCAL and $O(\max\{\log_\lambda \log n, \log\log n/\log\log\log n\})$ time in RandLOCAL, with no dependency on d. In the proof of our tree-structured LLL algorithm, first we developed a specialized network decomposition for trees that, with [61], yields a *deterministic* LLL algorithm with complexity $O(\max\{\log_\lambda n, \log n/\log\log n\})$, under LLL criterion $p(ed)^\lambda < 1$. Second, we developed a new method for shattering the dependency graph into $\text{poly}(\Delta)\log n$-size components,

in just $O(\log_\lambda \log n)$ time. Interestingly, the shattering routine is not concerned with the parameters of the LLL *per se*; it simply finds a stable state in a certain contagion process played out on the tree. By composing the graph shattering routine and the deterministic algorithm, we arrive at a final complexity of $O(\max\{\log_\lambda \log n, \log\log n/\log\log\log n\})$. We would like to see whether this approach can be applied to a broader graph class than trees.

For bounded degree general graphs, the current best upper bound [69] for T_{LLL} is $\exp^{(i)}\left(C\sqrt{\log^{(i+1)} n}\right)$, for any $1 \leq i \leq \log^* n - 2\log^*\log^* n$, where C is some universal constant. Here $\exp^{(i)}$ and $\log^{(i)}$ are iterated i-fold application of exp and log, respectively.

The Complexity Landscape on Bounded Degree Trees. The results of Balliu et al. [15] imply that the complexity hierarchies for bounded degree trees and bounded degree general graphs are definitely different. Whereas trees have no natural complexities between $\omega(\log n)$ and $n^{o(1)}$ (Theorem 3.5), there are an infinite number of such complexities on general graphs [15].

Are there any LCL problems whose complexity on bounded degree trees is in the range $\Omega(\log(\log^* n))$—$o(\log^* n)$? From the results of Balliu et al. [15] we know that the complexity landscape of this region is very dense for the case of bounded degree general graph, but the construction of the LCLs in [15] relies heavily on the fact that the graph class under consideration supports short cycles, so it is unlikely the result of [15] can be generalized to the case of trees.

Conjecture 2. *There exists no LCL problem whose* RandLOCAL *and* DetLOCAL *complexities are within* $\omega(1)$ *and* $o(\log^* n)$ *on bounded degree trees.*

We conjecture that the known gap $\omega(1)$—$o(\log(\log^* n))$ for bounded degree trees can be extended to $\omega(1)$—$o(\log^* n)$. In a recent work of Balliu et al. [16], they showed that the weak 2-coloring[46] problem has complexity $\Omega(\log^* n)$ on even degree regular trees, improving upon the previous $\Omega(\log\log^* n)$ lower bound. As a corollary of this result, they showed that Conjecture 2 is true for the case of even degree regular trees (without input labels). This offers a compelling evidence in favor of Conjecture 2.

Decidability of LCL Complexity. Our results in Chapter 4 implies that one can write a computer program that is able to automatically design an asymptotically optimal

[46]The weak 2-coloring problems asks for a 2-coloring of the vertices such that each vertex v has at least one neighbor $u \in N(v)$ that is colored differently than v.

distributed algorithm for any given LCL problem on paths or cycles. Is it possible to extend this result to a broader graph class?

Naor and Stockmeyer [116] showed that it is *undecidable* whether a given LCL problem can be solved in $O(1)$ time, even on grid graphs. The intuition underlying their proof is that one can encode a Turing machine as an LCL in such a way that an execution of the Turing machine corresponds to a valid labeling of the LCL on the grid graph. Extending such an undecidability proof to trees seems to be impossible, and so we have the following conjecture.

Conjecture 3. *The asymptotically optimal* RandLOCAL *and* DetLOCAL *complexities of an LCL problem on bounded degree trees are decidable.*

Recall that in Theorem 3.5 we proved that the $\omega(\log n)$—$n^{o(1)}$ gap on bounded degree trees is decidable, which is an evidence supporting Conjecture 3.

The Complexity of Distributed Edge Coloring. In Chapter 5 we have proved several new upper and lower bounds on the complexity of edge-coloring problems on general graphs and trees. Pedagogically, our simplified $\Omega(\log \log n)$ lower bound for sinkless orientation [31] and $(2\Delta - 2)$-edge coloring is appropriate for a single lecture in a distributed computing course. Our $(\Delta + \tilde{O}(\sqrt{\Delta}))$-edge coloring algorithm is simple, but tricky to analyze, and requires a general distributed LLL algorithm to be made efficient.

After our work [38], the palette size of distributed edge coloring has been significantly improved [76, 140]. In particular, Su and Vu recently showed that a $(\Delta + 2)$-edge coloring can be found in $O(\text{poly}(\Delta, \log n))$ time. Note that this palette size is only one color away from the existential bound of $\Delta + 1$ from Vizing's theorem! However, these improved algorithms [76, 140] have time complexity $O(\text{poly}(\Delta, \log n))$, which is inefficient when Δ is large. In Section 5.8 we proved that such a time complexity is necessary for a natural class of algorithms based on recoloring. It is still open whether a $(\Delta + \tilde{O}(1))$-edge coloring can be found in $O(\text{poly} \log n)$ time.

