The Complexity of Distributed Edge Coloring with Small Palettes*

Yi-Jun Chang † Qizheng He ‡ Wenzheng Li § Seth Pettie ¶ Jara Uitto ‖

Abstract

The complexity of distributed edge coloring depends heavily on the *palette size* as a function of the maximum degree Δ. In this paper we explore the complexity of edge coloring in the LOCAL model in different palette size regimes. Our results are as follows.

- We simplify the *round elimination* technique of Brandt et al. [9] and prove that $(2\Delta-2)$-edge coloring requires $\Omega(\log_\Delta \log n)$ time w.h.p. and $\Omega(\log_\Delta n)$ time deterministically, *even on trees*. The simplified technique is based on two ideas: the notion of an *irregular running time* (in which network components terminate the algorithm at prescribed, but irregular times) and some general observations that transform *weak* lower bounds into *stronger* ones.
- We give a randomized edge coloring algorithm that can use palette sizes as small as $\Delta + \tilde{O}(\sqrt{\Delta})$, which is a natural barrier for randomized approaches. The running time of the algorithm is at most $O(\log \Delta \cdot T_{LLL})$, where T_{LLL} is the complexity of a permissive version of the constructive Lovász local lemma.
- We develop a new distributed Lovász local lemma algorithm for *tree-structured dependency graphs*, which leads to a $(1+\epsilon)\Delta$-edge coloring algorithm for trees running in $O(\log \log n)$ time. This algorithm arises from two new results: a deterministic $O(\log n)$-time LLL algorithm for tree-structured instances, and a randomized $O(\log \log n)$-time *graph shattering* method for breaking the dependency graph into independent $O(\log n)$-size LLL instances.
- 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, e.g., via "augmenting 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 [32], which exploit the existence of $O(\log_\Delta n)$-length augmenting paths.

*Supported by NSF grants CCF-1514383 and CCF-1637546 and ERC Grant No. 336495 (ACDC).

†University of Michigan.

‡The Institute for Theoretical Computer Science (ITCS), Institute for Interdisciplinary Information Sciences, Tsinghua University.

§The Institute for Theoretical Computer Science (ITCS), Institute for Interdisciplinary Information Sciences, Tsinghua University.

¶University of Michigan.

‖ETH Zürich and University of Freiburg.

1 Introduction

We study *edge coloring*[1] problems in the well known LOCAL[2] model of distributed computation [28, 34], which for clarity we bifurcate into RandLOCAL and DetLOCAL depending on whether random bits are available. 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. In Section 1.1 we review previous edge coloring algorithms in descending order by palette size (see Table 1) and in Section 1.4 we summarize our contributions.

1.1 Edge Coloring Algorithms 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 [28] 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 [33, 20], $(2\Delta-1)$-edge coloring is solved in $\min\{2^{O(\sqrt{\log n})}, \tilde{O}(\sqrt{\Delta}) + \log^* n\}$ time. Barenboim, Elkin, and Maimon [6] gave deterministic algorithms for $(2^k\Delta)$-edge coloring ($k \geq 2$) in $\tilde{O}(k\Delta^{1/2k} + \log^* n)$.

Barenboim, Elkin, Pettie, and Schneider [7] proved that $O(\log \Delta)$ iterations of the natural randomized $(2\Delta-1)$-edge coloring algorithm effectively *shatters* the graph into uncolored components of poly$(\log n)$ ver-

[1] A k-edge coloring is a function $\phi : E \to \{1, \ldots, k\}$ such that edges sharing an endpoint are colored differently; "k" is called the *palette size*.

[2] The LOCAL model has the following features. The input graph $G = (V, E)$ is identical to the distributed network; vertices are identified with processors and edges with bi-directional communication links; each $v \in V$ initially knows $\deg(v)$, a port-numbering of its incident edges, and global parameters such as $n = |V|$ and $\Delta = \max_v \deg(v)$; *time* is divided into synchronized rounds, and in each round each processor can perform unlimited computation and communicate an unbounded-length message to each of its neighbors, which is delivered before the next round. Depending on the problem the vertices may carry additional input labels. The output of a LOCAL algorithm is typically a labeling of V or E that satisfies some constraints. In the RandLOCAL model the output labeling is correct w.h.p. $(1 - 1/\text{poly}(n))$. In the DetLOCAL model each vertex is assigned a unique $O(\log n)$-bit ID; the output labeling must always be correct.

tices, which can then be colored using a deterministic algorithm.[3] Elkin, Pettie, and Su [17] proved that when $\Delta > (\log n)^{1+\gamma}$, $(2\Delta-1)$-edge coloring can be solved in $O(\log^* n)$ time in RandLOCAL. Very recently Fischer, Ghaffari, and Kuhn proved that $(2\Delta-1)$-edge coloring can be solved in $O(\log^7 \Delta \log n)$ time in DetLOCAL. Together with [7] and [17], this implies a RandLOCAL algorithm running in $\min\{O((\log\log n)^8), O(\log^7 \Delta \log\log n)\}$ time. By using a slightly larger palette of $(2+\epsilon)\Delta$ colors, $\epsilon > 1/\log\Delta$, Ghaffari et al. [23] (cf. [24]) gave a faster DetLOCAL edge coloring algorithm running in time $O(\epsilon^{-1}\log^2\Delta \log\log\Delta(\log\log\log\Delta)^{1.71}\log n)$.

The "$2\Delta-1$" 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 [16] 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 [17] 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 1.3 for a discussion of the distributed LLL. The $(1+\epsilon)\Delta$-edge coloring problem is solved in $O(\log^* n \cdot \left\lceil \frac{\log n}{\Delta^{1-o(1)}} \right\rceil)$ time via the Chung-Pettie-Su LLL algorithm [13].[4] 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 natural limit for randomized coloring strategies is a $(\Delta+O(\sqrt{\Delta}))$-size palette,[5] which was achieved in 1987 by Karloff and Shmoys [26] 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 [36] proved that $\Delta+1$ suffices for any graph, and Holyer [25] proved that it is NP-hard to tell if a graph is Δ-colorable. The best *sequential* $(\Delta+1)$-edge coloring algorithms [1, 21] 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 [1, 21] 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 [32] in their distributed algorithm for Brooks' theorem.[6] 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.

1.2 Lower Bounds Linial's $\Omega(\log^* n)$ lower bound for $O(1)$-coloring the ring [28, 31] shows that $f(\Delta)$-edge coloring also cannot be computed in $o(\log^* n)$ time, for any function f.

None of the other published lower bounds apply 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/edge coloring problem. Linial's $O(\log_\Delta n)$ lower bound [28] (see [35, 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. [9] (in RandLOCAL) and Chang, Kopelowitz, and Pettie [11] (in DetLOCAL) for *sinkless orientation* and Δ-*vertex coloring trees* do not naturally generalize to edge coloring. Indeed, Brandt et al.'s lower bound technique oscillates between sinkless orientation and a closely related problem called sinkless coloring, *whose input already consists of a* Δ*-edge colored graph.*

1.3 The Distributed Lovász Local Lemma 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 $\mathrm{vbl}(A) \subset \mathcal{V}$. Define the dependency graph as $G_{\mathcal{E}} = (\mathcal{E}, \{(A,B) \mid \mathrm{vbl}(A) \cap \mathrm{vbl}(B) \neq \emptyset\})$. 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 $\Pr[\cap_{A\in\mathcal{E}} \overline{A}] > 0$. The (constructive) LLL problem is to assign values to all variables in $\mathcal{V}$ such that no event in $\mathcal{E}$ happens.

[3] The problem of coloring one of these components is a $(2\deg(v)-1)$-list edge coloring problem, i.e., v's palette includes an arbitrary set of $2\deg(v)-1$ colors.

[4] If Δ is sufficiently small ($\Delta \ll \log^\epsilon n$ for some specific $\epsilon > 0$), this algorithm can be sped up using a recent LLL algorithm of Fischer and Ghaffari [18].

[5] 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.

[6] Which states that any graph with $\Delta \geq 3$ having no $(\Delta+1)$-cliques is Δ-vertex colorable.

Palette Size	Time	(Rand)	Notes	References
$f(\Delta)$	Lower bound: $\Omega(\log^* n)$	R	$\Delta = O(1)$	[28, 31]
$O(\Delta^2)$	$\log^* n - \log^* \Delta + 1$	$\star$	Vertex coloring $L(G)$	[28]
$\Delta^{1+\epsilon}$	$C_\epsilon \log \Delta \log n$		Vertex coloring $L(G)$	[4]
$\Delta \log n$	$\log^4 n$			[14]
$C_\epsilon \Delta$	$\Delta^\epsilon \log n$		Vertex coloring $L(G)$	[4]
$2^k \Delta$	$k\Delta^{1/2k} + \log^* n$	$\star$	$k \geq 2$	[6, 24]
$(2+\epsilon)\Delta$	$\epsilon^{-3} \log^{11} n$			[24]
	$\epsilon^{-1} \log \Delta^{2+o(1)} \log n$	$\star$	$\epsilon > 1/\log \Delta$	[23]
$2\Delta - 1$	$2^{O(\sqrt{\log n})}$		Vertex coloring $L(G)$	[33]
	$\tilde{O}(\sqrt{\Delta}) + \log^* n$	$\star$	Vertex coloring $L(G)$	[20]
	$\log \Delta + 2^{O(\sqrt{\log \log n})}$		Vertex coloring $L(G)$	[7]
	$\log^* n$	R$\star$	$\Delta > (\log n)^{1+o(1)}$	[17]
	$2^{O(\sqrt{\log \log n})}$	R		[17]
	$\log^7 \Delta \log n$	$\star$		[19]
	$\min\{(\log \log n)^8, \log^7 \Delta \log \log n\}$	R$\star$		[7]+[17]+[19]
$2\Delta - 2$	Lower bound: $\Omega(\log_\Delta \log n)$	R		**new**
	Lower bound: $\Omega(\log_\Delta n)$			**new**
$(1+\epsilon)\Delta$	$\epsilon^{-1} \log \epsilon^{-1} + \log n$	R	$\Delta > (\log n)^{1+\gamma(\epsilon)}$	[16]
	$(\epsilon^{-2} \log \epsilon^{-1} + \log^* \Delta) \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil$	R	$\Delta > \Delta_\epsilon$	[17]
	$\log \epsilon^{-1} \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil + \log^* n$	R	$\epsilon\Delta > (\log n)^{1+o(1)}$	**new**
	$\log \epsilon^{-1} \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil + (\log \log n)^{3+o(1)}$	R$\star$	$\Delta > \Delta_\epsilon$	**new**
$\Delta + \tilde{O}(\sqrt{\Delta})$	$\log \Delta \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil + (\log \log n)^{3+o(1)}$	R$\star$		**new**
$\Delta + 1$	diameter(G)	$\star$		[36]

Table 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 [2, 27, 5, 3], 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 $\star$.

