Towards the Locality of Vizing's Theorem

Hsin-Hao Su
Boston College
suhx@bc.edu

Hoa T. Vu
Boston College
vuhd@bc.edu

ABSTRACT

Vizing [36] showed that it suffices to color the edges of a simple graph using $\Delta + 1$ colors, where Δ is the maximum degree of the graph. However, up to this date, no efficient distributed edge-coloring algorithm is known for obtaining such coloring, even for constant degree graphs. The current algorithms that get closest to this number of colors are the randomized $(\Delta + \tilde{\Theta}(\sqrt{\Delta}))$-edge-coloring algorithm that runs in $\text{polylog}(n)$ rounds by [8] and the deterministic $(\Delta + \text{polylog}(n))$-edge-coloring algorithm that runs in $\text{poly}(\Delta, \log n)$ rounds by [20].

We present two distributed edge-coloring algorithms that run in $\text{poly}(\Delta, \log n)$ rounds. The first algorithm, with randomization, uses only $\Delta + 2$ colors. The second algorithm is a deterministic algorithm that uses $\Delta + O(\log n / \log\log n)$ colors. Our approach is to reduce the distributed edge-coloring problem into an online and restricted version of balls-into-bins problem. If ℓ is the maximum load of the bins, our algorithm uses $\Delta + 2\ell - 1$ colors. We show how to achieve $\ell = 1$ with randomization and $\ell = O(\log n / \log\log n)$ without randomization.

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; **Distributed algorithms**.

KEYWORDS

distributed algorithms, edge coloring, randomized algorithms

ACM Reference Format:
Hsin-Hao Su and Hoa T. Vu. 2019. Towards the Locality of Vizing's Theorem. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3313276.3316393

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
STOC '19, June 23–26, 2019, Phoenix, AZ, USA
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6705-9/19/06…$15.00
https://doi.org/10.1145/3313276.3316393

1 INTRODUCTION

Problem description. Given a simple graph $G = (V, E)$, a k-edge-coloring is a mapping $\phi : E \longrightarrow \{1, \ldots, k\}$ that maps each edge to a color where no two adjacent edges are mapped to the same color.

We study the edge-coloring problem in the distributed LOCAL model. In this model, vertices host processors and operate in synchronized rounds. In each round, each vertex sends a message of arbitrary size to each of its neighbors, receives messages from its neighbors, and performs local computations. The time complexity of an algorithm is defined to be the number of rounds used. In the end, each vertex produces its own answer. In the edge-coloring problem, the coloring of an edge uv can either be computed by vertex u or vertex v.

The LOCAL model aims to investigate the *locality* of a problem. An r-round local algorithm in the LOCAL model implies that each vertex only uses information in its r-neighborhood to compute the answer, and vice versa. Therefore, a faster algorithm in the LOCAL model would mean that each vertex uses less local information to compute the answer.

Computing edge-coloring in the distributed setting has applications in scheduling problems of wireless networks [17, 34]. It is usually the case that the quality of the solution in these applications depends on the number of colors being used. Therefore, we hope to minimize the number of colors while keeping the coloring to be locally computable.

Let Δ denote the maximum degree of G. It takes at least Δ colors to color the edges of G, since the incident edges to a vertex must be colored with distinct colors. Vizing showed that the edges in every simple graph G can be colored with $\Delta + 1$ colors [36]. The best sequential algorithm for obtaining such a coloring runs in $O(\min(\Delta m \log n, m\sqrt{n \log n}))$ time [2, 16]. However, in order to obtain a $(\Delta + 1)$-edge-coloring in the LOCAL model, the only known method up to this date is a $O(\text{diameter(G)})$ solution − a leader collects the topology of the whole graph and then computes the answer.

"How close can we get to this [Vizing's edge-coloring], while remaining in polylogarithmic-time?"

This is an open problem raised in [14]. The followings are the thresholds on the number of colors that have been encountered or tackled by existing algorithms.

$2\Delta - 1$ *Threshold.* $(2\Delta - 1)$ is a natural threshold to be investigated because $(2\Delta - 1)$ is the minimum number of colors that can be obtained by the greedy algorithm. Panconesi and Rizzi [30] gave a deterministic $(2\Delta - 1)$-edge-coloring algorithm that runs in $O(\Delta + \log^* n)$ rounds.

The $(2\Delta - 1)$-edge coloring problem translates to the $(\hat{\Delta} + 1)$-vertex-coloring problem on its line graph, $L(G)$, where $\hat{\Delta} = 2\Delta - 2$ is an upper bound on the maximum degree of $L(G)$. There have been extensive studies on the $(\Delta + 1)$-vertex-coloring problem [1, 3, 5–7, 9, 15, 22–28, 31]. All the results can be applied to obtain a $(2\Delta - 1)$-edge-coloring.

Getting a polylog(n)-rounds *deterministic* algorithm for $(2\Delta - 1)$-edge-coloring had been a major open problem. After the progress made in [19, 21] for $(2 + o(1))\Delta$-edge-coloring, the problem was settled recently by [14] and later improved in [18].

$\Delta + \tilde{\Theta}(\sqrt{\Delta})$ *Threshold.* Based on a randomized approach, Panconesi and Srinivasan [32] first gave an algorithm that uses $1.6\Delta + O(\log^{1+\delta} n)$ colors and runs in $O(\log n)$ rounds w.h.p. [1], where $\delta > 0$ is a constant. Later, Dubhashi, Grable, and Panconesi [12] gave an algorithm that uses $(1 + \epsilon)\Delta$ colors and runs in $O(\log n)$ rounds w.h.p. for constant ϵ and $\Delta > \log^{1+\delta} n$. Later, Elkin, Pettie, and Su [13] removed the requirement that $\Delta \geq \log^{1+\delta} n$ and showed that for constant ϵ, there exists Δ_ϵ such that for all $\Delta \geq \Delta_\epsilon$, a $(1 + \epsilon)\Delta$-edge-coloring can be obtained w.h.p. by solving $O(\log^* \Delta)$ Lovasz Local Lemma (LLL) instances in the distributed setting. An LLL instance can be computed in polylog(n) rounds in the distributed setting [10]. This line of research culminated in the work of Chang et al. [8], who showed that for *any* $\epsilon = \tilde{\Omega}(1/\sqrt{\Delta})$ (not necessarily a constant), it is possible to obtain a $(1 + \epsilon)\Delta$-edge-coloring w.h.p. by $O(\log(1/\epsilon))$ invocations of distributed LLL with an additive poly(log log n) rounds.

The algorithm of [8] allows one to find a $\Delta+\tilde{O}(\sqrt{\Delta})$-edge-coloring in polylog(n) rounds. However, this number of colors seems to be the limit of such randomized approach. An intuitive reason is because in these coloring algorithms, each edge is colored with a (almost) random color out of the $(1 + \epsilon)\Delta$ colors. Once an edge is assigned a color, it is permanently colored. Consider a stage of the algorithm where an edge uv is still uncolored but all the incident edges are colored. Let $M(x)$ denote the colors that are not assigned to any incident edges of x. The size of $M(u)$ and $M(v)$ are about $\epsilon\Delta$. Note that $M(u) \cap M(v)$ must be non-empty in order to color uv. The $\epsilon\Delta$ colors in $M(u)$ and $M(v)$ are likely to be "randomly" sampled out of the $(1 + \epsilon)\Delta$ colors. Therefore, the expected number of colors in $M(u) \cap M(v)$ is $O(\epsilon^2\Delta)$. This implies ϵ has to be at least $\Omega(1/\sqrt{\Delta})$.

Very recently, in the dynamic setting, Duan et al. [11] gave a randomized $(1 + \epsilon)\Delta$-edge-coloring algorithm with $O(\text{poly}(1/\epsilon, \log n))$ amortized update time when $\epsilon = \Omega(\sqrt{\Delta}/\log n)$. Incidentally, the number of colors they are able to obtain is also capped at the $\Delta + \Theta(\sqrt{\Delta})$ threshold.

