Distributed Edge Coloring and a Special Case of the Constructive Lovász Local Lemma

YI-JUN CHANG, University of Michigan, USA
QIZHENG HE and WENZHENG LI, IIIS, Tsinghua University, China
SETH PETTIE, University of Michigan, USA
JARA UITTO, ETH Zürich, Switzerland and University of Freiburg, Germany

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

- **Lower Bounds:** First, we simplify the *round elimination* technique of Brandt et al. [16] and prove that $(2\Delta-2)$-edge coloring requires $\Omega(\log_\Delta \log n)$ time with high probability and $\Omega(\log_\Delta n)$ time deterministically, *even on trees*. Second, we show that a natural approach to computing $(\Delta+1)$-edge colorings (Vizing's theorem), namely, extending an arbitrary partial coloring by iteratively recoloring subgraphs, requires $\Omega(\Delta \log n)$ time.
- **Upper Bounds on General Graphs:** 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 our $(1+\epsilon)\Delta$-edge coloring algorithm is usually dominated by $O(\log \epsilon^{-1})$ calls to a distributed Lovász local lemma (LLL) algorithm. For example, using the Chung-Pettie-Su LLL algorithm, we compute a $(1+\epsilon)\Delta$-edge coloring in $O(\log n)$ time when $\epsilon \geq (\log^3 \Delta)/\sqrt{\Delta}$, or $O(\log_\Delta n) + (\log\log n)^{3+o(1)}$ time when $\epsilon = \Omega(1)$. When Δ is sublogarithmic in n the performance is improved with the Ghaffari-Harris-Kuhn LLL algorithm.
- **Upper Bounds on Trees:** We show that the $\Omega(\log_\Delta \log n)$ lower bound can be nearly matched on trees. To establish this result, we develop a new distributed Lovász local lemma algorithm for *tree-structured dependency graphs*, which arise naturally from $O(1)$-round probabilistic algorithms run on trees. Specifically, our $(1+\epsilon)\Delta$-edge coloring algorithm for trees takes $O(\log(1/\epsilon)) \cdot \max\{\frac{\log\log n}{\log\log\log n}, \log_{\log\Delta} \log n\}$ time when $\epsilon \geq (\log^3 \Delta)/\sqrt{\Delta}$, or $O(\max\{\frac{\log\log n}{\log\log\log n}, \log_\Delta \log n\})$ time when $\epsilon = \Omega(1)$.

CCS Concepts: • **Theory of computation → Distributed algorithms**;

Additional Key Words and Phrases: Distributed algorithms, edge coloring, LOCAL model, Lovász local lemma

A preliminary version of this article was presented at the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7–10, 2018. Supported by NSF Grants No. CCF-1514383, No. CCF-1637546, and No. CCF-1815316 and ERC Grant No. 336495 (ACDC).

Authors' addresses: Y.-J. Chang and S. Pettie, Department of EECS, University of Michigan, 2260 Hayward St., Ann Arbor, MI 48109, USA; emails: cyijun@umich.edu, pettie@umich.edu; Q. He and W. Li, Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, FIT Building 1-208, Tsinghua University, Beijing, China 100084; emails: {hqz14, wzli14}@mails.tsinghua.edu.cn; J. Uitto, ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland and University of Freiburg, Georges Köhler Allee, Gebäude 106, 79110 Freiburg im Breisgau, Germany; email: jara.uitto@inf.ethz.ch.

1549-6325/2019/11-ART8 $15.00
https://doi.org/10.1145/3365004

ACM Reference format:
Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. 2019. Distributed Edge Coloring and a Special Case of the Constructive Lovász Local Lemma. *ACM Trans. Algorithms* 16, 1, Article 8 (November 2019), 51 pages.
https://doi.org/10.1145/3365004

1 INTRODUCTION

In this article, we consider the complexity of the edge coloring problem in the well-known LOCAL model of distributed computation [48, 59]. A *k-edge coloring* of a graph $G = (V, E)$ is a function $\phi : E \to \{1, \ldots, k\}$ such that edges sharing an endpoint are colored differently; the parameter k is called the *palette size*. 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.

The LOCAL *Model.* In the LOCAL model, the input graph $G = (V, E)$ is identical to the underlying distributed network; vertices are identified with processors and edges with bi-directional communication links; *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 satisfying some constraints.

For clarity, we bifurcate the LOCAL model into RandLOCAL and DetLOCAL depending on whether random bits are available. In the RandLOCAL model the output labeling is correct with high probability (w.h.p.) (i.e., $1 - 1/\text{poly}(n)$). In the DetLOCAL model each vertex is assigned a unique $O(\log n)$-bit ID; the output labeling must always be correct.

We assume each $v \in V$ initially knows $\deg(v)$, a port-numbering of its incident edges, and global parameters such as $n = |V|$ and $\Delta = \max_{v \in V} \deg(v)$, or upper bounds on them if the exact parameters are not common knowledge. The assumption that global parameters are common knowledge can sometimes be removed; see Korman et al. [46].

Distributed Coloring. The two primary design objectives for distributed coloring algorithms are (i) minimizing the number of colors (palette size) and (ii) minimizing the number of rounds (time complexity). A modest standard for *efficient algorithm* in the LOCAL model is $O(\text{poly} \log n)$ time. However, there are now many examples of locally checkable labeling problems with $O(\text{poly}(\log \log n))$ randomized complexity [19, 35, 36, 40, 42, 60], and in some circumstances, $O(\log^* n)$ complexity [20, 29, 48, 61].

For the case of *vertex coloring*, it is well-known that a $(\Delta + 1)$-vertex coloring can be found in $O(\log n)$ time [1, 49] in RandLOCAL or $2^{O(\sqrt{\log n})}$ time in DetLOCAL [57]. The randomized complexity was recently improved to $O(\sqrt{\log \Delta}) + 2^{O(\sqrt{\log \log n})}$ [43], and then to $2^{O(\sqrt{\log \log n})}$ [20].

These randomized upper bounds imply that a vertex coloring with palette size $\Delta + 1$ can be computed efficiently. In general, the palette size of $\Delta + 1$ cannot be further reduced, since there exists a graph (a complete graph with $\Delta + 1$ vertices) that cannot be Δ-colored.

Edge Coloring. The case of *edge coloring* is more complicated. 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 $\hat{\Delta} = 2\Delta - 2$. Therefore, an edge coloring with palette size $\hat{\Delta} + 1 = 2\Delta - 1$ can be computed efficiently. The current state-of-the-art for $(2\Delta - 1)$-edge coloring is $\tilde{O}(\log^3 \log n)$ time[1] for all Δ [42], and $O(\log^* n)$ time when

[1]Here, $\tilde{O}(f(n)) = O(f(n) \cdot \text{poly}(\log f(n)))$.

$\Delta > \log^{1+o(1)} n$ [29]. Vizing's theorem [63] guarantees the existence of a $(\Delta+1)$-edge coloring for all graphs; but it is unknown whether such a coloring can be efficiently computed in LOCAL.

The number "$2\Delta-1$" is the *smallest* palette size with the property that any partial edge 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 et al. [26] 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 et al. [29] 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). The $(1+\epsilon)\Delta$-edge coloring problem is solved in $O(\log^* n) \cdot \lceil \frac{\log n}{\Delta^{1-o(1)}} \rceil$ time. The running time of the Dubhashi-Grable-Panconesi and Elkin-Pettie-Su algorithms depend *polynomially* on ϵ^{-1}. In both algorithms it is clear that ϵ need not be constant, but it is not self-evident how small it can be made *as a function of* Δ.

The $\lceil \frac{\log n}{\Delta^{1-o(1)}} \rceil$-factor in the time complexity of Reference [29] is due to the Chung-Pettie-Su LLL algorithm [22], which holds for all Δ. The Ghaffari-Harris-Kuhn [35] and Fischer-Ghaffari [30] LLL algorithms are faster when $\Delta = (\log n)^{o(1)}$; see Section 1.4 and Table 2.

New Results. In this article, we present new upper and lower bounds on the complexity of edge coloring in the regimes between palette size $\Delta+1$ and $2\Delta-2$, i.e., strictly below the "greedy" threshold $2\Delta-1$.

From the lower bound side, we 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*. This result is attained via the *round elimination* technique of Brandt et al. [16]. Second, we consider a natural approach to computing $(\Delta+1)$-edge colorings (Vizing's theorem) via extending partial colorings by iteratively recoloring parts of the graph via "alternating paths." We prove that this approach may be viable, but in the worst case requires recoloring subgraphs of diameter $\Omega(\Delta \log n)$. This stands in contrast to distributed algorithms for Brooks' theorem [56], which exploit the existence of $O(\log_\Delta n)$-length alternating paths.

From the upper bound side, we give an efficient 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. Notice that with a palette of size $\Delta + \Theta(\sqrt{\Delta})$, we have a constant probability of being able to color an arbitrary edge e, given a *random* feasible coloring of its neighborhood. Edge coloring with this palette size was achieved in 1987 by Karloff and Shmoys [45] in the context of parallel (PRAM) algorithms, but has not been achieved in the LOCAL model before. We also show that the $\Omega(\log_\Delta \log n)$ lower bound can be nearly matched on trees by developing a new distributed LLL algorithm for *tree-structured dependency graphs*.

1.1 Tools

Randomized distributed algorithms in the LOCAL model are often composed of iterations of $O(1)$-round routines that commit to a partial labeling [13, 26, 29, 60]. A vertex may proceed to the next iteration only if it satisfies some property or invariant, which typically holds with probability $1 - 1/\text{poly}(\Delta)$.

Graph Shattering. In the graph shattering framework [13, 14] of algorithm design, the *bad vertices* that violate the require property are temporarily *removed from consideration* in the subsequent iterations of the randomized algorithm. If it can be shown that at the end of the randomized

algorithm the connected components induced by the bad vertices have size at most $O(\text{poly}(\log n))$, then one can revert to the best available *deterministic* algorithm and solve the problem on each component of the "shattered" graph in parallel. The randomized part is called the *pre-shattering* phase; the deterministic part is called the *post-shattering* phase.

In some applications, we cannot tolerate the existence of a bad vertex. For example, when the palette size is below the greedy threshold $2\Delta - 1$, not every partial edge coloring can be extended to a total edge coloring. In this case, we need to resort to a distributed Lovász local lemma (LLL) algorithm, which can guarantee a global success (i.e., there is no bad vertex) with probability $1 - 1/\text{poly}(n)$ (using a randomized LLL algorithm) or even 1 (using a deterministic LLL algorithm).[2]

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 $\text{vbl}(A) \subset \mathcal{V}$. Define the dependency graph as $G_{\mathcal{E}} = (\mathcal{E}, \{(A, B) \mid \text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset)\})$. Symmetric versions of the Lovász local lemma are stated in terms of d, the maximum degree in $G_{\mathcal{E}}$, and $p = \max_{A \in \mathcal{E}} \Pr[A]$. A standard version of the LLL says that if $ep(d + 1) < 1$ then $\Pr[\cap_{A \in \mathcal{E}} \overline{A}] > 0$, i.e., it is *possible* to avoid all bad events. The constructive LLL problem is to assign values to all variables in $\mathcal{V}$ such that no event in $\mathcal{E}$ happens.

Distributed Lovász Local Lemma. In the *distributed* LLL problem the communications network is identical to $G_{\mathcal{E}}$. Every node A is identified with an event, which is aware of the distribution on the random variables $\text{vbl}(A) \subseteq \mathcal{V}$. The goal is to collectively assign values to all variables in $\mathcal{V}$ such that no event in $\mathcal{E}$ happens.

In distributed coloring algorithms it is typical to see $d = \text{poly}(\Delta)$ and $p = \exp(-d^{\Omega(1)})$, i.e., *any* polynomial LLL criterion of the form $p(ed)^c < 1$ where $c = O(1)$ is good enough. Chung, Pettie, and Su [22] provided an $O(\log_{1/epd^2} n)$ time algorithm under the LLL criterion $epd^2 < 1$. This remains the fastest distributed LLL algorithm under a polynomial criterion when d is arbitrary. There are faster LLL algorithms [30, 35] when d is small, and slower LLL algorithms [22, 34] under the stricter criterion $ep(d + 1) < 1$; see Section 1.4 and Table 2.

1.2 New Lower Bounds

Round Elimination. In Section 2, we show a lower bound on $(2\Delta - 2)$-edge coloring that follows the same lines as Brandt et al.'s [16] lower bound on Δ-vertex coloring. Both proofs establish hardness for a coloring problem by reduction from *sinkless orientation*,[3] but one subtlety here is that we are dealing with *two irreconcilable versions* of sinkless orientation. Brandt et al. [16] prove that sinkless orientation on a graph *that comes equipped with a Δ-edge coloring* is reducible to Δ-vertex coloring on the same graph. Hence, any lower bound on sinkless orientation (that is aware of the edge coloring) extends to Δ-vertex coloring. We show that sinkless orientation on a bipartite graph that comes equipped with (i) a 2-vertex coloring, and (ii) a $(2\Delta - 1)$-edge coloring, is reducible to $(2\Delta - 2)$-edge coloring on the same graph. We then prove that this version of sinkless orientation takes $\Omega(\log_\Delta \log n)$ time in RandLOCAL and $\Omega(\log_\Delta n)$ time in DetLOCAL, matching [16, 19].[4]

Roughly speaking, the idea of Brandt et al. [16] is to convert any randomized t-round algorithm with local error probability p into a $(t - 1)$-round algorithm with error probability $\approx p^{1/\Delta}$.

[2]However, applying an LLL algorithm does not mean we have circumvented the graph shattering method! All known distributed LLL algorithms with a sublogarithmic dependence on n ([30, 35] and Section 5) use graph shattering internally. One interpretation of Chang, Kopelowitz, and Pettie's derandomization [19, Theorem 3.1] is that graph shattering is intrinsic to fast randomized algorithms in the LOCAL model and cannot be completely avoided.

[3]Orient the edges of the (undirected) input graph so that no vertex is a sink.

[4]It is impossible to reconcile these two versions of sinkless orientation. The problem can be solved *without communication*, given a 2-vertex coloring and a k-edge coloring for any $k \in [\Delta, 2\Delta - 2]$.

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.

Our proof uses a simplified round elimination technique that *appears* to give quantitatively worse bounds but that can be *automatically* strengthened to match those of Reference [16]. 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 $(2\Delta - 1)$-edge colored. An algorithm has irregular time profile $t = (t_1, \ldots, t_{2\Delta-1})$ if edges with input color i choose their output color by examining only their t_i-neighborhood. In our round-elimination method, we show that any algorithm with time profile $(\underbrace{t, t, \ldots, t}_{i}, \underbrace{t-1, \ldots, t-1}_{(2\Delta-1)-i})$ and error probability p can be transformed into one with time profile $(\underbrace{t, t, \ldots, t}_{i-1}, \underbrace{t-1, \ldots, t-1}_{(2\Delta-1)-i+1})$ 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 Reference [16]. By following the proofs of Chang, Kopelowitz, and Pettie [19], any randomized lower bound of this type implies $\Omega(\log_\Delta n)$ lower bounds in DetLOCAL [19, Theorem 5], which then implies $\Omega(\log_\Delta \log n)$ lower bounds in RandLOCAL [19, Theorem 3].

Lower Bound for 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, then we prove in Section 3 that it takes $\Omega(\Delta \log n)$ time in the LOCAL model, with or without randomization. More generally, any $(\Delta + c)$-coloring that is based on recoloring subgraphs takes $\Omega(\frac{\Delta}{c} \log n)$ time. This establishes a quantitative difference between the "locality" of Vizing's theorem and Brooks' theorem [56].

Subsequent Work. Subsequent to the initial publication of Reference [18], Ghaffair, Kuhn, Maus, Uitto [39] showed that a $\Delta + O(\log n \cdot \log(2 + \Delta/\log n))$-edge coloring can be computed in DetLOCAL in $O(\text{poly}(\log n, \Delta))$ rounds. Very recently, Su and Vu [62] improved this bound and showed that in $O(\text{poly}(\log n, \Delta))$ rounds, it is possible to compute a $\Delta + O(\log_\Delta n)$-edge coloring in DetLOCAL or a $(\Delta + 2)$-edge coloring in RandLOCAL, which is only one color away from Vizing's theorem. All these upper bounds have time complexity of the form $O(\text{poly}(\log n, \Delta))$. It is still an intriguing open question as to whether an edge coloring with palette size significantly smaller than $\Delta + \tilde{O}(\sqrt{\Delta})$ can be computed in $O(\text{poly} \log n)$ time, regardless of Δ.

1.3 New Upper Bounds

Upper Bounds on General Graphs. The $(1 + \epsilon)\Delta$-edge coloring algorithms of References [26, 29] are slow (with a polynomial dependence on ϵ^{-1}) and have limits on how small ϵ can be, as a function of Δ. In Section 4, we prove that the most "natural" randomized algorithm (One-Shot-Coloring) 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 $\epsilon = \tilde{\Omega}(1/\sqrt{\Delta})$, we show that $(1 + \epsilon)\Delta$-edge coloring is reducible to $O(\log \epsilon^{-1})$ instances of the Lovász local lemma with local failure probability $p = \exp(-\epsilon^2 \Delta^{1-o(1)})$, plus one instance of $O(\Delta)$-edge coloring, which can be solved quickly using [13, 29, 37]. When $\epsilon^2 \Delta \gg \log n$ the local failure probability is already $1/\text{poly}(n)$; otherwise, we can invoke a distributed LLL algorithm [22, 30, 35, 53].

The running time of our algorithm varies for different choices of Δ and palette size. It can be shown that our algorithm computes a $(1+\epsilon)\Delta$-edge coloring in at most $O(\log n)$ time when $\epsilon \geq (\log^3 \Delta)/\sqrt{\Delta}$, or at most $O(\log_\Delta n) + (\log\log n)^{3+o(1)}$ time when $\epsilon = \Omega(1)$. These times reflect the use of Chung, Pettie, and Su's LLL algorithm [22]. Applying one of the Ghaffari-Harris-Kuhn LLL algorithms [35] leads to a $(1+\epsilon)\Delta$-edge coloring algorithm running in $O(\log \epsilon^{-1} \cdot \Delta^6 + 2^{O(\sqrt{\log\log n})})$ time.[5] Our $(\Delta + \tilde{O}(\sqrt{\Delta}))$-edge coloring algorithm is simple, but tricky to analyze, and requires a general distributed LLL algorithm to be made efficient. Resolving the complexity of the distributed LLL problem is a major open problem [21] but one that is unlikely to be completely settled any time soon, given its connection to computing general network decompositions [30, 38].

Upper Bounds on Trees. There is still a significant gap between the upper bound of our $(1+\epsilon)\Delta$-edge coloring RandLOCAL algorithm on general graphs in Section 4 and our $\Omega(\log_\Delta \log n)$ RandLOCAL lower bound in Section 2, which applies even to trees. We prove that this lower bound can be matched when the underlying network is a tree, at least when $\epsilon = \Omega(1)$ and $\Delta < \text{poly}(\log\log n)$. In particular, our $(1+\epsilon)\Delta$-edge coloring algorithm for trees takes $O(\log(1/\epsilon)) \cdot \max\{\frac{\log\log n}{\log\log\log n}, \log_{\log\Delta} \log n\}$ time when $\epsilon \geq (\log^3 \Delta)/\sqrt{\Delta}$, or $O(\max\{\frac{\log\log n}{\log\log\log n}, \log_\Delta \log n\})$ time when $\epsilon = \Omega(1)$.

This improvement is achieved by developing a new distributed LLL algorithm for *tree structured* dependency graphs, which appears in Section 5. Specifically, if $T = (V, E)$ is a tree and $r = O(1)$, then we say that $T^r = (V, \{(u, v) \mid \text{dist}_T(u, v) \leq r\})$ is *tree-structured*. This type of dependency graph arises naturally when we run $O(1)$-round probabilistic algorithms on trees.

Our new LLL algorithm is based on the graph shattering framework. We first apply a randomized algorithm that fixes the output of most of the vertices such that each connected component of the remaining part of the graph is small. We then apply a new deterministic LLL algorithm for tree-structured instances to each component in parallel.

Fischer and Ghaffari [30] showed that one can obtain a DetLOCAL LLL algorithm using a *network decomposition* algorithm as a black box. Based on this idea, we give a deterministic $O(\max\{\log_\lambda n, \log n/\log\log n\})$-time LLL algorithm for tree-structured instances under criterion $p(ed)^\lambda < 1, \lambda \geq 2$. The algorithm is based on two new network decomposition algorithms for tree-structured graphs, presented in Section 6.

For the randomized part of the graph shattering routine, the goal is to design an algorithm to compute a good partial assignment ϕ such that the connected components induced by the unassigned part of the dependency graph are small. We give an algorithm for tree-structured instances that achieves this goal in time $O(\log_\lambda \log n)$, improving the $O(d^2 + \log^* n)$-time shattering routine of Reference [30] when d is not too small. At a high level, our approach is to consider the following process. First, draw a total assignment ϕ to $\mathcal{V}$ according to the distribution of the variables. Whenever the probability that a bad event $E(v)$ occurs under the current partial assignment ϕ is higher than a certain threshold, 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. Vertices that have unset their variables are said to have been *infected*, and infected vertices can cause nearby neighbors to become infected. If this contagion process were actually simulated, then it would take $\Omega(\log n)$ parallel steps to reach a stable state, which is too slow. We develop a different method to achieve a stable state that is exponentially faster, by avoiding a direct simulation.

By composing these results, we obtain a randomized $O(\max\{\log_\lambda \log n, \log\log n/\log\log\log n\})$ LLL algorithm for tree-structured instances, when λ is at least a sufficiently large constant

[5] The running time of Reference [35] is (at least) quadratic in the degree d of the dependency graph, and in our case $d = \Theta(\Delta^3)$.

depending on r. Our upper bound essentially matches the RandLOCAL lower bound of Brandt et al. [16], which is of the form $\Omega(\log_{\log p^{-1}} \log n)$ under the LLL criterion $p \cdot f(d) \leq 1$ for any $f(d) \leq 2^d$.

A major open problem is to extend this *contagion dynamic* idea to general dependency graphs, and show that they, too, can be shattered in $O(\log \log n)$ time. In light of References [16, 19], this is a necessary first step towards proving Conjecture 1 from Chang and Pettie [21], namely, that the RandLOCAL complexity of the LLL under a polynomial criterion is $O(\log \log n)$.

Additional Results on Trees. In Section 7, we prove some additional results on the complexity of edge coloring trees. We design an $O(\log_\Delta n)$-time DetLOCAL algorithm for Δ-edge coloring a tree T with maximum degree $\Delta \geq 3$. A tree is said to be *oriented* if the tree is rooted and each vertex that is not the root knows its parent. We show that a $(\Delta + 1)$-edge coloring of an oriented tree can be found in $O(\log^* n)$ time, but Δ-edge coloring takes $\Omega(\log_\Delta n)$ time.

Remark. After the initial publication of this work in Reference [18], we learned that Molloy and Reed [51] also obtained a similar bound of $\Delta + O(\sqrt{\Delta} \log^4 \Delta)$ on the palette size for edge coloring. Their algorithm was more general in that it extends to k-uniform hypergraphs (with palette size $\Delta + O(\Delta^{1-1/k} \log^4 \Delta)$) and applies to *list* edge coloring. The main difference between our work and theirs [51] is the analysis. We use a concentration bound [27, Equation (8.5)] that takes into account the variance of each variable. The analysis of Reference [51] is based on Talagrand's concentration inequality. Our result is slightly better in terms of the polylog-factor, and it also improves the existential bound on the palette size for list edge coloring. Specifically, if each edge is given a list of $(1 + \epsilon)\Delta$ with $\epsilon = \omega((\log^{2.5} \Delta)/\sqrt{\Delta})$ colors, then the graph admits a proper list edge coloring.

1.4 Related Work

In this section, we walk though the rich history of distributed edge coloring and the distributed LLL.

We begin with reviewing previous edge coloring algorithms; see Table 1 for a summary. Edge coloring can be interpreted as a *vertex* coloring problem on the *line graph* $L(G)$, which has has maximum degree $\hat{\Delta} = 2\Delta - 2$. Applied to $L(G)$, Linial's [48] 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 [32, 57], $(2\Delta - 1)$-edge coloring is solved in $\min\{2^{O(\sqrt{\log n})}, \tilde{O}(\sqrt{\Delta}) + O(\log^* n)\}$ time. Barenboim et al. [12] gave deterministic algorithms for $(2^k\Delta)$-edge coloring $(k \geq 2)$ in $\tilde{O}(k\Delta^{1/2k}) + O(\log^* n)$ time.

Barenboim et al. [13] proved that $O(\log \Delta)$ iterations of the natural randomized $(2\Delta - 1)$-edge coloring algorithm effectively *shatters* the graph into uncolored components of $n' = \text{poly}(\log n)$ vertices; then we can employ a deterministic list coloring algorithm to color these components in $2^{O(\sqrt{\log n'})} = 2^{O(\sqrt{\log \log n})}$ time [57]. Thus, the total time complexity is $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$.

Elkin et al. [29] proved that when $\Delta > (\log n)^{1+\gamma}$ (for some constant γ), $(2\Delta - 1)$-edge coloring can be solved in $O(\log^* n)$ time in RandLOCAL. Recently, Fischer et al. [31] proved that $(2\Delta - 1)$-edge coloring can be solved in $O(\log^7 \Delta \log n)$ time in DetLOCAL. This bound was later improved to $O(\log^4 \Delta \log^2 n)$ by Ghaffari [35], and then to $\tilde{O}(\log^2 \Delta \log n)$ by Harris [42]. Together with References [13, 29], this implies a RandLOCAL algorithm taking $\tilde{O}(\log^3 \log n)$ time. Using a slightly larger palette of $(2 + \epsilon)\Delta$ colors, $\epsilon > 1/\log \Delta$, Ghaffari et al. [37] gave an $O(\epsilon^{-1} \log^2 \Delta \log \log \Delta (\log \log \log \Delta)^{1.71} \log n)$-time DetLOCAL edge coloring algorithm, improving a previous work [40].