The Complexity of Distributed Vertex Coloring. In Chapter 6 We have presented a randomized $(\Delta + 1)$-list coloring algorithm that requires $O(\mathsf{Det}_d(\text{poly} \log n))$ rounds of communication, which is *syntactically* close to the $\Omega(\mathsf{Det}(\text{poly} \log n))$ lower bound. Recall that Det and Det_d are the deterministic complexities of $(\Delta+1)$-list coloring and $(\deg+1)$-list coloring. When Δ is unbounded (relative to n), the best known algorithms for $(\Delta+1)$- and

$(\deg+1)$-list coloring are the same: they use Panconesi and Srinivasan's [120] $2^{O(\sqrt{\log n})}$-time construction of network decompositions. Even if optimal $(O(\log n), O(\log n))$-network decompositions could be computed *for free*, we still do not know how to solve $(\Delta+1)$-list coloring faster than $O(\log^2 n)$ time. Thus, reducing the $\mathsf{Det}_d(\text{poly}\log n)$ term in our running time below $O((\log\log n)^2)$ will require a radically new approach to the problem.

It is an open problem to generalize our algorithm to solve the $(\deg+1)$-list coloring problem, and here it may be useful to think about a problem of intermediate difficulty, at least conceptually. Define $(\deg+1)$-coloring to be the coloring problem when v's palette is $\{1,\ldots,\deg(v)+1\}$ (rather than an arbitrary set of $\deg(v)+1$ colors).[47] Whether the problem is $(\deg+1)$-coloring or $(\deg+1)$-*list* coloring, the difficulty is generalizing the notion of "ϵ-friend edge" and "ϵ-sparse vertex" to graphs with irregular degrees. See Figure 8.1 for an extreme example illustrating the difficulty of $(\deg+1)$-list coloring. Suppose $N(v)$ is partitioned into sets S_1, S_2 with $|S_1| = |S_2| = |N(v)|/2 = s$. The graphs induced by $S_1 \cup \{v\}$ and $S_2 \cup \{v\}$ are $(s+1)$-cliques and there are no edges joining S_1 and S_2. The palettes of vertices in S_1 and S_2 are, respectively, $[1, s+1]$ and $[s+1, 2s+1]$.

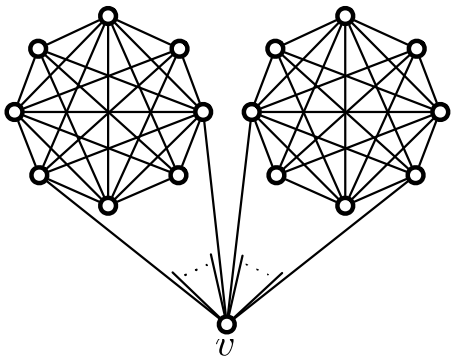

Figure 8.1: An example illustrating the difficulty of $(\deg+1)$-list coloring.

Notice that v is ϵ-sparse according to our definition (for any $\epsilon < 1/2$) and yet regardless of how we design the initial coloring step, we cannot hope to create more than *one* excess color at v since the two palettes $[1, s+1] \cap [s+1, 2s+1] = \{s+1\}$ only intersect at one color. Thus, it must be wrong to classify v as "ϵ-sparse" since it does not satisfy key properties of ϵ-sparse vertices. On the other hand, if v is to be classified as "ϵ-dense" then it is not clear whether we can recover any of the useful properties of ϵ-dense vertices from Lemma 6.4, e.g., that they form almost cliques with $O(1)$ weak diameter and have

[47] We are aware of one application [5] in distributed computing where the palettes are fixed in this way.

external degrees bounded by $O(\epsilon\Delta)$. This particular issue does not arise in instances of the $(\deg+1)$-coloring problem, which suggests that attacking this problem may be a useful conceptual stepping stone on the way to solving $(\deg+1)$-*list* coloring.

Distributed Expander Decomposition. In Chapter 7 we have presented a new approach of designing CONGEST algorithms using expander decomposition. Based on this approach, we showed that all variants of Triangle Detection, Enumeration, and Counting can be solved in $\tilde{O}(n^{1/2})$ rounds in the CONGEST model. In contrast, the previous state-of-the-art bounds for Triangle Detection and Enumeration were $\tilde{O}(n^{2/3})$ and $\tilde{O}(n^{3/4})$, respectively, due to Izumi and Le Gall [90].

The bottleneck in our Triangle Detection algorithm is not triangle-finding *per se*, but in the decomposition of the graph into expanding subgraphs. Our graph decomposition routine takes $\tilde{O}(n^{1/2})$ time and produces three edge sets, the third one inducing a subgraph with arboricity $O(n^{1/2})$. We believe that this third set is unnecessary, and that the running time can be improved substantially. Also, we would like to further investigate the trade-off between the runtime and the quality of the decomposition (e.g., the mixing time/conductance of the components in E_m, the size of E_r, and the arboricity of the subgraph induced by E_s).

If the runtime for the graph decomposition can be reduced to $n^{1/3+o(1)}$, then the time complexity of Triangle Enumeration in CONGEST can be improved to $n^{1/3+o(1)}$, which nearly matches the $\tilde{\Omega}(n^{1/3})$ lower bound in CONGESTED-CLIQUE [90, 123]. This will confirm that Triangle Enumeration is indeed a distributed problem where non-local communication does very little help.

Expander decomposition has found applications in various fields of theoretical computer science [9, 96, 114, 139]. Can this tool be applied to other distributed problems than Triangle Detection and Enumeration? It would be interesting to see more applications of this technique in the field of distributed computing.[48]

[48] We are aware of an application of expander decomposition in the GOSSIP model of distributed computing [33]. In their work [33], the expander decomposition is used in the analysis, not in the algorithm.