Distributed Lovász Local Lemma. The distributed LLL problem 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 the dependency graph $G_{\mathcal{E}}$ of the LLL system.

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 $1 - 1/\text{poly}(n)$ or even 1. Table 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.[7]

Chang and Pettie [12] conjectured that the RandLOCAL complexity of the LLL under some polynomial criterion (e.g., $p(ed)^c < 1$ for some fixed c) is $O(\log \log n)$, matching the Brandt et al. [9] lower bound. If this conjecture were true, the results of [11, Theorem 3] indicate what the optimal algorithm should look like: it must combine an $O(\log n)$-time DetLOCAL LLL algorithm and an $O(\log \log n)$-time *graph shattering* routine to break the dependency graph into $\text{poly}(\log n)$-size LLL instances. Fischer and Ghaffari [18] 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 $\log n$-size components.

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 [9] *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 [9]. 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 [9]. By following the proofs of Chang, Kopelowitz, and Pettie [11], any randomized lower bound of this type implies $\Omega(\log_\Delta n)$ lower bounds in DetLOCAL [11, Theorem 5], and hence $\Omega(\log_\Delta \log n)$ lower bounds in RandLOCAL [11, Theorem 3].

Faster $(1+\epsilon)\Delta$-edge Coloring. The $(1+\epsilon)\Delta$-edge coloring algorithms of [16, 17] 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 [7, 17, 23]. When $\epsilon^2 \Delta \gg \log n$ the error is $1/\text{poly}(n)$; otherwise we can invoke a distributed LLL algorithm [30, 13, 18]. The weird-looking term $\left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil$ in Table 1 is due to the $O(\log_{1/epd^2} n)$-time LLL algorithm of [13], with $1/epd^2 = \exp(\epsilon^2 \Delta^{1-o(1)})$.

Upper Bounds on Trees. Our lower bounds on $(2\Delta - 2)$-edge coloring apply 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 [18], 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 $O(\log_\lambda \log n)$, improv-

[7] In coloring algorithms it is typical to see $d = \text{poly}(\Delta)$ and $p = \exp(-\Omega(d^{\Omega(1)}))$.

Symmetric LLL Criterion	Time	Rand/Det	Notes	Reference
$ep(d+1) < 1$	$O(\mathsf{MIS} \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[30]
	$O(\mathsf{WeakMIS} \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[13]
	$O(\log d \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[22]+[13]
$epd^2 < 1$	$O(\log_{1/epd^2} n)$	Rand	also asymmetric criterion	[13]
$\mathrm{poly}(d)2^d < 1$	$O(\log n / \log\log n)$	Rand		[13]
$p(ed)^\lambda < 1$	$O(n^{1/\lambda} \cdot 2^{O(\sqrt{\log n})})$	Det	Any $\lambda \geq 1$	[18]
$p(ed)^{4\lambda} < 1$	$O(d^2 + (\log n)^{1/\lambda} \cdot 2^{O(\sqrt{\log\log n})})$	Rand	Any $\lambda \geq 8$	[18]
$p(ed)^{32} < 1$	$2^{O(\sqrt{\log\log n})}$	Rand	Requires $d < (\log\log n)^{1/5}$	[18]
$p(ed)^{d^2+1} < 1$	$O(d^2 + \log^* n)$	Det		[18]
Lower Bounds (apply to tree-structured instances)				
$p \cdot f(d) < 1$	$\Omega(\log^* n)$	Rand	Any f	[13]
$p \cdot f(d) \leq 1$	$\Omega(\log_{\log(1/p)} \log n)$	Rand	Any $f(d) \leq 2^d$	[9]
$p \cdot f(d) \leq 1$	$\Omega(\log_{\log(1/p)} n)$	Det	Any $f(d) \leq 2^d$	[11]
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 2: A survey of distributed LLL algorithms. $\mathsf{MIS} = O(\min\{d + \log^* n, \log d + 2^{O(\sqrt{\log\log n})}\})$ [5, 22] is the complexity of computing a maximal independent set in a graph with maximum degree d. $\mathsf{WeakMIS} = O(\log d)$ [22] is the task of finding an independent set I such that the probability that v is not in/adjacent to I is $1/\mathrm{poly}(d)$. All lower bounds apply even to tree-structured instances. The lower bounds of [9, 11] ($\Omega(\log_d \log n)$ randomized, $\Omega(\log_d n)$ deterministically) were for an LLL instance satisfying $p2^d \leq 1$. By a change of parameters, they also imply stronger lower bounds (substituting $\log(1/p)$ for d) under any LLL criterion $p \cdot f(d) \leq p2^d \leq 1$.

ing the $O(d^2 + \log^* n)$-time shattering routine of [18] 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 [9] and the conjectured upper bound for general instances [12, Conjecture 1]. See Table 2. We do not optimize the requirement $\lambda \geq 2(4^r + 8r)$.

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 [32].

Organization of the paper. In Section 2 we give lower bounds on $(2\Delta-2)$-edge coloring. In Section 3 we give a randomized $(1+\epsilon)\Delta$-edge coloring algorithm, In Section 4 we give new LLL algorithms for tree-structured dependency graphs. In Section 5 we show some results on edge-coloring trees deterministically, which may be folklore. In Section 6 we present the network decomposition algorithms on trees used in Section 4. In Section 7 we give lower bounds on a class of "recoloring" algorithms for Vizing's theorem. We conclude in Section 8. Some proof details in Section 3 and Section 5 are omitted; see [10] for the full proofs.

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

The *sinkless orientation* problem is to orient (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-colored bipartite graphs is reducible to $(2\Delta-2)$-edge coloring.

THEOREM 2.1. *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-colored bipartite 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 $\phi : E \to \{1,\dots,2\Delta-2,\perp\}$ 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 $\phi : E' \to \{1,\dots,2\Delta-2,\perp\}$, then orients the edges (in the direction $0 \to 1$ or $0 \leftarrow 1$) as follows. Let $\{u_0,u_1\} \in E$ be an edge with u_j colored $j \in \{0,1\}$.

- $\mathcal{A}_{s.o.}(\{u_0,u_1\}) = 0 \to 1$ if $\{u_0,u_1\} \in E'$ and $\phi(u_0,u_1) \in \{1,2,\dots,\Delta-1,\perp\}$, or if only u_0 selected $\{u_0,u_1\}$.
- $\mathcal{A}_{s.o.}(\{u_0,u_1\}) = 1 \to 0$ if $\{u_0,u_1\} \in E'$ and $\phi(u_0,u_1) \in \{\Delta,\dots,2\Delta-2\}$, or if only u_1 selected $\{u_0,u_1\}$.

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 $\phi(e) \in \{1,2,\dots,\Delta-1,\perp\}$. Criterion (iii) only occurs with probability at most p.

Thus, any lower bound for sinkless orientation on 2-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.[8]

For simplicity we suppose that the *edges* host processors (not vertices), 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* (edge coloring, vertex coloring, random bits, etc.) stored in their processors. By definition, a time-t algorithm has time profile $(t,t,t,\cdots,t)$.

LEMMA 2.1. **(Round Elimination Lemma)** *Suppose $\mathcal{A}_{s.o.}$ is a sinkless orientation algorithm for $\mathcal{T}_\Delta$ with error probability p and time profile $(\underbrace{t,t,\cdots,t}_{i},t-1,\cdots,t-1)$, i.e., edges colored $\{1,\dots,i\}$ halt after t rounds and the others after $t-1$ rounds. There exists a sinkless*

[8] 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.

orientation algorithm $\mathcal{A}'_{s.o.}$ with error probability $3p^{1/3}$ and time profile $(\underbrace{t, t, \cdots, t}_{i-1}, t-1, \cdots, t-1)$.

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 $\mathcal{A}_{s.o.}$.