Δ + poly log(n) *Threshold.* Recently, Ghaffari et al. [20] gave a poly($(1/\epsilon)$, log n)-round deterministic algorithm that uses $(1 + \epsilon)\Delta$ rounds provided that $\frac{\epsilon}{\log(1/\epsilon)} = \Omega(\log n/\Delta)$. A corollary of the result is that it is possible to obtain $\Delta + O(\log n \cdot \log\left(2 + \frac{\Delta}{\log n}\right))$ colors in poly(Δ, log n) rounds. This breaks the $\Delta + \tilde{\Theta}(\sqrt{\Delta})$ barrier, provided that $\Delta = \omega(\log^c n)$ for a large enough c. However, when Δ is small, say in a constant degree graph, it is still unclear what the minimum possible number of colors is to color the graph in polylog(n) rounds.

[1]W.h.p. denotes with high probability, which means with probability at least $1 - 1/n^c$ for some constant c.

1.1 Our Results

We show that by using poly(Δ, log n) rounds, the number of colors can be pushed down to $\Delta + 2$, which is merely just one more color than that in Vizing's Theorem.

Theorem 1.1. *There exists a randomized distributed $(\Delta + 2)$-edge-coloring algorithm that runs in* poly(Δ, log n) *rounds w.h.p. Furthermore, for any $\epsilon \geq (2/\Delta)$, this can be turned into a randomized distributed $(1 + \epsilon)\Delta$-edge-coloring algorithm that runs in* poly($(1/\epsilon)$, log n) *rounds w.h.p.*

Chang et al. [8] showed that 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. Our algorithms belongs to such category and so a polynomial dependency on Δ and log n is necessary.

Since the edge-coloring problem is locally-checkable, by using the derandomization result from [18], we can convert the randomized algorithm to a deterministic algorithm with a $2^{O(\sqrt{\log n})}$ factor blow-up.

Corollary 1.2. *For any $\epsilon \geq (2/\Delta)$, there exists a deterministic distributed $(1+\epsilon)\Delta$-edge-coloring algorithm that runs in* poly($(1/\epsilon)$) $\cdot$ $2^{O(\sqrt{\log n})}$ *time.*

For deterministic algorithms, we also show that it is possible to obtain a $\Delta + O(\log n/\log\log n)$ coloring in $O(\Delta^6 \cdot \log^{O(1)} n)$ rounds.

Theorem 1.3. *For any $\lambda > 1$, there exists a deterministic distributed $\Delta + O(\log n/(\log\lambda + \log\log n))$-edge-coloring algorithm that runs in $O(\lambda \cdot \Delta^6 \cdot \log^{O(1)} n)$ rounds. Furthermore, this can be turned into a deterministic distributed $(1 + \epsilon)\Delta$-edge-coloring algorithm that runs in $O(\lambda \cdot (1/\epsilon)^6 \cdot \log^{O(1)} n)$ rounds, provided $\epsilon \geq K \log n/(\Delta(\log\lambda + \log\log n))$ for some constant $K > 0$.*

Comparing Theorem 1.3 (with $\lambda = O(1)$) to the deterministic algorithms in [20], we are using fewer colors ($\Delta + O(\log n/\log\log n)$ v.s. $\Delta + O(\log n \cdot \log(2 + \frac{\Delta}{\log n}))$) and fewer number of rounds ($\tilde{O}(\Delta^6)$ v.s. $\tilde{O}(\Delta^9)$). For bipartite graphs, we show that it is possible to further improve the number of rounds to just $\tilde{O}(\Delta^4)$. We summarize what coloring can be obtained in poly(Δ, log n) in Table 1.

The second statements of Theorem 1.1 and Theorem 1.3 can be obtained by applying the degree splitting algorithm of [19] to recursively split the edges into subgraphs whose maximum degrees are upper bounded by $O(1/\epsilon)$ (see Lemma 3.4). Then we can apply the algorithm on each subgraph to obtain a poly($(1/\epsilon)$, log n) algorithm.

1.2 A High-Level Description of Our Method

To get a poly(Δ, log n)-round algorithm, we employ the construction of Vizing's coloring in the distributed setting. In Vizing's Theorem, it was shown that it is possible to align the endpoints of an uncolored edge with the same missing color. Thus, the edge can be colored with the missing color. The alignment can be done by shifting a fan and augmenting along an alternating path. The edges can be colored by this method one after another in a sequential manner.

	Previous	New	Range of Δ
Randomized	$\Delta + \tilde{O}(\sqrt{\Delta})$ [8]	$\Delta + 2$ Thm. 1.1	$O(\log^c n)$
	$\Delta + O(\log n \cdot \log(2 + \frac{\Delta}{\log n}))$ [20]		$\Omega(\log^c n)$
Deterministic	1.5Δ [20]		$O(\frac{\log n}{\log\log n})$
		$\Delta + O(\frac{\log n}{\log\log n})$ Thm. 1.3	$[\Omega(\frac{\log n}{\log\log n}), O(\log n)]$
	$\Delta + O(\log n \cdot \log(2 + \frac{\Delta}{\log n}))$ [20]	$\Delta + O(\log_\Delta n)$ Thm. 1.3	$[\Omega(\log n), 2^{O(\sqrt{\log n})}]$
		$\Delta + 2$ Cor. 1.2	$2^{\Omega(\sqrt{\log n})}$

Table 1: Minimum number of colors that can be obtained in poly$(\Delta, \log n)$ **rounds. The deterministic result on the fourth row is obtained by setting $\lambda = \Delta^\gamma$ for some constant $\gamma > 0$ in Theorem 1.3.**

However, in the distributed setting, there are two challenges. First, we have to color a large fraction of uncolored edges simultaneously. The difficulties arise when we color the edges together since there are dependencies among the coloring processes. Second, the number of rounds needed to augment along an alternating path must be at least proportional to its length. It is unclear if short alternating paths always exist (potentially the length of an alternating path can be $\Omega(n)$).

While the first challenge can be resolved by decomposing the coloring process into different phases and applying symmetry breaking techniques carefully, overcoming the second challenge relies on the fact all alternating paths are short. Instead of showing this is true, we chop up long alternating paths to stop the augmentation from propagating. This can be done by placing a blocking edge on it and augment on the portion of the path only from the starting vertex to the blocking edge. In the end, if each vertex is adjacent to at most ℓ blocking edges, then the remaining blocking edges can be colored with additional $2\ell - 1$ colors, because the maximum degree graph induced by the blocking edges is at most ℓ. Therefore, the problem reduces to the following load-balancing problem.

Balls into Bins v.s. Paths into Vertices. Consider the following version of balls-into-bins problem. Suppose there are n bins. The balls arrive in an online fashion. When a ball x arrives, it is given a size-T subset of the bins, B_x. We have to decide which bin in B_x to place x into. Each bin can only be the potential choice of at most t balls. The question is how to minimize the maximum load of the bins.

The problem of minimizing the maximum number of incident blocking edges per vertex can be thought as a balls-into-bins problem. In particular, each alternating path of length at least T is a ball and each vertex is a bin. Choosing the blocking edge on the first T edges is equivalent to choosing the bin (thus each ball occupies two bins, but we ignore this fact for now). We will show that each vertex can be passed by at most t alternating paths throughout the algorithm.

If $t \le T/\lambda$, then by placing each ball into a random bin, we can show that w.h.p. the maximum load is $O(\log n/(\log\lambda + \log\log n))$. Moreover, such randomized algorithm can be derandomized by letting each ball go to the bin with the smallest cost (we will specify the cost function later), breaking ties arbitrarily. This is how we obtain our deterministic algorithm that uses additional $O(\log n/(\log\lambda + \log\log n))$ colors in Theorem 1.3. Note that the $\Theta(\log n/(\log\lambda + \log\log n))$ bound is tight for the balls-into-bins problem. Additional constraints must be considered in order to further reduce the load.