We cannot hope to use fewer than $\Delta + 1$ colors on general graphs. Vizing [63] proved that $\Delta + 1$ suffices for any graph, and Holyer [44] proved that it is NP-hard to tell if a graph is Δ-colorable.

Table 1. A History of Notable Edge Coloring Algorithms and Lower Bounds, in Descending Order by Palette Size

Palette Size	Time	(Rand)	Notes	References
$f(\Delta)$	$\Omega(\log^* n)$	R	$\Delta = O(1)$	[48, 54]
$O(\Delta^2)$	$O(\log^* n - \log^* \Delta + 1)$	★	Vertex coloring $L(G)$	[48]
$\Delta^{1+\epsilon}$	$O(\log \Delta + \log^* n)$	★		[10]
$O(\Delta \log n)$	$O(\log^4 n)$			[24]
$t(2\Delta - 2)$	$(\Delta/t)^{O(1)} \cdot O(\log n)$		Vertex coloring $L(G)$	[9]
$2^k\Delta$	$\tilde{O}(k\Delta^{1/2k}) + O(\log^* n)$	★	$k \geq 2$	[12]
$(2+\epsilon)\Delta$	$O(\epsilon^{-3} \log^{11} n)$			[40]
	$O(\epsilon^{-1} \log \Delta^{2+o(1)} \log n)$		$\epsilon > 1/\log \Delta$	[37]
$2\Delta - 1$	$2^{O(\sqrt{\log n})}$		Vertex coloring $L(G)$	[57]
	$\tilde{O}(\sqrt{\Delta}) + O(\log^* n)$	★	Vertex coloring $L(G)$	[32]
	$O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$	R	Vertex coloring $L(G)$	[13]
	$O(\log^* n)$	R★	$\Delta > (\log n)^{1+o(1)}$	[29]
	$2^{O(\sqrt{\log \log n})}$	R		[29]
	$O(\log^7 \Delta \log n)$			[31]
	$O(\log^4 \Delta \log^2 n)$			[35]
	$\tilde{O}(\log^2 \Delta \log n)$	★		[42]
	$\tilde{O}((\log \log n)^3)$	R★		[13]+[29]+[42]
$2\Delta - 2$	$\Omega(\log_\Delta \log n)$	R		**new**
	$\Omega(\log_\Delta n)$			**new**
1.6Δ	$O(\log n)$	R	$\Delta > \log^{1+o(1)} n$	[58]
$(1+\epsilon)\Delta$	$O(\epsilon^{-1} \log \epsilon^{-1} + \log n)$	R	$\Delta > (\log n)^{1+\gamma(\epsilon)}$	[26]
	$O\left((\epsilon^{-2} \log \epsilon^{-1} + \log^* \Delta) \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil\right)$	R	$\Delta > \Delta_\epsilon$	[29]
	$O\left(\log \epsilon^{-1} \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil + \log^* n\right)$	R★	$\epsilon\Delta > (\log n)^{1+o(1)}$	**new**
	$O\left(\log \epsilon^{-1} \left\lceil \frac{\log n}{\epsilon^2 \Delta^{1-o(1)}} \right\rceil + (\log \log n)^{3+o(1)}\right)$	R★	$\epsilon = \omega((\log^{2.5} \Delta)/\sqrt{\Delta})$	**new**
$\Delta + O(\log_\Delta n)$	$O(\Delta^{6+\epsilon} \log^3 n)$	★		[62]
$\Delta + 2$	$O(\Delta^{13} \log^3 n)$	R★		[62]
$\Delta + 1$	diameter(G)	★		[63]

Some $(2\Delta - 1)$-edge coloring algorithms that follow from vertex coloring $L(G)$, such as References [3, 8, 11, 47], 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 ★.

The best *sequential* $(\Delta + 1)$-edge coloring algorithms [2, 33] 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 small a natural way to solve the coloring problem [2, 33] is to begin with any maximal partial coloring, and then iteratively *recolor* portions of the graph (e.g., along "alternating paths") so that at least one uncolored edge can be legally colored. This approach was successfully employed by Panconesi and Srinivasan [56] in their distributed algorithm for Brooks' theorem, which states that any graph with $\Delta \geq 3$ having no $(\Delta + 1)$-cliques is Δ-vertex colorable. They proved that for *any* partial coloring, there exists an alternating path with length $O(\log_\Delta n)$, and that given a $(\Delta + 1)$-vertex coloring, a Δ-vertex coloring could be computed in $O(\log^2 n \log_\Delta n)$ additional time. This bound was recently improved by Ghaffari et al. [36], which offers some improved Δ-vertex coloring algorithms.

Table 2. A Survey of Distributed LLL Algorithms (with a Symmetric LLL Criterion)

Criterion	Time	Rand/Det	Notes	Reference
$ep(d+1) < 1$	$O(\text{MIS} \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[53]
	$O(\text{WeakMIS} \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[22]
	$O(\log d \cdot \log_{1/ep(d+1)} n)$	Rand	also asymmetric criterion	[34]+[22]
$epd^2 < 1$	$O(\log_{1/epd^2} n)$	Rand	also asymmetric criterion	[22]
$p2^d \text{poly}(d) < 1$	$O(\log n / \log\log n)$	Rand		[22]
$p(ed)^\lambda < 1$	$O(n^{1/\lambda} \cdot 2^{O(\sqrt{\log n})})$	Det	Any $\lambda \geq 1$	[30]
$p(ed)^{4\lambda} < 1$	$O(d^2) + (\log n)^{1/\lambda} \cdot 2^{O(\sqrt{\log\log n})}$	Rand	Any $\lambda \geq 8$	[30]
$p(ed)^{32} < 1$	$2^{O(\sqrt{\log\log n})}$	Rand	$d \leq (\log\log n)^{1/5}$	[30]
$20000pd^8 < 1$	$\exp^{(i)}\left(O\left(\log d + \sqrt{\log^{(i+1)} n}\right)\right)$	Rand	$i \geq 1$.	[35]
$p(ed)^{d^2+1} < 1$	$O(d^2 + \log^* n)$	Det		[30]
Lower Bounds (apply to tree-structured instances)				
$p \cdot f(d) \leq 1$	$\Omega(\log^* n)$	Rand	Any f	[22]
$p \cdot f(d) \leq 1$	$\Omega(\log_{\log(1/p)} \log n)$	Rand	Any $f(d) \leq 2^d$	[16]
$p \cdot f(d) \leq 1$	$\Omega(\log_d n)$	Det	Any $f(d) \leq 2^d$	[19]
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**

MIS $= O(\min\{d + \log^* n, \log d + 2^{O(\sqrt{\log\log n})}\})$ [11, 34] is the complexity of computing a maximal independent set in a graph with maximum degree d. WeakMIS $= O(\log d)$ [34] is the task of finding an independent set I such that the probability that v is not in/adjacent to I is $1/\text{poly}(d)$. If $T = (V, E)$ is a tree, then $T^r = (V, \{(u, v) : \text{dist}_T(u, v) \leq r\})$ is *tree-structured*, where $r = O(1)$. All lower bounds apply even to tree-structured instances. We do not try to optimize the LLL criterion $\lambda \geq 2(4^r + 8r)$ in the last line.

Lower Bounds. Linial's $\Omega(\log^* n)$ lower bound for $O(1)$-coloring the ring [48, 54] implies that $f(\Delta)$-edge coloring also cannot be computed in $o(\log^* n)$ time, for any function f. To the best of our knowledge, none of the other published lower bounds applies directly to the edge coloring problem. Kuhn, Moscibroda, and Wattenhofer's $\Omega(\min\{\frac{\log\Delta}{\log\log\Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$ lower bounds apply to MIS and maximal matching, but not to any vertex or edge coloring problem. Linial's $\Omega(\log_\Delta n)$ lower bound [48] (see Reference [60, 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. [16] (RandLOCAL $\Omega(\log_\Delta \log n)$) and Chang, Kopelowitz, and Pettie [19] (DetLOCAL $\Omega(\log_\Delta n)$) for *sinkless orientation* and Δ-*vertex coloring trees* do not naturally generalize to edge coloring.

Distributed Lovász Local Lemma. 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. Chang and Pettie [21] conjectured that the RandLOCAL complexity of the LLL under some polynomial LLL criterion is $O(\log\log n)$, matching the Brandt et al. [16] lower bound. If this conjecture were true, due to the necessity of graph shattering [19, Theorem 3], then an optimal randomized LLL algorithm should be structured as follows. It must combine an $O(\log n)$-time deterministic LLL algorithm and an $O(\log\log n)$-time randomized *graph shattering* routine to break the dependency graph into $\text{poly}(\log n)$-size LLL instances. Fischer and

Ghaffari [30] 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 poly$(\log n)$-size components. More recently, Ghaffari et al. [35] developed a generic derandomization method for the LOCAL model that implies randomized LLL algorithms running in time $\exp^{(i)}(O(\log d + \sqrt{\log^{(i+1)} n}))$, for any $i \geq 1$. For example, when $d < 2^{O(\sqrt{\log \log n})}$, their LLL algorithm runs in $2^{O(\sqrt{\log \log n})}$ time.

1.5 Organization

In Section 2, we give lower bounds on $(2\Delta - 2)$-edge coloring. In Section 3, we give lower bounds on a class of "recoloring" algorithms for Vizing's theorem. In Section 4, we give a randomized $(1 + \epsilon)\Delta$-edge coloring algorithm, which requires a distributed LLL algorithm when $\epsilon^2\Delta$ is sufficiently small. In Section 5, we give new LLL algorithms for tree-structured dependency graphs. In Section 6, we present new network decomposition algorithms for trees, which are used in Section 5. In Section 7, we prove some new bounds on the complexity of Δ- and $(\Delta + 1)$-edge coloring trees, both in the oriented and unoriented cases. Much of the analysis of the randomized edge-coloring algorithm (Section 4) appears in Appendix A.

2 LOWER BOUND FOR $(2\Delta - 2)$-EDGE COLORING

The *sinkless orientation* problem [16] is to direct the edges such that no vertex has out-degree zero. Since this problem becomes *harder* with fewer edges, in this section, we write $\Delta_{\min}$ and $\Delta_{\max}$ to denote the *minimum* and the *maximum* degree. We follow the method of References [16, 19], who proved that Δ-coloring graphs (even trees) requires $\Omega(\log_\Delta \log n)$ in RandLOCAL and $\Omega(\log_\Delta n)$ in DetLOCAL. Brandt et al. [16] begin by reducing the sinkless orientation problem, *in which nodes initially know a Δ-edge coloring of the graph*, to Δ-vertex coloring. Having the Δ-edge coloring available is essential for making the reduction work, and intuitively it leaks no information helpful for solving either problem. In Theorem 1, we begin with a similar reduction, showing that sinkless orientation on bipartite graphs, *in which nodes initially know a proper 2-vertex coloring*, is reducible to $(2\Delta - 2)$-edge coloring. We then proceed to prove lower bounds on sinkless orientation, given the aforementioned 2-vertex coloring, and even given a proper $(2\Delta - 1)$-edge coloring. (By Theorem 1, reducing the edge-coloring palette to $2\Delta - 2$ would trivialize the sinkless orientation problem.)

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

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

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

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

Thus, any lower bound for sinkless orientation on 2-vertex colored bipartite graphs also applies to $(2\Delta-2)$-edge coloring.

Infinite Δ-regular Tree $\mathcal{T}_\Delta$. 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 $(2\Delta-1)$-edge 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.

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

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.

Irregular Time Profile. We say that an algorithm on a k-edge colored graph G has irregular time profile $\mathbf{t} = (t_1,\ldots,t_k)$ if edges with input color i decide their output by examining only their t_i-neighborhood. By definition, a time-t algorithm has time profile $(t,t,t,\ldots,t)$. In the subsequent discussion, we will apply this concept to $\mathcal{T}_\Delta$. Recall that the edges in $\mathcal{T}_\Delta$ are properly $(2\Delta-1)$-edge colored, and so an irregular time profile for an algorithm on $\mathcal{T}_\Delta$ is a $(2\Delta-1)$-tuple.

Lemma 1 (Round Elimination Lemma). *Suppose $\mathcal{A}_{s.o.}$ is a sinkless orientation algorithm for $\mathcal{T}_\Delta$ with local error probability p and time profile $(\underbrace{t,t,\ldots,t}_{i},\underbrace{t-1,\ldots,t-1}_{(2\Delta-1)-i})$, i.e., edges colored $\{1,\ldots,i\}$ halt after t rounds and the others after $t-1$ rounds. There exists a sinkless orientation algorithm $\mathcal{A}'_{s.o.}$ for $\mathcal{T}_\Delta$ with local error probability $3p^{1/3}$ and time profile $(\underbrace{t,t,\ldots,t}_{i-1},\underbrace{t-1,\ldots,t-1}_{(2\Delta-1)-(i-1)})$.*

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

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

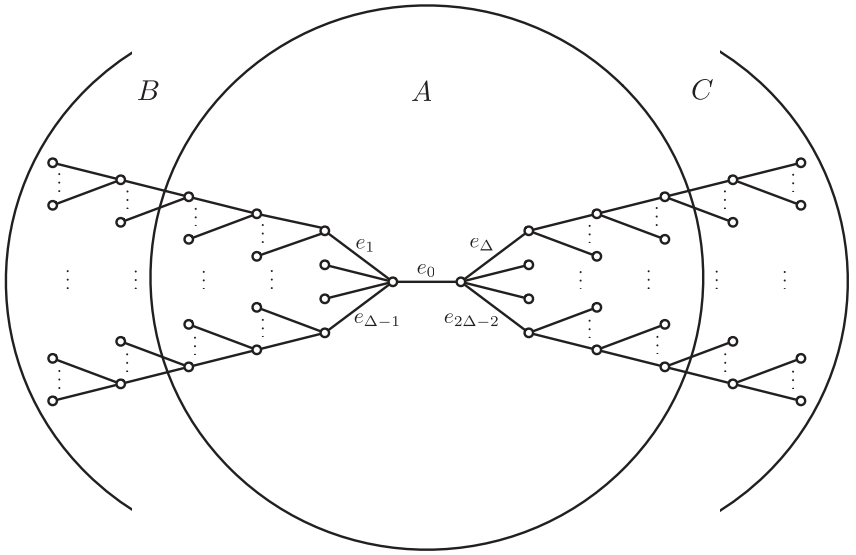

Fig. 1. The algorithm for e_0 inspects its $(t-1)$-neighborhood $A = N^{t-1}(e_0)$ and estimates the probability that $\mathcal{E}_0$ and $\mathcal{E}_1$ occur. Observe that $\mathcal{E}_0$ is a function of $\bigcup_{j\in[1,\Delta-1]} N^t(e_j)$, which is completely contained in $A \cup B$, and $\mathcal{E}_1$ is a function of $\bigcup_{j\in[\Delta,2\Delta-2]} N^t(e_j)$, which is completely contained in $A \cup C$. Hence, after conditioning on $\mathcal{E}_0^\star \cap \mathcal{E}_1^\star$ (which depends only on A), $\mathcal{E}_0$ and $\mathcal{E}_1$ become independent, since $B \cap C = \emptyset$.

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

$$\Pr[\mathcal{E}_0 \cap \mathcal{E}_1] \le 2p. \tag{1}$$

On edge e_0, $\mathcal{A}'_{s.o.}$ gathers its $(t-1)$-neighborhood $N^{t-1}(e_0)$ and evaluates whether the following two events $\mathcal{E}_0^\star, \mathcal{E}_1^\star$ occur. Intuitively, $\mathcal{E}_0^\star$ indicates that $\mathcal{E}_0$ is *dangerously likely* to happen, conditioned on $N^{t-1}(e_0)$, and likewise with $\mathcal{E}_1^\star$ and $\mathcal{E}_0$. See Figure 1.

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

Notice that if we inspect $N^{t-1}(e_0)$, and condition on the information seen in $N^{t-1}(e_0)$, the events $\mathcal{E}_0$ and $\mathcal{E}_1$ become independent, since they now depend on disjoint sets of random variables. Specifically, $\mathcal{E}_0$ depends on $\bigcup_{j\in[1,\Delta-1]} N^t(e_j)\backslash N^{t-1}(e_0)$ and $\mathcal{E}_1$ depends on $\bigcup_{j\in[\Delta,2\Delta-2]} N^t(e_j)\backslash N^{t-1}(e_0)$.[6] Thus,

$$\Pr[\mathcal{E}_0 \cap \mathcal{E}_1 \mid N^{t-1}(e_0)] = \Pr[\mathcal{E}_0 \mid N^{t-1}(e_0)] \cdot \Pr[\mathcal{E}_1 \mid N^{t-1}(e_0)]. \tag{2}$$

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

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

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

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

[6]Here, the analysis relies on the following fact, which is a consequence of how we generate the $(2\Delta-1)$-edge coloring of $\mathcal{T}_\Delta$. Conditioning on the colors of the edges in $N^{t-1}(e_0)$, the colors of the edges in $\bigcup_{j\in[1,\Delta-1]} N^t(e_j)\backslash N^{t-1}(e_0)$ and the colors of the edges in $\bigcup_{j\in[\Delta,2\Delta-2]} N^t(e_j)\backslash N^{t-1}(e_0)$ are independent.

We now calculate the failure probabilities of u_0 and u_1:

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

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

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

$$2^{3\Delta} \ge p_0^{-1} \;=\; \Omega\left((p^{-1})^{3^{-t(2\Delta-1)}}\right),$$

and taking logs twice, we have

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

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

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

Proof. Consider any sinkless orientation or $(2\Delta - 2)$-edge coloring algorithm with local probability of failure p. Lemma 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, 25], we get $\Omega(\min\{\Delta^{-1} \log\log p^{-1}, \log_\Delta n\})$ lower bounds. Following the same proof as Reference [19, Theorem 5], this implies an $\Omega(\log_\Delta n)$ lower bound in DetLOCAL, which, according to Reference [19, 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 *automatically* implies a stronger lower bound. □

3 LOWER BOUNDS FOR RECOLORING-TYPE ALGORITHMS

In this section, we show that for $c \in [1, \frac{\Delta}{3}]$, any algorithm for $(\Delta + c)$-edge coloring based on *extending partial colorings by recoloring subgraphs* needs $\Omega(\frac{\Delta}{c} \log \frac{cn}{\Delta})$ rounds.

Theorem 3. *Let Δ be the maximum degree and $c \in [1, \frac{\Delta}{3}]$. For any n, there exists an n-vertex graph $G = (V, E)$ and a partial edge coloring $\phi : E \to \{1, \ldots, \Delta + c, \perp\}$, with exactly one uncolored edge e_0*

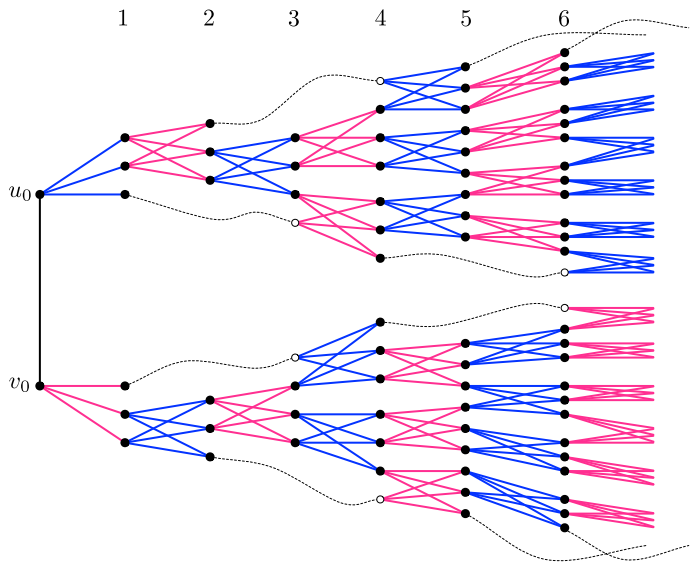

Fig. 2. An example of the construction when $\Delta = 5$, $c = 1$, $k = 3$, $k' = 2$, and $\ell \geq 7$. Edges colored by palette $S_0 = \{1, 2, 3\}$ are blue, and edges colored by palette $S_1 = \{4, 5, 6\}$ are pink. Leftover vertices in layer $i - 2$ are also depicted (hollow) in layer i, and joined by a dashed curve. They represent the same vertex, not two different vertices.

($\phi(e_0) = \perp$) satisfying the following property. For any total edge coloring $\phi' : E \to \{1, \ldots, \Delta + c\}$ of G, ϕ and ϕ' differ on a subgraph of diameter $\Omega(\frac{\Delta}{c} \log(\frac{cn}{\Delta}))$.

Suppose that G is a partially $(\Delta + c)$-edge colored graph, where an edge e_0 in uncolored. A natural approach to color e_0 is to find an "alternating path" $e_0 e_1 \cdots e_\ell$, and then recolor the path. That is, for $0 \leq i \leq \ell - 1$, let the new color of e_i be the old color of e_{i+1}, and then color the last edge e_ℓ by choosing any available color (if possible). This type of approach has successfully led to a distributed algorithm for Brooks' theorem [56]. Specifically, given a $(\Delta + 1)$-vertex coloring, it was shown in Reference [56] that a Δ-coloring can be computed in poly$(\log n)$ time, independent of Δ. See Ghaffari et al. [36] for several faster algorithms. However, Theorem 3 implies the existence of a graph where any alternating subgraph has diameter $\Omega(\frac{\Delta}{c} \log \frac{cn}{\Delta})$, which is expensive for large Δ. The remainder of this section is a proof of Theorem 3.

Construction. Without loss of generality, assume that $\Delta + c$ is even, and let $k = \frac{\Delta + c}{2}$. We divide the color palette $\{1, \ldots, \Delta + c\}$ into two equal-size sets $S_0 = \{1, \ldots, k\}$ and $S_1 = \{k + 1, \ldots, \Delta + c\}$. (One may refer to Figure 2 for an example, with $\Delta = 5, c = 1$. In the figure, blue edges are colored from palette S_0 and pink edges from S_1.) Let $k' = \Delta - k$.

The graph $G^*(\ell, \Delta, c)$ consists of one uncolored edge $e_0 = \{u_0, v_0\}$; all other vertices are arranged in layers $1, \ldots, \ell$ and all other edges connect two vertices in adjacent layers or layers i and $i + 3$, for some i. In $G^*(\ell, \Delta, c)$, e_0 is a bridge and the subgraphs attached to u_0 and v_0 are structurally isomorphic, but colored differently. Thus, we focus on the half of G^* attached to u_0.

Base Case. Layer 1 consists of k vertices attached to u_0. They are initially colored with distinct colors from S_0.

Inductive Step. The $(i + 1)$th layer is constructed as follows. We take all the vertices at layer i and the *leftover* vertices at layer $i - 2$ and partition them into groups of size k'; any ungrouped vertices

are called *leftovers* at level i. (In Figure 2, a leftover vertex in layer $i-2$ is drawn twice, solid in layer $i-2$ and hollow when it is *promoted* to layer i; they are connected by a dashed line.) The grouping is arbitrary, so long as all vertices promoted from layer $i-2$ are grouped. Each group forms the left-hand side of a complete bipartite graph $K_{k',k}$. Layer $i+1$ consists of the right-hand side of all the (disjoint) copies of $K_{k',k}$. All the edges in these graphs are properly colored with S_b where $b = i \bmod 2$. (The subgraph attached to v_0 is constructed in the same way, except that we flip the parity: the complete bipartite graphs are colored with S_b, $b = (i+1) \bmod 2$.)

Define n_i and l_i as the number of layer-i vertices and layer-i leftover vertices.[7] According to the construction, (n_i) and (l_i) satisfy the following recurrences:

$$\begin{aligned} n_1 &= k, \\ l_{-1} = l_0 &= 0, \\ n_{i+1} &= k \left\lfloor \frac{n_i + l_{i-2}}{k'} \right\rfloor && \text{for } i+1 \geq 2, \\ l_i &= (n_i + l_{i-2}) \bmod k' && \text{for } i \geq 1. \end{aligned}$$

Clearly, $n_i = \Theta((k/k')^i)$. Define $\epsilon = \frac{2c}{\Delta - c}$ so that $k/k' = \frac{\Delta + c}{\Delta - c} = 1 + \frac{2c}{\Delta - c} = 1 + \epsilon$. The total number of vertices in $G^*(\ell, \Delta, c)$ is $n = \Theta(\epsilon^{-1} n_\ell) = \Theta(\epsilon^{-1}(1+\epsilon)^\ell)$ and $\ell = \Theta(\log_{1+\epsilon}(\epsilon n)) = \Theta(\frac{\Delta}{c} \log \frac{cn}{\Delta})$. In particular, when c is constant and $\Delta < n^{1-\Omega(1)}$, $\ell = \Omega(\Delta \log n)$. The diameter of the graph is at least $\ell/3$, since, by construction, no edge crosses more than three layers. We remark that the purpose of the requirement $c \leq \Delta/3$ in the statement of Theorem 3 is to make $\epsilon \leq 1$. Our construction still applies to the case of $c > \Delta/3$, but it gives a worse bound on ℓ when c is close to Δ.