Appendix A

Concentration Bounds

A.1 Concentration Bounds

We make use of the following standard tail bounds [53]. Let X be binomially distributed with parameters (n, p), i.e., it is the sum of n independent 0-1 variables with mean p. We have the following bound on the lower tail of X:

$$\Pr[X \leq t] \leq \exp\left(\frac{-(\mu - t)^2}{2\mu}\right), \quad \text{where } t < \mu = np.$$

Chernoff Bound. We have the following multiplicative Chernoff bounds on the upper/lower tail of X with mean $\mu = np$.

$$\Pr[X \geq (1+\delta)\mu] \leq \exp\left(\frac{-\delta^2\mu}{3}\right) \qquad \text{if } \delta \in [0,1]$$

$$\Pr[X \geq (1+\delta)\mu] \leq \exp\left(\frac{-\delta\mu}{3}\right) \qquad \text{if } \delta > 1$$

$$\Pr[X \leq (1-\delta)\mu] \leq \exp\left(\frac{-\delta^2\mu}{2}\right) \qquad \text{if } \delta \in [0,1]$$

Chernoff bounds also hold when X is the sum of n *negatively correlated* 0-1 random variables [53, 54] with mean p, i.e., total independence is not required.

Hoeffding's Inequality. Consider the scenario where $X = \sum_{i=1}^{n} X_i$, and each X_i is an independent random variable bounded by the interval $[a_i, b_i]$. Let $\mu = \mathrm{E}[X]$. Then we have

the following concentration bound (Hoeffding's inequality) [87].

$$\Pr[X \geq (1+\delta)\mu] \leq \exp\left(\frac{-2(\delta\mu)^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Bibliography

[1] A. Abboud, K. Censor-Hillel, S. Khoury, and C. Lenzen. "Fooling Views: A New Lower Bound Technique for Distributed Computations under Congestion". In: *arXiv preprint arXiv:1711.01623* (2017).

[2] U. Agarwal, V. Ramachandran, V. King, and M. Pontecorvi. "A Deterministic Distributed Algorithm for Exact Weighted All-Pairs Shortest Paths in $\tilde{O}(n^{3/2})$ Rounds". In: *Proceedings 38th ACM Symposium on Principles of Distributed Computing (PODC)*. 2018, pp. 199–205.

[3] N. Alon, L. Babai, and A. Itai. "A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem". In: *J. Algor.* 7 (1986), pp. 567–583.

[4] N. Alon, M. Krivelevich, and B. Sudakov. "Coloring Graphs with Sparse Neighborhoods". In: *J. Comb. Theory, Ser. B* 77.1 (1999), pp. 73–82. DOI: 10.1006/jctb.1999.1910.

[5] A. Amir, O. Kapah, T. Kopelowitz, M. Naor, and E. Porat. "The Family Holiday Gathering Problem or Fair and Periodic Scheduling of Independent Sets". In: *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 2016, pp. 367–375.

[6] R. Andersen, F. R. K. Chung, and K. J. Lang. "Local Partitioning for Directed Graphs Using PageRank". In: *Internet Mathematics* 5.1 (2008), pp. 3–22.

[7] E. Arjomandi. "An Efficient Algorithm for Colouring the Edges of a Graph With $\Delta + 1$ Colours". In: *INFOR: Information Systems and Operational Research* 20.2 (1982), pp. 82–101.

[8] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[9] S. Arora, B. Barak, and D. Steurer. "Subexponential Algorithms for Unique Games and Related Problems". In: *J. ACM* 62.5 (Nov. 2015), 42:1–42:25. ISSN: 0004-5411.

[10] S. Arora, S. Rao, and U. Vazirani. "Expander Flows, Geometric Embeddings and Graph Partitioning". In: *J. ACM* 56.2 (Apr. 2009), 5:1–5:37. ISSN: 0004-5411.

[11] S. Assadi, Y. Chen, and S. Khanna. "Sublinear Algorithms for $(\Delta + 1)$ Vertex Coloring". In: *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 767–786.

[12] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. "Network Decomposition and Locality in Distributed Computation". In: *Proceedings 30th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1989, pp. 364–369.

[13] A. Balliu, S. Brandt, Y.-J. Chang, D. Olivetti, M. Rabie, and J. Suomela. "The Distributed Complexity of Locally Checkable Problems on Paths is Decidable". In: *ArXiv e-prints 1811.01672* (2018). arXiv: 1811.01672 [cs.DC].

[14] A. Balliu, S. Brandt, D. Olivetti, and J. Suomela. "Almost Global Problems in the LOCAL Model". In: *32nd International Symposium on Distributed Computing (DISC 2018)*. Ed. by U. Schmid and J. Widder. Vol. 121. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 9:1–9:16.

[15] A. Balliu, J. Hirvonen, J. H. Korhonen, T. Lempiäinen, D. Olivetti, and J. Suomela. "New Classes of Distributed Time Complexity". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2018. New York, NY, USA: ACM, 2018, pp. 1307–1318.

[16] A. Balliu, J. Hirvonen, D. Olivetti, and J. Suomela. "Hardness of Minimal Symmetry Breaking in Distributed Computing". In: *CoRR* abs/1811.01643 (2018). arXiv: 1811.01643.

[17] L. Barenboim. "Deterministic $(\Delta + 1)$-Coloring in Sublinear (in Δ) Time in Static, Dynamic and Faulty Networks". In: *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing (PODC)*. 2015, pp. 345–354. DOI: 10.1145/2767386.2767410.