To achieve maximum load of 1, we modify our algorithm such that we place the ball into one of the *non-empty* bins randomly. If there is always at least one non-empty bin in B_x for every ball x, it guarantees a maximum load of one. Recall that each ball is a path in our scenario. There are at most $n\Delta^{T-1}$ different paths of length T. If T is large enough (T = poly$(\Delta, \log n) \cdot t$), we can show that w.h.p. for *every* path, at most a constant fraction of the bins on the path are occupied (throughout the algorithm) by taking a union bound over all possible paths of length T. Since there are at least a constant fraction of empty bins, each ball is always guaranteed to land in an non-empty bin. In the end of the algorithm, the blocking edges induce a matching. After recoloring the matching with a new color, we obtain a $(\Delta + 2)$-edge-coloring.

2 NOTATION AND VIZING'S THEOREM

Consider a partial edge coloring $\phi : E \to \{\perp, 1, 2, \ldots, \Delta + 1, \star\}$ of a graph G. A partial coloring is *proper* if the subgraph induced by each color $i \in \{1, 2, \ldots, \Delta + 1\}$ is a matching. We say an edge e is uncolored if $\phi(e) = \perp$. The color $\star$ is a special color that denotes the color of blocking edges. Let $M(v) \subseteq \{1, 2, \ldots, \Delta + 1\}$ denote the *missing colors* of v (i.e. the colors that are not assigned to the incident edges of v). Note that $M(v)$ is always non-empty. Fix $m(v)$ to be any missing color of v.

Fix a partial edge-coloring ϕ. Suppose that v is missing α, i.e. $\alpha \in M(v)$, and there is at least one uncolored edge that is incident to v. The *fan* F_v with respect to ϕ is constructed as follow. Let $e_1 = vx_1$ be an uncolored edge incident to v. Our goal is to color the edge vx_1. If $m(x_1) \notin M(v)$, then there must be an edge vx_2 with the color $m(x_1)$. In particular, we construct vx_i as the edge with the color $m(x_{i-1})$. We stop this process if for some i, we have that $m(x_i) \in M(v)$ or some previous edge vx_j currently has color $m(x_i)$.

More formally, given $e_i = vx_i$, we construct edge e_{i+1} if both of the following two conditions do not occur: (i) $m(x_i) \in M(v)$. (ii) $m(x_i) \in \{m(x_1), \ldots, m(x_{i-1})\}$. Furthermore, e_{i+1} is constructed as the edge that is incident to v and colored in $m(x_i)$. We repeatedly construct edges $e_1 \ldots e_k$ for some $k \ge 1$ until it is not possible to do it further. The vertex set of F_v consists of the *center* v and the *leaves* $x_1, \ldots, x_k$. The edge set of F_v is $\{e_1, \ldots e_k\}$. We say k is the degree of F_v, $\deg(F_v)$. A fan F_v is an $\alpha\beta$-fan if $\alpha \in M(v)$ and

$m(x_{\deg(F_v)}) = \beta$. We again note that this procedure is well-defined since with $\Delta + 1$ colors in the palette, $m(x_i)$ is always non-empty.

We emphasize that when we use F_v to refer to a fan, that is a shorthand to say that it has v as the center. The notation F_v alone however does not uniquely define a fan since we may grow multiple fans centering at v throughout the algorithm.

The operation $\texttt{shift}(F_v, i)$ recolors the edges of F_v as follow. For each $1 \leq j \leq i - 1$, color edge e_j with $m(x_j)$. We leave the color of e_i to be the same. An $\alpha\beta$-alternating path is a path whose edges are colored in α and β alternatively and it is *maximal* of such paths. That is, no path whose edges are alternating between α and β contains it properly. Given an $\alpha\beta$-alternating path P, the operation $\texttt{augment}(P)$ switches the colors from α to β and from β to α for each edge on P. Note that two $\alpha\beta$-alternating paths cannot possibly intersect with each other.

2.1 Vizing's Theorem and Fan Repair

Vizing's Theorem shows that the uncolored edge in a fan can be properly colored if one recolors some other edges. This is done by using an augment operation and a shift operation. We call such a procedure *repairing* the fan. We adopt a version of the proof from [29], where it only uses one alternating path instead of potentially two. Given a fan F_v with edges $vx_1, \ldots, vx_k$. If $m(x_k) \in M(v)$, perform $\texttt{shift}(F_v, k)$ and color vx_k with $m(x_k)$ then every edge in F_v is now colored. Otherwise, it must be the case that there exists j, where $1 \leq j \leq k - 2$, such that $m(x_k) = m(x_j)$. Let P_v be the $\alpha\beta$-alternating path that starts at v. Note that the first edge of P_v is $e_{j+1} = vx_{j+1}$. There are two cases.

Case 1: P_v does not end at x_j. We perform $\texttt{shift}(F_v, j + 1)$ followed by $\texttt{augment}(P_v)$. After the $\texttt{shift}(F_v, j + 1)$ operation, there is only one conflict – both vx_j and vx_{j+1} are colored with β. This is resolved by $\texttt{augment}(P_v)$, which changes the color of vx_{j+1} to α. Moreover, $\texttt{augment}(P_v)$ does not create new conflicts, since its ending vertex must be missing either α or β (depending on the color of its last edge) before the $\texttt{shift}(F_v, j + 1)$ operation. The operation $\texttt{shift}(F_v, j + 1)$ does not color any edge with α or β except for xv_j. Since P_v does not end at x_j as assumed, the partial coloring remains proper after augmenting along P_v.

Case 2: P_v ends at x_j (Figure 1a), we perform $\texttt{augment}(P_v)$ first. After the augmentation, now v is missing β and x_j is missing α (Figure 1b). We let $m(x_j) = \alpha$ (there can be multiple missing colors, but we choose $m(x_j)$ to be α). Under the new partial coloring and $m(\cdot)$-values, F_v is still a fan of v by definition. Moreover, $m(x_k) = \beta \in M(v)$. We then perform $\texttt{shift}(F_v, k)$ and color vx_k with β (Figure 1c).

Therefore, given a fan, we can repair it and thus decrease the number of uncolored edges. Sequentially, if we fix fans one by one, we will be able to color all the edges. To be able to do this efficiently in the distributed setting, we need to address two challenges.

- We need to be able to color a large portion of uncolored edges at the same time.
- The number of rounds needed to augment along a path is proportional to the length of the path. It is unclear if all alternating paths are short (i.e. $\text{poly}(\Delta, \log n)$).

3 ALGORITHM

To tackle the first challenge, we divide the fans into different color classes so that the fans in the same color class can be fixed without interfering with each other. The rough idea is to take advantage of the fact that any two $\alpha\beta$-alternating paths do not intersect and the fact that if u and v are in the same color class of a 2-hop coloring then F_u and F_v do not intersect. Therefore, we can repair all $\alpha\beta$-fans whose centers are in the same color class of a 2-hop coloring together.

To overcome the second challenge, the idea is to truncate long alternating paths. Given an alternating path $P = (v_0, v_1, \ldots v_k)$, let $P(i)$ denote the truncated version of P defined as follow

$$P(i) = \begin{cases} P, & \text{if } k \leq i. \\ (v_0, v_1, \ldots, v_i), & \text{otherwise.} \end{cases}$$

We will truncate alternating paths at length $T = \text{poly}(\Delta, \log n)$, where the actual value of T will be specified later. Instead of doing the $\texttt{augment}$ operation on P, we will choose one edge in $P(T)$ to be the blocking edge to block the augmentation propagating down the path. Note that we only need to put a blocking edge on $P(T)$ if $|P| > T$, since if $|P| \leq T$, we can augment P in $O(T)$ rounds.