Let ϕ be the initial partial edge-coloring of $G^*(\ell, \Delta, c)$, with e_0 left uncolored, and ϕ' be any total edge-coloring. We claim that ϕ' recolors at least one edge in the subgraph induced by layers $\ell - 5, \ldots, \ell$. Suppose otherwise. Fix any vertex v in layer $\ell - 6$. It has exactly k neighbors in a higher layer, either $\ell - 5$ (if v is not a leftover vertex) or $\ell - 3$ (if v is a leftover vertex); each such neighbor u is adjacent to k edges to a higher layer, all of which are colored from the palette S_1 (without loss of generality, assume ℓ is even). That means that all edges connecting v to a higher layer must be colored from S_0. By a reverse induction from $\ell - 6$ down to 0, it follows that all edges from u_0 to layer 1 must be colored with S_0. A symmetric argument on v_0's side shows that all edges from v_0 to layer 1 must be colored with S_1; hence, e_0 cannot be properly colored by ϕ'.

4 RANDOMIZED EDGE COLORING ALGORITHM

Elkin et al. [29] showed that for any *constant* $\epsilon > 0$, there is a number Δ_ϵ (depending only on 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 Reference [29] (or its predecessor, Reference [26]) still works if we allow $\epsilon = o(1)$. For instance, is it possible to solve $(\Delta + \Delta^{0.7})$-edge coloring in RandLOCAL in $O(\text{poly} \log n)$ time?

[7] The leftover vertices at layer $i-2$ are still considered as layer i vertices, even though they have been promoted to layer i.

Challenges to Reducing the Number of Colors. The analysis of our algorithm is substantially more involved than all previous edge coloring algorithms [26, 29, 58]. Here, we give a short technical review of the types of issues faced in distributed edge coloring.

Previous algorithms [26, 29] are based on the *Rödl Nibble* method. In each round, every uncolored edge nominates itself to be colored with probability $O(\epsilon)$ and remains idle otherwise; a self-nominated edge picks a free color from its available palette and *permanently* colors itself if the colors selected by adjacent edges do not conflict with it. The goal is to show that natural quantities (palette size, degree of vertices in the uncolored graph, etc.) are sharply concentrated around their expectations. The first issue is finding the right concentration bound. Chernoff bounds are insufficient for several reasons, one of which is the need for independence (or negative dependence [27, 28]) between the events of interest. Azuma's inequality and variants fall short due to the weakness of Lipschitz properties (bounded differences).[8] The algorithm of Dubhashi, Grable, and Panconesi [26] used a specialized concentration inequality of Grable [41], whereas our algorithm and that of Elkin, Pettie, and Su [29] use one [27, Theorem (8.5)] that is syntactically closer to Chernoff/Hoeffding/Azuma-type inequalities. (It is restated as Theorem 13 in Appendix A.)

The purpose of the "self-nomination" step in References [26, 29] is to simplify certain aspects of the analysis. For example, the probability that an edge is successfully colored, *conditioned on it nominating itself*, is a very high $1 - O(\epsilon)$. Because of this, we can afford to toss out any color c from e's palette if any *self-nominated* edge e' adjacent to e selects c—regardless of whether e' successfully colors itself. This type of subtle change generally makes things simpler. Some events that would ordinarily be dependent become independent, and some variables (e.g., a vertex's c-degree) now depend on $\Theta(\Delta^2)$ variables rather than $\Theta(\Delta^3)$. The downside of this approach is that $\Omega(\epsilon^{-1})$ steps are necessary to color a large fraction of the graph, and *with each coloring step* the quantities we are monitoring (c-degree, palette size, etc.) deviate further from their expectations. When ϵ^{-1} is polynomial in Δ, the accumulated deviation errors make it impossible to achieve palette sizes as small as $\Delta + \tilde{O}(\sqrt{\Delta})$.

Our Approach. Our algorithm is more "natural" than References [26, 29]. Roughly speaking, in each step each edge chooses a color uniformly at random from its available palette and permanently colors itself if there are no local conflicts (One-Shot-Coloring). That is, we dispense with the low probability self-nomination step. Let p_i be a lower bound on the palette size after i such steps, and d_i, t_i be upper bounds on uncolored degree and c-degree of any vertex, respectively. It is straightforward to show that if everything behaves precisely according to expectation, the (d_i) sequence shrinks by a $(1 - e^{-2})$ factor in each step and both (p_i), (t_i) shrink by a $(1 - e^{-2})^2$ factor. In reality these quantities *do* deviate from their expectations, and even tiny, $(1 + o(1))$-factor deviations compound themselves and spin out of control. One reason our analysis is more complex than References [26, 29] is that we look at concentration up to *lower order terms*. For example, although $p_i \approx t_i$, we bound $\beta_i = \frac{p_i}{t_i} - 1$, which captures accumulated errors beyond the leading constants.

The Use of the Distributed LLL. As in Reference [29], we obtain good concentration on d_i, p_i, t_i with probability $1 - \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta)$, which is $1 - 1/\text{poly}(n)$ if Δ and ϵ are sufficiently large. If not, then we must invoke a distributed LLL algorithm to make sure each random coloring experiment introduces bounded deviation errors in d_i, p_i, t_i. A constant fraction of the edges are colored in each step. For many parameter regimes the running time is dominated by $O(\log \epsilon^{-1})$

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

calls to an distributed LLL algorithm, as our algorithm needs to run One-Shot-Coloring for this many iterations.

4.1 Our Result

In this section, we prove the following theorem, which improves upon the algorithms of References [26, 29]. Here, $T^*(n, \Delta')$ is the RandLOCAL complexity of the $5\Delta'$-edge coloring problem, and $T_{\text{LLL}}(n, p, d)$ is the complexity of distributed LLL with parameters p and d.

THEOREM 4. *Let $\epsilon = \omega(\frac{\log^{2.5}\Delta}{\sqrt{\Delta}})$ be a function of Δ. There is a* RandLOCAL *algorithm for $(1+\epsilon)\Delta$-edge coloring in time*

$$O\left(\log(1/\epsilon)\right) \cdot T_{LLL}\left(n, d, p\right) \; + \; T^*\left(n, O(\epsilon\Delta)\right),$$

where $p = \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta) = \exp(-\omega(\log\Delta))$ and $d = O(\text{poly}(\Delta))$.

The statement of Theorem 4 guarantees that whenever ϵ and Δ satisfy the specified condition, we always have $\exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta) = \exp(-\omega(\log\Delta))$, and 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 4. Selecting smaller ϵ allows us to use fewer colors, but it leads to a higher $p = \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta)$, which may increase the runtime of the LLL algorithm.

Runtime of $5\Delta'$-edge Coloring. It is known that $T^*(n, \Delta')$ is at most $O(\log\Delta')$ plus the DetLOCAL complexity of $3\Delta'$-edge coloring on $\text{poly}(\log n)$-size graphs. This is achieved by applying the $(\tilde{\Delta}+1)$-vertex coloring algorithm of Reference [13] 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 [29]. The state-of-the-art DetLOCAL algorithm [37] 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 References [13, 29, 37] with $x = 1$, we have

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

This is achieved as follows. If $\Delta' = \Omega(\log^2 n)$, then we run the $O(\log^* n)$-time RandLOCAL algorithm of Reference [29]. Otherwise, we run the RandLOCAL graph shattering phase of Reference [13] (using the first $2\Delta'$ colors) followed by the DetLOCAL algorithm of Reference [37] (using the remaining $3\Delta'$ colors) on each component.

4.2 Time Complexity Analysis

We calculate the time complexity for Theorem 4 in different parameter regimes of Δ and ϵ.

Running Time on General Graphs. Our algorithm computes a $(1+\epsilon)\Delta$-edge coloring in $O(\log n)$ time when $\epsilon = \omega(\log^{2.5}\Delta)/\sqrt{\Delta})$. Observe that in this parameter regime, we have $\log(1/\epsilon) = O(\log\Delta)$. Applying the distributed LLL algorithm of Chung, Pettie, and Su [22] under the criterion $epd^2 < 1$, we obtain $T_{\text{LLL}}(n, d, p) = O(\log_{1/ep(d+1)} n) = O(\log_\Delta n)$, as $\log(1/p) = \omega(\log\Delta)$. Therefore, $O(\log(1/\epsilon)) \cdot T_{LLL}(n, d, p) = O(\log n)$. By applying one of the Ghaffari-Harris-Kuhn LLL algorithms [30, 35], the cost per LLL is $O(d^2 + 2^{O(\sqrt{\log\log n})}) = O(\Delta^6 + 2^{O(\sqrt{\log\log n})})$.

The term $T^*(n, O(\epsilon\Delta)) = (\log\log n)^{3+o(1)}$ can become the dominant term when Δ is sufficiently large. In particular, when $\epsilon = \Omega(1)$, we have $\log(1/\epsilon) = O(1)$, and so our algorithm is able to finish in $O(\log_\Delta n) + (\log\log n)^{3+o(1)}$ time.

Runtime on Trees. Consider running our algorithm on a tree with palette size $(1+\epsilon)\Delta$, where $\epsilon = \Omega(\frac{\log^{2.5+x}\Delta}{\sqrt{\Delta}})$, 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 4, which satisfy the criterion $p(ed)^\lambda < 1$ with $\lambda = \Omega(\log^x \Delta)$. Using our randomized LLL algorithm for tree-structured dependency graphs (Section 5), we have

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

We claim that $T^*(n,\Delta') = O(\log^*\Delta' + \log_{\Delta'}\log n)$ on trees. This is achieved as follows. First, do a $O(\log^*\Delta')$-time randomized procedure to partially color the graph using the first $2\Delta'$ colors so that the remaining uncolored components have size $\text{poly}(\log n)$. This can be done using the algorithm of [29] without invoking any distributed LLL algorithm. Then, apply our deterministic $O(\log_{\Delta'}\tilde{n})$-time algorithm for Δ'-edge coloring trees (Section 7) to each uncolored component separately, using a set of Δ' fresh colors.

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

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

This nearly matches our $\Omega(\log_\Delta\log n)$ lower bound (Section 2).

For the case $\epsilon = \Omega(1)$, our algorithm runs faster, as we can use $\lambda = \Delta/\text{poly}\log\Delta$, and so the running time of the distributed LLL becomes $O(\max\{\frac{\log\log n}{\log\log\log n}, \log_\Delta\log n\})$. In this case, our algorithm finds a $(1+\epsilon)\Delta$-edge coloring in $O(\max\{\frac{\log\log n}{\log\log\log n}, \log_\Delta\log n\})$ time.

4.3 The Algorithm and Its Invariants

Our algorithm has two phases. The goal of the first phase is to color a subset of the edges using the colors from $C_1 \stackrel{\text{def}}{=} \{1,\ldots,\Delta(1+\xi)\}$ 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 $C_2 \stackrel{\text{def}}{=} \{\Delta(1+\xi)+1,\ldots,\Delta(1+\epsilon)\}$ 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 $\Psi_i(e) \subseteq C_1$. 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 $\Psi_i(e) = C_1$ for all edges. In the graph G_i, we maintain the following invariant $\mathcal{H}_i$.

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

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

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 ($i > 1$):

$$d_i \stackrel{\text{def}}{=} (1+\delta_{i-1})d^{\diamond}_{i-1}, \qquad d^{\diamond}_{i-1} \stackrel{\text{def}}{=} d_{i-1} \cdot \left(1-(1-1/p_{i-1})^{2(t_{i-1}-1)}\right),$$

$$t_i \stackrel{\text{def}}{=} (1+\delta_{i-1})t^{\diamond}_{i-1}, \qquad t^{\diamond}_{i-1} \stackrel{\text{def}}{=} t_{i-1} \cdot \left(1-\frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}\right)\left(1-(1-1/p_{i-1})^{2t_{i-1}}\right),$$

$$p_i \stackrel{\text{def}}{=} (1-\delta_{i-1})p^{\diamond}_{i-1}, \qquad p^{\diamond}_{i-1} \stackrel{\text{def}}{=} p_{i-1} \cdot \left(1-\frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}\right)^2.$$

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 One-Shot-Coloring 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 One-Shot-Coloring 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 One-Shot-Coloring) to color a subset of edges in G_i so that the next graph G_{i+1} induced by the uncolored edges satisfies $\mathcal{H}_{i+1}$.

Termination of the First Phase. The number of iterations of our algorithm will be $i^\star - 1 = O(\log(1/\epsilon))$ (Lemma 4). We will later see that after the $(i^\star-1)$th iteration, the degree of the vertices in the remaining uncolored part of the graph $G_{i^\star}$ satisfies the "terminating condition" $d_{i^\star} \le \frac{1}{5}(\epsilon-\xi)\Delta$. Then, we proceed to the second phase.

The purpose for requiring this condition is to create a sufficiently large gap between the maximum degree Δ' (in the remaining uncolored part of the graph) and the number of available colors (the colors $C_2 = \{\Delta(1+\xi)+1, \ldots, \Delta(1+\epsilon)\}$ reserved for the second phase), so that we can run a $5\Delta'$-edge coloring algorithm to color all remaining edges in the second phase of the algorithm.

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. The proof is deferred to Section 4.5.

Lemma 3. *Consider an index $i > 1$. Suppose* $\min\{d_{i-1}, t_{i-1}, p_{i-1}\} = \omega(\log\Delta)$, $\beta_{i-1} = o(1/\log\Delta)$, *and* $\delta_{i-1} = o(\beta_{i-1}/\log\Delta)$. *Then the following four equations hold:*

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

Based on Lemma 3, we have the following lemma.

Lemma 4. *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 is met for all indices $1 < i \le i^\star$. We prove this by an induction on i. By the induction hypothesis the four equations hold at index $i-1$. We show that the condition of Lemma 3 is met for the index i, and so the four equations also hold for index i. Due to $1/\eta = o(1/\log\Delta)$, we already have $\beta_{i-1} = o(1/\log\Delta)$ and $\delta_{i-1} = o(\beta_{i-1}/\log\Delta)$. It remains to prove that $\min\{d_{i-1}, t_{i-1}, p_{i-1}\} = \omega(\log\Delta)$.

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

□

It remains to show that (i) the terminating condition $d_{i^\star} \le \frac{1}{5}(\epsilon - \xi)\Delta$ is satisfied at the end of the $(i^\star-1)$th iteration, and (ii) in each iteration, in $T_{LLL}(n, \mathrm{poly}(\Delta), \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta))$ time, invariant $\mathcal{H}_i$ can be maintained. By Lemma 4, we have

$$\begin{aligned}
d_{i^\star} &= (1 \pm o(1))\Delta(1-e^{-2})^{i^\star-1} && \text{(Lemma 4 for } d_{i^\star})\\
&= (1 \pm o(1))\Delta \cdot \xi/\beta_{i^\star} && \text{(Lemma 4 for } \beta_{i^\star})\\
&\le (1 \pm o(1))\xi\eta\Delta. && (\beta_{i^\star} > 1/\eta)
\end{aligned}$$

For our choices of η and ξ, we have $d_{i^\star} \approx \xi\eta\Delta = \frac{\epsilon\Delta}{6}$. Since $\frac{1}{5}(\epsilon-\xi)\Delta > \frac{\epsilon\Delta}{6}$, the condition $d_{i^\star} \le \frac{1}{5}(\epsilon-\xi)\Delta$ is satisfied.

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

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

We will later see in Section 4.4 that this implies that any LLL algorithm with parameters $d = \mathrm{poly}(\Delta)$ 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.

4.4 Maintenance of the Invariant

In this section, we show how to apply a distributed LLL algorithm, with parameters $d = \text{poly}(\Delta)$ and $p = \exp(-\Omega(\delta_i^2 \cdot \min\{d_i, t_i, p_i\}))$, to achieve the following task: given a graph G_i meeting the property $\mathcal{H}_i$, color a subset of edges of G_i so that the graph induced by the remaining uncolored edges satisfies the property $\mathcal{H}_{i+1}$. We write $\Psi(e) = \Psi_i(e)$ for notational simplicity.

Achieving Uniform Progress. Consider the following modifications to the underlying graph G_i:

- Each edge e discards some arbitrary colors from its palette to achieve uniform palette size p_i.
- Each vertex v locally simulates some *imaginary* subtrees attached to v and obeying $\mathcal{H}_i$ to achieve uniform color degree t_i. That is, if a color c appears in the palette of some edge incident to a vertex v, then c must appear in the palette of *exactly* t_i edges incident to v.

These modifications to the underlying graph are introduced to enforce broadly *uniform* progress in every part of the graph.[9]

Observe that if $\mathcal{H}_i$ applies to the imaginary graph it also applies to the true graph as well, since we are concerned with *lower* bounds on palette sizes and *upper* bounds on c-degrees.

To increase the c-degree of each vertex v to t_i, we might need to add so many imaginary edges to v such that the degree of v exceeds d_i if we take into account these imaginary edges. This is fine, as we will later see that we only consider the real edges when we analyze the shrinking rate of the degree.[10]

One Shot Coloring. Our analysis focusses largely on how the following $O(1)$-round procedure affects the imaginary graph.

One-Shot-Coloring.

(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 $S^\diamond(v)$ (respectively, $N_c^\diamond(v)$) be the subset of $S(v)$ (respectively, $N_c^\diamond(v)$) that are still uncolored after One-Shot-Coloring. 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.

[9]The algorithm will likely work if it is run on the actual graph (that is, without hallucinating imaginary subtrees), but we do not see a way to enforce the same invariants. For example, suppose a vertex v in G_i has degree exactly d_i, but because the palettes in v's neighborhood happen to be advantageously configured, v's degree after one coloring step is likely to be *much less* than d_{i+1}. Surely, this is a good outcome! Yet, if more edges are colored than we expect, then the remaining edges will lose *more colors from their palettes* than we expect, possibly violating the lower bound on p_{i+1}. These concerns motivate us to enforce more uniform progress, hence the introduction of imaginary trees.

[10]In particular, if we want to increase the c-degree of v by k, then we can add k imaginary edges $e_1, \ldots, e_k$ incident to v such that the palette of each newly added edge contains c. Other than the color c, there is no overlap between the palettes of the newly added edges and other edges incident to v.

The following concentration bound implies that $\mathcal{H}_{i+1}$ holds with high probability in the graph induced by the real uncolored edges after One-Shot-Coloring, and thus we can apply a distributed LLL algorithm to obtain G_{i+1} that meets the invariant $\mathcal{H}_{i+1}$. See Appendix A for proof.

Lemma 5. *Suppose that $\mathcal{H}_i$ holds. The following concentration bounds hold for any $\delta > 0$.*

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

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 are both in $N^2(v)$, (ii) $N_c^\diamond(v)$ depends only on the colors selected by the edges whose endpoints are both in $N^3(v)$, and (iii) $\Psi^\diamond(e)$ depends only on the colors selected by the edges whose endpoints are both in $N^2(u) \cup N^2(v)$, where $e = \{u, v\}$.

Thus, the parameters for the LLL are $d = \text{poly}(\Delta)$ and $p = \exp(-\Omega(\delta_i^2 \cdot \min\{d_i, t_i, p_i\}))$ by Lemma 5. Recall from the calculation in Section 4.3 that $\delta_i^2 \cdot \min\{d_i, t_i, p_i\} = \Omega(\epsilon^2\Delta/\eta^4)$. We obtain the bound $p = \exp(-\epsilon^2\Delta/\log^{4+o(1)}\Delta) = \exp(-\omega(\log\Delta))$ required in the statement of Theorem 4 by selecting $\eta = \log^{1+o(1)}\Delta$.

4.5 Proof of Lemma 3

In this section, we prove Lemma 3. We assume $\min\{d_{i-1}, t_{i-1}, p_{i-1}\} = \omega(\log\Delta)$, $\beta_{i-1} = o(1/\log\Delta)$, and $\delta_{i-1} = o(\beta_{i-1}/\log\Delta)$. The two terms $(1-1/p_{i-1})^{2t_{i-1}}$ and $\frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}$ show up in the definition of $d_{i-1}^\diamond$, $t_{i-1}^\diamond$, and $p_{i-1}^\diamond$. We begin by showing that these two terms are both $e^{-2}(1 + o(1/\log\Delta))$. We use the fact that $\frac{t_{i-1}}{p_{i-1}} = \frac{1}{\beta_{i-1}+1}$ in the following calculation:

$$\begin{aligned}
(1-1/p_{i-1})^{2t_{i-1}} &= e^{-2t_{i-1}/p_{i-1}}(1 - O(t_{i-1}/p_{i-1}^2)) && \text{(Taylor expansion of } e^x\text{)}\\
&= e^{-2}\cdot e^{2(1-t_{i-1}/p_{i-1})}(1 - O(t_{i-1}/p_{i-1}^2))\\
&= e^{-2}\cdot e^{2(1-t_{i-1}/p_{i-1})}\left(1 - O\left(\frac{1}{(1+\beta_{i-1})p_{i-1}}\right)\right) && \text{(Defn. } \beta_{i-1}\text{)}\\
&= e^{-2}\cdot e^{2(1-t_{i-1}/p_{i-1})}(1 - o(1/\log\Delta)) && (p_{i-1} = \omega(\log\Delta))\\
&= e^{-2}\cdot e^{2\beta_{i-1}/(\beta_{i-1}+1)}(1 - o(1/\log\Delta)) && \text{(Defn. of } \beta_{i-1}\text{)}\\
&= e^{-2}\cdot (1 + O(2\beta_{i-1}/(\beta_{i-1}+1)))(1 - o(1/\log\Delta))\\
&= e^{-2}\cdot (1 + o(1/\log\Delta))(1 - o(1/\log\Delta))\\
&= e^{-2}(1 + o(1/\log\Delta)). && (*)\\
\frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}} &= e^{-2}\cdot\frac{t_{i-1}}{p_{i-1}}\cdot(1 + o(1/\log\Delta)) && \text{by (*)}\\
&= e^{-2}(1 + o(1/\log\Delta))/(1+\beta_{i-1})\\
&= e^{-2}(1 + o(1/\log\Delta))/(1 + o(1/\log\Delta))\\
&= e^{-2}(1 \pm o(1/\log\Delta)). && (**)
\end{aligned}$$

We are in a position to derive the first three equations in Lemma 3 (i.e., estimates of d_i, t_i, and p_i). Recall that $\delta_{i-1} = o(1/\log^2 \Delta)$ and $1/p_{i-1} = o(1/\log \Delta)$:

$$\begin{aligned} d_i &= d_{i-1} \cdot (1+\delta_{i-1}) \left(1 - (1-1/p_{i-1})^{2(t_{i-1}-1)}\right) \\ &= d_{i-1} \cdot (1+o(1/\log^2 \Delta)) \left(1 - e^{-2}(1+o(1/\log \Delta))/(1-1/p_{i-1})^2\right) && \text{By (*)} \\ &= d_{i-1} \cdot (1+o(1/\log^2 \Delta)) \left(1 - e^{-2}(1+o(1/\log \Delta))\right) \\ &= d_{i-1} \cdot (1 \pm o(1/\log \Delta))(1-e^{-2}). \end{aligned}$$

$$\begin{aligned} t_i &= t_{i-1} \cdot (1+\delta_{i-1}) \left(1 - \frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}\right) \left(1 - (1-1/p_{i-1})^{2t_{i-1}}\right) \\ &= t_{i-1} \cdot (1+o(1/\log^2 \Delta)) \left(1 - e^{-2}(1 \pm o(1/\log \Delta))\right)^2 && \text{By (**)} \\ &= t_{i-1} \cdot (1 \pm o(1/\log \Delta))(1-e^{-2})^2. \end{aligned}$$

$$\begin{aligned} p_i &= p_{i-1} \cdot (1-\delta_{i-1}) \left(1 - \frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}\right)^2 \\ &= p_{i-1} \cdot (1-o(1/\log^2 \Delta)) \left(1 - e^{-2}(1 \pm o(1/\log \Delta))\right)^2 && \text{By (**)} \\ &= p_{i-1} \cdot (1 \pm o(1/\log \Delta))(1-e^{-2})^2. \end{aligned}$$

Finally, we derive the last equation in Lemma 3: an estimate of the second-order error β_i.

$$\begin{aligned} \beta_i &= \frac{p_i}{t_i} - 1 \\ &= \frac{(1-\delta_{i-1})p^{\diamond}_{i-1}}{(1+\delta_{i-1})t^{\diamond}_{i-1}} - 1 \\ &= (1-O(\delta_{i-1})) \cdot \frac{p_{i-1}}{t_{i-1}} \cdot \frac{1 - \frac{t_{i-1}}{p_{i-1}}(1-1/p_{i-1})^{2t_{i-1}}}{1-(1-1/p_{i-1})^{2t_{i-1}}} - 1 && \text{Definition of } p^{\diamond}_{i-1} \text{ and } t^{\diamond}_{i-1} \\ &= (1-O(\delta_{i-1})) \cdot \frac{\frac{p_{i-1}}{t_{i-1}} - (1-1/p_{i-1})^{2t_{i-1}}}{1-(1-1/p_{i-1})^{2t_{i-1}}} - 1 \\ &= \frac{\left(\frac{p_{i-1}}{t_{i-1}} - 1\right) + O(\delta_{i-1})\left(-\frac{p_{i-1}}{t_{i-1}} + (1-1/p_{i-1})^{2t_{i-1}}\right)}{1-(1-1/p_{i-1})^{2t_{i-1}}} \\ &= \frac{\left(\frac{p_{i-1}}{t_{i-1}} - 1\right) + O(\delta_{i-1})\left(-\frac{p_{i-1}}{t_{i-1}} + (1-1/p_{i-1})^{2t_{i-1}}\right)}{1-e^{-2}(1+o(1/\log \Delta))} && \text{by (*)} \\ &= \frac{\beta_{i-1} - O(\delta_{i-1})}{(1-e^{-2})(1-o(1/\log \Delta))} && -\frac{p_{i-1}}{t_{i-1}} + (1-1/p_{i-1})^{2t_{i-1}} = -\Theta(1) \\ &= \frac{\beta_{i-1}(1-o(1/\log^2 \Delta))}{(1-e^{-2})(1-o(1/\log \Delta))} && \delta_{i-1} = o(1/\log^2 \Delta) \\ &= \beta_{i-1} \cdot (1 \pm o(1/\log \Delta))/(1-e^{-2}). \end{aligned}$$

5 DISTRIBUTED LOVÁSZ LOCAL LEMMA ON TREES

Tree-structured Dependency Graphs. In this section, we study the distributed LLL on tree-structured dependency graphs, which we define as follows. Let T be a tree. Each vertex v holds

some variables $\mathcal{V}(v)$ and is associated with a bad event $E(v)$ that depends only on variables within distance $r/2$ of v; that is, $\text{vbl}(E(v)) = \bigcup_{u \in N^{r/2}(v)} \mathcal{V}(u)$. If S is a subset of the vertices, then we use $\text{vbl}(S)$ to be short for $\bigcup_{v \in S} \text{vbl}(E(v)) = \bigcup_{v \in S} \bigcup_{u \in N^{r/2}(v)} \mathcal{V}(u)$. We assume that r is a constant, and we do not analyze the dependence on r in the time complexity.