[18] L. Barenboim and M. Elkin. "Deterministic Distributed Vertex Coloring in Polylogarithmic Time". In: *J. ACM* 58.5 (2011), p. 23.

[19] L. Barenboim and M. Elkin. "Sublogarithmic Distributed MIS Algorithm for Sparse Graphs using Nash-Williams Decomposition". In: *Distributed Computing* 22.5-6 (2010), pp. 363–379.

[20] L. Barenboim, M. Elkin, and U. Goldenberg. "Locally-Iterative Distributed $(\Delta + 1)$-Coloring below Szegedy-Vishwanathan Barrier, and Applications to Self-Stabilization and to Restricted-Bandwidth Models". In: *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*. 2018, pp. 437–446.

[21] L. Barenboim, M. Elkin, and F. Kuhn. "Distributed $(\Delta + 1)$-Coloring in Linear (in Δ) Time". In: *SIAM J. Comput.* 43.1 (2014), pp. 72–95.

[22] L. Barenboim, M. Elkin, and T. Maimon. "Deterministic Distributed $(\Delta + o(\Delta))$-Edge-Coloring, and Vertex-Coloring of Graphs with Bounded Diversity". In: *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC)*. 2017.

[23] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. "The Locality of Distributed Symmetry Breaking". In: *J. ACM* 63.3 (2016), 20:1–20:45. ISSN: 0004-5411.

[24] R. Bar-Yehuda, K. Censor-Hillel, and G. Schwartzman. "A Distributed $(2+\epsilon)$-Approximation for Vertex Cover in $O(\log\Delta/\epsilon\log\log\Delta)$ Rounds". In: *Proceedings 35th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 2016, pp. 3–8.

[25] J. Beck. "An Algorithmic Approach to the Lovász Local Lemma. I". In: *Random Struct. Algorithms* 2.4 (1991), pp. 343–366.

[26] M. Ben-Or. "Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols". In: *Proceedings of the 2nd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*. 1983, pp. 27–30. DOI: 10.1145/800221.806707.

[27] T. Bisht, K. Kothapalli, and S. V. Pemmaraju. "Brief Announcement: Super-fast t-ruling Sets". In: *Proceedings 33rd ACM Symposium on Principles of Distributed Computing (PODC)*. 2014, pp. 379–381.

[28] B. Bollobás. "Chromatic Number, Girth and Maximal Degree". In: *Discrete Mathematics* 24.3 (1978), pp. 311–314. DOI: http://dx.doi.org/10.1016/0012-365X(78)90102-4.

[29] B. Bollobás. *Extremal graph theory*. Vol. 11. London Mathematical Society Monographs. London: Academic Press Inc. [Harcourt Brace Jovanovich Publishers], 1978.

[30] G. Bracha. "An Asynchronous $(n-1)/3$-Resilient Consensus Protocol". In: *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing (PODC)*. 1984, pp. 154–162. DOI: 10.1145/800222.806743.

[31] 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: *Proceedings 48th ACM Symposium on the Theory of Computing (STOC)*. 2016, pp. 479–488.

[32] S. Brandt, J. Hirvonen, J. H. Korhonen, T. Lempiäinen, P. R. Östergård, C. Purcell, J. Rybicki, J. Suomela, and P. Uznański. "LCL Problems on Grids". In: *Proceedings of the ACM Symposium on Principles of Distributed Computing*. PODC '17. Washington, DC, USA, 2017, pp. 101–110. ISBN: 978-1-4503-4992-5.

[33] K. Censor-Hillel, B. Haeupler, J. Kelner, and P. Maymounkov. "Rumor Spreading with No Dependence on Conductance". In: *SIAM Journal on Computing* 46.1 (2017), pp. 58–79.

[34] K. Censor-Hillel, P. Kaski, J. H. Korhonen, C. Lenzen, A. Paz, and J. Suomela. "Algebraic Methods in the Congested Clique". In: *Distributed Computing* (2016). DOI: https://doi.org/10.1007/s00446-016-0270-2.

[35] Y.-J. Chang, V. Dani, T. P. Hayes, Q. He, W. Li, and S. Pettie. "The Energy Complexity of Broadcast". In: *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*. 2018, pp. 95–104.

[36] Y.-J. Chang, W. Jin, and S. Pettie. "Simple Contention Resolution via Multiplicative Weight Updates". In: *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*. Ed. by J. T. Fineman and M. Mitzenmacher. Vol. 69. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 16:1–16:16.

[37] Y.-J. Chang, M. Fischer, M. Ghaffari, J. Uitto, and Y. Zheng. "The Complexity of $(\Delta + 1)$ Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation". In: *CoRR* abs/1808.08419 (2018). arXiv: 1808.08419.

[38] Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto. "The Complexity of Distributed Edge Coloring with Small Palettes". In: *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2018, pp. 2633–2652.

[39] Y.-J. Chang, T. Kopelowitz, and S. Pettie. "An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model". In: *SIAM Journal on Computing* 48.1 (2019), pp. 122–143.

[40] Y.-J. Chang, T. Kopelowitz, S. Pettie, R. Wang, and W. Zhan. "Exponential Separations in the Energy Complexity of Leader Election". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. New York, NY, USA: ACM, 2017, pp. 771–783.

[41] Y.-J. Chang, W. Li, and S. Pettie. "An Optimal Distributed $(\Delta+1)$-coloring Algorithm?" In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2018. New York, NY, USA: ACM, 2018, pp. 445–456.