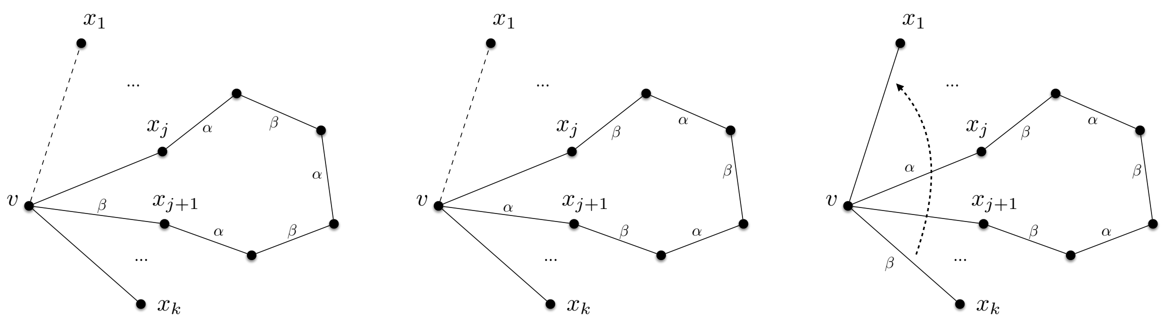
3.1 The Framework

We describe our algorithm in Algorithm 1. Let G^2 denote the distance-2 graph of G where $(u, v) \in G^2$ if $\text{dist}_G(u, v) \leq 2$. Since the maximum degree of G^2 is at most $\Delta + \Delta \cdot (\Delta - 1) = \Delta^2$, we can color G^2 using $\Delta^2 + 1$ colors. This can be done in $O(\Delta^2 + \log^* n)$ rounds [6].

Algorithm 1 Distributed Fan Repair

```
1:  Obtain a 2-hop coloring using $\Delta^2 + 1$ colors.
2:  while there exists uncolored edges do
3:      for $i = 1, 2, \ldots, \Delta^2 + 1$ do
4:          for $\alpha$ = color $1, 2, \ldots, \Delta + 1$ do
5:              Let $V_{i,\alpha}$ denote color-$i$ vertices that misses $\alpha$ and have at
                least one uncolored edges.
6:              Let $\mathcal{F}_{i,\alpha}$ denote the set of fans grown from each
                vertex $V_{i,\alpha}$.
7:              for $\beta$ = color $1, 2, \ldots, \Delta + 1$ do
8:                  Let $\mathcal{F}_{i,\alpha,\beta}$ denote the set of $\alpha\beta$-fans in $\mathcal{F}_{i,\alpha}$.
9:                  Build a conflict graph $G_{\mathcal{F}_{i,\alpha,\beta}}$ for $\mathcal{F}_{i,\alpha,\beta}$ and
                    color $G_{\mathcal{F}_{i,\alpha,\beta}}$ using $O(1)$ colors.
10:                 Let $\mathcal{F}_{i,\alpha,\beta,j} \subseteq \mathcal{F}_{i,\alpha,\beta}$ denote the fans that are
                    colored in $j$.
11:                 for $j$ = color $1, 2, \ldots, O(1)$ do
12:                     Repair each fan in $\mathcal{F}_{i,\alpha,\beta,j}$ that has not
                        been destroyed.
```

The outermost while loop repeats until there is no uncolored edge. We will show in Lemma 3.1 that each iteration colors a constant fraction of the uncolored edges. Thus, the while loop uses $O(\log n)$ iterations. Inside the while loop, we iterate through each color of G^2 (Line 3). For each color class, we further iterate through all possible values for α (Line 4). Let $V_{i,\alpha}(\phi)$ denote the set of color-i vertices that have at least one incident uncolored edge and miss α w.r.t. the partial coloring ϕ. We often omit the parameter ϕ and write it as $V_{i,\alpha}$ when the reference is w.r.t. the current coloring.



(a) P_v **ends at** x_j **(b)** $\mathtt{augment}(P_v)$ **(c)** $\mathtt{shift}(F_v, k)$ **and color** vx_k **with** β

Figure 1

For each $v \in V_{i,\alpha}$, we grow a fan F_v centered at v. Let $\mathcal{F}_{i,\alpha}$ be a collection of all such fans. It is guaranteed that the fans do not intersect, since $V_{i,\alpha}$ is an independent set in G^2. Next, we iterate β from color 1 to $\Delta + 1$. Let $\mathcal{F}_{i,\alpha,\beta}$ denote all the $\alpha\beta$-fans in $\mathcal{F}_{i,\alpha}$. Ideally, we want to be able to repair all fans in $\mathcal{F}_{i,\alpha,\beta}$ simultaneously, since their alternating paths do not intersect. However, it is possible that an alternating path of a fan F_v ends at a node of another fan $F_w \in \mathcal{F}_{i,\alpha,\beta}$. In this case, we might not be able to repair them together, because the $\mathtt{augment}(P_v)$ step may *destroy* the structure of F_w (i.e. change $M(x)$ for $x \in F_w$ or change the color of the edges in F_w). Note that if P_v crosses a node in F_w and does not end at a node in F_w, augmenting along P_v does not affect the structure of F_w, since in this case, P_v cannot contain any edge of F_w. In other words, if P_v crosses but does not end at a node $x \in F_w$, then $m(x) \neq \alpha$ and $m(x) \neq \beta$. Therefore, augmenting $P(v)$ or the prefix of P_v before the blocking edge (if $|P_v| > T$) cannot change $m(x)$.

To resolve this issue, we consider the directed graph $G_{\mathcal{F}_{i,\alpha,\beta}} = (V_{\mathcal{F}_{i,\alpha,\beta}}, E_{\mathcal{F}_{i,\alpha,\beta}})$, where $V_{\mathcal{F}_{i,\alpha,\beta}} = \{F_v \mid F_v \in \mathcal{F}_{i,\alpha,\beta}\}$ and $(F_u, F_w) \in E_{\mathcal{F}_{i,\alpha,\beta}}$ if $|P_v| \leq T$ and P_v ends at any nodes of F_w. Note that we only consider $|P_v| \leq T$, because if $|P_v| > T$, we will only do modification on $P_v(T)$. Such modification will not affect the fans that intersect with $P_v(T)$.

Since the out-degree of every vertex in $G_{\mathcal{F}_{i,\alpha,\beta}}$ is at most 1, the arborcity of $G_{\mathcal{F}_{i,\alpha,\beta}}$ is $O(1)$. If we consider $G_{\mathcal{F}_{i,\alpha,\beta}}$ as an undirected graph, we can color $G_{\mathcal{F}_{i,\alpha,\beta}}$ using $O(1)$ colors in $O(\log n)$ rounds [4], if $G_{\mathcal{F}_{i,\alpha,\beta}}$ were the underlying communication graph. Since each edge in $G_{\mathcal{F}_{i,\alpha,\beta}}$ is stretched by a factor of at most T in G, we can simulate the coloring algorithm in $O(T \cdot \log n)$ rounds in G. Now we iterate through each color j of the coloring and process the fans colored in j together (Line 11). Since the alternating path of each fan colored in j does not end at any other fan colored in j, nor do the alternating paths intersect, we can repair them simultaneously.

Therefore, it is possible to repair all undestroyed fans in $\mathcal{F}_{i,\alpha,\beta,j}$ simultaneously, using a $\mathtt{shift}$ operation and possibly an $\mathtt{augment}$ operation for each fan. For a fan F_v with its alternating path P_v whose first edge is the edge between v and the $(j+1)$'th leave of F_v, if $|P_v| \leq T$, we repair the fan as outlined in Section 2. Otherwise, we select an edge of $P_v(T) = (v_0, v_1, \ldots, v_T)$ to be the *blocking edge*, say (v_{i-1}, v_i), where $1 \leq i \leq T$. We color (v_{i-1}, v_i) with a special color $\star$ and then perform $\mathtt{shift}(v, j+1)$ followed by $\mathtt{augment}(P_v(i-1))$ (Case 1 of the repairing step). After the repairing step, the uncolored edge in the fan becomes colored and the partial coloring remains proper. We will show that after $O(\log n)$ iterations of the outermost while loop, all the edges become colored (some possibly in color $\star$).