The dependency graph for the set of bad events $\mathcal{E}$ is exactly T^r, which is the graph obtained by adding edges to all pairs of vertices of distance at most r in T. Thus, the maximum degree of the dependency graph is at most Δ^r, where Δ is the maximum degree of T. We fix the parameter $d = \Delta^r$.

The tree-structured dependency graphs (with parameter r) arise naturally from any $r/2$-time RandLOCAL experiment that is run on a tree T. Throughout this section, we assume $r/2 \geq 1$ is an integer and that $\Delta \geq 3$.

5.1 Deterministic LLL Algorithm

Network Decomposition. A (λ, γ)*-network decomposition* is a partition of the vertex set into $V_1, \ldots, V_\lambda$ such that each connected component induced by each V_i has diameter at most γ. Fischer and Ghaffari [30] showed that given a (λ, γ)-decomposition of $G_{\mathcal{E}}^2$, an LLL instance satisfying $p(ed)^\lambda < 1$ is solvable in $O(\lambda(\gamma + 1))$ time. We use a slight generalization of standard network decompositions. A $(\lambda_1, \gamma_1, \lambda_2, \gamma_2)$-network decomposition is a partition of the vertices into $V_1, \ldots, V_{\lambda_1}, U_1, \ldots, U_{\lambda_2}$ such that connected components induced by V_i have diameter at most γ_1 and those induced by U_i have diameter at most γ_2.

Strong and Weak Diameter. For a subgraph $H = (V', E')$ of G, its *weak* diameter is defined as $\max_{u,v \in V'} \text{dist}_G(u, v)$, whereas its *strong* diameter is $\max_{u,v \in V'} \text{dist}_H(u, v)$. Though strong diameter is used in the above definition of a (λ, γ)-network decomposition, we remark that weak diameter suffices for the purpose of applying Lemma 6.

Lemma 6 (Fischer and Ghaffari [30]). *Suppose that a $(\lambda_1, \gamma_1, \lambda_2, \gamma_2)$-network decomposition of $G_{\mathcal{E}}^2$ is given. Any LLL instance on $G_{\mathcal{E}}$ satisfying $p(ed)^{\lambda_1+\lambda_2} < 1$ can be solved in* DetLOCAL *in $O(\lambda_1(\gamma_1 + 1) + \lambda_2(\gamma_2 + 1))$ time.*

The proof of Theorem 5 is based on the network decompositions for trees found in Section 6. A *distance-d dominating set* of a graph G is a vertex set S such that for each vertex v in the graph G, there exists $u \in S$ such that $\text{dist}(u, v) \leq d$.

Theorem 5. *Any tree-structured LLL satisfying $p(ed)^\lambda < 1$ with $\lambda \geq 2$ can be solved in* DetLOCAL *in $O(\max\{\log_\lambda s, \frac{\log s}{\log\log s}\} + \log^* n)$ time, where $s \leq n$ is the size of any distance-$O(1)$ dominating set of the tree T.*

Proof. Recall that the dependency graph is T^r for some tree T and constant r. In Section 6, we show that a standard $(2, O(\log s))$-decomposition for $(T^r)^2 = T^{2r}$ is computable in $O(\log s + \log^* n)$ time, and if $\lambda = \Omega(1)$ is sufficiently large, a $(1, O(\log_\lambda s), O(\lambda^2), 0)$-decomposition for T^{2r} is computable in $O(\log_\lambda s + \log^* n)$ time, i.e., one part of the partition has diameter $O(\log_\lambda s)$ and the remaining graph is properly $O(\lambda^2)$-colored.

If we want to use Lemma 6 to solve the given LLL instance satisfying $p(ed)^\lambda < 1$, then we need a $(\lambda_1, \gamma_1, \lambda_2, \gamma_2)$-network decomposition of T^{2r} satisfying $\lambda_1 + \lambda_2 \leq \lambda$, i.e., the number of parts is at most λ.

When $\lambda = O(1)$ is sufficiently small, we apply Lemma 6 with the first network decomposition. Because the decomposition has two parts, this works with LLL criterion $p(ed)^\lambda < 1$ for any $\lambda \geq 2$. The resulting LLL algorithm takes time $O(\log s + \log^* n)$.

When λ is sufficiently large, we compute a $(1, O(\log_{\hat{\lambda}} s), O(\hat{\lambda}^2), 0)$-decomposition in $O(\log_{\hat{\lambda}} s + \log^* n)$ time, where $\hat{\lambda}$ is chosen as the largest number such that $\hat{\lambda} \leq \sqrt{\frac{\log s}{\log\log s}}$ and the number parts $\lambda_1 + \lambda_2 = O(\hat{\lambda}^2)$ in the decomposition is at most λ. We have $\hat{\lambda} = \min\{O(\sqrt{\lambda}), \sqrt{\frac{\log s}{\log\log s}}\}$. We solve the LLL by applying Lemma 6, which takes time $O(\hat{\lambda}^2 + \log_{\hat{\lambda}} s + \log^* n) = O(\max\{\log_\lambda s, \frac{\log s}{\log\log s}\} + \log^* n)$. Observe that because of the $\hat{\lambda}^2$ term, we cannot benefit from LLL instances with $\lambda \gg \frac{\log s}{\log\log s}$. □

Notice that the time bound for Theorem 5 is in terms of s rather than n. We will apply Theorem 5 after performing a graph shattering step, the output of which creates many disjoint tree-structured instances with size $\Delta^{O(1)} \cdot O(\log n)$, each of them admitting a distance-$O(1)$ dominating set of size at most $s = O(\log n)$. We want the time bound to be in terms of $s = O(\log n)$, independent of Δ.

For a given LLL instance with criterion $p(ed)^\lambda < 1$, the shattering routine of Fischer and Ghaffari [30] achieves the above requirement in time $O(d^2 + \log^* n)$ in such a way that the resulting LLL instances after the shattering routine satisfy the criterion $p(ed)^{\lambda/2} < 1$. If we combine this with Theorem 5, then we obtain a $O(d^2 + \max\{\log_\lambda \log n, \frac{\log\log n}{\log\log\log n}\})$-time RandLOCAL LLL algorithm for criterion $p(ed)^\lambda < 1$, $\lambda \geq 4$, which is efficient only when d is small. Notice that we need $\lambda/2 \geq 2$ to apply Theorem 5 on LLL instances with criterion $p(ed)^{\lambda/2} < 1$.

In Section 5.2, we present a new method (Lemma 7) for computing a partial assignment to the variables that effectively shatters a large dependency graph into many independent subproblems, each satisfying a polynomial LLL criterion w.r.t. the unassigned variables.

5.2 Randomized LLL Algorithm

Consider a tree-structured LLL instance T^r with LLL criterion $p(ed)^\lambda < 1$. In subsequent discussion, unless otherwise stated, the underlying graph is, by default, assumed to be T. Our shattering routine will work towards finding a *good* partial assignment.

Definition 1. A partial assignment ϕ to the variables in the LLL system is *good* if it satisfies the following two properties.

(1) Conditioned on the partial assignment ϕ, the probability of any bad event $E(v)$ is at most $p' = \sqrt{p}$.
(2) Let V' be the set of all vertices v such that $\mathrm{vbl}(E(v))$ contains some unassigned variables. Each connected component C induced by V' has size at most $\Delta^{O(1)} \cdot O(\log n)$, and C contains a distance-$2r$ dominating set with size at most $O(\log n)$.

Due to Definition 1(1), conditioned on a good partial assignment ϕ, the bad events in each connected component C induced by V' form an LLL system with the LLL criterion $p'(ed)^{\lambda/2} < 1$. Definition 1(2) guarantees that each component is of small size. Thus, a good partial assignment ϕ is able to shatter the tree T into small components, each of which is an independent LLL system. In Sections 5.3–5.5, we prove the following efficient "shattering lemma."

Lemma 7. *Suppose we are given a tree-structured LLL instance T^r satisfying LLL criterion $p(ed)^\lambda < 1$, where $\lambda \geq 2(4^r + 8r)$. There is a* RandLOCAL *algorithm that computes a good partial assignment ϕ in $O(\log_\lambda \log n)$ time.*

The overall algorithm is obtained by composing Lemma 7 and Theorem 5, which is summarized in Theorem 6. In particular, the algorithm has the usual two-phase graph shattering structure.

Shattering. Given the LLL instance with dependency graph T^r, find a *good* partial assignment ϕ' using Lemma 7. Each component induced by events having at least one unset variable has size $\text{poly}(\Delta) \cdot O(\log n)$ and contains a distance-$2r$ dominating set with size $O(\log n)$. Moreover, each such component is an LLL instance with parameters d and $p' = \sqrt{p}$ satisfying criterion $p'(ed)^{\lambda/2} < 1$.

Post-shattering. We extend ϕ' to a total assignment by independently fixing the variables in each component of the shattered LLL instance. By Theorem 5, this can be done in $O(\max\{\log_{\lambda/2} s, \frac{\log s}{\log\log s}\})$ time, where in our case $s = O(\log n)$.

THEOREM 6. *Let T^r be a tree-structured LLL instance satisfying criterion $p(ed)^\lambda < 1$ with $\lambda \geq 2(4^r + 8r)$. This LLL can be solved in* RandLOCAL *in $O(\max\{\log_\lambda \log n, \frac{\log\log n}{\log\log\log n}\})$ time.*

We briefly overview the ideas behind the proof of Lemma 7. The goal is to design an algorithm to compute a good partial assignment ϕ. Consider the following process. First, draw a total assignment ϕ to $\mathcal{V}$ according to the distribution of the variables in the underlying LLL instance. If any bad event $E(v)$ occurs under ϕ, then update ϕ by *unsetting* all variables in $\text{vbl}(E(v))$. More generally, 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. At the end of the contagion process, we obtain a partial assignment satisfying Definition 1(1).

If this contagion process were actually simulated, then it would take $\Omega(\log n)$ parallel steps to reach a stable state, which is too slow. We will provide a different method to compute a stable state (i.e., a partial assignment satisfying Definition 1(1)) that is exponentially faster, by avoiding a direct simulation.

The proof of Lemma 7 appears at the end of Section 5.4. It uses Lemma 9, which concerns the problem of finding a stable state in a contagion process, and Lemma 10, which connects the problem of shattering a dependency graph T^r to a contagion played out on T^r.

5.3 Criterion for Infection

Let u be a vertex in the undirected tree T. Then $T - \{u\}$ consists of $\deg(u)$ subtrees $T_1, \ldots, T_{\deg(u)}$; we call T_k the *kth subtree* of u. Define $C_u(k, [i, j])$ to be the set of vertices in the kth subtree of u whose distance to u lies in the interval $[i, j]$. For example, $C_u(k, [1, 1])$ only contains the kth neighbor of u. For any vertex set S, define $\widehat{\deg}_S(u)$ as follows:

$$\widehat{\deg}_S(u) = |\{k \ : \ C_u(k, [1, r]) \cap S \neq \emptyset\}| .$$

In other words, it is the number of *distinct* subtrees of u containing at least one S-vertex within distance r.

Let $\mu \geq 4$ and $\lambda' \geq 1$ be two integers such that $\lambda \geq 2(\mu^r + \lambda')$. The following bad events $B(S, v)$ and $B(v)$ are defined w.r.t. the following process. First, we fix a total assignment ϕ to the variables, then progressively add vertices to the set S. All variables in $\text{vbl}(S)$ are considered *unset*; for example, conditioning on "$\text{vbl}(E(v))\setminus \text{vbl}(S)$" means keeping ϕ's assignment to $\text{vbl}(E(v))\setminus \text{vbl}(S)$ and *resampling* $\text{vbl}(S)$ according to their distribution in the underlying LLL instance:

$$B(S, v) : \left[\Pr\left[E(v) \mid \text{vbl}(E(v))\setminus \text{vbl}(S)\right] \geq (ed)^{-\lambda/2} \right],$$

$$B(v) : \left[\bigcup_{S \subset N^r(v),\ |S| \leq \mu^r} B(S, v) \right].$$

In other words, $B(S, v)$ is the event that, if we *were* to resample $\text{vbl}(S)$, then the probability that $E(v)$ occurs is at least $(ed)^{-\lambda/2}$. The event $B(v)$ occurs if it is *possible* to find a subset S of cardinality at most μ^r such that $B(S, v)$ occurs.

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

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

By a union bound over the $|N^r(v)|^{\mu^r} \leq d^{\mu^r}$ choices of S (recall that $d = \Delta^r$),

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

Intuitively, $B(v)$ is the event that $E(v)$ is *too close* to happening. That is, relatively few variables need to be resampled to give $E(v)$ a likely probability of happening. Lemma 8 shows that the criterion for infection "$\widehat{\deg}_S(v) > \mu$" is a good proxy for the harder-to-analyze criterion "$E(v)$ is too close to happening."

Lemma 8. *Fix a total variable assignment ϕ. Let S be any vertex set such that, for each vertex v, if $B(v)$ occurs under ϕ or $\widehat{\deg}_S(v) > \mu$, then v must be in S. Then* $\Pr[E(v) \mid \text{vbl}(E(v)) \setminus \text{vbl}(S)] < (ed)^{-\lambda/2}$ *for each vertex v.*

Proof. If $v \in S$, then the probability of seeing $E(v)$ after resampling $\text{vbl}(S)$ is, according to the original LLL criterion, at most $p < (ed)^{-\lambda}$. In what follows, we assume $v \notin S$.

To prove the lemma, it suffices to show that there exists a vertex set S' such that (i) $S' \subset N^r(v)$, (ii) $|S'| \leq \mu^r$, and (iii) $\text{vbl}(S') \cap \text{vbl}(E(v)) = \text{vbl}(S) \cap \text{vbl}(E(v))$. Notice that (iii) implies that resampling $\text{vbl}(S')$ is equivalent to resampling $\text{vbl}(S)$ from v's point of view. Since $v \notin S$, by assumption, event $B(v)$ does not occur. Since $|S'| \leq \mu^r$, event $B(S', v)$ does not occur. Hence, $\Pr[E(v) \mid \text{vbl}(E(v)) \setminus \text{vbl}(S')] < (ed)^{-\lambda/2}$, as desired.

Root the tree at v. We call a vertex $u \in S$ "highest" if u is in $N^r(v)$ and no ancestor of u is in S. Observe that if H is the set of highest vertices, then $\text{vbl}(S) \cap \text{vbl}(E(v)) = \text{vbl}(H) \cap \text{vbl}(E(v))$. To see this, observe that if $u' \in S$ is not highest, and is a descendant of some highest $u \in S$, that $\text{vbl}(E(u')) \cap \text{vbl}(E(v))$ is contained in $\text{vbl}(E(u)) \cap \text{vbl}(E(v))$.

Thus, we only need to bound $|H|$ by μ^r. Suppose, for the sake of contradiction, that $|H| \geq \mu^r + 1$. Define the path $(v = v_0, v_1, \ldots, v_r)$ by selecting v_i as the child of v_{i-1} that maximizes the number of vertices in H contained in the subtree rooted at v_i. We prove by induction that the subtree rooted at v_i contains at least $\mu^{r-i} + 1$ H-vertices. The base case $i = 0$ holds by assumption. If there are $\mu + 1$ subtrees of v_i containing H-vertices, then v_i would be infected. Thus, by the pigeonhole principle, the number of H-vertices in the subtree rooted at v_{i+1} must be at least $\lceil(\mu^{r-i} + 1)/\mu\rceil = \mu^{r-(i+1)} + 1$. Hence, the subtree rooted at v_r contains $\mu^0 + 1 = 2$ H-vertices; this is a contradiction, since the only vertex in this subtree eligible to be in H is v_r itself. □

5.4 Contagion Process

A (q_0, r, μ)-*contagion process* on an n-vertex tree T is played out as follows. Initially, each vertex is infected with probability q_0, and these events are independent for vertices at distance greater than r. If S is the set of infected vertices at some time and $\widehat{\deg}_S(v) > \mu$, then v becomes infected. In this section our goal is, given the initially infected vertices, to compute a superset of those vertices that is *stable* and *small*, defined as follows.

Definition 2. Let S_0 be the initially infected vertices and $S \supset S_0$.

- S is called *stable* if it causes no more infection.
- S is called *small* if each connected component induced by $\bigcup_{v\in S} N^r(v)$ contains a distance-$2r$ dominating set of size at most $O(\log n)$.

In Lemma 9, we show that one can efficiently compute a set S that is both *stable* and *small*.

Lemma 9. *Consider a (q_0, r, μ)-contagion process played on an n-vertex tree T with maximum degree Δ. There is a* RandLOCAL *algorithm that computes a small stable set S in $O(\log_\mu \log n)$ time, where r is constant, $q_0 \le (ed)^{-8r}$, $d = \Delta^r$, and $\mu \ge 4$.*

The proof of Lemma 9 is deferred to Section 5.5. Lemma 10 connects the contagion problem to finding a good partial assignment.

Lemma 10. *Suppose there is a τ-round* RandLOCAL *algorithm for finding a small stable set S for a $((ed)^{-\lambda'}, r, \mu)$-contagion process. Then there exists a $(\tau + O(1))$-round* RandLOCAL *algorithm for finding a good partial assignment ϕ to a tree-structured LLL instance with criterion $p(ed)^\lambda < 1$, where $\lambda \ge 2(\mu^r + \lambda')$.*

Proof. Let $q_0 = (ed)^{-\lambda'}$. Consider the (q_0, r, μ)-contagion process defined by choosing a random assignment ϕ' to the variables in the LLL system and initially infecting all vertices v such that $B(v)$ occurs. The lower bound on λ implies $\Pr[B(v)] \le q_0 = (ed)^{-\lambda'}$. Given the small stable set S, we let ϕ be the result of unassigning all variables in $\mathrm{vbl}(S) = \bigcup_{v\in S} \mathrm{vbl}(E(v)) = \bigcup_{v\in S} \bigcup_{u\in N^{r/2}(v)} \mathcal{V}(u)$.

We now verify that ϕ is a good partial assignment. Since S is stable, for each vertex v, if $B(v)$ occurs under ϕ or $\widehat{\deg}_S(v) > \mu$, then v must be in S. By Lemma 8, $\Pr[E(v) \mid \mathrm{vbl}(E(v)) \setminus \mathrm{vbl}(S)] < (ed)^{-\lambda/2} < \sqrt{p}$ for each vertex v, and so Definition 1(1) is satisfied. Let $V' = \bigcup_{v\in S} N^r(v)$ be the set of all vertices v such that $\mathrm{vbl}(E(v))$ contains some unassigned variables. Since S is small, each connected component C induced by V' contains a distance-$2r$ dominating set with size at most $O(\log n)$. Since $2r = O(1)$, the cardinality of C is at most $\mathrm{poly}(\Delta) \cdot O(\log n)$. Hence, Definition 1(2) is also satisfied. □

We are now in a position to prove Lemma 7.

Proof. Recall that the LLL criterion of in Lemma 7 is $\lambda \ge 2(4^r + 8r)$. We pick the largest *even* integer μ such that $\lambda \ge 2(\mu^r + 8r)$, and we set $\lambda' = 8r$. Notice that $\mu \ge 4$ and $\log \mu = \Theta(\log \lambda)$. By Lemma 9, a small stable set S for the $((ed)^{-8r}, r, \mu)$-contagion process can be computed in $O(\log_\mu \log n) = O(\log_\lambda \log n)$ time. By Lemma 10, this implies a $O(\log_\lambda \log n)$-time RandLOCAL algorithm to finding a good partial assignment ϕ under the LLL criterion $p(ed)^\lambda < 1$. □

5.5 Finding a Small Stable Set

We prove Lemma 9 in this section. The algorithm for Lemma 9 simulates a more virulent contagion process for τ steps using threshold $\mu/2$ rather than μ, then simulates a reverse-contagion for τ steps, where vertices become *uninfected* if they were not initially infected and they have nearby infected vertices in at most μ subtrees. We prove that when $\tau = \Theta(\log_\mu \log n)$, the final infected set $S = L_\tau$ is both stable and small. This process is called Find-Small-Stable-Set. The sets generated by this process satisfy that $U_0 \subseteq \cdots \subseteq U_\tau = L_0 \supseteq \cdots \supseteq L_\tau$.

Find-Small-Stable-Set.

(1) $U_0 \leftarrow \{u \in V \mid u \text{ is initially infected}\}$. That is, $u \in U_0$ if $B(u)$ occurs initially.
(2) For $1 \le i \le \tau$, do $U_i \leftarrow U_{i-1} \cup \{u \in V \mid \widehat{\deg}_{U_{i-1}}(u) > \mu/2\}$.
(3) $L_0 \leftarrow U_\tau$.
(4) For $1 \le i \le \tau$, do $L_i \leftarrow L_{i-1} \setminus \{u \in L_{i-1} \setminus U_0 \mid \widehat{\deg}_{L_{i-1}}(u) \le \mu\}$.
(5) Return L_τ.

We show that $S = L_\tau$ is stable in Lemma 15. Let $L_{\tau+1}$ be the set of all vertices u such that $\widehat{\deg}_{L_\tau}(u) > \mu$. Our goal is to show that if $u \notin L_\tau$, then $\widehat{\deg}_{L_\tau}(u) \le \mu$ (i.e., $u \notin L_{\tau+1}$) with high probability.

Root T at an arbitrary vertex, and let T' refer to the rooted version. Define T'_u to be the subtree of T' rooted at u, and define $C'_u(k,[i,j])$ as $C_u(k,[i,j]) \cap T'_u$. Given a vertex set W, define $\deg'_W(u)$ as the number of different k such that $C'_u(k,[1,r]) \cap W \neq \emptyset$. Although the original contagion process is played on T, it is easier to analyze a similar process played on T', where only descendants can cause a vertex to become infected.

In general, if $\{X(u)\}_{u\in V}$ is an ensemble of events associated with vertices and W a subset of vertices, then we write $X(W)$ to denote the event $\bigcup_{u\in W} X(u)$; i.e., there exists $u \in W$ such that $X(u)$ occurs. We write X to denote the set of vertices $\{u \in V \mid X(u) \text{ occurs}\}$. For any two events A and B, we write $A \Rightarrow B$ to denote $A \subseteq B$, i.e., A implies B. With respect to a vertex u, consider the following three sequences of events:

$(F_i(u))$: for each $0 \le i \le \tau$, let $F_i(u)$ be $(u \notin U_i) \wedge (u \in L_{i+1})$.

$(H_i(u))$: let $H_0(u)$ be $(u \in U_0)$; for each $0 \le i < \tau$, let $H_{i+1}(u)$ be $H_0(u) \vee (\deg'_{H_i}(u) \ge \mu/2)$.

$(\tilde{F}_i(u))$: let $\tilde{F}_0(u)$ be $H_\tau(u)$; for each $0 \le i < \tau$, let $\tilde{F}_{i+1}(u)$ be $\deg'_{\tilde{F}_i}(u) \ge \mu/2$.

Lemma 11. *No vertex can belong to both $U_\tau \setminus L_\tau$ and $L_{\tau+1}$.*

Proof. Suppose there were such a vertex u. If $u \in L_{\tau+1}$, then it must have more than μ neighbors in L_τ, which were also in $L_{\tau-1} \subseteq \cdots \subseteq L_0 = U_\tau$. But if $u \in U_\tau$, then it would also remain in $L_0, \ldots, L_\tau$, contradicting the assumption that $u \in U_\tau \setminus L_\tau$. □

By Lemma 11, to prove that $S = L_\tau$ is stable, it suffices to prove that

$$\Pr[F_\tau(u)] = \Pr[(u \notin L_\tau) \wedge (u \in L_{\tau+1})] = 1/\mathrm{poly}(n).$$

Lemma 12 connects the true contagion process on T to an imagined one played on T'.

Lemma 12. *For each vertex u in T, and for each $0 \le i \le \tau$, we have $F_i(u) \Rightarrow \tilde{F}_i(u)$.*

Proof. We first show that $(u \in U_i) \Rightarrow H_i(u)$, for each $0 \le i \le \tau$. The base case ($i = 0$) follows from the definition of $H_0(u)$. Assume by inductive hypothesis that $(u \in U_{i-1}) \Rightarrow H_{i-1}(u)$. We have

$$(u \in U_i \setminus U_0) \Rightarrow \left(\widehat{\deg}_{U_{i-1}}(u) > \mu/2\right) \Rightarrow \left(\deg'_{U_{i-1}}(u) \ge \mu/2\right) \Rightarrow \left(\deg'_{H_{i-1}}(u) \ge \mu/2\right).$$

This implies $(u \in U_i) \Rightarrow H_i(u)$, since $(u \in U_0) \Rightarrow H_0(u) \Rightarrow H_i(u)$.