[42] Y.-J. Chang and S. Pettie. "A Time Hierarchy Theorem for the LOCAL Model". In: *SIAM Journal on Computing* 48.1 (2019), pp. 33–69.

[43] Y.-J. Chang, S. Pettie, and H. Zhang. "Distributed Triangle Detection via Expander Decomposition". In: *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2019, pp. 821–840.

[44] K.-M. Chung, S. Pettie, and H.-H. Su. "Distributed Algorithms for the Lovász Local Lemma and Graph Coloring". In: *Distributed Computing* 30 (2017), pp. 261–280.

[45] R. Cole and U. Vishkin. "Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking". In: *Information and Control* 70.1 (1986), pp. 32–53.

[46] A. Czumaj and C. Konrad. "Detecting Cliques in CONGEST Networks". In: *Proceedings 32nd International Symposium on Distributed Computing (DISC)*. Ed. by U. Schmid and J. Widder. Vol. 121. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 16:1–16:15. ISBN: 978-3-95977-092-7.

[47] A. Czygrinow, M. Hanckowiak, and M. Karonski. "Distributed $O(\Delta \log n)$-edge-coloring algorithm". In: *Proc. ESA 2001*. 2001, pp. 345–355.

[48] X. Dahan. "Regular graphs of large girth and arbitrary degree". In: *Combinatorica* 34.4 (2014), pp. 407–426. DOI: 10.1007/s00493-014-2897-6.

[49] A. Das Sarma, S. Gollapudi, and R. Panigrahy. "Sparse Cut Projections in Graph Streams". In: *Proceedings 17th European Symposium on Algorithms (ESA)*. 2009, pp. 480–491.

[50] D. Dolev, C. Lenzen, and S. Peled. ""Tri, Tri Again": Finding Triangles and Small Subgraphs in a Distributed Setting". In: *Proceedings 26th International Symposium on Distributed Computing (DISC)*. 2012, pp. 195–209.

[51] A. Drucker, F. Kuhn, and R. Oshman. "On the power of the congested clique model". In: *Proceedings 33rd ACM Symposium on Principles of Distributed Computing (PODC)*. 2014, pp. 367–376.

[52] D. P. Dubhashi, D. A. Grable, and A. Panconesi. "Near-Optimal, Distributed Edge Colouring via the Nibble Method". In: *Theor. Comput. Sci.* 203.2 (1998), pp. 225–251. DOI: 10.1016/S0304-3975(98)00022-X.

[53] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[54] D. P. Dubhashi and D. Ranjan. "Balls and bins: A study in negative dependence". In: *J. Random Structures and Algs.* 13.2 (1998), pp. 99–124.

[55] M. Elkin, S. Pettie, and H.-H. Su. "$(2\Delta - 1)$-Edge Coloring is Much Easier than Maximal Matching in the Distributed Setting". In: *Proceedings 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2015, pp. 355–370.

[56] M. Elkin. "A Simple Deterministic Distributed MST Algorithm, with Near-Optimal Time and Message Complexities". In: *Proceedings 37th ACM Symposium on Principles of Distributed Computing (PODC)*. 2017, pp. 157–163.

[57] M. Elkin. "Distributed exact shortest paths in sublinear time". In: *Proceedings 49th Annual ACM Symposium on Theory of Computing (STOC)*. 2017, pp. 757–770.

[58] G. Even, O. Fischer, P. Fraigniaud, T. Gonen, R. Levi, M. Medina, P. Montealegre, D. Olivetti, R. Oshman, I. Rapaport, and I. Todinca. "Three Notes on Distributed Property Testing". In: *Proceedings 31st International Symposium on Distributed Computing (DISC)*. Vol. 91. Leibniz International Proceedings in Informatics (LIPIcs). 2017, 15:1–15:30.

[59] L. Feuilloley and P. Fraigniaud. "Randomized Local Network Computing". In: *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 2015, pp. 340–349.

[60] M. J. Fischer, N. A. Lynch, and M. Paterson. "Impossibility of Distributed Consensus with One Faulty Process". In: *J. ACM* 32.2 (1985), pp. 374–382. DOI: 10.1145/3149.214121.

[61] M. Fischer and M. Ghaffari. "Sublogarithmic Distributed Algorithms for Lovász Local Lemma with Implications on Complexity Hierarchies". In: *Proceedings 31st International Symposium on Distributed Computing (DISC)*. 2017, 18:1–18:16.

[62] M. Fischer, M. Ghaffari, and F. Kuhn. "Deterministic Distributed Edge Coloring via Hypergraph Maximal Matching". In: *Proceedings 58th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 180–191.

[63] M. Fischer. "Improved Deterministic Distributed Matching via Rounding". In: *31st International Symposium on Distributed Computing (DISC 2017)*. Ed. by A. W. Richa. Vol. 91. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 17:1–17:15.

[64] O. Fischer, T. Gonen, F. Kuhn, and R. Oshman. "Possibilities and Impossibilities for Distributed Subgraph Detection". In: *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures (SPAA)*. Vienna, Austria: ACM, 2018, pp. 153–162. ISBN: 978-1-4503-5799-9.

[65] P. Fraigniaud, M. Heinrich, and A. Kosowski. "Local conflict coloring". In: *Proceedings 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2016, pp. 625–634.

[66] P. Fraigniaud. "Distributed Computational Complexities: Are You Volvo-addicted or Nascar-obsessed?" In: *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*. PODC '10. New York, NY, USA: ACM, 2010, pp. 171–172. DOI: 10.1145/1835698.1835700. URL: http://doi.acm.org/10.1145/1835698.1835700.