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

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

$$\Pr(\mathcal{E}_0 \cap \mathcal{E}_1) \leq 2p \tag{2.1}$$

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

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

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

Note that if we inspect $N^{t-1}(e_0)$ (and condition on the information seen), the events $\mathcal{E}_0$ and $\mathcal{E}_1$ become independent since they now depend on disjoint sets of random variables.[9] Thus,

$$\begin{aligned} &\Pr(\mathcal{E}_0 \cap \mathcal{E}_1 \mid N^{t-1}(e_0)) \\ &= \Pr(\mathcal{E}_0 \mid N^{t-1}(e_0)) \cdot \Pr(\mathcal{E}_1 \mid N^{t-1}(e_0)) \end{aligned} \tag{2.2}$$

Since $\mathcal{E}_0^\star, \mathcal{E}_1^\star$ are determined by $N^{t-1}(e_0)$, (2.2) implies that $\Pr(\mathcal{E}_0 \cap \mathcal{E}_1 \mid \mathcal{E}_0^\star \cap \mathcal{E}_1^\star) \geq p^{2/3}$, and with (2.1) we deduce that

$$\Pr(\mathcal{E}_0^\star \cap \mathcal{E}_1^\star) \leq 2p^{1/3} \tag{2.3}$$

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

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

The failure probability at a vertex not adjacent to any edge colored i is the same under $\mathcal{A}_{s.o.}$ and $\mathcal{A}'_{s.o.}$.

[9] $\mathcal{E}_0$ depends on $\bigcup_{j\in[1,\Delta-1]} N^t(e_j) \backslash N^{t-1}(e_0)$ and $\mathcal{E}_1$ depends on $\bigcup_{j\in[\Delta,2\Delta-2]} N^t(e_j) \backslash N^{t-1}(e_0)$.

We calculate the failure probabilities of the remaining vertices now.

$$\begin{aligned} \Pr(u_0 \text{ is a sink}) &= \Pr(\overline{\mathcal{E}_0^\star} \cap \mathcal{E}_0) \\ &\leq \Pr(\mathcal{E}_0 \mid \overline{\mathcal{E}_0^\star}) \ \leq \ p^{1/3}, \end{aligned}$$

which follows from the definition of $\mathcal{E}_0^\star$.

$$\begin{aligned} \Pr(u_1 \text{ is a sink}) &= \Pr(\mathcal{E}_0^\star \cap \mathcal{E}_1) \\ &\leq \Pr(\mathcal{E}_0^\star \cap \mathcal{E}_1^\star) + \Pr(\mathcal{E}_1 \cap \overline{\mathcal{E}_1^\star}) \\ &\leq 2p^{1/3} + p^{1/3} \ = \ 3p^{1/3}, \end{aligned}$$

which follows from (2.3) and the definition of $\mathcal{E}_1^\star$.

LEMMA 2.2. *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 2.1 $t(2\Delta-1)$ 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 $0 \rightarrow 1$. 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 2.2. *Even on 2-colored trees or 2-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/$(2\Delta-2)$-edge coloring algorithm with local probability of failure p. Lemma 2.2 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 $\Omega(\log_\Delta n)$ [15, 8],

we get $\Omega(\min\{\Delta^{-1}\log\log p^{-1}, \log_\Delta n\})$ lower bounds. Following the same proof as [11, Theorem 5], this implies an $\Omega(\log_\Delta n)$ lower bound in DetLOCAL, which, according to [11, Theorem 3], 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 2.2 *automatically* implies a stronger lower bound.

3 Randomized Edge Coloring Algorithm

Elkin, Pettie, and Su [17] 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 [17] 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.

The algorithm of [17] is based on the Rödl Nibble method as follows. After the ith iteration of the algorithm, a certain invariant $\mathcal{H}_{i+1}$ holds at each vertex w.h.p. in Δ, given that $\mathcal{H}_i$ holds everywhere beforehand. To ensure that the invariant $\mathcal{H}_{i+1}$ is met, a distributed Lovász Local Lemma algorithm is applied in each iteration of the algorithm. Their algorithm requires $O(\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon}))$ iterations,[10] which is inefficient if ϵ is small, e.g., $1/\text{poly}(\Delta)$. In this section, we prove the following theorem, which improves upon the algorithm of [17].

THEOREM 3.1. *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*

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

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 3.1. 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 an 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 $\text{poly}(\log n)$-size graphs. This is achieved by applying the $(\tilde{\Delta}+1)$-vertex coloring algorithm of [7] 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 [17]. The state-of-the-art DetLOCAL algorithm [23] 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 [17, 7, 23], we have

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

This is achieved as follows. If $\Delta' = \Omega(\log^2 n)$, we run the $O(\log^* n)$-time RandLOCAL algorithm of [17]. Otherwise, we run the RandLOCAL graph shattering phase of [7] (using the first $2\Delta'$ colors) followed by the DetLOCAL algorithm of [23] (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 3.1, which satisfy the criterion $p(ed)^\lambda < 1$ with $\lambda = \Omega(\log^x\Delta)$. Using our randomized LLL algorithm for trees (Section 4), we have

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

We claim that $T^*(n,\Delta') = O(\log^*\Delta' + \log_{\Delta'}\log n)$ on trees. This is achieved as follows. First, do

[10] To design a $\Delta(1+\epsilon)$-edge coloring algorithm for very small ϵ, one cannot to afford to have too many iterations. Observe that each iteration necessarily incur at least a $1 \pm O(1/\sqrt{\Delta})$ factor of drift to the parameters (e.g., palette size), since these parameters are upper bounded by $O(\Delta)$. If the number of iterations is much higher than $\sqrt{\Delta}$, then the effect of the drift becomes non-negligible. For instance, if the number of iterations is $\Omega(1/\epsilon^2)$ (which is the case of [17]), then we cannot make $\epsilon = o(\Delta^{-1/4})$.

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 [17] without invoking distributed LLL. Then, apply our deterministic $O(\log_{\Delta'} \tilde{n})$-time algorithm for Δ'-edge coloring trees (Section 5) 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

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

This matches our $\Omega(\log_\Delta \log n)$ lower bound (Section 2) when $1/\epsilon, \Delta = O(1)$.

3.1 Main 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, for each vertex v, each edge e, and each color c, we maintain the following invariant $\mathcal{H}_i$.

Invariant $\mathcal{H}_i$: (i) $\deg_i(v) \le d_i$, (ii) $\deg_{c,i}(v) \le t_i$, and (iii) $|\Psi_i(e)| \ge p_i$.

Parameters. Given two numbers $\eta \ge 1$ 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 ($i > 1$):

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

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 graph induced by the uncolored edges (i.e., G_{i+1}) 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 $\epsilon \ge \frac{\eta^{2.5}}{\sqrt{\Delta}}$. 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. See [10] for a proof.

LEMMA 3.1. *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 3.1, we have the following lemma.

LEMMA 3.2. *Let $i^\star = O(\log(1/\epsilon)) = O(\log\Delta)$ be the largest index such that $\beta_{i^\star-1} \le 1/\eta$. Then the following four equations hold for any $1 < i \le i^\star$.*

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

Proof. To prove the lemma, it suffices to show that the condition of Lemma 3.1 is met for all indices $1 < i \le i^\star$. We prove this by an induction on i. Suppose by the induction hypothesis the four equations hold at index $i-1$. We show that the condition of Lemma 3.1 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 $\min\{d_{i-1}, t_{i-1}, p_{i-1}\} = \omega(\log\Delta)$.

$$\begin{aligned} &\min\{d_{i-1}, t_{i-1}, p_{i-1}\} \\ &\ge (1\pm o(1))\Delta(1-e^{-2})^{2(i-1)} \\ &\qquad\qquad \text{(Induction hypothesis for } d_{i-1}, t_{i-1}, p_{i-1}) \\ &= (1\pm o(1))\Delta\cdot\left(\tfrac{(1\pm o(1))\xi}{\beta_{i-1}}\right)^2 \\ &\qquad\qquad \text{(Induction hypothesis for } \beta_{i-1}) \\ &\ge (1\pm o(1))\xi^2\eta^2\Delta \qquad (\beta_{i-1} \le 1/\eta) \\ &= \Omega(\eta^5) = \omega(\log\Delta) \end{aligned}$$

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 3.2, we have:

$$\begin{aligned} d_{i^\star} &= (1\pm o(1))\Delta(1-e^{-2})^{i^\star-1} && \text{(Lemma 3.2 for } d_{i^\star}) \\ &= (1\pm o(1))\Delta\cdot\xi/\beta_{i^\star} && \text{(Lemma 3.2 for } \beta_{i^\star}) \\ &\le (1\pm o(1))\xi\eta\Delta && (\beta_{i^\star} > 1/\eta) \end{aligned}$$

For our choices of η and ξ, we have $d_{i^\star} \approx \xi\eta\Delta = \frac{\epsilon\Delta}{6}$. Thus, the terminating condition $d_i \le \frac{1}{5}(\epsilon-\xi)\Delta$ 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 3.2 for d_i. For each $1 < i \le i^\star$, we have:

$$\begin{aligned} &\delta_i^2\cdot\min\{d_i, t_i, p_i\} \\ &= \beta_i^2 t_i/\eta^2 && \text{(Definition of } \delta_i) \\ &= (1\pm o(1))\cdot\left(\xi/(1-e^{-2})^{i-1}\right)^2 \\ &\quad\cdot\left(\Delta(1-e^{-2})^{2(i-1)}\right)/\eta^2 && \text{(Lem. 3.2 for } t_i,\ \beta_i) \\ &= (1\pm o(1))\cdot\Delta(\xi/\eta)^2 \\ &= \Omega(\epsilon^2\Delta/\eta^4) && \text{(Definition of } \xi) \\ &= \omega(\log\Delta). && \text{(Definition of } \epsilon) \end{aligned}$$

We will later see in Section 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.