Next, we prove by induction that $F_i(u) \Rightarrow \tilde{F}_i(u)$, for each $0 \le i \le \tau$. The base case $i = 0$ follows from the above result:

$$F_0(u) \Rightarrow (u \in L_1) \Rightarrow (u \in L_0 = U_\tau) \Rightarrow H_\tau(u) \Rightarrow \tilde{F}_0(u).$$

Assume inductively that $F_{i-1}(u) \Rightarrow \tilde{F}_{i-1}(u)$. Let u be any vertex in $L_{i+1} \setminus U_i$; i.e., the event $F_i(u)$ occurs. Since $u \notin U_i \supseteq U_0$, the only way Find-Small-Stable-Set could put $u \in L_{i+1} \setminus U_i$ is if

$$\widehat{\deg}_{L_i}(u) > \mu,$$
$$\text{and } \widehat{\deg}_{U_{i-1}}(u) \le \mu/2,$$

which implies

$$\widehat{\deg}_{F_{i-1}}(u) = \widehat{\deg}_{L_i}(u) - \widehat{\deg}_{U_{i-1}}(u) > \mu/2, \text{ and hence}$$

and hence

$$\deg'_{F_{i-1}}(u) \ge \mu/2.$$

By inductive hypothesis, we have

$$\left(\deg'_{F_{i-1}}(u) \geq \mu/2\right) \Rightarrow \left(\deg'_{\tilde{F}_{i-1}}(u) \geq \mu/2\right) \Rightarrow \tilde{F}_i(u),$$

which completes the induction. □

For brevity, define $p_i = \max_u \Pr[\tilde{F}_i(u)]$ and $q_i = \max_u \Pr[H_i(u)]$. We prove two auxiliary lemmas.

Lemma 13. $p_\tau \leq (\Delta^{2((r^2/2)+1)} p_0)^{(\frac{\mu}{2})^{\tau/(r/2)}}$.

Proof. Suppose that u is a vertex such that $\tilde{F}_i(u)$ occurs. Then, by definition of $\tilde{F}_i(u)$, there exist $\mu/2$ different indices k such that $\tilde{F}_{i-1}(C'_u(k,[1,r]))$ occurs. A consequence of this observation is that

$$\tilde{F}_{i-1}(C'_u(k,[1,r])) \Rightarrow \tilde{F}_{i-2}(C'_u(k,[2,2r])) \Rightarrow \tilde{F}_{i-3}(C'_u(k,[3,3r])) \cdots \Rightarrow \tilde{F}_{i-(r/2)}(C'_u(k,[r/2,r^2/2])).$$

Therefore, if $\tilde{F}_i(u)$ occurs, then there must exist $\mu/2$ indices k such that $\tilde{F}_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))$ occurs. The $\mu/2$ events $\{\tilde{F}_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))\}$ are independent, since $\tilde{F}_i(v)$ depends only on $\mathrm{vbl}(T'_v) = \bigcup_{w \in N^{r/2}(v) \cup T'_v} \mathcal{V}(w)$. This independence property is one reason why it is easier to analyze a contagion on T' rather than T.

By a union bound over all vertices in $C'_u(k,[r/2,r^2/2])$, we have

$$\Pr\left[\tilde{F}_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))\right] \leq \Delta^{r^2/2-1} p_{i-(r/2)}.$$

Taking a union bound over at most $\binom{\Delta}{\mu/2}$ choices of $\mu/2$ distinct indices k, we infer that

$$p_i \leq \Delta^{\mu/2} \left(\Delta^{r^2/2-1} p_{i-(r/2)}\right)^{\mu/2} \leq \left(\Delta^{(r^2/2)} p_{i-(r/2)}\right)^{\mu/2}$$

for each $r/2 \leq i \leq \tau$. Assume τ is a multiple of $r/2$, and recall $\mu/2 \geq 2$. We can bound p_τ as follows:

$$p_\tau \leq p_0^{(\frac{\mu}{2})^{\tau/(r/2)}} \cdot \prod_{j=1}^{\tau/(r/2)} \left(\Delta^{(r^2/2)}\right)^{(\frac{\mu}{2})^j} \leq \left(\Delta^{r^2} p_0\right)^{(\frac{\mu}{2})^{\tau/(r/2)}}.$$ □

Lemma 14. $p_0 = q_\tau \leq \Delta^{r/2} q_0$.

Proof. Recall that $H_i(u)$ is $(u \in H_0) \vee (\deg'_{H_{i-1}}(u) \geq \mu/2)$. This implies that

$$H_{i-1}(C'_u(k,[1,r])) \Rightarrow H_0(C'_u(k,[1,r])) \vee H_{i-2}(C'_u(k,[2,2r])).$$

Repeating this $(r/2)-1$ times, $H_{i-1}(C'_u(k,[1,r]))$ implies that

$$H_0(C'_u(k,[1,r(r/2-1)]) \vee H_{i-(r/2)}(C'_u(k,[r/2,r^2/2])).$$

Since $H_0(C'_u(k,[1,r(r/2-1)]) \Rightarrow H_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))$, we conclude that

$$H_{i-1}(C'_u(k,[1,r])) \Rightarrow H_0(C'_u(k,[1,r/2-1]) \vee H_{i-(r/2)}(C'_u(k,[r/2,r^2/2])).$$

Thus, if $H_i(u)$ occurs, then either (i) $H_0(N^{r/2-1}(u))$ occurs, or (ii) there exist $\mu/2$ different indices k such that $H_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))$ occurs. The events $H_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))$ for all k are independent, since $H_i(v)$ depends only on $\mathrm{vbl}(T'_v) = \bigcup_{w \in N^{r/2}(v) \cup T'_v} \mathcal{V}(w)$.

By a union bound, $\Pr[H_{i-(r/2)}(C'_u(k,[r/2,r^2/2]))] \le \Delta^{r^2/2-1} q_{i-r/2}$. Suppose that τ is a multiple of $r/2$. Taking a union bound over at most $\binom{\Delta}{\mu/2}$ choices of $\mu/2$ distinct indices k, we have

$$\begin{aligned}
q_\tau &\le \Pr\left[H_0(N^{r/2-1}(u))\right] + \binom{\Delta}{\mu/2} \cdot \Delta^{r^2/2-1} q_{\tau-(r/2)} \\
&\le \Delta^{r/2-1} q_0 + \Delta^{\mu/2} \left(\Delta^{r^2/2-1} q_{\tau-(r/2)}\right)^{\mu/2} \\
&\le \Delta^{r/2-1} q_0 + \left(\Delta^{r^2/2} q_{\tau-(r/2)}\right)^{\mu/2} \\
&\le \Delta^{r/2-1} q_0 + q_0^{(\frac{\mu}{2})^{\tau/(r/2)}} \cdot \prod_{j=1}^{\tau/(r/2)} \left(\Delta^{r^2/2}\right)^{(\frac{\mu}{2})^j} \\
&\le \Delta^{r/2-1} q_0 + \left(\Delta^{2(r^2/2)} q_0\right)^{(\frac{\mu}{2})^{\tau/(r/2)}} && (\mu/2 \ge 2) \\
&\le \Delta^{r/2-1} q_0 + \left(\Delta^{2(r^2/2)} q_0\right)^{2} && ((\mu/2)^{\tau/(r/2)} \ge 2) \\
&\le \Delta^{r/2-1} q_0 + \Delta^{4(r^2/2)-8r^2} q_0 && (q_0 \le (ed)^{-8r} \text{ and } d = \Delta^r) \\
&\le \Delta^{r/2} q_0. && \square
\end{aligned}$$

We are now ready to prove that $S = L_\tau$ is stable.

Lemma 15. *For each vertex $u \notin L_\tau$, $\widehat{\deg}_{L_\tau}(u) \le \mu$ with high probability, and so L_τ is stable.*

Proof. It suffices to show that $\Pr[F_\tau(u)] = 1/\text{poly}(n)$. By Lemma 12, $\Pr[F_\tau(u)] \le \Pr[\tilde{F}_\tau(u)] = p_\tau$. We show that $p_\tau = 1/\text{poly}(n)$.

$$\begin{aligned}
p_\tau &\le \left(\Delta^{r^2} p_0\right)^{(\frac{\mu}{2})^{\tau/(r/2)}} && \text{(Lemma 13)} \\
&\le \left(\Delta^{r^2+r/2} q_0\right)^{(\frac{\mu}{2})^{\tau/(r/2)}} && \text{(Lemma 14)} \\
&\le \left(\Delta^{r^2+r/2-8r^2}\right)^{(\frac{\mu}{2})^{\tau/(r/2)}} && (q_0 \le (ed)^{-8r} \text{ and } d = \Delta^r) \\
&\le \left(\Delta^{-27}\right)^{(\frac{\mu}{2})^{\tau/(r/2)}} && (r \ge 2) \\
&\le \left(\Delta^{-27}\right)^{\Theta(\log n)} && (\tau = \Theta(\log_\mu \log n) \text{ and } r = O(1)) \\
&\le 1/\text{poly}(n). && \square
\end{aligned}$$

In Lemma 17, we prove that U_τ is small, which implies that $S = L_\tau \subseteq U_\tau$ is also small. We write $T^{[a,b]}$ to denote the graph defined by the vertex set $V(T)$ and the edge set $\{\{u,v\} \mid \text{dist}_T(u,v) \in [a,b]\}$. We first prove an auxiliary lemma.

Lemma 16. *Fix a $c \ge 1$. With probability $1 - n^{-\Omega(c)}$, the graph $H = T^{[r+1,4r]}$ has no connected subgraph D such that (i) $|D| \ge c \log n$, and (ii) there is a subset $D' \subseteq D \cap U_0$ containing at least half of the vertices in D, and $\text{dist}_T(u,v) > r$ for distinct $u, v \in D'$.*

Proof. The proof is similar to that of Reference [13, Lemma 3.3]. Suppose that such D exists, and consider a tree $\hat{T}$ in H spanning D. There are at most $4^{c \log n}$ different rooted unlabeled $c \log n$-node trees; and each of them can be embedded into H in less that $n \cdot \Delta^{4r(c \log n - 1)}$ ways. Moreover, there are at most $2^{c \log n}$ ways of selecting a subset $D' \subseteq D$. Since $|D'| \ge c \log n/2$ and $\text{dist}_T(u,v) > r$ for distinct $u, v \in D'$, the probability that such $\hat{T}$ exists is at most $q_0^{c \log n/2}$.

Recall that $q_0 \le (ed)^{-8r}$, $d = \Delta^r$, $r \ge 2$, and $\Delta \ge 3$. A union bound over all possibilities of $\hat{T}$ implies that such D exists with probability at most

$$\begin{aligned} p' &= 4^{c\log n} \cdot n \cdot \Delta^{4r(c\log n - 1)} \cdot 2^{c\log n} \cdot q_0^{c\log n/2} \\ &\le n^{3c+1}\Delta^{-4c(r^2-r)\log n} e^{-4cr\log n} \\ &\le n^{(4-4(r^2-r)\log\Delta - 4\log e)c} \\ &\le n^{-14c}. \end{aligned}$$

□

Recall from Definition 2 that U_i is small if each connected component induced by $\bigcup_{v\in U_i} N^r(v)$ contains a distance-$2r$ dominating set of size at most $O(\log n)$.

Lemma 17. *With high probability, each connected component induced by $\bigcup_{v\in U_\tau} N^r(v)$ contains a distance-$2r$ dominating set of size at most $O(\log n)$, and so U_τ is small.*

Proof. Let C be any connected component induced by $\bigcup_{v\in U_\tau} N^r(v)$. We pick a distance-$2r$ dominating set D of C greedily, preferring vertices in U_0 over U_1, and U_1 over U_2, and so on. Each time a vertex v is picked, we remove from consideration all vertices in $N^r(v)$. Recall that $U_0 \subseteq \cdots \subseteq U_\tau$. The set D is obviously a distance-r dominating set of $U_\tau \cap C$. Since $U_\tau \cap C$ is itself a distance-r dominating set of C, the set D is a distance-$2r$ dominating set of C.

We write u_i to denote the ith vertex added to D, and define $D_i = \{u_1, \ldots, u_i\}$. Let m_i denote the number of connected components induced by D_i in the graph $T^{[r+1,2r]}$ (rather than T). We claim that if $u_i \notin U_0$, then $m_i < m_{i-1}$. This implies that at least half of the vertices in D belong to U_0. Observe that the set D is connected in $H = T^{[r+1,4r]}$ (since D is a distance-$2r$ dominating set of C), and so by Lemma 16, $|D| = O(\log n)$ with high probability.

We prove the above claim in the remainder of the proof. Consider the moment some $u_i \notin U_0$ is added to D. We will show that the connected component of D_i in the graph $T^{[r+1,2r]}$ that contains u_i is formed by merging u_i with at least two connected components of D_{i-1} in the graph $T^{[r+1,2r]}$.

The algorithm Find-Small-Stable-Set added u_i to U_j, because u_i had at least $\mu/2 \ge 2$ subtrees containing U_{j-1}-vertices that are within $N^r(u_i)$. Let T_1 and T_2 be any two such subtrees. For each $k = 1, 2$, let v_k be a U_{j-1}-vertex contained in both T_k and $N^r(u_i)$. Then there must be a vertex $w_k \in N^r(v_k)$ such that w_k has been already added to D, since otherwise the greedy algorithm should prefer v_k over u_i. Observe that w_1 and w_2 belong to separate connected components of D_{i-1} in the graph $T^{[r+1,2r]}$, since $u_i \notin N^r(w_1) \cup N^r(w_2)$; but w_1, w_2, and u_i are in the same component of D_i in the graph $T^{[r+1,2r]}$, since $w_k \in N^r(v_k) \subseteq N^{2r}(u_i)$, for both $k = 1, 2$. □

We have proven (Lemmas 15 and 17) that the algorithm Find-Small-Stable-Set computes a set $S = L_\tau$ that is *stable* and *small*, in $O(\log_\mu \log n)$ time. Lemma 10 shows that any such algorithm can be used to find a *good* partial assignment to the variables in any tree-structured LLL instance with $p(ed)^\lambda < 1$ and $\lambda \ge 2(4^r + 8r)$.[11] The *stability* criterion is used to show that the derived LLL instances satisfy $p'(ed)^{\lambda/2} < 1$ and $p' = \sqrt{p}$. The *smallness* criterion implies that the instances have size $\mathrm{poly}(\Delta)\log n$ and $\log n$-size, distance-$O(1)$ dominating sets. Because $\log\mu = \Theta(\log\lambda)$, the time to find the good partial assignment is $O(\log_\lambda \log n)$.

[11] It is possible to replace $2(4^r + 8r)$ with $2(4^r + cr)$ for some smaller c, but not too small. We do not attempt to optimize this coefficient.

6 NETWORK DECOMPOSITION OF TREES

Our interest in network decompositions stems from Lemma 6 due to Reference [30], which shows that they imply non-trivial *deterministic* LLL algorithms. Most work on network decompositions [57] has focussed on arbitrary graphs.

Recall that a (λ, γ)-network decomposition is a partition of the vertices into λ parts $V_1, \ldots, V_\lambda$ such that each V_i induces connected components with diameter at most γ; and a $(\lambda_1, \gamma_1, \lambda_2, \gamma_2)$-network decomposition is a partition of the vertices into $\lambda_1 + \lambda_2$ parts $V_1, \ldots, V_{\lambda_1}, U_1, \ldots, U_{\lambda_2}$ such that each V_i (respectively, U_i) induces connected components with diameter γ_1 (respectively, γ_2).

In this section, we present 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 to be given as input.

We emphasize that the network decomposition that we would like to compute is with respect to T^k, but the communication network is T.

6.1 A Simple Network Decomposition

We first design a simple decomposition that partitions any tree-structured graph T^k into 2 parts.

Theorem 7. *Let T be a tree containing a distance-d dominating set of size s. There is a* DetLOCAL *algorithm $\mathcal{A}$ that computes a $(2, O(\log s + d/k))$-network decomposition of T^k in $O(k \log s + d + k \log^* n)$ time, i.e., $O(\log s + \log^* n)$ time when $d = O(1)$ and $k = O(1)$.*

In what follows, we prove Theorem 7. We assume the underlying communications network is T rather than T^k. Consider the following two tree operations. They are similar to the ones described in Reference [21], which are inspired by Miller and Reif [50]. The second operation is parameterized by an integer $\ell \geq 2$. In our application, we set $\ell = \Theta(k)$.

Rake: Remove all leaves and isolated vertices.
Compress: Remove all vertices that belong to some path P such that (i) all vertices in P have degree at most 2, and (ii) the number of vertices in P is at least ℓ.

Let $\mathcal{A}'$ be the algorithm on the tree T defined as follows. (1) Do $3d + 1$ Rake operations; (2) repeat the following sequence $\log s$ times: perform one Compress and then $\ell - 1$ Rake operations.

Lemma 18. *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 T_v 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 $\text{size}(v) \leq 1$, observe that the height of the subtree T_v rooted at v is at most $3d$. Suppose the height of T_v is at least $3d + 1$, then there is a path P connecting v and a leaf that has at least $3d + 2$ vertices. We claim that for any distance-d dominating set S of T, we need to have $|S \cap T_v| \geq 2$. For each $u \in S \cap T_v$, u can dominate at most $2d + 1$ vertices in P, and so there must be at least one vertex x in P that is not dominated by u and its distance to v is at least d. To dominate x, we need another vertex in $S \cap T_v$, and so $|S \cap T_v| \geq 2$, contradicting the assumption $\text{size}(v) \leq 1$. Therefore, the entire subtree T_v (including v) must be removed after the initial $3d + 1$ Rake operations.

Consider the case $2^{i-1} < \text{size}(v) \leq 2^i$. By the inductive hypothesis, 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 T_v rooted at v. The set V' induces a path with one endpoint at v, since otherwise $\text{size}(v) > 2 \cdot 2^{i-1} = 2^i$. Let C be a connected component induced by vertices in V' that are not removed yet. If $|C| \geq \ell$, then all vertices in C are removed after 1 Compress. Otherwise, all vertices in C are removed after $\ell - 1$ Rake operations. □

In the following discussion, the notions of connected components and degrees are with respect to T. To compute a $(2, O(\log s + d/k))$-network decomposition of T^k, it suffices to compute a partition $V = V_1 \cup V_2$ meeting the following two conditions.

- (C1) For both labels $c \in \{1, 2\}$, any two vertices u and v in two distinct connected components of V_c must have $\text{dist}_T(u, v) > k$. This guarantees that the set of connected components of V_c remains unaltered if we change the underlying graph from T to T^k.
- (C2) For both labels $c \in \{1, 2\}$, each connected component of V_c has diameter at most $O(k \log s + d)$. This implies the diameter upper bound of $O(\log s + d/k)$ when the underlying graph is T^k.

Recall that $\mathcal{A}'$ performs $L_r = (3d + 1) + (\ell - 1) \log s$ Rake and $L_c = \log s$ Compress operations; let $L = L_r + L_c = (3d + 1) + \ell \log s$. We write U_i to denote the set of all vertices that are removed during the ith operation. We are now in a position to present the algorithm $\mathcal{A}$. The algorithm $\mathcal{A}$ begins by computing the decomposition $V = \bigcup_{i=1}^{L} U_i$ using $\mathcal{A}'$. Then, for $i = L$ down to 1, label all vertices $v \in U_i$ by $\{1, 2\}$ as follows.

Case 1. If the ith operation is Rake, then label U_i as follows. Let $v \in U_i$. For the case that v is of degree-1 in the subgraph induced by $\bigcup_{j=i}^{L} U_j$, let u be the unique neighbor of v in $\bigcup_{j=i}^{L} U_j$. If $u \notin U_i$, then v adopts the same label as u. Otherwise, $u \in U_i$ must also be of degree-1 in $\bigcup_{j=i}^{L} U_j$; we give both u and v the same label $c \in \{1, 2\}$. For the case that v is an isolated vertex of $\bigcup_{j=i}^{L} U_j$, we label v by any $c \in \{1, 2\}$.

Case 2. If the ith operation is Compress, then label U_i as follows. Let P be a path that is a connected component of U_i. The number of vertices in P is at least $\ell = \Theta(k)$. Compute a labeling of the vertices in P meeting the following conditions: (i) each connected component induced by vertices of the same label has size within $[k, 7k]$, (ii) if v is an endpoint of P that is adjacent to a vertex $u \in \bigcup_{j=i+1}^{L} U_j$, then the label of v is the same as the label of u.

Such a labeling of P can be computed in $O(k)$ time if we are given an independent set I of P such that each connected component of $P \setminus I$ has size within $[3k, 6k]$. Suppose that we already have such a set I. For each $v \in I$, we find an arbitrary subpath $P_v \subseteq P$ that contains v and has exactly k vertices. All vertices in $\bigcup_{v \in I} P_v$ are labeled 1, and the remaining vertices in P are labeled 2. At this moment, each connected component induced by vertices of label 1 has size k, and each connected component induced by vertices of label 2 has size within $[3k - 2(k - 1), 6k] = [k + 2, 6k]$. If there is a component C violating Condition (ii) of the previous paragraph, then we flip the label of all vertices in C (i.e., from 1 to 2 or from 2 to 1). If $\ell \geq ck$ for some large enough universal constant c, then we obtain a labeling satisfying both Condition (i) and Condition (ii).

The computation of the independent set I can be done in $O(k \log^* n)$ time, as we explain below. Suppose that we have an independent set I' of P such that each connected component of $P \setminus I$ has size within $[\alpha, 2\alpha]$. We show that in $O(\alpha \log^* n)$ time we can compute an independent set I'' of P such that each connected component of $P \setminus I$ has size within $[\beta, 2\beta]$, for any prescribed number $\beta \leq 2\alpha + 1$. Let $\tilde{P}$ be the "imaginary path" formed by contracting all vertices in $P \setminus I$. A maximal independent set $\tilde{I}$ of $\tilde{P}$ can be computed in $O(\alpha \log^* n)$ time. At this point, each connected component C of $P \setminus \tilde{I}$ has size within $[2\alpha + 1, 4\alpha + 2]$. The component size constraint $[\beta, 2\beta]$ can be

met by adding new vertices to $\tilde{I}$ to subdivide the oversized components. The desired independent set I can be computed by $\log k$ iterated applications of the above procedure, and the runtime is $\sum_{i=1}^{\log k} O(2^i \log^* n) = O(k \log^* n)$.

Time Complexity. The total running time of $\mathcal{A}$ is $O(L_r + kL_c) + O(k \log^* n) = O(k \log s + d + k \log^* n)$, since the independent set computation of paths removed by the Compress operation can be computed in $O(k \log^* n)$ time *in parallel*.

Validity of Labeling. We now verify that the labeling resulting from $\mathcal{A}$ satisfies the two conditions (C1) and (C2). Consider two distinct connected components C and C' induced by V_1. In view of Case 2 of algorithm $\mathcal{A}$, any path P' connecting a vertex in C and a vertex in C' in T must contain a subpath P'' consisting of k vertices in V_2. The same is true if we swap V_1 and V_2, and so (C1) holds. Consider a connected component C by V_1 or V_2. Let $i^\star$ be the largest index i such that $U_i \cap C \neq \emptyset$, and let $v^\star$ be any vertex in $C \cap U_{i^\star}$. We show that for any vertex $u \in C$, the unique path P connecting u and $v^\star$ in T contains $O(L_r + kL_c) = O(k \log s + d)$ vertices, and so (C2) holds. Consider any index $i \in [1, i^\star]$. If the ith operation is Rake, then we have $|P \cap U_i| \leq 2$ (in view of Case 1). If the ith operation is Compress, then we have $|P \cap U_i| \leq 7k$ (in view of Case 2). Thus, indeed $|P| = O(L_r + kL_c)$.

6.2 A Mixed-Diameter Network Decomposition

In this section, we show how to compute a network decomposition where one part has diameter roughly $\log_\lambda s$ and the remaining portion of the graph is properly $O(\lambda^2)$-colored, i.e., they form $O(\lambda^2)$ parts with diameter zero. Here $\lambda = \Omega(k)$ is a sufficiently large parameter.

Theorem 8. *Let T be a tree containing a distance-d dominating set of size s. There is a* DetLOCAL *algorithm $\mathcal{A}$ that computes a $(1, O(\log_{\lambda/k} s + (d/k)), O(\lambda^2), 0)$-network decomposition of T^k in $O(k \log_{\lambda/k} s + d + k \log^* n)$ time, where $\lambda = \Omega(k)$ is sufficiently large, i.e., $\lambda \geq ck$ for some universal constant c. When $k = O(1)$ and $d = O(1)$ the time bound is $O(\log_\lambda s + \log^* n)$.*

In what follows, we prove Theorem 8. We write T_i to denote the set of vertices that are not removed during the first $i - 1$ tree operations. Consider the following two tree operations applied to T_i.

Rake: Remove all leaves and isolated vertices.
Compress: Remove all vertices v such that $|N^{2.5k}(v) \cap T_i| \leq \lambda$.

We set $m = \frac{\lambda}{2.5k} - 1$. Let $\mathcal{A}^*$ be the algorithm on the tree T defined as follows. (1) Do $3d + 1$ Rake operations; (2) repeat the following sequence $\log_m s$ times: do one Compress followed by $2.5k$ Rake operations.

Lemma 19. *Algorithm $\mathcal{A}^*$ removes all vertices in T.*

Proof. Let S be any size-s distance-d dominating set of T. Root T at an arbitrary vertex, and let $\text{size}(v)$ 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 $1 < \text{size}(v) \leq m^i$, then v is removed within the first i iterations in Step (2) of $\mathcal{A}^*$.

For the case of $\text{size}(v) \leq 1$, the height of the subtree rooted at v is at most $3d$, and so the entire subtree (including v) must be removed after $3d + 1$ Rake operations. For the case of $m^{i-1} < \text{size}(v) \leq m^i$, we assume by induction that all vertices u with $\text{size}(u) \leq m^{i-1}$ have been removed within the first $i - 1$ iterations of Step (2). Let v be any vertex with $\text{size}(v) \in (m^{i-1}, m^i]$, and define V' to be the set of all vertices u such that (i) $\text{size}(u) > m^{i-1}$, and (ii) u is in the subtree rooted at v.