Lemma 3.1. *In each iteration of the while loop, at least a constant fraction of uncolored edges become colored. Thus, it takes $O(\log n)$ iterations to color all the edges.*

Proof. Consider an iteration of the while loop. Let $\phi_0^{-1}(\perp)$ denote the set of uncolored edges at the beginning of the while loop. Let $\phi_{i,\alpha}$ be the partial coloring during the i'th iteration of the outer **for loop** at Line 3 and at the beginning of α'th iteration of the inner **for loop** at Line 4. Let ϕ_i denote the partial coloring at the beginning of i'th iteration of the outer **for loop** at Line 3.

For each $uv \in \phi_0^{-1}(\perp)$, we orient the edge from u to v if the color of u is smaller than that of v; otherwise we orient it from v to u. We will show that

$$\sum_{\alpha=1}^{\Delta+1} |\mathcal{F}_{i,\alpha}| \geq \sum_{u \in V_i} \mathrm{outdeg}(u)/2,$$

where V_i denotes the set of color-i vertices.

For each $v \in V_i$, let $C(v)$ denote the first outdeg(v) missing colors of v w.r.t. ϕ_i so that $|C(v)| = \mathrm{outdeg}(v)$. For each $\alpha \in C(v)$, if $v \in V_{i,\alpha}(\phi_{i,\alpha})$, then a fan centered at v will be created during iteration α (Recall that $V_{i,\alpha}(\phi_{i,\alpha})$ denotes the set of color i-vertices that have at least one uncolored incident edge and are missing α with respect to the coloring $\phi_{i,\alpha}$). Otherwise, if $v \notin V_{i,\alpha}(\phi_{i,\alpha})$, it must be caused by the repairing step of some color-i vertex u that changes the color of an incident edge of v to α during iteration α' for some $\alpha' < \alpha$. In other words, there is a fan F_u that was grown in iteration $\alpha' < \alpha$ and repairing F_u causes $v \notin V_{i,\alpha}(\phi_{i,\alpha})$. Repairing a fan F_u can only change the missing color set (i.e. $M(\cdot)$) of its leaves, its center, and the other endpoint of the alternating path. Since u and v are both in V_i, they cannot be neighbors. Therefore, v must be either the endpoint of the alternating path P_u or $v = u$.

Case 1: v is the end of the alternating path of F_u. It must be the $\mathtt{augment}$ operation of F_u that changes the missing color of v during the repairing step of F_u.

Case 2: $u = v$. Let $x_1 \ldots x_k$ be the leaves of F_u. If an edge incident to u becomes colored in α during iteration α', it must be the case that $\alpha = m(x_k)$ and $\alpha \in M(u)$. We perform $\texttt{shift}(F_u, k)$ operation and color ux_k in α. Note that in this case the augment operation is not performed.

Therefore, if $v \notin V_{i,\alpha}(\phi_{i,\alpha})$, we can blame it to some fan F_u. Note that each fan F_u can be blamed at most once, since Case 1 and Case 2 are mutually exclusive. Since for each v and $\alpha \in C(v)$, either it creates a fan or it blames another fan, we have

$$\sum_{u \in V_i} \text{outdeg}(u) = \sum_{u \in V_i} |C(u)| \le \sum_{\alpha=1}^{\Delta+1} |\mathcal{F}_{i,\alpha}| + (\#\text{blames}) \le 2 \sum_{\alpha=1}^{\Delta+1} |\mathcal{F}_{i,\alpha}| \, .$$

The last inequality follows since (#blames) $\le \sum_{\alpha=1}^{\Delta+1} |\mathcal{F}_{i,\alpha}|$.

Note that at most half of the fans are destroyed, since repairing a fan can only destroy at most one other fan. If a fan is not destroyed, then the uncolored edge in the fan must become colored. Therefore, at least $\sum_{i=1}^{\Delta^2+1} \sum_{\alpha=1}^{\Delta+1} |\mathcal{F}_{i,\alpha}|/2$ uncolored edges are colored. The total number of edges that are colored during the while loop is at least

$$\sum_{i=1}^{\Delta^2+1} \sum_{\alpha=1}^{\Delta+1} |\mathcal{F}_{i,\alpha}|/2 \ge \sum_{i=1}^{\Delta^2+1} \sum_{v \in V_i} \text{outdeg}(v)/4 = |\phi_0^{-1}(\perp)|/4 \, .$$

This implies that at least 1/4 of the uncolored edges become colored during the iteration. Thus, after $O(\log n)$ iterations, all edges are colored.

□

3.2 Load Balancing the Blocking Edges

By Lemma 3.1, Line 12 of Algorithm 1 is executed for at most $t = K\Delta^4 \log n$ times for some constant $K > 0$. Note that in each execution of Line 12, the alternating path of each fan in $\mathcal{F}_{i,\alpha,\beta,j}$ does not intersect alternating paths of other fans in $\mathcal{F}_{i,\alpha,\beta,j}$.

Let $\ell(v)$ denote the *load* of v, which is defined to be the number of incident edges to v colored in $\star$. Also, let $\ell(G) = \max_{v \in V} \ell(v)$. The hope is that at the end of the algorithm, $\ell(G)$ will be small. Since the maximum degree induced by $\star$-edges is $\ell(G)$, we can recolor $\star$-edges using $2\ell(G) - 1$ additional colors using the algorithm of Panconesi and Rizzi [30] in $O(\ell(G) + \log^* n)$ rounds.

In each execution of Line 12 of Algorithm 1, each vertex can only increase its $\star$ degree by one. This is because the alternating paths of the fans in $\mathcal{F}_{i,\alpha,\beta,j}$ do not intersect, and thus each vertex can be passed by at most one alternating path. Recall that t is the number of times Line 12 of Algorithm 1 is executed. Therefore, each vertex can be passed by at most t alternating paths throughout the algorithm.

On the other hand, for each alternating path P of length at least T, it has T choices to place the blocking edge on $P(T)$. If we place the blocking edge at uniformly random on each path, the expected load for each vertex v is, $\mathbb{E}[\ell(v)] \le \frac{2t}{T} = O(1/\lambda)$, provided $T \ge t \cdot \lambda$. We can show by Chernoff bound that $\ell(G) = O(\log n/(\log \lambda + \log\log n))$ w.h.p. However, to obtain a $(\Delta + 2)$-edge coloring, we need to make $\ell(G) = 1$ so that the $\star$-edges form a matching. In the rest of this section, we show how to achieve $\ell(G) = 1$ using randomization and $\ell(G) = O(\log n/(\log \lambda + \log\log n))$ without randomization.

Achieving $\ell(G) = 1$. Let $T = K\Delta^7 \cdot t = K\Delta^{11} \log n$ for some sufficiently large constant K. Instead of randomly placing the blocking edge, we do the following. Given a path P, let $E_0(P)$ be the set of edges whose endpoints have zero loads. In each alternating path P, we choose an edge from $E_0(P)$ uniformly at random to be the blocking edge. We will show that at any time during the algorithm, for every path P of length T (P is not necessarily an alternating path), $|E_0(P)| \ge T/15$ w.h.p. Therefore, given an alternating path of length at least T, it is always possible to choose a blocking edge whose both endpoints have zero loads. If some alternating path of length at least T with $|E_0(P)| < T/15$ is processed during Line 12, we say the algorithm *fails* and let our graph and coloring *freeze* there. If the algorithm did not fail, then we have achieved $\ell(G) = 1$.

Lemma 3.2. *After the last execution of Line 12 of Algorithm 1, it holds w.h.p. that for every path P of length T, we have that $|E_0(P)| \ge T/15$.*

Proof. Fix a path P of length T. We will show that the probability $\Pr(|E_0(P)| < T/15) \le 1/(\text{poly}(n) \cdot \Delta^T)$ after the t'th execution of Line 12 of Algorithm 1. By taking the union bound over all possible paths of length T, we will conclude that w.h.p. for every path P of length T, $|E_0(P)| < T/15$. This also implies the algorithm did not fail throughout execution $1 \ldots t$ w.h.p.

We say a vertex v is *occupied* if $\ell(v) \ge 1$. Otherwise, it is *empty*. Similarly, we say an edge is occupied if any of its endpoints is occupied. Otherwise, it is empty.