3.2 Maintenance of Invariant In this section we show how to apply a distributed LLL algorithm, with parameters $d = \text{poly}(\Delta)$ and $p = \exp(-\Omega\left(\delta_i^2\cdot\min\{d_i, t_i, p_i\}\right)$, 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.

Consider the following 1-round coloring procedure on the modified 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 [10] for a proof.

LEMMA 3.3. *Suppose that $\mathcal{H}_i$ holds. The following concentration bounds hold for any $\delta > 0$.*

$$\Pr\left[|S^\diamond(v)| > (1+\delta)d_i^\diamond\right] < \exp\left(-\Omega(\delta^2 d_i)\right)$$
$$\Pr\left[|N_c^\diamond(v)| > (1+\delta)t_i^\diamond \mid N_c^\diamond(v) \neq \emptyset\right] < \exp\left(-\Omega(\delta^2 t_i)\right)$$
$$\Pr\left[|\Psi^\diamond(e)| < (1-\delta)p_i^\diamond \mid e \text{ remains uncolored}\right] < \exp\left(-\Omega(\delta^2 p_i)\right)$$

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 both are 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.

4 Lovász Local Lemma on Trees

In this section, we study distributed LLL on tree-structured instances. 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, i.e., $\text{vbl}(E(v)) = \bigcup_{u \in N^{r/2}(v)} \mathcal{V}(u)$. If S is a subset of the vertices, we use $\text{vbl}(S)$ to be short for $\bigcup_{v \in S} \text{vbl}(E(v)) = \bigcup_{v \in S} \bigcup_{u \in N^{r/2}(v)} \mathcal{V}(u)$. The dependency graph for this set $\mathcal{E}$ of bad events is exactly T^r, obtained by connecting vertices at distance at most r in T, so $d \leq (\Delta(T))^r$. *Tree-structured* dependency graphs arise naturally from any constant-time ($r/2$ time) RandLOCAL experiment that is run on a tree topology.

4.1 Deterministic LLL Algorithms A (λ, γ)-*network decomposition* is a partition of the vertex set into $V_1, \ldots, V_\lambda$ such that connected components induced by V_i have diameter at most γ. Fischer and Ghaffari [18] 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 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 4.1. (FISCHER AND GHAFFARI [18]) *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 4.1 uses new network decompositions for trees; see Section 6.

THEOREM 4.1. *Any tree-structured LLL satisfying $p(ed)^\lambda < 1$ with $\lambda \geq 2$ can be solved in* DetLOCAL *in $O(\max\{\log_\lambda s, \frac{\log s}{\log\log s}\})$ time, where $s \leq n$ is the size of any distance-$O(1)$ dominating set of the tree.*

Proof. Recall that the dependency graph is T^r for some tree T and constant r. In Section 6 we show that a standard $(2, O(\log s))$-decomposition for $(T^r)^2 = T^{2r}$ is computable in $O(\log s)$ 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 $O(\log_\lambda s)$ time.

When $\lambda = O(1)$ is sufficiently small, we apply Lemma 4.1 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 $O(\log_{\hat{\lambda}} s)$ time, where $\hat{\lambda} = \min\{\lambda, \sqrt{\frac{\log s}{\log\log s}}\}$. We solve the LLL by applying Lemma 4.1, which takes time $O(\hat{\lambda}^2 + \log_{\hat{\lambda}} s) = O(\max\{\log_\lambda s, \frac{\log s}{\log\log s}\})$. Observe that because of the $\hat{\lambda}^2$ term, we cannot benefit from LLL instances with $\lambda \gg \sqrt{\frac{\log s}{\log\log s}}$.

Combining Theorem 4.1 with the $O(d^2 + \log^* n)$ graph shattering routine of [18] 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 very small. In Section 4.2 we

give 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.

4.2 Shattering the Dependency Graph In this section we prove the following efficient shattering lemma.

LEMMA 4.2. *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 partial assignment in $O(\log_\lambda \log n)$ time with the following properties.*

1. *No bad event occurs under the assignment.*
2. *The connected components of T^r induced by events with unassigned variables have size at most $\text{poly}(d) \log n$. Moreover, each such component contains a distance-$\frac{3r}{2}$ dominating set (in the tree T) with size at most $\log n$.*
3. *Conditioned on the partial assignment, the probability of any bad event is at most $p' = \sqrt{p}$ and each component satisfies LLL criterion $p'(ed)^{\lambda/2} < 1$.*

By applying Lemma 4.2 and then Theorem 4.1 to each component, we can now efficiently solve tree-structured LLL instances in $O(\log \log n)$ time or faster, independent of the dependency graph degree d.

THEOREM 4.2. *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 4.2 suggests an algorithm to compute such a 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 $\text{vbl}(E(v))$. In general, whenever $\Pr[E(v)|\phi]$ exceeds $\sqrt{p}$, update ϕ by unsetting all variables in $\text{vbl}(E(v))$. 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 stablize. We give an algorithm that computes a stable set (satisfying the other requirements of Lemma 4.2) exponentially faster, by avoiding a direct simulation.

Let u be a vertex in the unoriented tree T. Define $C_u(k, [i, j])$ to be the set of vertices that belong to kth subtree of u such that the distance to u lies in the interval $[i, j]$. For example, $C_u(k, [1, 1])$ is the kth neighbor of u. For any vertex set S, define $\widehat{\deg}_S(u)$ to be the number of different k s.t. $C_u(k, [1, r]) \cap S \neq \emptyset$.

Choose $\mu \geq 4$ and $\lambda' \geq 1$ to be any integers such that $\lambda \geq 2(\mu^r + \lambda')$. The following bad events are defined with respect to a fixed total assignment ϕ to the variables.

$$B(S, v) : \Pr\left[E(v) \mid \text{vbl}(E(v)) \setminus \text{vbl}(S)\right] \geq (ed)^{-\lambda/2}$$
$$B(v) : \bigcup_{S \subset N^r(v), |S| \leq \mu^r} B(S, v).$$

In other words, $B(S, v)$ is the event that, if we *were* to resample $\text{vbl}(S)$ (but leave other variables in $\text{vbl}(E(v))$ set according to ϕ), 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 the right cardinality such that $B(S, v)$ occurs.

We can now consider the probability that these events occur, over a randomly selected total assignment ϕ.

$$\Pr_\phi[B(S, v)] \leq \frac{\Pr_\phi[E(v)]}{\Pr_\phi[E(v) \mid B(S, v)]} \leq \frac{(ed)^{-\lambda}}{(ed)^{-\lambda/2}} \leq (ed)^{-(\mu^r + \lambda')}$$

and, by a union bound over the d^{μ^r} choices of S,

$$\Pr_\phi[B(v)] \leq \sum_S \Pr_\phi[B(S, v)] \leq (ed)^{-\lambda'}.$$

Intuitively $B(v)$ is the event that $E(v)$ is *too close* to happening, i.e., relatively few variables need to be resampled to give $E(v)$ a likely probability of happening. We imagine a contagion process that samples a total variable assignment ϕ, and initially infects S consisting of all v such that $B(v)$ occurs. An uninfected vertex w becomes infected (joins S) if $\widehat{\deg}_S(w) > \mu$ until S is *stable*. Lemma 4.3 proves that the criterion for infection "$\widehat{\deg}_S(v) > \mu$" is a good proxy for the harder-to-analyze criterion "$E(v)$ is dangerously close to happening."

LEMMA 4.3. *Fix a total variable assignment ϕ, and let S be any vertex set such that (i) $v \in S$ if $B(v)$ occurs under ϕ, and (ii) $v \in S$ if $\widehat{\deg}_S(v) > \mu$. Then for* any *$v$, $\Pr[E(v) \mid \text{vbl}(E(v)) \setminus \text{vbl}(S)] < (ed)^{-\lambda/2}$.*

Proof. If $v \in S$, then the probability of seeing $E(v)$ after resampling $\text{vbl}(S)$ is $p < (ed)^{-\lambda}$, so 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) $\text{vbl}(S') \cap \text{vbl}(E(v)) = \text{vbl}(S) \cap \text{vbl}(E(v))$, i.e., (iii) says that resampling $\text{vbl}(S')$ is equivalent to resampling $\text{vbl}(S)$, from v's point of view. Since $v \notin S$, we know $B(S', v)$ does not occur, and so $\Pr[E(v) \mid \text{vbl}(E(v)) \setminus \text{vbl}(S')] < (ed)^{-\lambda/2}$.

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 be the child of v_{i-1} whose subtree contains the largest number of vertices in H. We prove by induction that v_i contains at least $\mu^{r-i} + 1$ H-vertices in its subtree. The base case $i = 0$ holds by assumption. If v_i has $\mu + 1$ subtrees containing H-vertices, then v_i would be infected, so by the pigeonhole principle v_{i+1} must have at least $\lceil (\mu^{r-i} + 1)/\mu \rceil = \mu^{r-(i+1)} + 1$ H-vertices in its subtree. Hence the subtree of v_r contains $\mu^0 + 1 = 2$ H-vertices, which is a contradiction since the only vertex eligible to be in H is v_r itself.

4.3 Contagion Process A (q_0, r, μ)-contagion process on an n-vertex tree T is defined 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. Our goal is to quickly compute a set S that is both *stable* and *small*.

DEFINITION 1. *A set S is* stable *if it causes no more infection and* small *if each connected component induced by $N^{r/2}(S)$ contains a distance-$\frac{3r}{2}$ dominating set (in the tree T) of size at most* $\log n$.

Lemma 4.4 connects the contagion problem to finding a partial assignment satisfying the criteria of Lemma 4.2. Theorem 4.3 shows that one can efficiently compute small stable sets in contagion processes.