Notice that all descendants of v other than those in V' have been removed within the first $i-1$ iterations of Step (2). Therefore, the set V' induces a subtree rooted at v having at most $m-1$ leaves. For those vertices $u \in V'$ with $\text{dist}_T(u,v) \geq 2.5k$, we have $|N^{2.5k}(u) \cap T_i| \leq m(2.5k)+1 \leq \lambda$, so they will be removed after one Compress. The rest of the vertices in V' will be removed during the next $2.5k$ Rake operations.

The above inequality $|N^{2.5k}(u) \cap T_i| \leq m(2.5k)+1$ can be derived as follows. Consider the subgraph induced by the vertices in V' that are within distance $2.5k$ to u. This subgraph can be seen as a tree rooted at u of height at most $2.5k$ with at most $(m-1)+1 = m$ leaves, which clearly has at most $m(2.5k)+1$ vertices. Notice that this analysis relies on the assumption that $u \in V'$ satisfies $\text{dist}_T(u,v) \geq 2.5k$, since otherwise $N^{2.5k}(u) \cap T_i$ may contain vertices that are ancestors of v. □

Now, we present our network decomposition algorithm $\mathcal{A}$. First, we run $\mathcal{A}^*$ on T. Then, for any vertex v removed by Compress, we mark all vertices in $N^{k/2}(v)$; i.e.,

$$\mathcal{M} = \{u \mid \exists v \text{ removed by Compress}, u \in N^{k/2}(v)\}$$

is the set of all marked vertices. We let $\tilde{T}$ be the graph defined as $V(\tilde{T}) = \mathcal{M}$, and $\{u,v\} \in E(\tilde{T})$ if $\text{dist}_T(u,v) \leq k$.

The $(1, O(k \log_{\lambda/k} s + d), O(\lambda^2), 0)$ network decomposition of T^k is computed by assigning color 0 to all unmarked vertices, and coloring the remaining vertices in $\tilde{T}$ with $\{1, \ldots, O(\lambda^2)\}$. We next show that (i) $\Delta(\tilde{T}) \leq \lambda$, and so the $O(\lambda^2)$-coloring can be computed using Linial's algorithm [48] in $O(k \log^* n)$ time, and (ii) each connected component induced by unmarked vertices (in T^k) has diameter $O(\log_{\lambda/k} s + (d/k))$. Thus, $\mathcal{A}$ indeed computes a $(1, O(\log_{\lambda/k} s + (d/k)), O(\lambda^2), 0)$-network decomposition of T^k in $O(k \log_{\lambda/k} s + d + k \log^* n)$ time.

Proof of (i). For any marked vertex v, we claim that $|N^k(v) \cap \mathcal{M}| \leq \lambda$ (in T), and so $\Delta(\tilde{T}) \leq \lambda$. Let u be the first vertex marked in $N^k(v)$. The vertex u is added to $\mathcal{M}$ due to the removal of a vertex $w \in N^{k/2}(u)$ in a Compress operation (it is possible that $u = w$). Suppose that w was removed in $i^\star$th tree operation. Then, we have $|N^{2.5k}(w) \cap T_{i^\star}| \leq \lambda$. We claim that $N^k(v) \cap \mathcal{M} \subseteq N^k(v) \cap T_{i^\star} \subseteq N^{2.5k}(w) \cap T_{i^\star}$, and this implies $|N^k(v) \cap \mathcal{M}| \leq \lambda$, and so $\Delta(\tilde{T}) \leq \lambda$. Since the $i^\star$th tree operation is the first iteration such that a vertex in $N^k(v)$ is marked due to the removal of another vertex during the $i^\star$th tree operation, $N^k(v) \cap T_{i^\star}$ contains all marked vertices within distance k of v. Since $\text{dist}(v,w) \leq \text{dist}(v,u) + \text{dist}(u,w) \leq 1.5k$, we have $N^k(v) \cap T_{i^\star} \subseteq N^{2.5k}(w) \cap T_{i^\star}$.

Proof of (ii). The diameter of each connected component (in T) induced by the unmarked vertices is $O(k \log_{\lambda/k} s + d)$, since the total number of Rakes is $O(k \log_{\lambda/k} s) + 3d + 1$, and all vertices removed by Compress are marked. We show that the set of connected components induced by the unmarked vertices remains the same if we change the underlying graph from T to T^k. This implies the diameter upper bound $O(\log_{\lambda/k} s + (d/k))$ when the underlying graph is T^k.

Consider any pair of unmarked vertices u and v. Notice that u and v must be removed by Rakes. Suppose that u and v are not connected in T after deleting those vertices removed by Compress from T. Assume the first time they become disconnected in T is iteration i, which is due to the removal of a vertex w in Compress. Since all vertices in $N^{k/2}(w)$ are marked, the unique shortest path in T connecting u and v must have 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.

Discussion. We briefly discuss how we choose the parameters $r_1 = 2.5k$ used in the Compress operation and $r_2 = 0.5k$ used in defining $\mathcal{M}$. Notice that the correctness of Lemma 19 is independent of the choice of these parameters. The proof of (i) relies on the fact that $r_1 \geq 2k + r_2$. The proof of (ii) relies on the fact that $2r_2 + 1 \geq k$. We select the smallest possible values of r_1 and r_2 to make these proofs work.

7 DETERMINISTIC ALGORITHMS FOR EDGE COLORING TREES

Let $T = (V, E)$ be a tree with n vertices and $N^+(v) = N(v) \cup \{v\}$ be the inclusive neighborhood of v. We decompose T using another variation on Miller and Reif's [50] rake and compress operations, the second of which is parameterized by an integer $k \geq 2$.

Rake: Remove all leaves and isolated vertices from T.
Compress: Remove the set $\{v \in V \mid \text{for every } u \in N^+(v), \deg_T(u) \leq k\}$ from T.

THEOREM 9. *Alternately applying Compress and Rake* $1 + \log_k n$ *times removes all vertices from any* n*-vertex tree* T*.*

PROOF. Root T at an arbitrary vertex and let $\text{size}(v)$ be the number of vertices in the subtree rooted at v. We prove by induction that if $\text{size}(v) \leq k^i$, v will be removed after the first $i + 1$ rounds of Compress and Rake. The claim is trivially true when $i = 0$. Assume the claim is true for $i - 1$. Let v be any vertex with $\text{size}(v) \in (k^{i-1}, k^i]$, and define V' to be the set of all vertices u such that (i) $\text{size}(u) \in (k^{i-1}, k^i]$ and (ii) u is in the subtree rooted at v. Notice that each vertex $u \in V'$ has $\deg_{V'}(u) \leq k$, since otherwise $\text{size}(u) > k^i$. By the inductive hypothesis, all descendants of v that are not in V' have been removed after i rounds of Compress and Rake. The $(i + 1)$th Compress will remove all remaining vertices in $V' - \{v\}$. Hence, all descendants of v have been removed after the $(i + 1)$th Compress. However, the degree of the parent of v is unbounded, so v may not be removed. If v still remains, then the $(i + 1)$th Rake will remove it. □

THEOREM 10. *There is an* $O(\log_\Delta n)$*-time* DetLOCAL *algorithm for* Δ*-edge coloring a tree* T *with maximum degree* $\Delta \geq 3$*.*

PROOF. Let β be the constant such that Linial's algorithm [48] finds a $\beta\Delta^2$-edge coloring in $O(\log^* n - \log^* \Delta + 1)$ time. We begin by decomposing T with Compress and Rake steps, using parameter $k = \max\{2, \lfloor(\Delta/\beta)^{1/3}\rfloor\}$. Define $T_i = (V_i, E_i)$ to be the forest before the ith round of Compress and Rake, and let V_i^c and V_i^r be those vertices removed by the ith Compress and Rake, respectively.

We edge color the trees $T_{1+\log_k n}, \ldots, T_1 = T$ in this order. Given a coloring of T_{i+1}, we need to color the remaining uncolored edges in T_i. Let $u \in T_{i+1}$ be a vertex, and let $v_1, \ldots, v_x \in V_i^r$ be the vertices adjacent to u removed by the ith Rake. At this point u is incident to at most $\Delta - x$ colored edges. We assign to $\{u, v_1\}, \ldots, \{u, v_x\}$ any distinct available colors from their palettes.

We now turn to the vertices removed by the ith Compress. First, suppose that Δ is large enough such that $k = \lfloor(\Delta/\beta)^{1/3}\rfloor$. Let ϕ be a βk^2-edge coloring of the (as yet uncolored) subgraph of T_i (i.e., the edges that are incident to some vertices in V_i^c). We argue that this subgraph has maximum degree at most k, and so we are able to apply Linial's algorithm [48] to find a βk^2-edge coloring. Suppose $e = \{u, v\}$ is in this subgraph, but either $\deg_{T_i}(u) > k$ or $\deg_{T_i}(v) > k$. If this were true, then neither u nor v could have been removed by the ith Compress, contradicting the fact that e is incident to some vertices in V_i^c.

Partition the palette $\{1, \ldots, \Delta\}$ into βk^2 parts $P_1, \ldots, P_{\beta k^2}$. Each part has size $\Delta/(\beta k^2) \geq k$. Each $v \in V_i^c$ colors each edge $\{v, u\}$ any available color in $P_{\phi(\{v,u\})}$. Since $\deg_{T_i}(u) \leq k$, at most $k - 1$ of its incident edges may already be colored, and so there must be at least one available color in $P_{\phi(\{v,u\})}$ for $\{v, u\}$ to use. All calls to Linial's βk^2-edge coloring algorithm can be executed in parallel, so the overall time is $O(\log_k n + \log^* n - \log^* k) = O(\log_\Delta n)$.

When $k = 2$, the subgraph induced by $V_1^c \cup \cdots \cup V_{1+\log_k n}^c$ consists of a set of paths. In $O(\log^* n)$ time, we find an *initial* 3-edge coloring of these paths. We now color $T_{1+\log_k n}, \ldots, T_1$ in this order. Coloring the edges removed during a Rake is done as before. The set V_i^c removed in one Compress induces some paths, each end-edge of which may be adjacent to one (previously colored) edge in

T_{i+1}. If the initial color of an end-edge conflicts with the coloring of T_{i+1}, then we recolor it any available color. When $k = 2$ this procedure takes $O(\log^* n + \log_k n) = O(\log_\Delta n)$ time. □

An *oriented* tree is a rooted tree where each vertex that is not the root knows its parent. We show that a $(\Delta + 1)$-edge coloring of an oriented tree can be found in $O(\log^* n)$ time, but Δ-edge coloring takes $\Omega(\log_\Delta n)$ time.

Theorem 11. *Any oriented tree T can be $(\Delta + 1)$-edge colored in $O(\log^* n)$ time.*

Proof. Initially pick color $\phi_0(\{u, \text{parent}(u)\}) = i$ if $\text{ID}(u)$ is the ith largest ID among its siblings. Observe that for any i, $\phi_0^{-1}(i)$ is a subgraph consisting of oriented paths, and that $\phi_0^{-1}(\Delta)$ is at most one edge, attached to the root. For each $i \in \{1, \ldots, \Delta - 1\}$, in parallel, recolor $\phi_0^{-1}(i)$ using the color set $\{i, \Delta, \Delta + 1\}$ in such a way that the most ancestral edge in each path remains colored i. This takes $O(\log^* n)$ time [23, 48].

The result is a legal $(\Delta + 1)$-edge coloring. It is clear that for each $i \in \{1, \ldots, \Delta - 1\}$, no two edges with color i are adjacent. Now consider $i \in \{\Delta, \Delta + 1\}$. Suppose there exist two adjacent edges $e = \{u, v\}$ and $e' = \{v, w\}$ that are both colored i. Let $j \in \{1, \ldots, \Delta\}$ be the original color of e before recoloring, and let P be the j-color (before recoloring) oriented path containing e. Similarly, let $j' \in \{1, \ldots, \Delta\}$ be the original color of e' before recoloring, and let P' be the j'-color (before recoloring) oriented path containing e'. Then the two paths P and P' intersect only at v, and so at least one of e and e' is the most ancestral edge of the corresponding path. This contradicts the assumption that they are colored by $i \in \{\Delta, \Delta + 1\}$ (after recoloring). Thus, all edges colored $i \in \{\Delta, \Delta + 1\}$ are not adjacent to each other. □

Theorem 12. *Any Δ-edge coloring algorithm for oriented trees takes $\Omega(\log_\Delta n)$ time in* RandLOCAL.

Proof. Let T be an oriented Δ-regular tree with height $h = \Theta(\log_\Delta n)$ and $\mathcal{A}$ be an edge coloring algorithm running in $h/3$ time. The color of $\{u, \text{parent}(u)\}$ is uniquely determined by the colors of the edges incident to leaf-descendants of u. Let V' denote the set of leaf-descendants of u. In general, $N^{h/3}(u)$ and $\bigcup_{v \in V'} N^{h/3}(v)$ do not intersect. In this case, u only has a $1/\Delta$ chance of guessing the correct edge color; if it guesses incorrectly, there must be a violation somewhere in the subtree rooted at u. □

8 CONCLUDING REMARKS

The focus of this article has been on the complexity of distributed *edge-coloring*, on general graphs and trees, with and without randomization. Nonetheless, we took several extended detours into apparently unrelated topics such as the distributed Lovász local lemma (Section 5) and network decompositions (Section 6). A recent line of work on developing a *complexity theory* for the LOCAL model [4–7, 16, 17, 19, 21, 30, 35, 38, 55] explains why these particular detours are *natural* and perhaps unavoidable in the pursuit of optimal LOCAL algorithms.

The appearance of the distributed Lovász local lemma (LLL) is no surprise at all, given that it generalizes a problem related to $(1 + \epsilon)\Delta$-edge coloring, namely, sinkless orientation (Theorem 1), is complete for sublogarithmic time [21, Theorem 4.1], and is a generally useful tool for finding objects that cannot be generated by a greedy algorithm [22, 29, 30, 52, 60]. The *structure* of the LLL algorithm in Sections 5 and 6 also turns out to be quite natural. Chang, Kopelowitz, and Pettie's derandomization [19, Theorem 3.1] justifies why we *must* apply the graph shattering method to solve the LLL in randomized $O(\log \log n)$ time, and that any such algorithm must contain within it a deterministic $O(\log n)$-time algorithm. Our $O(\log n)$-time deterministic LLL algorithm for trees (Theorem 5) follows Fischer and Ghaffari [30], who showed how to solve LLL instances using

network decompositions. Ghaffari, Kuhn, and Maus [38] show that this choice also turns out to be *natural*, in the sense that you cannot solve the LLL deterministically *without* computing good network decompositions deterministically.[12]

APPENDIX

A PROOF OF LEMMA 5

In this section, we prove the concentration bounds of Lemma 5. For notational simplicity, we ignore all subscripts i, i.e., p, d, t are the palette size, degree, and c-degree before the ith round of coloring, all of which satisfy invariant $\mathcal{H}_i$. Recall that we introduce imaginary edges, if necessary, to ensure that the entire graph has uniform c-degree t and uniform palette size p. $S(v)$ is the set of real edges incident to v, $|S(v)| \leq d$, and $N_c(v)$ the set of real and imaginary edges incident to v with c in their palettes. The arguments of this section do not differentiate between real and imaginary edges. From Lemma 3, we use the fact that $t = \Theta(p)$; i.e., t and p are interchangeable in those parts of the proof that are not sensitive to the leading constant.

We make extensive use of Theorem 13 and Lemma 20 to prove Lemma 5. Theorem 13 is from Dubhashi and Panconesi's book [27] on the concentration of measure, where it is called the *method of bounded variances*. Ignoring the leading constant in the exponent, Theorem 13 is strictly more powerful than Chernoff-Hoeffding and Azuma-type inequalities, and is best suited in applications that have the following two features:

- We are interested in deviations of $f(\mathbf{X}_n)$ from its expectation (up to $\pm s$) that are significantly smaller than the number of underlying random variables (n) times the Lipschitz bound satisfied by the martingale (M). This feature renders Azuma's inequality too weak to be of any use.[13]
- The Lipschitz bound is pessimistic: although $D_i = \mathrm{E}[f|\mathbf{X}_i] - \mathrm{E}[f|\mathbf{X}_{i-1}]$ can be as large as M, its variance (σ_i^2) conditioned on <u>any</u> $\mathbf{X}_{i-1}$ is substantially smaller.

For example, in the first round of coloring, the c-degree of a vertex v depends on $\Theta(\Delta^3)$ random variables (colors chosen by edges in the 3-neighborhood), but we are interested in deviations from the expected c-degree that are $s = O(\Delta)$. Any single edge could have a significant effect on v's c-degree ($M = \Theta(1)$), but the *variances* of these effects are substantially smaller. In particular, the sum of variances $\sum_i \sigma_i^2$ will be $O(\Delta)$.

Theorem 13 ([27, Equation (8.5)]). *Let $X_1, \ldots, X_n$ be an arbitrary set of random variables. Let $f(X_1, \ldots, X_n)$ be such that $\mathrm{E}[f]$ is finite. We write $D_i \stackrel{\text{def}}{=} \mathrm{E}[f|\mathbf{X}_i] - \mathrm{E}[f|\mathbf{X}_{i-1}]$. Suppose that there exist M and values $\{\sigma_i^2\}_{1 \leq i \leq n}$ meeting the following conditions.*

- *For any assignment to the random variables $\mathbf{X}_{i-1}$, $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] \leq \sigma_i^2$.*
- *For any assignment to the random variables $\mathbf{X}_i$, $|D_i| \leq M$.*

Then $\Pr[f > \mathrm{E}[f] + s] \leq \exp\left(-\frac{s^2}{2(\sum_{i=1}^n \sigma_i^2 + Ms/3)}\right)$.

Lemma 20 follows from straightforward calculation.

Lemma 20. *Let X be a random variable such that (i) $\mathrm{E}[X] = 0$, (ii) $\Pr[X = a] = \alpha$ and $\Pr[X = b] = 1 - \alpha$, and (iii) $|a - b| \leq k$. Then, we have the following.*

[12] In particular, the distributed LLL is PSLOCAL-hard as it generalizes the PSLOCAL-complete problem of *Weak Local Splitting* [38, Theorem 1.4]. As a consequence, any deterministic poly(log n) LLL algorithm can also be used to compute (poly(log n), poly(log n))-network decompositions deterministically.

[13] A vector $(X_1, \ldots, X_i)$ of random variables is written $\mathbf{X}_i$.

- $\mathrm{Var}[X] \le \alpha(1-\alpha)k^2 \le \alpha k^2$.
- $|b| \le \alpha k$.
- $|a| \le (1-\alpha)k \le k$.

Throughout this section, we use the following notation. For each edge e and each color c, define $z_{e,c}$ as the indicator random variable that e *successfully* colors itself c, thus $z_{e,c} = 0$ if $c \notin \Psi(e)$.

A.1 Concentration of Vertex Degree

Let $v^\bullet$ be a vertex. We claim that $\mathrm{E}[|S^\diamond(v^\bullet)|] \le d^\diamond$. An edge e successfully colors itself with probability $(1-1/p)^{2(t-1)}$, since there are $2(t-1)$ edges competing with e for $\mathrm{Color}^\star(e)$, and each of these $2(t-1)$ edges selects $\mathrm{Color}^\star(e)$ with probability $1/p$. Thus, by linearity of expectation,

$$\mathrm{E}[|S^\diamond(v^\bullet)|] = (1-(1-1/p)^{2(t-1)})|S(v^\bullet)| \le (1-(1-1/p)^{2(t-1)})d = d^\diamond.$$

For brevity, we write $S \stackrel{\text{def}}{=} S(v^\bullet)$, $S^\diamond \stackrel{\text{def}}{=} S^\diamond(v^\bullet)$, and $z \stackrel{\text{def}}{=} |S| - |S^\diamond|$. The goal of this section is to show that $\Pr[z < \mathrm{E}[z] - s] = \exp(-\Omega(s^2/|S|))$, which implies the desired concentration bound $\Pr[|S^\diamond(v^\bullet)| > (1+\delta)d^\diamond] = \exp(-\Omega(\delta^2 d))$, by setting $s = \delta d^\diamond$.

Notations. We write $z_e \stackrel{\text{def}}{=} \sum_{c\in\Psi(e)} z_{e,c}$ and $z_c \stackrel{\text{def}}{=} \sum_{e\in S} z_{e,c}$. In other words, z_e is the indicator random variable that e successfully colors itself; z_c is the indicator random variable that some edge in S successfully colors itself by c. We can express z as $z = \sum_{e\in S} z_e$ or $z = \sum_c z_c$, where the summation is over all colors $c \in \bigcup_{e\in S}\Psi(e)$.

Let S' denote the set of edges such that $e' \in S'$ if there exists $e = \{v^\bullet, u\} \in S$ such that (i) $\Psi(e) \cap \Psi(e') \neq \emptyset$, and (ii) e' is incident to e. For each edge $e' \in S'$ and for each color $c \in \Psi(e')$, we define $R(e',c)$ as the subset of S such that $e \in R(e',c)$ if (i) e is incident to e', and (ii) $c \in \Psi(e)$. We write $w(e',c) = |R(e',c)|$ and $w(e') = \sum_{c\in\Psi(e')} w(e',c)$. Notice that the value $w(e',c)$ may exceed 2 when $e' \notin S$ is an imaginary edge incident to $v^\bullet$. Intuitively, $w(e')$ measures the influence of $\mathrm{Color}^\star(e')$ on z. Notice that $\sum_{e'\in S'} w(e') \le 2|S|pt$.

We consider the sequence of random variables $(X_1, \ldots, X_{|S|+|S'|})$, where the initial $|S'|$ variables are the colors selected by the edges in S', in arbitrary order, and the remaining $|S|$ variables are the colors selected by the edges in S, in arbitrary order. We let $z = f(X_1, \ldots, X_{|S|+|S'|})$ in Theorem 13. To prove the desired concentration bound, it suffices to show that we can set $M = O(1)$ and σ_i^2 to achieve $\sum_{i=1}^{|S|+|S'|}\sigma_i^2 = O(|S|)$. In what follows, we analyze the effect of exposing the value of the random variable X_i, given that all variables in $\mathbf{X}_{i-1}$ have been fixed.

Exposing an Edge in S'. Consider the case where $X_i = \mathrm{Color}^\star(e^\star)$ is the color selected by the edge $e^\star \in S'$. Recall $D_i = \mathrm{E}[z|\mathbf{X}_i] - \mathrm{E}[z|\mathbf{X}_{i-1}]$. Our goal is to show that $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(w(e)/(pt))$ and $|D_i| = O(1)$. Hence, we set $\sigma_i^2 = O(w(e)/(pt))$, which implies $\sum_{1\le i\le|S'|}\sigma_i^2 = O(|S|)$, as desired.

By linearity of expectation, $D_i = \sum_c(\mathrm{E}[z_c|\mathbf{X}_i] - \mathrm{E}[z_c|\mathbf{X}_{i-1}])$, where the summation ranges over all colors c that appear in $\bigcup_{e\in S}\Psi(e)$. We write $D_{i,c} = \mathrm{E}[z_c|\mathbf{X}_i] - \mathrm{E}[z_c|\mathbf{X}_{i-1}]$, and make the following observations:

- $D_{i,c} \neq 0$ only if $c \in \Psi(e^\star)$. For each $c \in \Psi(e^\star)$, $D_{i,c}$ depends only on whether $e^\star$ selects the color c, which occurs with probability $1/p$. In particular, $D_{i,c} < 0$ only if $e^\star$ selects c, and $D_{i,c} > 0$ only if $e^\star$ does not select c. Thus, $\mathrm{Cov}[D_{i,c}, D_{i,c'}|\mathbf{X}_{i-1}] \le 0$ for all color pairs $\{c, c'\}$.
- For each $e \in S$, both $\mathrm{E}[z_{e,c}|\mathbf{X}_i]$ and $\mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]$ are within $[0, 1/p]$, since $z_{e,c} = 1$ only if $c \in \Psi(e)$ and e selects c, which occurs with probability $1/p$. Thus, $\max_{X_i} D_{i,c} - \min_{X_i} D_{i,c} \le w(e^\star, c)/p$.

By Lemma 20 (with $k \le w(e^\star, c)/p$ and $\alpha = 1/p$), we have $\mathrm{Var}[D_{i,c}|\mathbf{X}_{i-1}] \le (1/p)(w(e^\star, c)/p)^2$. We bound the variance $\mathrm{Var}[D_i|\mathbf{X}_{i-1}]$ as follows:

$$\begin{aligned}
\mathrm{Var}[D_i|\mathbf{X}_{i-1}] &= \sum_c \mathrm{Var}[D_{i,c}|\mathbf{X}_{i-1}] + \sum_{c,c'} \mathrm{Cov}[D_{i,c}, D_{i,c'}|\mathbf{X}_{i-1}] \\
&= \sum_c O((w(e^\star, c)/p)^2/p) && \mathrm{Cov}[D_{i,c}, D_{i,c'}|\mathbf{X}_{i-1}] \le 0 \\
&= \sum_c O(w(e^\star, c)/p^2) && w(e^\star, c) < t = \Theta(p) \\
&= O(w(e^\star)/p^2) \\
&= O(w(e^\star)/(pt)).
\end{aligned}$$

We bound $|D_i|$ as follows. Consider $c \in \Psi(e^\star)$. Recall that we already have the bound $|D_{i,c}| \le w(e^\star, c)/p \le (t-1)/p$. If c is not selected by $e^\star$, which occurs with probability $1 - 1/p$, then we have a tighter bound $|D_{i,c}| \le w(e^\star, c)/p^2 \le (t-1)/p^2$ by Lemma 20 with $k \le w(e^\star, c)/p$ and $\alpha = 1/p$. Therefore,

$$|D_i| \le \sum_c |D_{i,c}| \le 1 \cdot \frac{t-1}{p} + (p-1) \cdot \frac{t-1}{p^2} = O(1).$$

Exposing an Edge in S. Consider the case where $X_i = \mathrm{Color}^\star(e^\star)$ is the color selected by the edge $e^\star \in S$. Suppose that $X_i = c^\star$. Recall $D_i = \sum_c D_{i,c}$. It is straightforward to see that (i) $|D_{i,c}| \le 1$ if $c = c^\star$, (ii) $|D_{i,c}| \le 1/p$ if $c \in \Psi(e^\star) - \{c^\star\}$, and (iii) $|D_{i,c}| = 0$ otherwise. Thus, $|D_i| = O(1)$, and $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(1)$. We set $\sigma_i^2 = O(1)$, and so $\sum_{|S'|<i\le|S|+|S'|} \sigma_i^2 = O(|S|)$.