[67] H. N. Gabow, T. Nishizeki, O. Kariv, D. Leven, and O. Terada. *Algorithms for edge-coloring graphs*. Technical Report TRECIS-8501. Tohoku University, 1985.

[68] M. Ghaffari. "An improved distributed algorithm for maximal independent set". In: *Proceedings 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2016, pp. 270–277. DOI: 10.1137/1.9781611974331.ch20.

[69] M. Ghaffari, D. G. Harris, and F. Kuhn. "On Derandomizing Local Distributed Algorithms". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 662–673.

[70] M. Ghaffari, J. Hirvonen, F. Kuhn, Y. Maus, J. Suomela, and J. Uitto. "Improved Distributed Degree Splitting and Edge Coloring". In: *Proceedings 31st International Symposium on Distributed Computing (DISC)*. 2017, 19:1–19:15.

[71] M. Ghaffari and H.-H. Su. "Distributed Degree Splitting, Edge Coloring, and Orientations". In: *Proceedings 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2017, pp. 2505–2523. DOI: 10.1137/1.9781611974782.166.

[72] M. Ghaffari. "Distributed MIS via All-to-All Communication". In: *Proceedings of the ACM Symposium on Principles of Distributed Computing*. PODC '17. Washington, DC, USA: ACM, 2017, pp. 141–149. ISBN: 978-1-4503-4992-5. DOI: 10.1145/3087801.3087830. URL: http://doi.acm.org/10.1145/3087801.3087830.

[73] M. Ghaffari and B. Haeupler. "Distributed Algorithms for Planar Networks I: Planar Embedding". In: *Proceedings 36th ACM Symposium on Principles of Distributed Computing (PODC)*. 2016, pp. 29–38.

[74] M. Ghaffari and B. Haeupler. "Distributed Algorithms for Planar Networks II: Low-Congestion Shortcuts, MST, and Min-Cut". In: *Proceedings 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2016, pp. 202–219.

[75] M. Ghaffari, J. Hirvonen, F. Kuhn, and Y. Maus. "Improved Distributed Delta-Coloring". In: *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. PODC '18. New York, NY, USA: ACM, 2018, pp. 427–436.

[76] M. Ghaffari, F. Kuhn, Y. Maus, and J. Uitto. "Deterministic distributed edge-coloring with fewer colors". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. ACM. 2018, pp. 418–430.

[77] M. Ghaffari, F. Kuhn, and H.-H. Su. "Distributed MST and Routing in Almost Mixing Time". In: *Proceedings 37th ACM Symposium on Principles of Distributed Computing (PODC)*. 2017, pp. 131–140. DOI: 10.1145/3087801.3087827. URL: http://doi.acm.org/10.1145/3087801.3087827.

[78] M. Ghaffari and J. Li. "New Distributed Algorithms in Almost Mixing Time via Transformations from Parallel Algorithms". In: *Proceedings 32nd International Symposium on Distributed Computing (DISC)*. Ed. by U. Schmid and J. Widder. Vol. 121. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 31:1–31:16. ISBN: 978-3-95977-092-7.

[79] A. V. Goldberg and S. A. Plotkin. "Parallel $(\Delta + 1)$-coloring of constant-degree graphs". In: *Information Processing Letters* 25.4 (1987), pp. 241–245.

[80] A. Goldberg, S. Plotkin, and G. Shannon. "Parallel symmetry-breaking in sparse graphs". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM. 1987, pp. 315–324.

[81] T. Gonen and R. Oshman. "Lower Bounds for Subgraph Detection in the CONGEST Model". In: *Proceedings 21st International Conference on Principles of Distributed Systems (OPODIS)*. Vol. 95. Leibniz International Proceedings in Informatics (LIPIcs). 2018, 6:1–6:16.

[82] M. Göös, J. Hirvonen, R. Levi, M. Medina, and J. Suomela. "Non-local Probes Do Not Help with Many Graph Problems". In: *Distributed Computing*. Ed. by C. Gavoille and D. Ilcinkas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 201–214.

[83] D. A. Grable. "A Large Deviation Inequality for Functions of Independent, Multi-Way Choices". In: *Combinatorics, Probability & Computing* 7.1 (1998), pp. 57–63.

[84] R. L. Graham, B. L. Rothschild, and J. H. Spencer. *Ramsey Theory*. 2nd. New York: John Wiley and Sons, 1990.

[85] M. Hańćkowiak, M. Karoński, and A. Panconesi. "On the distributed complexity of computing maximal matchings". In: *SIAM J. Discrete Mathematics* 15.1 (2001), 41–57 (electronic).

[86] D. G. Harris, J. Schneider, and H.-H. Su. "Distributed $(\Delta + 1)$-Coloring in Sublogarithmic Rounds". In: *J. ACM* 65.4 (Apr. 2018), 19:1–19:21. ISSN: 0004-5411. DOI: 10.1145/3178120. URL: http://doi.acm.org/10.1145/3178120.

[87] W. Hoeffding. "Probability Inequalities for Sums of Bounded Random Variables". In: *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30.

[88] I. Holyer. "The NP-Completeness of Edge-Coloring". In: *SIAM Journal on Computing* 10.4 (1981), pp. 718–720.

[89] C. Huang, D. Nanongkai, and T. Saranurak. "Distributed Exact Weighted All-Pairs Shortest Paths in $\tilde{O}(n^{5/4})$ Rounds". In: *Proceedings 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 168–179.