LEMMA 4.4. *Suppose there is a τ-round* RandLOCAL *algorithm for finding a small stable set for a $((ed)^{-\lambda'}, r, \mu)$-contagion process. Then there exists a $(\tau + O(1))$-round* RandLOCAL *algorithm for finding a partial assignment to a tree-structured LLL instance satisfying Lemma 4.2(1–3), under criterion $p(ed)^\lambda < 1$, where $\lambda \geq 2(\mu^r + \lambda')$.*

Proof. Consider the contagion process defined by choosing a partial assignment to the variables and initially infecting any vertex v where $B(v)$ occurs. The lower bound on λ implies $\Pr[B(v)] \leq q_0 = (ed)^{-\lambda'}$ and according to Lemma 4.3, any stable set satisfies Lemma 4.2(3).

THEOREM 4.3. *Consider a (q_0, r, μ)-contagion process played on an n-vertex tree T with maximum degree Δ. In* RandLOCAL*, we can compute a* small stable set *S in $O(\log_\mu \log n)$ time, where r is constant, $q_0 \leq e^{-1} d^{-8r}$, $d = \Delta^r$ and $\mu \geq 4$.*

Combining Lemma 4.4 and Theorem 4.3 together, Lemma 4.2 is proved as follows. Pick the largest μ such that $\lambda \geq 2(\mu^r + 8r)$. By Lemma 4.4 we only need to show a $O(\log_\lambda \log n)$-round RandLOCAL algorithm to find a small stable set for a $((ed)^{-8r}, r, \mu)$-contagion process. Since $\lambda \geq 2(4^r + 8r)$ and $\mu \geq 4$, by Theorem 4.3, a small stable set can be computed in $O(\log_\mu \log n) = O(\log_\lambda \log n)$ rounds.

Theorem 4.3 is proved in the remainder of this section. The algorithm for Theorem 4.3 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 $\tau = O(\log_\mu \log n)$, the final infected set L_τ is both stable and small. This process is called Find Small Stable Set. Observe that the sets generated by this process have the following containment.

$$U_0 \subseteq \cdots \subseteq U_\tau = L_0 \supseteq \cdots \supseteq L_\tau.$$

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 $1 \leq i \leq \tau$, 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 $1 \leq i \leq \tau$, do $L_i \leftarrow L_{i-1} \setminus \{u \in L_{i-1} \setminus U_0 \mid \widehat{\deg}_{L_{i-1}}(u) \leq \mu\}$.

(5) Return L_τ.

LEMMA 4.5. *For each vertex $v \notin L_\tau$, $\widehat{\deg}_{L_\tau}(v) \leq \mu$ with high probability, and so L_τ is stable.*

Proof. Root the tree at v. Define $S(u)$ to be the subtree rooted at u. Define $C'_u(k, [i, j])$ to be $C_u(k, [i, j]) \cap S(u)$ and $\deg'_S(u)$ to be $\widehat{\deg}_{S \cap S(u)}(u)$. Let $L_{\tau+1}$ be the set of all vertices u s.t. $\widehat{\deg}_{L_\tau}(u) > \mu$. In general, if $\{X(u)\}$ is an ensemble of events associated with vertices and S a subset of vertices, we define $X(S)$ to be the event $\bigcup_{u \in S} X(u)$ and X to be the set of

vertices $\{v \mid X(v) \text{ occurs}\}$. Consider the following three sequences of events $\{F_i\}_{0\le i\le\tau}$, $\{H_i\}_{0\le i\le\tau}$, and $\{\tilde{F}_i\}_{0\le i\le\tau}$.

$$\begin{aligned} F_i(u):&\ (u\notin U_i)\wedge(u\in L_{i+1}),\\ H_0(u):&\ (u\in U_0),\\ H_{i+1}(u):&\ H_0(u)\vee(\deg'_{H_i}(u)\ge\mu/2),\\ \tilde{F}_0(u):&\ H_\tau(u),\\ \tilde{F}_{i+1}(u):&\ \deg'_{\tilde{F}_i}(u)\ge\mu/2. \end{aligned}$$

Notice that v cannot belong to both $U_\tau\backslash L_\tau$ and $L_{\tau+1}$, since otherwise v is contained in $L_1,\ldots,L_\tau$. Therefore, $F_\tau(v)=(v\notin L_\tau)\wedge(v\in L_{\tau+1})$, and so it suffices to bound the probability of event $F_\tau(v)$ by $1/\text{poly}(n)$ to prove this lemma. We make the following two observations.

Observation 1: $(u\in U_i)\Rightarrow H_i(u)$.

The base case ($i=0$) follows from the definition of H_i. Assume that $(u\in U_{i-1})\Rightarrow H_{i-1}(u)$. We have:

$$\begin{aligned} u\in U_i\backslash U_0 &\Longrightarrow \widehat{\deg}_{U_{i-1}}(u)>\mu/2\\ &\Longrightarrow \deg'_{U_{i-1}}(u)\ge\mu/2\\ &\Longrightarrow \deg'_{H_{i-1}}(u)\ge\mu/2, \end{aligned}$$

and this implies $(u\in U_i)\Rightarrow H_i(u)$.

Observation 2: $F_i(u)\Rightarrow\tilde{F}_i(u)$.

The base case ($i=0$) follows from Observation 1: $F_0(u)\Rightarrow(u\in L_1)\Rightarrow(u\in U_\tau)\Rightarrow H_\tau(u)\Rightarrow\tilde{F}_0(u)$. Assume that $F_{i-1}(u)\Rightarrow\tilde{F}_{i-1}(u)$. By definition, $u\in L_{i+1}$ implies $\widehat{\deg}_{L_i}(u)>\mu$ while $u\notin U_i$ implies $\widehat{\deg}_{U_{i-1}}(u)\le\mu/2$. That is,

$$\begin{aligned} F_i(u) &\Longrightarrow \widehat{\deg}_{F_{i-1}}(u)>\mu/2\\ &\Longrightarrow \deg'_{F_{i-1}}(u)\ge\mu/2\\ &\Longrightarrow \deg'_{\tilde{F}_{i-1}}(u)\ge\mu/2, \end{aligned}$$

and so $F_i(u)\Rightarrow\tilde{F}_i(u)$.

Since $F_\tau(u)\Rightarrow\tilde{F}_\tau(u)$, it remains to bound the probability of event $\tilde{F}_\tau(v)$ by $1/\text{poly}(n)$. For convenience, we write $p_i=\max_u\Pr[\tilde{F}_i(u)]$ and $q_i=\max_u\Pr[H_i(u)]$. We will show that (i) $p_\tau\le(\Delta^{2(r^2+1)}p_0)^{(\frac{\mu}{2})^{\tau/r}}$ and (ii) $p_0=q_\tau\le\Delta^{r^2+2}q_0$. Therefore,

$$\begin{aligned} p_\tau &\le(\Delta^{2(r^2+1)}p_0)^{(\frac{\mu}{2})^{\tau/r}}\\ &\le(\Delta^{3r^2+4}q_0)^{(\frac{\mu}{2})^{\tau/r}}\le 1/\text{poly}(n), \end{aligned}$$

as desired. In the remainder of the proof we derive these two inequalities.

Upper Bound of p_τ. Notice that the event $\tilde{F}_i(u)$ is contained in the following event: "there exist $\mu/2$ many different k such that $\tilde{F}_{i-1}(C'_u(k,[1,r]))$ occurs". A consequence of this observation is that $\tilde{F}_i(C'_u(k,[1,r]))\Rightarrow\tilde{F}_{i-1}(C'_u(k,[2,2r]))$. Similarly, we have:

$$\begin{aligned} \tilde{F}_{i-1}(C'_u(k,[1,r])) &\Longrightarrow \tilde{F}_{i-2}(C'_u(k,[2,2r]))\\ &\Longrightarrow\cdots\\ &\Longrightarrow \tilde{F}_{i-r}(C'_u(k,[r,r^2])). \end{aligned}$$

Thus, the $\tilde{F}_i(u)$ is contained in the following event: "there exists $\mu/2$ many different k such that $\tilde{F}_{i-r}(C'_k(r,r^2))$ occurs". Notice that the events $\tilde{F}_{i-r}(C'_u(k,[r,r^2]))$ for all k are independent, since $\tilde{F}_i(u)$ only depends on variables associated with vertices in $N^r(u)$. By a union bound, $\Pr[\tilde{F}_{i-r}(C'_u(k,[r,r^2]))]\le\Delta^{r^2}p_{i-r}$. Taking a union bound over all $\binom{\Delta}{\mu/2}$ choices of indices for k, we have

$$p_i\le\Delta^{\mu/2}(\Delta^{r^2}p_{i-r})^{\mu/2}\le(\Delta^{r^2+1}p_{i-r})^{\mu/2}.$$

Since $\mu/2\ge 2$, we have

$$\begin{aligned} \Delta^{2(r^2+1)}p_i &\le(\Delta^{2(r^2+1)}p_{i-r})^{\mu/2}\\ &\le\cdots\\ &\le(\Delta^{2(r^2+1)}p_0)^{(\frac{\mu}{2})^{i/r}}, \end{aligned}$$

which implies $p_\tau\le(\Delta^{2(r^2+1)}p_0)^{(\frac{\mu}{2})^{\tau/r}}$.