A.2 Concentration of Palette Size

Let $e^\bullet = \{u, v\}$ be an edge, and let $c^\bullet = \mathrm{Color}^\star(e^\bullet)$ be the color selected by $e^\bullet$. We do not consider $c^\bullet$ as a random variable in the analysis (i.e., we expose the color selected by $e^\bullet$ first). Let $\mathcal{E}$ be the event that $e^\bullet$ does not successfully color itself. Since $e^\bullet$ remains uncolored with at least a constant probability, we are allowed to ignore the condition "$e^\bullet$ remains uncolored" in Lemma 5 in the subsequent calculation. To prove the desired concentration bound regarding palette size $\Pr[|\Psi^\diamond(e)| < (1-\delta)p^\diamond \mid e\text{remains uncolored}] = \exp(-\Omega(\delta^2 p))$, it suffices to show that (i) $|\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|] - p^\diamond| = O(1)$, and (ii) $\Pr[|\Psi^\diamond(e^\bullet)| < (1-\delta)\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|]] = \exp(-\Omega(\delta^2\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|]))$.

Notations. We write S_u (respectively, S_v) to denote the set of edges e incident to $e^\bullet$ on u (respectively, v) such that $\Psi(e) \cap \Psi(e^\bullet) - \{c^\bullet\} \ne \emptyset$. We write S' to denote the set of edges such that $e' \in S'$ if there exists $e \in S_u \cup S_v$ meeting the following conditions: (i) e' is incident to e, (ii) $e' \notin S_u \cup S_v \cup \{e^\bullet\}$, and (iii) $\Psi(e) \cap \Psi(e') \cap \Psi(e^\bullet) - \{c^\bullet\} \ne \emptyset$. Notice that $\Psi^\diamond(e^\bullet)$ is determined by the colors selected by the edges in $S_u \cup S_v \cup S'$. We have $|S_u| \le (p-1)(t-1) < pt$, $|S_v| \le (p-1)(t-1) < pt$, and $|S'| \le 2(p-1)(t-1)^2 < 2pt^2$.

Expected Value. In what follows, consider a color $c \in \Psi(e^\bullet) - \{c^\bullet\}$.

- Let $e \in S_u \cup S_v$ such that $c \in \Psi(e)$. We have $\mathrm{E}[z_{e,c}] = \frac{1}{p}(1-\frac{1}{p})^{2t-3}$. Notice that $e^\bullet$ selects $c^\bullet \ne c$, so there are $2t-3$ (rather than $2t-2$) edges competing with e for the color c.
- Let $e' = \{u, x\} \in S_u$ and $e'' = \{v, y\} \in S_v$ such that $c \in \Psi(e') \cap \Psi(e'')$. We define $z_{e',e'',c} \stackrel{\mathrm{def}}{=} z_{e',c} \cdot z_{e'',c}$. If $x = y$, then $z_{e',e'',c} = 0$. Otherwise, $x \ne y$ and $\mathrm{E}[z_{e',e'',c}] = \frac{1}{p^2}(1-\frac{1}{p})^{4t-6-b(e',e'')}$, where $b(e', e'') \le 3$ is the number of edges e such that (i) $e \ne e^\bullet$, and (ii) e is incident to both e' and e''.

Let z_c be the indicator random variable that some edge incident to $e^\bullet$ successfully colors itself by c, that is,

$$z_c \stackrel{\text{def}}{=} \sum_{e\,:\,e\in S_u\cup S_v,\ c\in\Psi(e)} z_{e,c} - \sum_{e',e''\,:\,e'\in S_u,\ e''\in S_v,\ c\in\Psi(e')\cap\Psi(e'')} z_{e',e'',c}.$$

The number of edges $e \in S_u \cup S_v$ such that $c \in \Psi(e)$ is exactly $2t-2$. The number of pairs $(e' = \{u,x\} \in S_u, e'' = \{v,y\} \in S_v)$ such that $c \in \Psi(e') \cap \Psi(e'')$ and $x \neq y$ is at least $(t-1)^2 - (t-1)$ and at most $(t-1)^2$. By linearity of expectation (recall $t = \Theta(p)$),

$$\mathrm{E}[z_c] = \frac{2t}{p}(1-1/p)^{2t} - \frac{t^2}{p^2}(1-1/p)^{4t} \pm O(1/p).$$

Define $z \stackrel{\text{def}}{=} \sum_{c\in\Psi(e^\bullet)-\{c^\bullet\}} z_c$. Then, we have

$$\begin{aligned}
\mathrm{E}[|\Psi^\diamond(e^\bullet)|] &= |\Psi(e^\bullet)| - \mathrm{E}[z] && |\Psi^\diamond(e^\bullet)| = |\Psi(e^\bullet)| - z\\
&= p\cdot\left(1 - \frac{2t}{p}(1-1/p)^{2t} + \frac{t^2}{p^2}(1-1/p)^{4t} \pm O(1/p)\right)\\
&= p\cdot\left(1 - \frac{2t}{p}(1-1/p)^{2t} + \frac{t^2}{p^2}(1-1/p)^{4t}\right) \pm O(1)\\
&= p^\diamond \pm O(1). && \text{Definition of } p^\diamond
\end{aligned}$$

Hence, $|\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|] - p^\diamond| = O(1)$.

Concentration Bound. Consider the sequence of random variables $(X_1,\ldots,X_{|S_u|+|S_v|+|S'|})$, where the initial $|S'|$ variables are the colors selected by the edges in S', in arbitrary order, and the remaining $|S_u|+|S_v|$ variables are the colors selected by the edges in $S_u \cup S_v$, in arbitrary order. Let $z = f(X_1,\ldots,X_{|S_u|+|S_v|+|S'|})$ in Theorem 13. To prove the desired concentration bound $\Pr[|\Psi^\diamond(e^\bullet)| < (1-\delta)\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|]] = \exp(-\Omega(\delta^2\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|]))$, it suffices to show that $\Pr[z > \mathrm{E}[z]+s] = \exp(-\Omega(s^2/p))$, by setting $s = \delta\,\mathrm{E}[|\Psi^\diamond(e^\bullet)|]$, and recall that $\mathrm{E}[|\Psi^\diamond(e^\bullet)|] = p^\diamond \pm O(1) = \Theta(p)$. In view of Theorem 13, we only need to show that we can set $M = O(1)$ and σ_i^2 such that $\sum_{i=1}^{|S_u|+|S_v|+|S'|}\sigma_i^2 = O(p)$.

Exposing an Edge in S'. Consider the case where $X_i = \mathrm{Color}^\star(e^\star)$ is the color selected by the edge $e^\star \in S'$. Our goal is to show that $|D_i| = O(1/t)$. This implies $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(1/t^2)$, and so we may set $\sigma_i^2 = O(1/t^2)$. Since $|S'| = O(pt^2)$, we have $\sum_{i=1}^{|S'|}\sigma_i^2 = O(p)$.

Let R denote the set of edges in $S_u \cup S_v$ that are incident to $e^\star$. Notice that $1 \le |R| \le 2$. We define

$$z_c^{(i)} \stackrel{\text{def}}{=} \sum_{e'\,:\,e'\in R,\ c\in\Psi(e')} z_{e',c} - \sum_{e',e''\,:\,e'\in S_u,\ e''\in S_v,\ c\in\Psi(e')\cap\Psi(e''),\ \{e,e''\}\cap R\neq\emptyset} z_{e',e'',c}.$$

Intuitively, $z_c^{(i)}$ is the result of subtracting all terms from the definition of z_c not involving edges in R. We now argue that $\mathrm{E}[z_c|\mathbf{X}_i] - \mathrm{E}[z_c|\mathbf{X}_{i-1}] = \mathrm{E}[z_c^{(i)}|\mathbf{X}_i] - \mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]$. This is due to the two observations: (i) If $e \notin R$, then $\mathrm{E}[z_{e,c}|\mathbf{X}_i] = \mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]$. (ii) If $\{e',e''\}\cap R = \emptyset$, then $\mathrm{E}[z_{e',e'',c}|\mathbf{X}_i] = \mathrm{E}[z_{e',e'',c}|\mathbf{X}_{i-1}]$.

Consider a color $c \in \Psi(e^\star)\cap\Psi(e^\bullet) - \{c^\bullet\}$. The probability that some edge in R selects c is at most $|R|/p \le 2/p$. Thus, the conditional expectations $\mathrm{E}[z_c^{(i)}|\mathbf{X}_i]$ and $\mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]$ must be within $[0,2/p]$, and so $|\,\mathrm{E}[z_c^{(i)}|\mathbf{X}_i] - \mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]| \le 2/p$. For the case of $c \neq X_i$, which occurs with probability $1-1/p$, we have a tighter bound $|\,\mathrm{E}[z_c^{(i)}|\mathbf{X}_i] - \mathrm{E}[z_c^{(i)}|\mathbf{X}_{i-1}]| \le 2/p^2$ by Lemma 20 with

$k \le 2/p$ and $\alpha = 1/p$. We bound $|D_i|$ as follows:

$$\begin{aligned}|D_i| &\le \sum_{c \in \Psi(e^\bullet) - \{c^\bullet\}} |\operatorname{E}[z_c|\mathbf{X}_i] - \operatorname{E}[z_c|\mathbf{X}_{i-1}]| \\ &= \sum_{c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}} |\operatorname{E}[z_c^{(i)}|\mathbf{X}_i] - \operatorname{E}[z_c^{(i)}|\mathbf{X}_{i-1}]| \\ &\le (2/p) + (2/p^2)(|\Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}| - 1) \\ &= O(1/p) = O(1/t).\end{aligned}$$

Exposing an Edge in $S_u \cup S_v$. Consider the case where $X_i = \text{Color}^\star(e^\star)$ is the color selected by the edge $e^\star \in S_u \cup S_v$. We define $w(e^\star) \stackrel{\text{def}}{=} |\Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}|$. The goal is to show that (i) $|D_i| = O(1)$ and (ii) $\operatorname{Var}[D_i|\mathbf{X}_{i-1}] = O(w(e^\star)/p)$. By setting $\sigma_i^2 = O(w(e^\star)/p)$, we achieve

$$\sum_{i=|S'|+1}^{|S'|+|S_u|+|S_v|} \sigma_i^2 = \sum_{e \in S_u \cup S_v} O(w(e)/p) = O(pt/p) = O(t) = O(p).$$

By the linearity of expectation, $D_i = \sum_{c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}} D_{i,c}$, where $D_{i,c} = \operatorname{E}[z_c|\mathbf{X}_i] - \operatorname{E}[z_c|\mathbf{X}_{i-1}]$. Since both $\operatorname{E}[z_c|\mathbf{X}_i]$ and $\operatorname{E}[z_c|\mathbf{X}_{i-1}]$ are within $[0,1]$, we have $|D_{i,c}| \le 1$. We have a tighter bound $|D_{i,c}| \le 1/p$ in the event that $\text{Color}^\star(e^\star) \neq c$ (by Lemma 20 with $k \le 1$ and $\alpha = 1/p$). Thus, $|D_i| \le 1 + (w(e^\star) - 1)/p = O(1)$.

To prove that $\operatorname{Var}[D_i|\mathbf{X}_{i-1}] = O(w(e^\star)/p)$, we need the following two observations.

- Consider a color $c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}$. Recall that $|D_{i,c}| \le 1/p$ for the case c is not selected by $e^\star$, which occurs with probability $1 - 1/p$. Thus, $\operatorname{E}[D_{i,c} \cdot D_{i,c}|\mathbf{X}_{i-1}] \le (1/p) \cdot 1 + (1 - 1/p) \cdot 1/p^2 = O(1/p)$.
- Consider two distinct colors c and c' in $\Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}$. If $e^\star$ selects c or c' (which occurs with probability $2/p$), then $D_{i,c} \cdot D_{i,c'} \le 1 \cdot (1/p)$. Otherwise, $D_{i,c} \cdot D_{i,c'} \le (1/p) \cdot (1/p)$. Therefore, $\operatorname{E}[D_{i,c} \cdot D_{i,c'}|\mathbf{X}_{i-1}] \le (2/p) \cdot 1/p + (1 - 2/p) \cdot 1/p^2 = O(1/p^2)$.

We now bound $\operatorname{Var}[D_i|\mathbf{X}_{i-1}]$ as follows:

$$\begin{aligned}\operatorname{Var}[D_i|\mathbf{X}_{i-1}] &\le \sum_{c \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}} \sum_{c' \in \Psi(e^\star) \cap \Psi(e^\bullet) - \{c^\bullet\}} \operatorname{E}[D_{i,c} \cdot D_{i,c'}|\mathbf{X}_{i-1}] \\ &\le w(e^\star) \cdot O(1/p) + w(e^\star)(w(e^\star) - 1) \cdot O(1/p^2) \\ &= O(w(e^\star)/p).\end{aligned}$$

A.3 Concentration of Color Degree

For the remainder of this section, fix a vertex $v^\bullet$ and a color $c^\bullet$ in the palette $\Psi(e)$ for some e incident to $v^\bullet$. For convenience, we write $R \stackrel{\text{def}}{=} N_{c^\bullet}(v^\bullet)$. Define $R^\diamond$ as the subset of R such that $e = \{v^\bullet, u\} \in R^\diamond$ if (i) e is not successfully colored by a color in $\Psi(e) - \{c^\bullet\}$, and (ii) no edge incident to e on u successfully colors itself $c^\bullet$. We write $z \stackrel{\text{def}}{=} |R \setminus R^\diamond|$. Let $\mathcal{E}'$ be the event that $N^\diamond_{c^\bullet}(v^\bullet) \neq \emptyset$. Observe that if $\mathcal{E}'$ occurs, then no edge incident to $v^\bullet$ successfully colors itself $c^\bullet$. Thus, conditioning on $\mathcal{E}'$ happening, $R \setminus R^\diamond$ equals $N^\diamond_{c^\bullet}(v^\bullet)$.

Our goal is to show that (i) $\Pr[z < \operatorname{E}[z] - s] = \exp(-\Omega(s^2/t))$, and (ii) $\operatorname{E}[|R^\diamond|] = |R| - \operatorname{E}[z] = t^\diamond \pm O(1)$. Since $\mathcal{E}'$ occurs with constant probability, the above (i) and (ii) together imply the desired

concentration bound $\Pr[|N^{\diamond}_{c^\bullet}(v^\bullet)| > (1+\delta)t^\diamond \mid \mathcal{E}'] = \exp(-\Omega(\delta^2 t))$, by setting $s = \delta t^\diamond \pm O(1)$. Recall that $t^\diamond = \Theta(t)$.

Expected Value. With respect to an edge $e = \{v^\bullet, u\} \in R$, we define the following notations based on parts (i) and (ii) of the definition of $R^\diamond$.

- Define z^a_e as the indicator random variable that some edge incident to e on u successfully colors itself $c^\bullet$. We have $\mathrm{E}[z^a_e] = (t-1)\cdot\frac{1}{p}(1-\frac{1}{p})^{2t-2} = \frac{t}{p}(1-\frac{1}{p})^{2t} \pm O(1/p)$.
- Define z^b_e as the indicator random variable that e is successfully colored by a color in $\Psi(e) - \{c^\bullet\}$. We have $\mathrm{E}[z^b_e] = (p-1)\cdot\frac{1}{p}(1-\frac{1}{p})^{2t-2} = (1-\frac{1}{p})^{2t} \pm O(1/p)$.

Let $z^{a,b}_e \stackrel{\text{def}}{=} z^a_e \cdot z^b_e$. Notice that z^a_e and z^b_e are nearly independent but not independent. Let $z_e \stackrel{\text{def}}{=} z^a_e + z^b_e - z^{a,b}_e$, and so we have $z = |R \setminus R^\diamond| = \sum_{e\in R} z_e$. We calculate $\mathrm{E}[z^{a,b}_e]$ as follows. Let e' be any edge incident to e such that $c^\bullet \in \Psi(e')$, and let c be any color in $\Psi(e) - \{c^\bullet\}$. With respect to (e, e', c), we define the following two sets:

- S_a is the set of all edges e'' such that (i) $e'' \neq e, e'$, (ii) e'' is incident to e', and (iii) $c^\bullet \in \Psi(e'')$. Intuitively, S_a is the set of all edges other than e that contend with e' for the color $c^\bullet$. Notice that $|S_a| = 2t-3$, since $\Psi(e)$ must contain $c^\bullet$.
- S_b is the set of all edges e'' such that $e'' \in S_b$ if (i) $e'' \neq e, e'$, (ii) e'' is incident to e, and (iii) $c \in \Psi(e'')$. Intuitively, S_b is the set of all edges other than e' that contend with e for the color c. Notice that $2t-3 \le |S_b| \le 2t-2$, since $\Psi(e')$ may or may not contain c. The extent to which S_a and S_b intersect is unknown.

Fixing the edge e incident to $v^\bullet$, let $x(c, e')$ denote the probability that (i) e' successfully colors itself $c^\bullet$ and (ii) e successfully colors itself c. In view of the definition of S_a and S_b, we have

$$\begin{aligned}
x(c,e') &= \frac{1}{p^2}\prod_{e''\in S_a\setminus S_b}(1-1/p)\prod_{e''\in S_b\setminus S_a}(1-1/p)\prod_{e''\in S_a\cap S_b}(1-2/p)\\
&= \frac{1}{p^2}(1-1/p)^{|S_a\setminus S_b|}(1-1/p)^{|S_b\setminus S_a|}(1-2/p)^{|S_a\cap S_b|}\\
&= \frac{1}{p^2}(1-1/p)^{|S_a\setminus S_b|}(1-1/p)^{|S_b\setminus S_a|}(1-1/p)^{2|S_a\cap S_b|}\left(1-O\left(\frac{|S_a\cap S_b|}{p^2}\right)\right)\\
&= \frac{1}{p^2}(1-1/p)^{|S_a|+|S_b|}(1-O(1/p)) \qquad \text{(Notice that } |S_a\cap S_b| < t = \Theta(p).\text{)}\\
&= \frac{1}{p^2}(1-1/p)^{4t-O(1)}(1-O(1/p))\\
&= \frac{1}{p^2}(1-1/p)^{4t} \pm O(1/p^3).
\end{aligned}$$

We now calculate $\mathrm{E}[z^{a,b}_e]$ and show that $\mathrm{E}[|R^\diamond|] = |R| - \mathrm{E}[z] = t^\diamond \pm O(1)$.

$$\begin{aligned}
\mathrm{E}[z^{a,b}_e] &= \sum_{\substack{(c,e')\,:\,e' \text{ incident to } e,\\ c^\bullet\in\Psi(e'),\, c\in\Psi(e)-\{c^\bullet\}}} x(c,e') \qquad \text{(union of disj. events)}\\
&= (t-1)(p-1)\cdot\left(\frac{1}{p^2}(1-1/p)^{4t} \pm O(1/p^3)\right)\\
&= \frac{t}{p}(1-1/p)^{4t} \pm O(1/p).
\end{aligned}$$

$$\begin{aligned}
\mathrm{E}[|R^{\diamond}|] &= |R| - \mathrm{E}[z] \\
&= t - \sum_{e \in R} \left(\mathrm{E}[z_e^a] + \mathrm{E}[z_e^b] - \mathrm{E}[z_e^{a,b}] \right) \\
&= t \cdot \left(1 - \frac{t}{p}(1-1/p)^{2t} - (1-1/p)^{2t} + \frac{t}{p}(1-1/p)^{4t} \pm O(1/p) \right) \\
&= t \cdot \left(1 - \frac{t}{p}(1-1/p)^{2t} - (1-1/p)^{2t} + \frac{t}{p}(1-1/p)^{4t} \right) \pm O(1) \\
&= t^{\diamond} \pm O(1). && \text{Definition of } t^{\diamond}
\end{aligned}$$

Concentration Bound. We have established that $|R^{\diamond}|$ has the correct expectation and now need to prove that it has sufficiently good concentration around that expectation. The analysis here becomes more complicated, because we have to consider the colors selected in some 3-neighborhood. The *palette size* and *degree* analyses focussed only on 2-neighborhoods.

Based on the definition of z_e^a and z_e^b, we define the following sets.

- Recall that $R = N_{c^\bullet}(v^\bullet)$. Let R_1 be the set of all edges e such that (i) $e \notin R$, (ii) $c^\bullet \in \Psi(e)$, and (iii) e is incident to some edge in R. Similarly, let R_2 be the set of all edges e such that (i) $e \notin R \cup R_1$, (ii) $c^\bullet \in \Psi(e)$, and (iii) e is incident to some edge in R_1. Notice that the value z_e^a, for any $e \in R$, is determined by the information about which edges in $R \cup R_1 \cup R_2$ select $c^\bullet$. We write $\alpha = |R \cup R_1 \cup R_2|$.
- Let R' be the set of all edges e' such that (i) $e' \notin R$ and (ii) there exists $e \in R$ such that $\Psi(e) \cap \Psi(e') - \{c^\bullet\} \neq \emptyset$. Notice that the the value z_e^b, for any $e \in R$, is determined by the colors selected by the edges in $R \cup R'$. We write $\beta = |R \cup R'|$.

For each $e \in R$, z_e^a is simply the summation of $z_{e',c^\bullet}$ over all edges $e' \in R_1$ incident to e. For each $e'' \in R_2$, we write $w(e'')$ to denote the number of edges in R_1 incident to e''. Intuitively, $w(e'')$ measures the influence of $\mathrm{Color}^\star(e'')$ on $\sum_{e \in R} z_e^a$.

We consider the sequence of random variables $(X_1, \ldots, X_{\alpha+\beta})$, where the initial α random variables reveal which edges in $R \cup R_1 \cup R_2$ select the color $c^\bullet$ according to the ordering R_2, R_1, R, and the remaining β random variables reveal the colors selected by the edges in $R \cup R'$ according to the ordering R', R. We let $z = f(X_1, \ldots, X_{\alpha+\beta})$ in Theorem 13. To prove the desired concentration bound $\Pr[z < \mathrm{E}[z] - s] = \exp(-\Omega(s^2/t))$, it suffices to show that we can set $M = O(1)$ and σ_i^2 such that $\sum_{i=1}^{\alpha+\beta} \sigma_i^2 = O(t)$. In what follows, we analyze the effect of exposing the value of X_i, given that all variables in $\mathbf{X}_{i-1}$ have been fixed.

Revealing whether $c^\bullet$ is Selected by an Edge in $R \cup R_1 \cup R_2$. Consider the case where X_i reveals whether $c^\bullet$ is selected by the edge $e^\star \in R \cup R_1 \cup R_2$. Notice that X_i is binary, and recall that $D_i = \mathrm{E}[z|\mathbf{X}_i] - \mathrm{E}[z|\mathbf{X}_{i-1}]$. There are at most two distinct outcomes of $D_i|\mathbf{X}_{i-1}$, in which one occurs with probability $1/p$. Thus, by Lemma 20, we have

$$\mathrm{Var}[D_i|\mathbf{X}_{i-1}] \leq \left(\max_{X_i} D_i|\mathbf{X}_{i-1} - \min_{X_i} D_i|\mathbf{X}_{i-1} \right)^2 /p = O(\max_{\mathbf{X}_i} |D_i|^2/p).$$

Thus, to achieve $\sum_{i=1}^{\alpha} \sigma_i^2 = O(t)$ and $M = O(1)$ it suffices to show the following:

- For the case $e^\star \in R_2$, we must prove $|D_i| = O(w(e^\star)/p)$.[14] Since $w(e^\star) < t = \Theta(p)$, $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O((w(e^\star)/p)^2/p) = O(w(e^\star)/p^2)$, so we can set $\sigma_i^2 = O(w(e^\star)/p^2)$.

[14] Intuitively, if $e^\star$ chooses color $c^\bullet$, it prevents $w(e^\star)$ edges in R_1 from successfully coloring themselves $c^\bullet$, but the prior probability of these edges coloring themselves $c^\bullet$ was only $O(1/p)$, hence the total influence on the expectation of z should be $O(w(e^\star)/p)$.

- For the case $e^\star \in R \cup R_1$, we must prove $|D_i| = O(1)$. Hence, we may set $\sigma_i^2 = \mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(1/p)$.