[90] T. Izumi and F. Le Gall. "Triangle Finding and Listing in CONGEST Networks". In: *Proceedings 37th ACM Symposium on Principles of Distributed Computing (PODC)*. 2017, pp. 381–389. DOI: 10.1145/3087801.3087811. URL: http://doi.acm.org/10.1145/3087801.3087811.

[91] M. Jerrum and A. Sinclair. "Approximating the Permanent". In: *SIAM Journal on Computing* 18.6 (1989), pp. 1149–1178.

[92] Ö. Johansson. "Simple Distributed $\Delta + 1$-coloring of Graphs". In: *Info. Proc. Lett.* 70.5 (1999), pp. 229–232.

[93] T. Jurdziński and K. Nowicki. "MST in $O(1)$ Rounds of Congested Clique". In: *Proceedings 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2018, pp. 2620–2632.

[94] R. Kannan, S. Vempala, and A. Vetta. "On Clusterings: Good, Bad and Spectral". In: *J. ACM* 51.3 (May 2004), pp. 497–515. ISSN: 0004-5411. DOI: 10.1145/990308.990313. URL: http://doi.acm.org/10.1145/990308.990313.

[95] H. J. Karloff and D. B. Shmoys. "Efficient Parallel Algorithms for Edge Coloring Problems". In: *J. Algorithms* 8.1 (1987), pp. 39–52. DOI: 10.1016/0196-6774(87)90026-5.

[96] K.-I. Kawarabayashi and M. Thorup. "Deterministic Edge Connectivity in Near-Linear Time". In: *J. ACM* 66.1 (Dec. 2018), 4:1–4:50. ISSN: 0004-5411. DOI: 10.1145/3274663. URL: http://doi.acm.org/10.1145/3274663.

[97] V. King and J. Saia. "Byzantine Agreement in Expected Polynomial Time". In: *J. ACM* 63.2 (Mar. 2016), 13:1–13:21. ISSN: 0004-5411. DOI: 10.1145/2837019. URL: http://doi.acm.org/10.1145/2837019.

[98] J. H. Korhonen and J. Rybicki. "Deterministic Subgraph Detection in Broadcast CONGEST". In: *Proceedings 21st International Conference on Principles of Distributed Systems (OPODIS)*. Vol. 95. Leibniz International Proceedings in Informatics (LIPIcs). 2018, 4:1–4:16.

[99] A. Korman, J.-S. Sereni, and L. Viennot. "Toward more localized local algorithms: removing assumptions concerning global knowledge." In: *Distributed Computing* 26.5–6 (2013), pp. 289–308.

[100] K. Kothapalli and S. V. Pemmaraju. "Super-Fast 3-Ruling Sets". In: *Proceedings IARCS Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. Vol. 18. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012, pp. 136–147.

[101] S. Krinninger and D. Nanongkai. "A Faster Distributed Single-Source Shortest Paths Algorithm". In: *Proceedings 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 686–697.

[102] F. Kuhn, T. Moscibroda, and R. Wattenhofer. "Local Computation: Lower and Upper Bounds". In: *J. ACM* 63.2 (2016), 17:1–17:44.

[103] F. Kuhn and R. Wattenhofer. "On the Complexity of Distributed Graph Coloring". In: *Proceedings 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 2006, pp. 7–15.

[104] F. Kuhn and A. R. Molla. "Distributed Sparse Cut Approximation". In: *Proceedings 19th International Conference on Principles of Distributed Systems (OPODIS)*. 2015, 10:1–10:14.

[105] C. Lenzen. "Optimal Deterministic Routing and Sorting on the Congested Clique". In: *Proceedings 33rd ACM Symposium on Principles of Distributed Computing (PODC)*. 2013, pp. 42–50. ISBN: 978-1-4503-2065-8. DOI: 10.1145/2484239.2501983. URL: http://doi.acm.org/10.1145/2484239.2501983.

[106] C. Lenzen and R. Wattenhofer. "Brief Announcement: Exponential Speed-up of Local Algorithms Using Non-local Communication". In: *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*. PODC '10. New York, NY, USA: ACM, 2010, pp. 295–296.

[107] N. Linial. "Locality in Distributed Graph Algorithms". In: *SIAM J. Comput.* 21.1 (1992), pp. 193–201.

[108] M. Luby. "A simple parallel algorithm for the maximal independent set problem". In: *SIAM J. Comput.* 15.4 (1986), pp. 1036–1053.

[109] G. L. Miller and J. H. Reif. "Parallel Tree Contraction–Part I: Fundamentals". In: *Advances in Computing Research* 5 (1989), pp. 47–72.

[110] M. Molloy and B. A. Reed. "A Bound on the Strong Chromatic Index of a Graph". In: *J. Comb. Theory, Ser. B* 69.2 (1997), pp. 103–109. DOI: 10.1006/jctb.1997.1724.

[111] M. Molloy and B. Reed. "Near-optimal list colorings". In: *Random Structures & Algorithms* 17.3-4 (2000), pp. 376–402.

[112] R. A. Moser and G. Tardos. "A constructive proof of the general Lovász local lemma". In: *J. ACM* 57.2 (2010). DOI: 10.1145/1667053.1667060.

[113] G. Moshkovitz and A. Shapira. "Decomposing a graph into expanding subgraphs". In: *Random Struct. Algorithms* 52.1 (2018), pp. 158–178.

[114] D. Nanongkai, T. Saranurak, and C. Wulff-Nilsen. "Dynamic minimum spanning forest with subpolynomial worst-case update time". In: *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 950–961.