We say an alternating path is adjacent to an edge uv in P if u or v is in the alternating path. We divide all the alternating paths from the beginning up to after the t'th execution of Line 12 of Algorithm 1 that intersect P into two groups:

$$\mathcal{P}_1 = \{\text{alt. paths that are adjacent to at least } \frac{T}{1000} \text{ edges in } P\}$$

$$\mathcal{P}_2 = \{\text{alt. paths that are adjacent to fewer than } \frac{T}{1000} \text{ edges in } P\}.$$

Because the alternating paths in each execution of Line 12 of Algorithm 1 are vertex-disjoint, for each edge in P, at most 2 alternating paths are adjacent to it in each execution. Thus, at most $2t$ alternating paths are adjacent to that edge in total. This implies the number of pairs $(e, \tilde{P}) \in P \times \mathcal{P}_1$ where P is adjacent to e is at most $2tT$. Since each alternating path in $\mathcal{P}_1$ consumes at least $T/1000$ pairs, we have

$$|\mathcal{P}_1| \le \frac{2tT}{T/1000} = 2000t \, .$$

Note that we only place at most one blocking edge on an alternating path (i.e., if the path is truncated). Each blocking edge can cause at most three edges in P to be occupied (this corresponds to the case the blocking edge is in P). Thus, the alternating paths in $\mathcal{P}_1$ can cause at most $6000t$ edges in P to be occupied.

Now we bound the number of occupied edges in P caused by the blocking edges of alternating paths in $\mathcal{P}_2$. Let $P = u_0 \ldots u_T$. Let $E'(P) = \{u_0u_1, u_2u_3, \ldots, u_{2\cdot(\lceil T/2\rceil-1)}u_{2\cdot(\lceil T/2\rceil-1)+1}\}$ denote the set of every other edge in P so that $|E'(P)| = \lceil T/2\rceil$. We say an edge is **adjacent** to $E'(P)$ if it shares an endpoint with an edge in $E'(P)$.

We say a vertex u is **in** $E'(P)$ if it is an endpoint of an edge in $E'(P)$. We orient every edge (including those not in P) $e = uv$ that is adjacent to $E'(P)$ as follows: Without loss of generality, let u be the vertex that is in $E'(P)$. If v is not in $E'(P)$, we orient e from u to v. Otherwise, we orient the edge from the vertex that appears earlier in P to the vertex that appears later in P.

Given an edge $e = uv \in E'(P)$, we say e is **outwardly occupied** if any of the **outgoing edges** from u or v is a blocking edge of some alternating path in $\mathcal{P}_2$. Next, we will bound the number of outwardly occupied edges in $E'(P)$.

The reason we are using "outwardly occupied" instead of "occupied" is because it resolves some dependency issues. In particular, each alternating path in $\mathcal{P}_2$ can only make at most one edge in $E'(P)$ outwardly occupied, while it can make two edges occupied. Moreover, as we will see later, the number of occupied edges in $E'(P)$ is at most 3 times the number of outwardly occupied edges.

Now we hope to upper bound the total number of outwardly occupied edges $X = \sum_{e \in E'(P)} X_e$. Consider an edge e, the following is a rough argument to bound $\Pr(X_e = 1)$. The probability that an alternating path makes e to be outwardly occupied is $O(1/T)$, since w.h.p. it has $\Omega(T)$ empty edges to choose from. Moreover, there are at most $O(t)$ alternating paths that are adjacent to e throughout the algorithm. Thus, by taking a union bound over these paths, we have $\Pr(X_e = 1) = O(t) \cdot O(1/T) = O(t/T)$. As a result, $\mathbb{E}[X] = O(T \cdot (t/T)) = O(t)$. However, it will be difficult to obtain a tail bound on X directly due to dependency issues.

Since we need to apply the strongest form of Chernoff Bound in order to obtain a very sharp bound $(1/(\text{poly}(n) \cdot \Delta^T))$ on the probability that X deviates from $\mathbb{E}[X]$, we couple random variables $\{X_e\}_{e \in E'(P)}$ with independent Bernoulli random variables $\{Y_e\}_{e \in E'(P)}$ with success probability at most $150t/T$ in the following lemma whose proof is deferred to the end of this proof.

Lemma 3.3. *For every $e \in E'(P)$, let $X_e = 1$ if e is outwardly occupied by a blocking edge in an alternating path in $\mathcal{P}_2$ and $X_e = 0$ otherwise. Let $\{Y_e\}_{e \in E'(P)}$ be independent Bernoulli random variables with success probability at most $150t/T$. Let $X = \sum_{e \in E'(P)} X_e$ and $Y = \sum_{e \in E'(P)} Y_e$. For any $M \geq 0$, we have that $\Pr(X > M) \leq \Pr(Y > M)$.*

Therefore,

$$\mathbb{E}[Y] \leq \lceil T/2 \rceil \cdot \frac{150t}{T} < 100t \ .$$

Let $\delta = (T/700t) - 1$, and so $(1 + \delta) = (T/700t)$. Let $M = 100t$ be an upper bound of $\mathbb{E}[Y]$. We can apply a variant of Chernoff bound (where one could replace $\mathbb{E}[Y]$ by M if $M \geq \mathbb{E}[Y]$, see Corollary 4 of [33]) to obtain:

$$\begin{aligned}
\Pr(Y > T/7) &= \Pr(Y > (T/700t) \cdot 100t) \\
&= \Pr(Y > (1 + \delta) \cdot M) \\
&\leq \Pr(Y > \mathbb{E}[Y] + \delta M) \\
&\leq \left(\frac{e}{1+\delta}\right)^{(1+\delta)\cdot M} && \text{Corollary 4 of [33]} \\
&= \left(\frac{e}{1+\delta}\right)^{T/7} \\
&\leq \left(\frac{1}{e\Delta^7}\right)^{T/7} && (1+\delta) = \frac{T}{700t} > e^2 \cdot \Delta^7 \\
&\leq \frac{1}{\text{poly}(n)} \cdot \frac{1}{\Delta^T} && T = \Omega(\log n) \ .
\end{aligned}$$

By Lemma 3.3, $\Pr(X > T/7) \leq \Pr(Y > T/7) \leq 1/\text{poly}(n) \cdot 1/\Delta^T$.

Suppose that the number of outwardly occupied edges in $E'(P)$ is at most $T/7$. Consider an occupied edge that is not outwardly occupied. Its adjacent blocking edge must be adjacent to an edge $e \in E'(P)$ that is outwardly occupied. Moreover, each $e \in E'(P)$ can only be adjacent to at most two blocking edges. Therefore, the number of occupied edges is at most $T/7 + 2(T/7) + 6000t = 3T/7 + 6000t$. The number of empty edges in $E'(P)$ must be at least $\lceil T/2 \rceil - 3T/7 - 6000t > T/15$. Thus, the probability that there are fewer than $T/15$ empty edges on P is at most $1/(\text{poly}(n) \cdot \Delta^T)$.

There are at most $n \cdot \Delta^{T-1}$ paths of length T. By taking a union bound all length-T paths, the probability that there is a length-T path with fewer than $T/15$ empty edges is at most $(n \cdot \Delta^{T-1})/(\text{poly}(n) \cdot \Delta^T) = 1/\text{poly}(n)$. This also implies that the algorithm did not fail w.h.p. □

Proof of Lemma 3.3. First, we create i.i.d. random variables $\theta_{e,i}$ for each edge $e \in E'(P)$ and $1 \leq i \leq 2t$, where $\Pr(\theta_{e,i} = 1) = 75/T$. For each $e \in E'(P)$, we let $Y_e = \bigvee_{1 \leq i \leq 2t} (\theta_{e,i} = 1)$. Clearly, $\Pr(Y_e = 1) \leq 2t \cdot (75/T) = 150t/T$ and $\{Y_e\}_{e \in E'(P)}$ are independent.