Upper Bound of p_0. We use a similar argument to derive the bound of $p_0=q_\tau$. Notice that $H_i(u)$ is contained in the event $(u\in U_0)\vee(\deg'_{H_{i-1}}(u)\ge\mu/2)$. A simple consequence of this is that $H_i(C'_u(k,[1,r]))\Rightarrow H_0(C'_u(k,[1,1]))\vee H_{i-1}(C'_u(k,[2,2r]))$. Similarly, we have:

$$\begin{aligned} &H_i(C'_u(k,[1,r]))\\ &\Longrightarrow H_0(C'_u(k,[1,1]))\vee H_{i-1}(C'_u(k,[2,2r]))\\ &\Longrightarrow\cdots\\ &\Longrightarrow H_0(C'_u(k,[1,r-1]))\vee H_{i-r}(C'_u(k,[r,r^2])). \end{aligned}$$

Thus, $H_i(u)$ is contained in the union of the event $H_0(N^r(u))$ and the following event: "there exist $\mu/2$ many different k such that $H_{i-r}(C'_k(r,r^2))$ occurs". The events $H_{i-r}(C'_u(k,[r,r^2]))$ for all k are independent, since $\tilde{F}_i(u)$ only depends on the variables associated with vertices in $N^r(u)$. By a union bound, $\Pr[H_{i-r}(C'_u(k,[r,r^2]))]\le\Delta^{r^2}q_{i-r}$. By taking a union bound over $N^r(u)$ and another union bound over all $\binom{\Delta}{\mu/2}$ choices of indices for k, we have:

$$q_i\le\Delta^{r+1}q_0+\Delta^{\mu/2}(\Delta^{r^2}q_{i-r})^{\mu/2}.$$

We show by induction that $q_i \leq \Delta^{r^2+2} q_0$ for each i. Suppose that $q_{i-r} \leq \Delta^{r^2+2} q_0$, then

$$\begin{aligned} q_i &\leq \Delta^{r+1} q_0 + \Delta^{\mu/2} (\Delta^{r^2} q_{i-r})^{\mu/2} \\ &\leq \Delta^{r^2+1} q_0 + (\Delta^{2r^2+3} q_0)^{\mu/2}. \end{aligned}$$

Since $q_0 \leq e^{-1}\Delta^{-8r^2}$ and $\mu \geq 4$, we have $(\Delta^{2r^2+3} q_0)^{\mu/2} \ll \Delta^{r^2+1} q_0$, and so $q_i \leq 2\Delta^{r^2+1} q_0 \leq \Delta^{r^2+2} q_0$.

LEMMA 4.6. *With high probability, each connected component in the subgraph of T induced by $N^{r/2}(U_\tau)$ contains a distance-$\frac{3r}{2}$ dominating set (in the tree T) of size at most $\log n$, and so L_τ is small.*

Proof. Define $H = T^{[r+1,3r]}$ to be the graph obtained by connecting vertices in T at distance in the range $[r+1, 3r]$. We first show that, with high probability, H has no connected component D such that (i) $|D| \geq \log n$, and (ii) at least half of vertices in D belong to U_0. The existence of such a component implies that H contains a subtree of $\log n$ vertices with at least half of them in U_0. There are at most $4^{\log n}$ different such tree topologies and each can be embedded into H in less that $n \cdot \Delta^{2r \log n}$ ways. Moreover, there are at most $2^{\log n}$ ways of choosing the positions of vertices in U_0 on trees and the probability that such a particular tree occurs in H is at most $q_0^{\log n/2}$. A union bound over all trees lets us conclude that such component exists with probability at most

$$4^{\log n} \cdot n \cdot \Delta^{2r \log n} \cdot 2^{\log n} \cdot q_0^{\log n/2} \leq 1/\mathrm{poly}(n).$$

Let S be a connected component in T induced by $N^{r/2}(U_\tau)$. We pick a distance-$\frac{3r}{2}$ dominating set D of S (in the tree T) greedily, preferring vertices in U_0 over U_1, and U_1 over U_2, etc. Each time a vertex is picked we remove from consideration all vertices in its r-neighborhood. Recall that $U_0 \subseteq \ldots \subseteq U_\tau$, and notice that D does not contain any vertex not in U_τ. The reason that D is a distance-$\frac{3r}{2}$ dominating set is as follows. The set D is obviously a distance-r dominating set of $U_\tau \cap S$ (in the tree T), and $U_\tau \cap S$ is a distance-$r/2$ dominating set of S (in the tree T).

We write u_i to denote the ith vertex added to D, and write $D_i = \{u_1, \ldots, u_i\}$. Let m_i denote the number of connected components induced by D_i in the graph $T^{[r+1,2r]}$ (which is the graph obtained by connecting vertices in T at distance in the range $[r+1, 2r]$). 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, and so $|D| < \log n$ with high probability.

We prove this 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 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 to U_j because u had at least $\mu/2 \geq 2$ subtrees containing U_{i-1}-vertices that are within distance-r to u. 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 $N^r(u)$. 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 we should pick v_k instead of u. Observe that w_1 and w_2 belong to separate connected components of D_{i-1} in the graph $T^{[r+1,2r]}$, but w_1, w_2, and u are in the same component of D_i in the graph $T^{[r+1,2r]}$ since $w_k \in N^r(v_k) \subseteq N^{2r}(u)$ for $k = 1, 2$.

5 Deterministic Algorithms for Edge-coloring Trees

Let $T = (V, E)$ be a tree with n vertices and $N^+(v)$ be the inclusive neighborhood of v. We decompose T using two operations inspired by Miller and Reif [29], the second of which is parameterized by an integer $k \geq 2$.

Rake: Remove all leaves and isolated vertices from T.

Compress: Remove the following set from T:

$$\{v \in V \mid \text{for every } u \in N^+(v), \deg_T(u) \leq k\}.$$

THEOREM 5.1. *Alternately applying compress and rake $\log_k n + 1$ times removes all vertices from any n-vertex tree.*

Proof. Root T at an arbitrary vertex and let $\mathrm{size}(v)$ be the number of vertices in the subtree rooted at v. We prove by induction that if $\mathrm{size}(v) \leq k^i$, 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 $\mathrm{size}(v) \in (k^{i-1}, k^i]$ and define V' to be the set of u such that (i) $\mathrm{size}(u) \in (k^{i-1}, k^i]$ 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 $\mathrm{size}(u) > k^i$. By the inductive hypothesis, all descendants of v except V' have been removed after i rounds of compress/rake. The $(i+1)$th compress will remove any remaining vertices in $V' - \{v\}$ (the degree of the parent of v is unbounded, so v may not be removed), and if v still remains, the $(i+1)$th rake will remove it.

THEOREM 5.2. *There is an $O(\log_\Delta n)$-time* DetLOCAL *algorithm for Δ-edge coloring a tree, where $\Delta \geq 3$.*

Proof. Let β be the constant such that Linial's algorithm [28] finds a $\beta\Delta^2$-edge coloring in $O(\log^* n - \log^* \Delta + 1)$ time. We begin by decomposing T with compress/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 $T_{\log_k n}, \ldots, T_0 = T$ 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 $v_1, \ldots, v_x \in V_i^r$ be the vertices incident to u removed by a rake. At this point u is incident to at most $\Delta - x$ colored edges. It assigns to $\{u, v_1\}, \ldots, \{u, v_x\}$ distinct available colors from its palette. We now turn to the vertices removed by a compress operation. 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 incident to 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. 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 $V_0^c \cup \cdots \cup V_{\log_k n}^c$ consists of a set of paths. In $O(\log^* n)$ time, find an *initial* 3-edge coloring of these paths. We now color $T_{\log_k n}, \ldots, T_0$ 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}, recolor it any available color. When $k = 2$ this procedure takes $O(\log^* n + \log_k n) = O(\log_\Delta n)$ time.

The proof of the following two theorems are omitted due to space limitation. See [10] for the proofs.

THEOREM 5.3. *Any* oriented *tree T can be $(\Delta+1)$-edge colored in $O(\log^* n)$ time.*

THEOREM 5.4. *Any Δ-edge coloring algorithm for oriented trees takes $\Omega(\log_\Delta n)$ time in* RandLOCAL.

6 Network Decomposition of Trees

Recall that a (λ, γ)-network decomposition is a decomposition of the vertices of a graph into λ parts $V_1, \ldots, V_\lambda$ such that each connected component in the graph induced by each V_i has diameter at most γ. We generalize this notation as follows. A $(\lambda_1, \gamma_1, \lambda_2, \gamma_2)$-network decomposition is a decomposition of the vertices of a graph into $\lambda_1 + \lambda_2$ parts $V_1, \ldots, V_{\lambda_1}, U_1, \ldots, U_{\lambda_2}$ such that each connected component in the graph induced by each V_i has diameter at most γ_1 and each connected component in the graph induced by each U_i has diameter at most γ_2.

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 $\text{dist}(u, v) \leq d$. 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.

6.1 A $(2, O(k \log s + d))$-Network Decomposition

THEOREM 6.1. *Let T be a tree containing a distance-d dominating set of size s. There is a* DetLOCAL *algorithm $\mathcal{A}$ that computes a $(2, O(k \log s + d))$-network decomposition of T^k in $O(k \log s + d + k \log^* n)$ time.*

In what follows we prove Theorem 6.1. Consider the following two tree operations. They are similar to the ones described in [12], which are inspired by Miller and Reif [29]. The second operation is parameterized by an integer $\ell \geq 2$. In our application we set $\ell = 3k$.

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 $3d+1$ rakes; (2) repeat $\log s$ times: do 1 compress and then $\ell - 1$ rakes.