[115] M. Naor. "A Lower Bound on Probabilistic Algorithms for Distributive Ring Coloring". In: *SIAM J. Discrete Mathematics* 4.3 (1991), pp. 409–412. DOI: 10.1137/0404036.

[116] M. Naor and L. J. Stockmeyer. "What Can be Computed Locally?" In: *SIAM J. Comput.* 24.6 (1995), pp. 1259–1277. DOI: 10.1137/S0097539793254571.

[117] L. Orecchia and N. K. Vishnoi. "Towards an SDP-based Approach to Spectral Methods: A Nearly-Linear-Time Algorithm for Graph Partitioning and Decomposition". In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2011, pp. 532–545.

[118] L. Orecchia and Z. A. Zhu. "Flow-based Algorithms for Local Graph Clustering". In: *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '14. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2014, pp. 1267–1286.

[119] A. Panconesi and R. Rizzi. "Some simple distributed algorithms for sparse networks". In: *Distributed Computing* 14.2 (2001), pp. 97–100.

[120] A. Panconesi and A. Srinivasan. "On the Complexity of Distributed Network Decomposition". In: *J. Algor.* 20.2 (1996), pp. 356–374.

[121] A. Panconesi and A. Srinivasan. "Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding Bounds". In: *SIAM J. Comput.* 26.2 (1997), pp. 350–368. DOI: 10.1137/S0097539793250767.

[122] A. Panconesi and A. Srinivasan. "The Local Nature of Δ-Coloring and its Algorithmic Applications". In: *Combinatorica* 15.2 (1995), pp. 255–280. DOI: 10.1007/BF01200759. URL: https://doi.org/10.1007/BF01200759.

[123] G. Pandurangan, P. Robinson, and M. Scquizzato. "On the Distributed Complexity of Large-Scale Graph Computations". In: *Proceedings of the 30th ACM Symposium on Parallelism in Algorithms and Architecture (SPAA)*. 2018, pp. 405–414.

[124] M. Parter. "$(\Delta + 1)$ coloring in the congested clique model". In: *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Ed. by I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella. Vol. 107. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 160:1–160:14.

[125] M. Parter and H.-H. Su. "Randomized $(\Delta+1)$ coloring in $O(\log^* \Delta)$ congested clique rounds". In: *32nd International Symposium on Distributed Computing (DISC 2018)*. Ed. by U. Schmid and J. Widder. Vol. 121. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 39:1–39:18.

[126] M. Pătraşcu and M. Thorup. "Planning for Fast Connectivity Updates". In: *Proceedings 48th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2007, pp. 263–271.

[127] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

[128] D. Peleg and V. Rubinovich. "A Near-tight Lower Bound on the Time Complexity of Distributed Minimum-weight Spanning Tree Construction". In: *SIAM J. Comput.* 30.5 (2000), pp. 1427–1442.

[129] S. Pettie and H.-H. Su. "Distributed algorithms for coloring triangle-free graphs". In: *Information and Computation* 243 (2015), pp. 263–280.

[130] M. O. Rabin. "Randomized Byzantine Generals". In: *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1983, pp. 403–409. DOI: 10.1109/SFCS.1983.48.

[131] P. Raghavendra and D. Steurer. "Graph Expansion and the Unique Games Conjecture". In: *Proceedings 42nd ACM Symposium on Theory of Computing (STOC)*. 2010, pp. 755–764.

[132] T. Saranurak and D. Wang. "Expander Decomposition and Pruning: Faster, Stronger, and Simpler". In: *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2019, pp. 2616–2635.

[133] A. D. Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. "Distributed verification and hardness of distributed approximation". In: *SIAM J. Comput.* 41.5 (2012), pp. 1235–1265.

[134] A. D. Sarma, A. R. Molla, and G. Pandurangan. "Distributed Computation of Sparse Cuts via Random Walks". In: *Proceedings 16th International Conference on Distributed Computing and Networking (ICDCN)*. 2015, 6:1–6:10.

[135] J. Schneider and R. Wattenhofer. "A new technique for distributed symmetry breaking". In: *Proceedings 29th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 2010, pp. 257–266.

[136] J. Schneider and R. Wattenhofer. "An optimal maximal independent set algorithm for bounded-independence graphs". In: *Distributed Computing* 22.5-6 (2010), pp. 349–361.

[137] M. Sipser. *Introduction to the Theory of Computation*. 3rd. International Thomson Publishing, 2012.

[138] D. A. Spielman and S.-H. Teng. "A Local Clustering Algorithm for Massive Graphs and Its Application to Nearly Linear Time Graph Partitioning". In: *SIAM J. Comput.* 42.1 (2013), pp. 1–26.

[139] D. A. Spielman and S.-H. Teng. "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In: *Proceedings 36th Annual ACM Symposium on Theory of Computing (STOC)*. 2004, pp. 81–90.

[140] H.-H. Su and H. T. Vu. "Towards the Locality of Vizing's Theorem". In: *Proceedings of the 51th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. ACM. 2019.

[141] L. Trevisan. "Approximation Algorithms for Unique Games". In: *Theory of Computing* 4.5 (2008), pp. 111–128.

[142] V. G. Vizing. "On an estimate of the chromatic class of a p-graph". In: *Diskret. Analiz No.* 3 (1964), pp. 25–30.

[143] V. H. Vu. "A General Upper Bound On The List Chromatic Number Of Locally Sparse Graphs". In: *Combinatorics, Probability & Computing* 11.1 (2002), pp. 103–111. DOI: 10.1017/S0963548301004898.