The next step is to couple $\{X_e\}_{e \in E'(P)}$ with $\{\theta_{e,i}\}_{e,i}$. Recall that in our algorithm, each path $\tilde{P} \in \mathcal{P}_2$ selects the blocking edge uniformly at random from $E_0(\tilde{P})$ (unoccupied edges at the execution when $\tilde{P}$ occurs). Now we consider a modified algorithm that has the same outcome with the original algorithm. It allows us to do the coupling more easily. Given $e \in E'(P)$, let $out_{\tilde{P}}(e)$ denote the set of edges that are in $E_0(\tilde{P})$ and are outgoing edges from any endpoint of e. Note that if $e, e' \in E'(P)$ and $e \neq e'$, then $out_{\tilde{P}}(e)$ and $out_{\tilde{P}}(e')$ are disjoint.

Let $E'_{\tilde{P}}(P) \subseteq E'(P)$ denote the set of edges with non-empty $out_{\tilde{P}}(e)$. Since $\tilde{P} \in \mathcal{P}_2$, $|E'_{\tilde{P}}(P)| \leq T/1000$. Also, since $|out_{\tilde{P}}(e)| \leq 4$, we have $\sum_{e \in E'_{\tilde{P}}(P)} |out_{\tilde{P}}(e)| \leq 4T/1000$.

The modified algorithm maintains a counter, *occurred*, which is initially set to 0. Then it goes through each edge $e \in E'_{\tilde{P}}(P)$ and does the following: Throw a coin independently with success probability $|out_{\tilde{P}}(e)|/(|E_0(\tilde{P})| - \textit{occurred})$. If it succeeds, it selects a blocking edge randomly from $out_{\tilde{P}}(e)$ and then it is done with processing $\tilde{P}$. Otherwise, it sets $\textit{occurred} \leftarrow \textit{occurred} + |out_{\tilde{P}(e)}|$ and continues to process the next edge in $E'_{\tilde{P}}(P)$. If the algorithm did not successfully select a blocking edge after it processed every edge in $E'_{\tilde{P}}(P) \subseteq E'(P)$, it selects an edge from $E_0(\tilde{P}) \setminus (\bigcup_{e \in E'_{\tilde{P}}(P)} out_{\tilde{P}}(e))$ uniformly at random as the blocking edge.

It is easy to see that in the modified algorithm, the probability that any edge in $E_0(\tilde{P})$ is selected as the blocking edge is exactly $1/|E_0(\tilde{P})|$. Therefore, the modified algorithm is equivalent to the original algorithm where $\tilde{P}$ selects a random edge in $E_0(\tilde{P})$ as the blocking edge.

Now we couple the randomness of the modified algorithm with $\{\theta_{e,i}\}_{i,e}$. When the modified algorithm processes an edge $e \in E'_{\tilde{P}}(P)$, instead of throwing a new coin with probability $|out_{\tilde{P}}(e)|/(|E_0(\tilde{P})| - \textit{occurred})$, we will couple it with the randomness from $\theta_{e,i}$. For each $e \in E'(P)$, we maintain a counter $q(e)$ that points to the smallest index such that $\theta_{e,q(e)}$ is not yet exposed. Initially, $q(e)$ are 1 for every $e \in E'(P)$. Now when the algorithm processes $\tilde{P} \in \mathcal{P}_2$ and an edge $e \in E'_{\tilde{P}}(P)$, we will expose $\theta_{e,q(e)}$ and let $\mathcal{E}_1$ denote the event that $\theta_{e,q(e)} = 1$. Also, we set $q(e) \leftarrow q(e) + 1$. Let $\mathcal{E}_2$ denote the event that a coin with success probability $\frac{1}{\Pr(\mathcal{E}_1)} \cdot \frac{|out_{\tilde{P}}(e)|}{(|E_0(\tilde{P})| - occurred)}$ succeeds. Note that $\Pr(\mathcal{E}_1) \cdot \Pr(\mathcal{E}_2) = |out_{\tilde{P}}(e)|/(|E_0(\tilde{P})| - occurred)$. The original coin with probability $|out_{\tilde{P}}(e)|/(|E_0(\tilde{P})| - occurred)$ will be simulated by the trial that both the two events $\mathcal{E}_1$ and $\mathcal{E}_2$ happen.

Note that given $e \in E'(P)$, at most $2t$ alternating paths in $\mathcal{P}_2$ may be adjacent to e in total. This implies $\theta_{e,q(e)}$ is always well-defined when the algorithm refers to it. It remains to show that $\Pr(\mathcal{E}_2)$ is a valid probability, namely, $|out_{\tilde{P}}(e)|/(|E_0(\tilde{P})| - occurred) \le \Pr(\mathcal{E}_1)$.

$$\begin{aligned}
\Pr(\mathcal{E}_1) &= 75/T \\
&\ge \left(\frac{|out_{\tilde{P}}(e)|}{T/15 - 4T/1000}\right) && |out_{\tilde{P}}(e)| \le 4 \\
&\ge \left(\frac{|out_{\tilde{P}}(e)|}{T/15 - occurred}\right) && occurred \le 4T/1000 \\
&\ge \left(\frac{|out_{\tilde{P}}(e)|}{|E_0(\tilde{P})| - occurred}\right) && |E_0(\tilde{P})| \ge T/15\ .
\end{aligned}$$

The second to the last line holds since $occurred \le \sum_{e \in E'_{\tilde{P}}(P)} |out_{\tilde{P}}(e)| \le 4T/1000$. Also, the last line holds since if $|E_0(\tilde{P})| < T/15$, the algorithm freezes and $\tilde{P}$ will not be processed.

Finally, we point out that the coupling argument implies whenever $X_e = 1$, it must be the case that $\theta_{e,i} = 1$ for some $1 \le i \le 2t$. This in turns implies that $Y_e = 1$ and thus $X_e \le Y_e$. Therefore, for any $M > 0$, $\Pr(X > M) \le \Pr(Y > M)$. □

By Lemma 3.1, Line 9 is executed $O(\Delta^4 \log n)$ times. Each execution uses $O(T \cdot \log n)$ time, where $T = O(\Delta^{11} \log n)$. By Lemma 3.1 again, Line 12 is also executed $O(\Delta^4 \log n)$ times. Each execution takes $O(T)$ time. Therefore, the number of rounds used by our algorithm to obtain a $(\Delta+2)$-edge-coloring is $O(\Delta^{15} \log^3 n)$. The following lemma (whose proof we postpone to the end of the section) completes the proof of Theorem 1.1, which shows the algorithm can be converted to a $(1+\epsilon)\Delta$-edge-coloring algorithm that runs in $O((1/\epsilon)^{15} \log^3 n)$ rounds for any $\epsilon \ge 2/\Delta$.

Lemma 3.4. *Suppose that there is a deterministic (randomized) $(\Delta + z)$-edge-coloring algorithm that runs in $O(\Delta^x \log^y n)$ rounds. Then there exists a deterministic (randomized) $(1+\epsilon)\Delta$-edge-coloring algorithm that runs in $O((1/\epsilon)^x \cdot (z+1)^x \cdot \log^y n) + \tilde{O}(\log^2 \Delta \cdot \epsilon^{-1} \cdot \log \epsilon^{-1} \cdot \log n)$ rounds for any $\epsilon \ge z/\Delta$, where $\tilde{O}(\cdot)$ hides a* poly$(\log\log n)$ *factor.*

Achieving $\ell(G) = O(\log n/(\log \lambda + \log\log n))$ deterministically. We will choose $T = 2t \cdot \lambda$. Let $(1+\delta) = K \cdot \lambda \cdot \log n/(\log \lambda + \log\log n)$ for some constant $K > 0$. Given an alternating path P with length at least T, let $\ell_q(v)$ denote the load of v after q'th execution of Line 12 of Algorithm 1.

During the q'th execution of Line 12 of Algorithm 1, we select the blocking edge as follows. Given an edge $e = uv$, define the cost $c(e)$ to be $(1+\delta)^{\ell_{q-1}(v)} + (1+\delta)^{\ell_{q-1}(u)}$. Given an alternating path P, we select the edge $e \in P(T)$ with the minimum cost $c(e)$ to be the blocking edge. We will show that this greedy algorithm achieves a maximum load of $O(\log n/(\log \lambda + \log\log n))$.