LEMMA 6.1. *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 $\text{size}(v)$ 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 $\text{size}(v) \leq 1$, then v is removed in Step (1) of $\mathcal{A}'$, and (ii) if $1 < \text{size}(v) \leq 2^i$, then v is removed on or before the ith iteration of Step (2) of $\mathcal{A}'$.

For the case $\text{size}(v) \leq 1$, the height of the subtree rooted at v is at most $3d$, and so the entire subtree (including v) must be removed after $3d+1$ rakes. For the case $2^{i-1} < \text{size}(v) \leq 2^i$, we assume by inductive hypothesis that all vertices u with $\text{size}(u) \leq 2^{i-1}$ 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 such that v is an endpoint, 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 $\ell - 1$ rakes.

Notice that a $(2, \cdot)$-network decomposition of T^k is simply a partition $V = V_1 \cup V_2$ such that for both $c = 1, 2$, any two vertices u and v in two separate components of V_c must have $\text{dist}_T(u, v) > k$. Recall that $\mathcal{A}'$ performs $L = (3d + 1) + \ell \log s$ rake/compress operations. We write U_i to denote the set of all vertices that are removed during the ith rake/compress operation. For any labeling $\bigcup_{j=i}^{L} U_j \to \{1, 2\}$, define the property $\mathcal{P}_i$ as follows.

- Each connected component induced by vertices of the same label have diameter at most $2t_r^i + (6k - 2)t_c^i$, where t_r^i (resp., t_c^i) is the total number of rakes (resp., compresses) after the ith tree operation.
- Let u and v be any two vertices in $\bigcup_{j=i}^{L} U_j$ with the same label $c \in \{1, 2\}$. If u and v belong to separate components induced by the vertices in $\bigcup_{j=i}^{L} U_j$ labeled c, then the distance between u and v in $\bigcup_{j=i}^{L} U_j$ is strictly more than k.

Notice that any labeling $V \to \{1, 2\}$ with the property $\mathcal{P}_1$ is a $(2, O(k \log s + d))$-network decomposition of T^k. This is because $2t_r^1 + (6k - 2)t_c^1 \leq 6kL = O(k \log s + d)$.

We are now in a position to present the algorithm $\mathcal{A}$. The algorithm $\mathcal{A}$ begins by computing $V = \bigcup_{i=1}^{L} U_i$ using $\mathcal{A}'$. Then, for $i = L$ down to 1, label all vertices $v \in U_i$ as follows.

- (Case: the ith operation is rake) Let $v \in U_i$. For the case that v is a leaf of $\bigcup_{j=i}^{L} U_j$, 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 a leaf of $\bigcup_{j=i}^{L} U_j$, 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: the ith operation is compress) Let P be a path that is a connected component of U_i. Notice that the number of vertices in P is at least $\ell = 3k$. 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 $[k, 3k)$, (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. If a $(2k + 1, 2k)$-ruling set of P is given, such a labeling can be computed in $O(k)$ time.

It is straightforward to prove by an induction that the above labeling for $\bigcup_{j=i}^{L} U_j$ has the property $\mathcal{P}_i$. The total running time of $\mathcal{A}$ is $O(k \log s + d + k \log^* n)$, since the ruling set computation of paths removed by compress operation can be computed in $O(k \log^* n)$ time in parallel.

6.2 A $(1, O(k \log_\lambda s + d), O(\lambda^2), 0)$-Network Decomposition

THEOREM 6.2. *Let T be a tree containing a distance-d dominating set of size s. There is a* DetLOCAL *algorithm $\mathcal{A}$ that computes a $(1, O(k \log_\lambda s + d), O(\lambda^2), 0)$-network decomposition of T^k in $O(k \log_\lambda s + d + k \log^* s)$ time, where $\lambda = \Omega(k)$ is sufficiently large (i.e., $\lambda \geq ck$ for some universal constant c).*

In what follows we prove Theorem 6.2. Consider the following two tree operations. We denote the underlying graph before the ith tree operation as T_i.

Rake: Remove all leaves and isolated vertices.

Compress: Remove every v such that $|N_{T_i}^{2.5k}(v)| \leq \lambda$.

Set $m = \lambda/(2.5k)$. Let $\mathcal{A}^*$ be the algorithm on the tree T defined as follows: (1) do $3d+1$ rakes; (2) repeat $\log_m s$ times: do 1 compress and then $2.5k$ rakes.

LEMMA 6.2. *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 $\text{size}(v)$ be the number of vertices in the subtree rooted at v that belong to S. We prove by induction that (i) if $\text{size}(v) \leq 1$, 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 $\text{size}(v) \leq 1$, the height of subtree rooted at v is at most $3d$, and so the entire subtree (including v) must be removed after $3d+1$ rakes. For the case $m^{i-1} < \text{size}(v) \leq m^i$, Assume the claim is true for $i-1$. Let v be any vertex with $\text{size}(v) \in (m^{i-1}, m^i]$ and define V' to be the set of u such that (i) $\text{size}(u) > m^{i-1}$ and (ii) u is in the subtree rooted at v. By the inductive hypothesis, all descendants of v except those in V' have been removed after $i - 1$ iterations of Step (2). Therefore, the set V' induces a subtree rooted at v having at most $m - 1$ leaves, which means the degree of each vertex in V' is at most m. For those vertices $u \in V'$ whose depth is at least $2.5k$ from v, $|N^{2.5k}(u)| \leq m(2.5k) \leq \lambda$, so they will be removed after one compress. Then the rest will also be removed during the next $2.5k$ rakes.

Now we present our algorithm $\mathcal{A}$ for network decomposition. First apply $\mathcal{A}^*$ to T, decomposing it with rakes and compresses. Next, 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. For any two vertices u, $v \in \mathcal{M}$, add an edge between them if $d(u,v) \le k$ in T. Denote the resulting graph by G.

The $(1, O(k \log_\lambda s + d), O(\lambda^2), 0)$ network decomposition of T^k is computed by assigning color 0 to all unmarked vertices, and color the remaining vertices in G with $\{1, \ldots, O(\lambda^2)\}$. We will show that (i) $\Delta(G) \le \lambda$, and so the coloring can be computed using Linial's algorithm [28] in $O(k \log^* s)$ time, (ii) each connected component induced by unmarked vertices (in T^k) has diameter $O(k \log_\lambda s + d)$.

- After deleting those vertices removed by compress from T, the diameter of each connected component (in T) is $O(k \log_\lambda s + d)$, since the total number of rakes is $O(k \log_\lambda s) + 3d + 1$. We show that the diameter of each connected component of unmarked vertices in T^k is still $O(k \log_\lambda s + d)$. 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 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, any 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.

- For any marked vertex v, we claim that $|N^k(v) \cap \mathcal{M}| \le \lambda$ (in T), and so $\Delta(G) \le \lambda$. 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 iteration i. Then we have $|N^{2.5k}_{T_i}(w)| \le \lambda$. Notice that $N^k(v) \cap \mathcal{M} \subseteq N^k_{T_i}(v) \subseteq N^{2.5k}_{T_i}(w)$, since $d(v,w) \le d(v,u) + d(u,w) \le 1.5k$, and since $N^k_{T_i}(v)$ contains all possible marked vertices within distance-k of v.

7 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 recoloring subgraphs to extend partial colorings needs $\Omega(\frac{\Delta}{c} \log n)$ rounds.

THEOREM 7.1. *Let Δ be the maximum degree and $c \in [1, \frac{\Delta}{3}]$. For any n, there exist an n-vertex graph $G = (V, E)$ and a partial edge coloring $\phi : E \to \{1, \ldots, \Delta + c, \perp\}$ with the following property. For any coloring $\phi' : E \to \{1, \ldots, \Delta + c, \perp\}$ that colors a strict superset of the edges colored by ϕ, ϕ and ϕ' differ on a subgraph of diameter $\Omega(\frac{\Delta}{c} \log n)$.*

As a special case, suppose that G is a partially $(\Delta + c)$-edge colored graph, with exactly one uncolored edge e_0. A natural approach to color G 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 approach leads to a distributed algorithm for Brooks' theorem [33]. However, Theorem 7.1 implies the existence of a graph where any augmenting path has length $\Omega(\frac{\Delta}{c} \log n)$, and so any "augmenting paths"-based algorithm for constructive Vizing's theorem must take at least $\Omega(\frac{\Delta}{c} \log n)$ time, which is inefficient for large Δ.

Construction. The construction of the partially edge-colored graph for Theorem 7.1 is as follows. Without loss of generality, assume that $\Delta + c$ is even, and let $k = \frac{\Delta + c}{2}$. Divide the color palette $\{1, \cdots, \Delta + c\}$ into two sets $S_1 = \{1, \cdots, k\}$ and $S_2 = \{k + 1, \cdots, \Delta + c\}$. Let $k' = \Delta - k$. Let $e_0 = \{u_0, v_0\}$ be the uncolored edge. We construct a graph consists of ℓ layers of vertices, which is denoted as $G^*(\ell, \Delta, c)$.

Layer 0 consists of only u_0 and v_0. The vertex u_0 has k neighbors other than v_0, which form layer 1. The edges between u_0 and these k vertices are colored by $S_1 = \{1, \cdots, k\}$. Suppose that layers $0 \ldots i$ have been constructed. Layer $i + 1$ is constructed as follows.