Notice that $\sum_{e^\star \in R_2} w(e^\star) < t^3$, $|R_1| < t^2$, and $|R| = t$. Thus, $\sum_{i=1}^{\alpha} \sigma_i^2 = O(t)$. With respect to the edge $e^\star \in R \cup R_1 \cup R_2$, we make the following definitions:

$$Y^a \stackrel{\text{def}}{=} \{e' \in R_1 \ : \ e' = e^\star \text{ or } e' \text{ is incident to } e^\star\}, \quad D_i^a \stackrel{\text{def}}{=} \sum_{e' \in Y^a} \left(\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] + \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}]\right),$$

$$Y^b \stackrel{\text{def}}{=} \{e \in R \ : \ e = e^\star \text{ or } e \text{ is incident to } e^\star\}, \qquad D_i^b \stackrel{\text{def}}{=} \sum_{e \in Y^b} |\,\mathrm{E}[z_e^b|\mathbf{X}_i] - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}]|.$$

Intuitively, Y^a and Y^b are the subsets of R_1 and R that are "relevant" to D_i in the following sense:

$$\mathrm{E}[z_{e'',c^\bullet}|\mathbf{X}_i] = \mathrm{E}[z_{e'',c^\bullet}|\mathbf{X}_{i-1}] \qquad \text{for all } e'' \in R_1 \setminus Y^a,$$
$$\mathrm{E}[z_{e'}^b|\mathbf{X}_i] = \mathrm{E}[z_{e'}^b|\mathbf{X}_{i-1}] \qquad \text{for all } e' \in R \setminus Y^b.$$

Our plan of bounding $|D_i|$ is as follows. First, we show that $|D_i| \le 4D_i^a + D_i^b$ in Claim 1, and then we bound D_i^a and D_i^b separately in Claims 2 and 3. The three claims together establish a desired bound on $|D_i|$.

Claim 1. $|D_i| \le 4D_i^a + D_i^b$.

Proof. We define the following notations:

$$P_1 \stackrel{\text{def}}{=} \{(e,e') \ : e \in R \setminus Y^b, e' \in Y^a, \ e \text{ is incident to } e'\},$$
$$P_2 \stackrel{\text{def}}{=} \{(e,e') \ : e \in Y^b, e' \in R_1 \setminus Y^a, \ e \text{ is incident to } e'\},$$
$$P_3 \stackrel{\text{def}}{=} \{(e,e') \ : e \in Y^b, e' \in Y^a, \ e \text{ is incident to } e'\},$$

$$Q_j \stackrel{\text{def}}{=} -\sum_{(e,e') \in P_j} \left(\mathrm{E}[z_{e',c^\bullet} \cdot z_e^b|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} \cdot z_e^b|\mathbf{X}_{i-1}]\right) \text{(for each } j = 1,2,3),$$
$$F_j \stackrel{\text{def}}{=} \sum_{e \in R} \left(\mathrm{E}[z_e^j|\mathbf{X}_i] - \mathrm{E}[z_e^j|\mathbf{X}_{i-1}]\right) \text{(for each } j = a,b).$$

The definitions of P_1, P_2, and P_3 depend on Y^a and Y^b, which depend on the edge $e^\star$. For instance, if $e^\star \in R$, then $Y^b = R$, which implies that $P_1 = \emptyset$. Recall that the edge $e^\star$ can be any edge in $R \cup R_1 \cup R_2$, and the proof of this claim applies to all choices of $e^\star \in R \cup R_1 \cup R_2$.

Notice that for any pair $(e \in R, e' \in R_1)$ such that e is incident to e' but $(e,e') \notin P_1 \cup P_2 \cup P_3$, we must have $\mathrm{E}[z_{e',c^\bullet} \cdot z_e^b|\mathbf{X}_i] = \mathrm{E}[z_{e',c^\bullet} \cdot z_e^b|\mathbf{X}_{i-1}]$ due to the definition of Y^a and Y^b. We rewrite the term D_i as follows:

$$\begin{aligned} D_i &= \mathrm{E}[z|\mathbf{X}_i] - \mathrm{E}[z|\mathbf{X}_{i-1}] \\ &= \sum_{e \in R} \left(\mathrm{E}[z_e|\mathbf{X}_i] - \mathrm{E}[z_e|\mathbf{X}_{i-1}]\right) \\ &= \sum_{e \in R} \left(\left(\mathrm{E}[z_e^a|\mathbf{X}_i] - \mathrm{E}[z_e^a|\mathbf{X}_{i-1}]\right) + \left(\mathrm{E}[z_e^b|\mathbf{X}_i] - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}]\right) - \left(\mathrm{E}[z_e^a \cdot z_e^b|\mathbf{X}_i] - \mathrm{E}[z_e^a \cdot z_e^b|\mathbf{X}_{i-1}]\right)\right) \end{aligned}$$

(recall that z_e^a is the summation of $z_{e',c^\bullet}$ over all edges $e' \in R_1$ incident to e)

$$= F_a + F_b - \sum_{(e,e') \,:\, e \in R,\ e' \in R_1,\ e' \text{ incident to } e} \left(\mathrm{E}[z_{e',c^\bullet} \cdot z_e^b|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet} \cdot z_e^b|\mathbf{X}_{i-1}]\right)$$

(any pair $(e, e') \notin P_1 \cup P_2 \cup P_3$ contributes zero to this summation)

$$= F_a + F_b + Q_1 + Q_2 + Q_3.$$

To prove this claim it suffices to show that (i) $|F_a + Q_1| \leq 2D_i^a$, (ii) $|F_b + Q_2| \leq D_i^b$, and (iii) $|Q_3| \leq 2D_i^a$. We expand F_a using the fact that z_e^a is the summation of $z_{e',c^\bullet}$ over all edges $e' \in R_1$ incident to e:

$$|F_a + Q_1| \leq \left| Q_1 + \sum_{(e,e') \,:\, e \in R,\ e' \in R_1,\ e' \text{ incident to } e} \left(\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}]\right) \right|.$$

Since any pair $(e, e') \notin P_1 \cup P_3$ contributes 0 in the summation,

$$\leq \left| Q_1 + \sum_{(e,e') \in P_1 \cup P_3} \left(\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}]\right) \right|,$$

and by definition of Q_1,

$$\leq \sum_{(e,e') \in P_1} \left| \mathrm{E}[z_{e',c^\bullet}(1 - z_e^b)|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}(1 - z_e^b)|\mathbf{X}_{i-1}] \right| + \sum_{(e,e') \in P_3} \left| \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}] \right|.$$

When $e \notin R \setminus Y^b$, $\mathrm{E}[z_e^b|\mathbf{X}_{i-1}] = \mathrm{E}[z_e^b|\mathbf{X}_i]$, so

$$\leq \sum_{(e,e') \in P_1} (1 - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}]) \left| \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}] \right| + \sum_{(e,e') \in P_3} \left| \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}] \right|,$$

and since $0 \leq 1 - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}] \leq 1$,

$$\leq \sum_{(e,e') \in P_1 \cup P_3} \left| \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}] \right|.$$

Finally, any edge $e' \in R_1$ is incident to at most two edges in R, so

$$\leq 2 \sum_{e' \in Y^a} \left| \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i] - \mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}] \right| \leq 2D_i^a.$$

For each $e \in Y^b$, we write $B(e)$ to denote the set of all edges $e' \in R_1 \setminus Y^a$ that are incident to e, i.e., $\{e\} \times B(e) \subseteq P_2$. Notice that $0 \leq \mathrm{E}[\sum_{e' \in B(e)} z_{e',c^\bullet}|\mathbf{X}_{i-1}] = \mathrm{E}[\sum_{e' \in B(e')} z_{e',c^\bullet}|\mathbf{X}_i] \leq 1$, since $e = \{v^\bullet, u\}$ and all edges in $B(e)$ share the vertex u, and so at most one could be successfully colored $c^\bullet$. By definition, none are incident to $e^\star$. We can now bound $|F_b + Q_2|$ as follows:

$$|F_b + Q_2| \leq \left| Q_2 + \sum_{e \in Y^b} \mathrm{E}[z_e^b|\mathbf{X}_i] - \mathrm{E}[z_e^b|\mathbf{X}_{i-1}] \right|.$$

According to the definition of $B(e)$ and Q_2,

$$\leq \sum_{e\in Y^b}\left|\mathrm{E}\left[z_e^b\left(1-\sum_{e'\in B(e)} z_{e',c^\bullet}\right)\middle|\mathbf{X}_i\right]-\mathrm{E}\left[z_e^b\left(1-\sum_{e'\in B(e)} z_{e',c^\bullet}\right)\middle|\mathbf{X}_{i-1}\right]\right|.$$

For every $e'\in R_1\setminus Y^a$, we have $\mathrm{E}\left[z_{e',c^\bullet}|\mathbf{X}_i\right]=\mathrm{E}\left[z_{e',c^\bullet}|\mathbf{X}_{i-1}\right]$, which implies

$$\begin{aligned}
&\leq \sum_{e\in Y^b}\left(1-\mathrm{E}\left[\sum_{e'\in B(e)} z_{e',c^\bullet}\middle|\mathbf{X}_{i-1}\right]\right)\cdot\left|\mathrm{E}[z_e^b|\mathbf{X}_i]-\mathrm{E}[z_e^b|\mathbf{X}_{i-1}]\right|\\
&\leq \sum_{e\in Y^b}\left|\mathrm{E}[z_e^b|\mathbf{X}_i]-\mathrm{E}[z_e^b|\mathbf{X}_{i-1}]\right|\\
&= D_i^b.
\end{aligned}$$

Our last task is to bound the absolute value of Q_3:

$$\begin{aligned}
|Q_3| &\leq \sum_{(e,e')\in P_3}\left(\mathrm{E}[z_{e',c^\bullet}\cdot z_e^b|\mathbf{X}_i]+\mathrm{E}[z_{e',c^\bullet}\cdot z_e^b|\mathbf{X}_{i-1}]\right)\\
&\leq \sum_{(e,e')\in P_3}\left(\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i]+\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}]\right).
\end{aligned}$$

Since any edge $e'\in R_1$ is incident to at most 2 edges in R,

$$\begin{aligned}
&\leq 2\sum_{e'\in Y^a}\left(\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_i]+\mathrm{E}[z_{e',c^\bullet}|\mathbf{X}_{i-1}]\right)\\
&\leq 2D_i^a.
\end{aligned}$$

□

Claim 2. *If $e^\star\in R_2$, then $D_i^a=O(w(e^\star)/p)$. If $e^\star\in R\cup R_1$, then $D_i^a=O(1)$.*

Proof. We first consider the case that $e^\star\in R_2$. In this case $|Y^a|=w(e^\star)$. Recall that $Y^a\subseteq R_1$, and so all $e\in Y^a$ have not yet decided whether to select $c^\bullet$ when X_i is revealed. Therefore, both $\mathrm{E}[z_{e,c^\bullet}|\mathbf{X}_i]$ and $\mathrm{E}[z_{e,c^\bullet}|\mathbf{X}_{i-1}]$ are within the range $[0,1/p]$, and so $D_i^a=O(w(e^\star)/p)$. Next, consider the case that $e^\star\in R\cup R_1$. All edges in Y^a must share a vertex with $e^\star$, and so at most two edges in Y^a can successfully color themselves by $c^\bullet$. Hence,

$$D_i^a\leq\sum_{e\in Y^a}\left(\mathrm{E}[z_{e,c^\bullet}|\mathbf{X}_i]+\mathrm{E}[z_{e,c^\bullet}|\mathbf{X}_{i-1}]\right)\leq 2+2=4=O(1).$$

□

Claim 3. *If $e^\star\in R_1\cup R_2$, then $D_i^b=O(1/p)$. If $e^\star\in R$, then $D_i^b=O(1)$.*

Proof. Recall that $z_e^b=\sum_{c\in\Psi(e)-\{c^\bullet\}} z_{e,c}$ for any edge $e\in Y^b$, and so

$$D_i^b\leq\sum_{e\in Y^b}\sum_{c\in\Psi(e)-\{c^\bullet\}}|\mathrm{E}[z_{e,c}|\mathbf{X}_i]-\mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]|.$$

We first show that $|\mathrm{E}[z_{e,c}|\mathbf{X}_i]-\mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]|=O(1/p^2)$ if $e^\star\neq e$. We write k_1 (respectively, k_2) to denote the number of edges incident to e that have decided to select $c^\bullet$ (respectively, have decided to not select $c^\bullet$) by the time X_i is revealed:

$$\mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]=\begin{cases}0 & (e\text{ has decided to select }c^\bullet)\\ \frac{1}{p-1}\cdot(1-1/p)^{2t-1-k_1-k_2}\cdot(1-1/(p-1))^{k_2} & (e\text{ has decided to not select }c^\bullet)\\ \frac{1}{p}\cdot(1-1/p)^{2t-1-k_1-k_2}\cdot(1-1/(p-1))^{k_2} & (e\text{ has not made any decision}).\end{cases}$$

In any case, $\mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}] = O(1/p)$. There are two possibilities of $\mathrm{E}[z_{e,c}|\mathbf{X}_i]$ based on X_i, i.e., whether $e^\star$ selects $c^\bullet$:

$$\mathrm{E}[z_{e,c}|\mathbf{X}_i] = \begin{cases} \mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]/(1-1/p) & (e^\star \text{ selects } c^\bullet) \\ \mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}] \cdot (1-1/(p-1))/(1-1/p) & (e^\star \text{ does not select } c^\bullet). \end{cases}$$

In any case, $|\mathrm{E}[z_{e,c}|\mathbf{X}_i] - \mathrm{E}[z_{e,c}|\mathbf{X}_{i-1}]| = O(1/p^2)$. We are now in a position to bound D_i^b. For the case that $e^\star \in R_1 \cup R_2$, we have $|Y^b| \le 2$ and $e^\star \notin Y^b$, and so $D_i^b \le 2 \cdot (p-1) \cdot O(1/p^2) = O(1/p)$. For the case that $e^\star \in R$, we have $|Y^b| = |R| = t$ and $e^\star \in Y^b$, and so $D_i^b \le 1 + (t-1) \cdot (p-1) \cdot O(1/p^2) = O(1)$. □

Revealing the Color Selected by an Edge in $R \cup R'$. Next, we analyze the effect of exposing the value of X_i, where $\alpha < i \le \alpha + \beta$, given that all variables in $\mathbf{X}_{i-1}$ have been fixed.

Observe that z_e^a, for all $e \in R$, are already determined by $\{X_j : j \in [\alpha]\}$. If $z_e^a = 1$, then $z_e = 1$ regardless of the value of z_e^b; if $z_e^a = 0$, then $z_e = z_e^b$. For those edges $e \in R$ such that z_e is not determined by $\{X_j : j \in [\alpha]\}$, the random variable $z_e = z_e^b$ behaves the same as z_e in the analysis of concentration of vertex degree, so the analysis in Appendix A.1 can be applied here (think of $S = R$ and $S' = R'$).

In more detail, for each edge $e' \in R'$, we define $w'(e')$ as $\sum_{e \in R,\ e' \text{ incident to } e} |\Psi(e') \cap \Psi(e) - \{c^\bullet\}|$. We have $\sum_{e' \in R'} w'(e') \le |R|(p-1)(t-1) < pt^2$. Now consider the color $X_i = \mathrm{Color}^\star(e^\star)$ selected by the edge $e^\star \in R \cup R'$. From the analysis in Appendix A.1, we infer the following:

- If $e^\star \in R'$, then $|D_i| = O(1)$ and $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(w'(e^\star)/(pt))$. Hence, we can set $\sigma_i^2 = O(w'(e^\star)/(pt))$.
- If $e^\star \in R$, then $|D_i| = O(1)$ and $\mathrm{Var}[D_i|\mathbf{X}_{i-1}] = O(1)$. Hence, we can set $\sigma_i^2 = O(1)$.

Thus, $\sum_{j=\alpha+1}^{\alpha+\beta} \sigma_i^2 = O(t)$, as desired.

REFERENCES

[1] N. Alon, L. Babai, and A. Itai. 1986. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algor.* 7, 4 (1986), 567–583.

[2] E. Arjomandi. 1982. An efficient algorithm for colouring the edges of a graph with $\Delta + 1$ colours. *Info. Syst. Operat. Res.* 20, 2 (1982), 82–101.

[3] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. 1989. Network decomposition and locality in distributed computation. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science (FOCS'89)*. 364–369.

[4] A. Balliu, S. Brandt, Y.-J. Chang, D. Olivetti, M. Rabie, and J. Suomela. 2019. The distributed complexity of locally checkable problems on paths is decidable. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC'19)*.

[5] A. Balliu, S. Brandt, D. Olivetti, and J. Suomela. 2018. Almost global problems in the LOCAL model. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC'18)*. 9:1–9:16. DOI: https://doi.org/10.4230/LIPIcs.DISC.2018.9

[6] A. Balliu, J. Hirvonen, J. H. Korhonen, T. Lempiäinen, D. Olivetti, and J. Suomela. 2018. New classes of distributed time complexity. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC'18)*. 1307–1318. DOI: https://doi.org/10.1145/3188745.3188860

[7] A. Balliu, J. Hirvonen, D. Olivetti, and J. Suomela. 2019. Hardness of minimal symmetry breaking in distributed computing. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC'19)*.

[8] L. Barenboim. 2015. Deterministic $(\Delta + 1)$-coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'15)*. 345–354. DOI: https://doi.org/10.1145/2767386.2767410

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

[10] L. Barenboim and M. Elkin. 2013. Distributed deterministic edge coloring using bounded neighborhood independence. *Distrib. Comput.* 26, 5 (Oct. 2013), 273–287.

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

[12] L. Barenboim, M. Elkin, and T. Maimon. 2017. Deterministic distributed $(\Delta + o(\Delta))$-edge-coloring, and vertex-coloring of graphs with bounded diversity. In *Proceedings of the 36th ACM Symposium on Principles of Distributed Computing (PODC'17)*. 175–184.

[13] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. 2016. The locality of distributed symmetry breaking. *J. ACM* 63, 3 (2016).

[14] J. Beck. 1991. An algorithmic approach to the Lovász local lemma. I. *Random Struct. Algor.* 2, 4 (1991), 343–366.

[15] B. Bollobás. 1978. *Extremal Graph Theory*. London Mathematical Society Monographs, Vol. 11. Academic Press, London.

[16] S. Brandt, O. Fischer, J. Hirvonen, B. Keller, T. Lempiäinen, J. Rybicki, J. Suomela, and J. Uitto. 2016. A lower bound for the distributed Lovász local lemma. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC'16)*. 479–488.

[17] S. Brandt, J. Hirvonen, J. H. Korhonen, T. Lempiäinen, P. R. J. Östergård, C. Purcell, J. Rybicki, J. Suomela, and P. Uznanski. 2017. LCL problems on grids. In *Proceedings of the 36th ACM Symposium on Principles of Distributed Computing (PODC'17)*. 101–110. DOI: https://doi.org/10.1145/3087801.3087833

[18] Y.-J. Chang, Q. He, W. Li, S. Pettie, and J. Uitto. 2018. The complexity of distributed edge coloring with small palettes. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'18)*. 2633–2652.

[19] Y.-J. Chang, T. Kopelowitz, and S. Pettie. 2019. An exponential separation between randomized and deterministic complexity in the LOCAL model. *SIAM J. Comput.* 48, 1 (2019), 122–143.

[20] Y.-J. Chang, W. Li, and S. Pettie. 2018. An optimal distributed $(\Delta + 1)$-coloring algorithm? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*. ACM, New York, NY, 445–456.

[21] Y.-J. Chang and S. Pettie. 2019. A time hierarchy theorem for the LOCAL model. *SIAM J. Comput.* 48, 1 (2019), 33–69.

[22] K.-M. Chung, S. Pettie, and H.-H. Su. 2017. Distributed algorithms for the Lovász local lemma and graph coloring. *Distrib. Comput.* 30 (2017), 261–280. Issue 4.

[23] R. Cole and U. Vishkin. 1986. Deterministic coin tossing with applications to optimal parallel list ranking. *Info. Control* 70, 1 (1986), 32–53.

[24] A. Czygrinow, M. Hanckowiak, and M. Karonski. 2001. Distributed $O(\Delta \log n)$-edge-coloring algorithm. In *Proceedings of the European Symposium on Algorithms (ESA'01)*. 345–355.

[25] X. Dahan. 2014. Regular graphs of large girth and arbitrary degree. *Combinatorica* 34, 4 (2014), 407–426. DOI: https://doi.org/10.1007/s00493-014-2897-6

[26] D. P. Dubhashi, D. A. Grable, and A. Panconesi. 1998. Near-optimal, distributed edge colouring via the nibble method. *Theor. Comput. Sci.* 203, 2 (1998), 225–251. DOI: https://doi.org/10.1016/S0304-3975(98)00022-X

[27] D. P. Dubhashi and A. Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press.

[28] D. P. Dubhashi and D. Ranjan. 1998. Balls and bins: A study in negative dependence. *J. Random Struct. Algs.* 13, 2 (1998), 99–124.

[29] M. Elkin, S. Pettie, and H.-H. Su. 2015. $(2\Delta - 1)$-edge coloring is much easier than maximal matching in the distributed setting. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*. 355–370.

[30] M. Fischer and M. Ghaffari. 2017. Sublogarithmic distributed algorithms for Lovász local lemma with implications on complexity hierarchies. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC'17)*. 18:1–18:16.

[31] M. Fischer, M. Ghaffari, and F. Kuhn. 2017. Deterministic distributed edge coloring via hypergraph maximal matching. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS'17)*. 180–191.

[32] P. Fraigniaud, M. Heinrich, and A. Kosowski. 2016. Local conflict coloring. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS'16)*. 625–634.

[33] H. N. Gabow, T. Nishizeki, O. Kariv, D. Leven, and O. Terada. 1985. *Algorithms for Edge-coloring Graphs*. Technical Report TRECIS-8501. Tohoku University.

[34] M. Ghaffari. 2016. An improved distributed algorithm for maximal independent set. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*. 270–277. DOI: https://doi.org/10.1137/1.9781611974331.ch20

[35] M. Ghaffari, D. G. Harris, and F. Kuhn. 2018. On derandomizing local distributed algorithms. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS'18)*. 662–673.

[36] M. Ghaffari, J. Hirvonen, F. Kuhn, and Y. Maus. 2018. Improved distributed delta-coloring. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC'18)*. 427–436.

[37] M. Ghaffari, J. Hirvonen, F. Kuhn, Y. Maus, J. Suomela, and J. Uitto. 2017. Improved distributed degree splitting and edge coloring. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC'17)*. 19:1–19:15.

[38] M. Ghaffari, F. Kuhn, and Y. Maus. 2017. On the complexity of local distributed graph problems. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC'17)*. 784–797. DOI: https://doi.org/10.1145/3055399.3055471

[39] M. Ghaffari, F. Kuhn, Y. Maus, and J. Uitto. 2018. Deterministic distributed edge-coloring with fewer colors. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*. ACM, New York, NY, 418–430. DOI: https://doi.org/10.1145/3188745.3188906

[40] M. Ghaffari and H.-H. Su. 2017. Distributed degree splitting, edge coloring, and orientations. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*. 2505–2523. DOI: https://doi.org/10.1137/1.9781611974782.166

[41] D. A. Grable. 1998. A large deviation inequality for functions of independent, multi-way choices. *Combinator. Prob. Comput.* 7, 1 (1998), 57–63.

[42] D. G. Harris. 2018. Distributed approximation algorithms for maximum matching in graphs and hypergraphs. *CoRR* abs/1807.07645.

[43] D. G. Harris, J. Schneider, and H.-H. Su. 2018. Distributed $(\Delta + 1)$-coloring in sublogarithmic rounds. *J. ACM* 65, 4 (Apr. 2018). DOI: https://doi.org/10.1145/3178120

[44] I. Holyer. 1981. The NP-completeness of edge-coloring. *SIAM J. Comput.* 10, 4 (1981), 718–720.

[45] H. J. Karloff and D. B. Shmoys. 1987. Efficient parallel algorithms for edge coloring problems. *J. Algor.* 8, 1 (1987), 39–52. DOI: https://doi.org/10.1016/0196-6774(87)90026-5

[46] A. Korman, J.-S. Sereni, and L. Viennot. 2013. Toward more localized local algorithms: Removing assumptions concerning global knowledge. *Distrib. Comput.* 26, 5–6 (2013), 289–308.

[47] F. Kuhn and R. Wattenhofer. 2006. On the complexity of distributed graph coloring. In *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing (PODC'06)*. 7–15.

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

[49] M. Luby. 1986. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.* 15, 4 (1986), 1036–1053.

[50] G. L. Miller and J. H. Reif. 1989. Parallel tree contraction—Part I: Fundamentals. *Adv. Comput. Res.* 5 (1989), 47–72.

[51] M. Molloy and B. Reed. 2000. Near-optimal list colorings. *Random Struct. Algor.* 17, 3–4 (2000), 376–402.

[52] M. Molloy and B. Reed. 2001. *Graph Colouring and the Probabilistic Method*. Springer.

[53] R. A. Moser and G. Tardos. 2010. A constructive proof of the general Lovász local lemma. *J. ACM* 57, 2 (2010). DOI: https://doi.org/10.1145/1667053.1667060

[54] M. Naor. 1991. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM J. Discrete Math.* 4, 3 (1991), 409–412. DOI: https://doi.org/10.1137/0404036

[55] M. Naor and L. J. Stockmeyer. 1995. What can be computed locally? *SIAM J. Comput.* 24, 6 (1995), 1259–1277. DOI: https://doi.org/10.1137/S0097539793254571

[56] A. Panconesi and A. Srinivasan. 1995. The local nature of Δ-coloring and its algorithmic applications. *Combinatorica* 15, 2 (1995), 255–280. DOI: https://doi.org/10.1007/BF01200759

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

[58] A. Panconesi and A. Srinivasan. 1997. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM J. Comput.* 26, 2 (1997), 350–368. DOI: https://doi.org/10.1137/S0097539793250767

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

[60] S. Pettie and H.-H. Su. 2015. Distributed algorithms for coloring triangle-free graphs. *Info. Comput.* 243 (2015), 263–280.

[61] J. Schneider and R. Wattenhofer. 2010. A new technique for distributed symmetry breaking. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing (PODC'10)*. 257–266.

[62] H.-H. Su and H. T. Vu. 2019. Towards the locality of Vizing's theorem. In *Proceedings of the 51st ACM Symposium on Theory of Computing (STOC'19)*. 355–364.

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

Received April 2018; revised June 2019; accepted August 2019