Let n_i be the current number of vertices in layer i. We divide these n_i vertices into $\lfloor \frac{n_i}{k'} \rfloor$ groups of size k', and the remaining $n_i \mod k'$ vertices are *ungrouped*. Each group forms a complete bipartite graph $K_{k',k}$ with k corresponding *new* vertices in layer $i + 1$. In total, $\lfloor \frac{n_i}{k'} \rfloor k$ new vertices are added to layer $i + 1$. Notice that a complete bipartite graph $K_{k',k}$ can be properly k-edge colored. If i is even, we use the palette S_1 to color these complete bipartite graphs; otherwise we use S_2. The $n_i \mod k'$ ungrouped vertices in layer i are *promoted* to layer $i + 2$. When we assigning groups to the vertices in layer $i + 2$, we prioritize these vertices to ensure that each vertex is promoted at most twice. We have the formula:

$$n_{i+1} = \left\lfloor \left(\lfloor \frac{n_i}{k'} \rfloor k + (n_{i-1} \mod k') \right) / k' \right\rfloor \cdot k', \ i \ge 2.$$

The construction of v_0's side is similar. The only difference is that we switch the roles of S_1 and S_2 for

coloring complete bipartite graphs. We call the partial edge coloring ϕ. See Figure 1 for a concrete example which shows the construction of layers $0, \ldots, 5$. Notice that some vertices in layer 4 have been promoted to layer 6 (during the construction of layer 5).

Notice that the ratio of the number of vertices in two consecutive layers is $\Theta(\log_{k/k'} n) = \Theta(\frac{\Delta}{c} \log n)$, and so we can set $\ell = \Theta(\frac{\Delta}{c} \log n)$. The distance from the uncolored edge to any vertex in layer ℓ is at least $\frac{\ell}{3} = \Theta(\frac{\Delta}{c} \log n)$, since any edge can skip at most two layers. To prove Theorem 7.1, it suffices to prove the following lemma.

LEMMA 7.1. *It is impossible to extend the partial edge coloring ϕ of $G^*(\ell, \Delta, c)$ to a total edge coloring by only recoloring edges within layer 0 to layer $\ell - 1$.*

Proof. Without loss of generality, we only consider u_0's side. Let us refer to the edges connecting layer i and higher layers as the *edges of layer i*. For an odd (resp., even) ℓ, fix the colors of edges of layer ℓ using only the color palette S_1 (resp., S_2). Consider the case ℓ is odd (the other case is similar). Observe that each grouped vertex in layer $\ell-1$ has k neighbors in layer ℓ. Therefore, the edges of layer $\ell-1$ must have their colors picked from S_2. By an induction, edges of layer 1 must be colored using S_1. Similarly, edges of layer 1 at v_0's side must be colored using S_2, and so no available color is left for $e_0 = \{u_0, v_0\}$. Thus, even if all edges within layer 0 to layer $\ell-1$ are allowed to be recolored, we cannot obtain a proper edge coloring of $G^*(\ell, \Delta, c)$.

8 Conclusion

Consider this strange phenomenon. The black-box transformations of [11] and [12] imply the existence of efficient algorithms that are *not entirely constructive*, i.e., they do not have short, coherent descriptions. For example, Fischer and Ghaffari's randomized $2^{O(\sqrt{\log \log n})}$ LLL algorithm (for $d < (\log \log n)^{1/5}$) implies [11, Theorem 3] that the deterministic complexity of LLL (for $d < (\log n)^{1/5}$) is $2^{O(\sqrt{\log n})}$. One could then apply the transformation of [12, Theorem 5] to obtain a deterministic $O(\log n)$-time LLL algorithm for tree-structured dependency graphs (for $d = O(1)$). Algorithms constructed in this way are neither useful nor enlightening, but *they exist* and suggest that humanly comprehensible algorithms for these problems also exist. (See, e.g., our $O(\log n)$-time LLL algorithm for trees in Section 4 and Section 6.)

Our randomized $(1+\epsilon)\Delta$-edge coloring algorithm, together with [18, 11], implies the existence of a deterministic algorithm for $\Delta < (\log n)^{\alpha}$, $\alpha > 0$, running in $2^{O(\sqrt{\log n})}$ time. This suggests the existence of a simple, direct algorithm using network decompositions [32], but to date, network decompositions have only been used for problems that admit "greedy" algorithms, e.g., MIS or $(\Delta+1)$-vertex coloring, but *not* more difficult problems like k-edge coloring, $k < 2\Delta - 1$. Finding simple, explicit, and deterministic algorithms for $(1+\epsilon)\Delta$-coloring is a challenging open problem.

We have proved that the LLL on tree-structured dependency graphs can be solved in $O(\log \log n)$ time w.h.p. (or faster), confirming [12, Conjecture 1] for this case. It is unclear if there is any hope of extending this type of algorithm to general LLL instances, absent a breakthrough in network decomposition technology [18, 33].

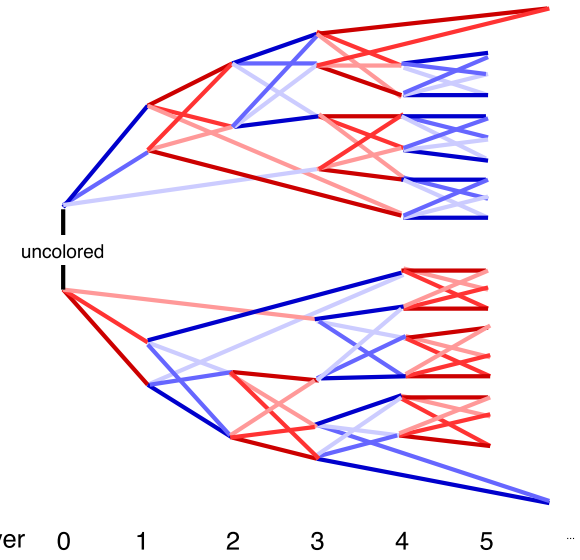

Figure 1: An example of the construction, where $\Delta = 5$, $c = 1$, $k = 3$ and $k' = 2$. Edges with palette $S_1 = \{1, 2, 3\}$ are colored in red, and edges with palette $S_2 = \{4, 5, 6\}$ are colored in blue.

References

[1] E. Arjomandi. An efficient algorithm for colouring the edges of a graph with $\Delta+1$ colours. *INFOR: Information Systems and Operational Research*, 20(2):82–101, 1982.

[2] 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)*, pages 364–369, 1989.

[3] 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 Princi-*

ples of Distributed Computing (PODC), pages 345–354, 2015.

[4] L. Barenboim and M. Elkin. Deterministic distributed vertex coloring in polylogarithmic time. *J. ACM*, 58(5):23, 2011.

[5] L. Barenboim, M. Elkin, and F. Kuhn. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. *SIAM J. Comput.*, 43(1):72–95, 2014.

[6] 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)*, pages 175–184, 2017.

[7] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. *J. ACM*, 63(3):20:1–20:45, 2016.

[8] B. Bollobás. *Extremal graph theory*, volume 11 of *London Mathematical Society Monographs*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1978.

[9] 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)*, pages 479–488, 2016.

[10] Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto. The complexity of distributed edge coloring with small palettes. *CoRR*, abs/1708.04290, 2017.

[11] Y.-J. Chang, T. Kopelowitz, and S. Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. In *Proceedings 57th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 615–624, 2016.

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

[13] K.-M. Chung, S. Pettie, and H.-H. Su. Distributed algorithms for the Lovász local lemma and graph coloring. *Distributed Computing*, 30:261–280, 2017.

[14] A. Czygrinow, M. Hanckowiak, and M. Karonski. Distributed $O(\Delta \log n)$-edge-coloring algorithm. In *Proc. ESA 2001*, pages 345–355, 2001.

[15] X. Dahan. Regular graphs of large girth and arbitrary degree. *Combinatorica*, 34(4):407–426, 2014.

[16] D. P. Dubhashi, D. A. Grable, and A. Panconesi. Near-optimal, distributed edge colouring via the nibble method. *Theor. Comput. Sci.*, 203(2):225–251, 1998.

[17] 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)*, pages 355–370, 2015.

[18] 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)*, pages 18:1–18:16, 2017.

[19] 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)*, pages 180–191, 2017.

[20] P. Fraigniaud, M. Heinrich, and A. Kosowski. Local conflict coloring. In *Proceedings 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 625–634, 2016.

[21] H. N. Gabow, T. Nishizeki, O. Kariv, D. Leven, and O. Terada. Algorithms for edge-coloring graphs. Technical Report TRECIS-8501, Tohoku University, 1985.

[22] M. Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 270–277, 2016.

[23] 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)*, pages 19:1–19:15, 2017.

[24] M. Ghaffari and H.-H. Su. Distributed degree splitting, edge coloring, and orientations. In *Proceedings 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2505–2523, 2017.

[25] I. Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981.

[26] H. J. Karloff and D. B. Shmoys. Efficient parallel algorithms for edge coloring problems. *J. Algorithms*, 8(1):39–52, 1987.

[27] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. In *Proceedings 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 7–15, 2006.

[28] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[29] G. L. Miller and J. H. Reif. Parallel tree contraction–Part I: fundamentals. *Advances in Computing Research*, 5:47–72, 1989.

[30] R. A. Moser and G. Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2), 2010.

[31] M. Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM J. Discrete Mathematics*, 4(3):409–412, 1991.

[32] A. Panconesi and A. Srinivasan. The local nature of Δ-coloring and its algorithmic applications. *Combinatorica*, 15(2):255–280, 1995.

[33] A. Panconesi and A. Srinivasan. On the complexity of distributed network decomposition. *J. Algor.*, 20(2):356–374, 1996.

[34] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

[35] S. Pettie and H.-H. Su. Distributed algorithms for coloring triangle-free graphs. *Information and Computation*, 243:263–280, 2015.

[36] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz No.*, 3:25–30, 1964.