Lemma 3.5. *If $T = 2t\lambda$, the greedy algorithm achieves a load of $O(\log n/(\log \lambda + \log\log n))$.*

Proof. We use the method of conditional expectation to show that the greedy algorithm is a derandomization of the randomized algorithm. Define the pessimistic estimator

$$\Phi(q) = \sum_{v \in V} \frac{(1+\delta)^{\ell_q(v)} \cdot (1 + \delta \cdot (2/T))^{t-q}}{(1+\delta)^{(1+\delta)/\lambda}}\ .$$

We show that $\Phi(q+1) \le \Phi(q)$ by the choice of the greedy algorithm. Consider the $(q+1)$'th iteration of Line 12 of Algorithm 1.

Let B denote a set of blocking edges chosen during $(q+1)$'th execution. Given an edge e, if v is an endpoint of e, we write $v \in e$. Moreover, we write $v \in B$, if there exists $e \in B$ such that $v \in e$. We use $[v \in B]$ denote the indicator variable for the event $v \in B$. Let $Q(B)$ denote the following quantity:

$$Q(B) = \sum_{v \in V} (1+\delta)^{\ell_q(v) + [v \in B]}\ .$$

Suppose that we choose each blocking edge randomly in $(q+1)$'th execution of Line 12, we have

$$\begin{aligned}
&\mathbb{E}[Q(B)] \\
&= \sum_{v \in V} \left(\Pr(v \in B) \cdot (1+\delta)^{\ell_q(v)+1} + \Pr(v \notin B) \cdot (1+\delta)^{\ell_q(v)}\right) \\
&= \sum_{v \in V} (1+\delta)^{\ell_q(v)} \cdot (\Pr(v \in B) \cdot (1+\delta) + (1 - \Pr(v \in B))) \\
&= \sum_{v \in V} (1+\delta)^{\ell_q(v)} \cdot (1 + \delta \cdot \Pr(v \in B)) \\
&\le \sum_{v \in V} (1+\delta)^{\ell_q(v)} \cdot \left(1 + \delta \cdot \frac{2}{T}\right)\ .
\end{aligned}$$

Observe that given e and e', if $c(e) < c(e')$, then $\sum_{v \in V} (1+\delta)^{\ell_q(v) + [v \in e]} < \sum_{v \in V} (1+\delta)^{\ell_q(v) + [v \in e']}$. Moreover, the alternating paths generated in the $(q+1)$'s execution of Line 12 do not intersect. Therefore, our greedy strategy is always choosing a set of blocking edges to minimize $Q(B)$. Thus,

$$\begin{aligned}
\Phi(q+1) &= \frac{(1 + \delta \cdot (2/T))^{t-q-1}}{(1+\delta)^{(1+\delta)/\lambda}} \cdot \min_B Q(B) \\
&\le \frac{(1 + \delta \cdot (2/T))^{t-q-1}}{(1+\delta)^{(1+\delta)/\lambda}} \cdot \mathbb{E}_B[Q(B)] \\
&\le \frac{(1 + \delta \cdot (2/T))^{t-q-1}}{(1+\delta)^{(1+\delta)/\lambda}} \cdot \sum_{v \in V} (1+\delta)^{\ell_q(v)} (1 + \delta \cdot (2/T)) \\
&= \Phi(q)\ .
\end{aligned}$$

Let $X = \log\lambda + \log\log n$. We have

$$\begin{aligned}
&\sum_{v\in V}(1+\delta)^{\ell_t(v)-(1+\delta)/\lambda} = \Phi(t) \le \Phi(t-1) \le \ldots \le \Phi(0)\\
&= \sum_{v\in V}\frac{(1+\delta\cdot(2/T))^t}{(1+\delta)^{(1+\delta)/\lambda}}\\
&\le \sum_{v\in V}\frac{e^{\delta\cdot(2t/T)}}{(1+\delta)^{(1+\delta)/\lambda}} \qquad \text{(since } 1+x\le e^x\text{)}\\
&\le \sum_{v\in V}\frac{e^{\delta/\lambda}}{(1+\delta)^{(1+\delta)/\lambda}}\\
&\le \sum_{v\in V}\left[\frac{e}{K\lambda\log n/(\log\lambda+\log\log n)}\right]^{K\log n/(\log\lambda+\log\log n)}\\
&= \sum_{v\in V}\exp(-(X+\log K-\log(X)-1)\cdot K\log n/X)\\
&\le \sum_{v\in V}\exp(-(K/2)\log n) \qquad \text{(for sufficiently large } X\text{)}\\
&\le \sum_{v\in V}\frac{1}{\text{poly}(n)} < 1\ .
\end{aligned}$$

This implies $\ell_t(v) \le (1+\delta)/\lambda = O(\log n/(\log\lambda + \log\log n))$ for all v. Thus, we have achieved $\ell(G) = O(\log n/(\log\lambda+\log\log n))$. □

The running time is dominated by Line 9, which is executed $O(\Delta^4\log n)$ times. Each execution uses $O(T\cdot\log n)$ time. The total running time for obtaining a $\Delta+O(\log n/(\log\lambda+\log\log n))$ coloring is $O(\Delta^4 T\cdot\log^2 n) = O(\lambda\cdot\Delta^8\log^3 n)$. We show how to reduce the running time to $O(\lambda\cdot\Delta^6\log^3 n)$ in the next section. By Lemma 3.4, it can be converted to a deterministic $(1+\epsilon)\Delta$-edge-coloring algorithm that runs in $O(\lambda\cdot(1/\epsilon)^6\log^9 n)$ algorithm for $\epsilon = \Omega(\log n/(\Delta(\log\lambda + \log\log n)))$.

Proof of Lemma 3.4. W.L.O.G. we may assume $\epsilon \le 1/8$. If $\Delta \le (16(z+8))/\epsilon$, then by applying the $(\Delta+z)$-edge-coloring algorithm, we obtain an $(1+\epsilon)\Delta$-edge-coloring in $O(\Delta^x\log^y n) = O((1/\epsilon)^x\cdot(z+1)^x\log^y n)$ rounds, since $z\le\epsilon\Delta$ and $\epsilon \le (16(z+8))/\Delta$.

Otherwise, let $\epsilon' = \epsilon/(32\log_{4/3}\Delta)$. By using the splitting algorithm of [19], we obtain two subgraphs of maximum degree $(1/2+\epsilon')\Delta+4$ in $\tilde{O}(\epsilon'^{-1}\cdot\log\epsilon'^{-1}\cdot\log n)$ rounds. Let $\Delta_0 = \Delta$ and $\Delta_i = (1/2+\epsilon')\Delta_{i-1}+4$. If we apply the algorithm recursively for h levels, we obtain 2^h subgraphs whose maximum degree are bounded by Δ_h. We choose h to be the smallest number that $(1/2+\epsilon')^h\cdot\Delta \in [(8(z+8))/\epsilon, (16(z+8))/\epsilon]$. Note that $h\le\log_{4/3}\Delta$, since $\epsilon'\le\epsilon\le 1/4$. Note that

$$\begin{aligned}
\Delta_h &= (1/2+\epsilon')^h\cdot\Delta + \sum_{i=0}^{h-1}(1/2+\epsilon')^i\cdot 4\\
&\le (1/2+\epsilon')^h\cdot\Delta + \sum_{i=0}^{h-1}(3/4)^i\cdot 4\\
&\qquad \text{(since } \epsilon' < 1/4\text{)}\\
&\le (1/2+\epsilon')^h\cdot\Delta + 16\\
&\le (1+\epsilon/4)(1/2+\epsilon')^h\cdot\Delta\\
&\qquad \text{(since } (1/2+\epsilon')^h\cdot\Delta \ge (8(z+8))/\epsilon \ge 64/\epsilon\text{)}\ .
\end{aligned}$$

We run the (Δ_h+z)-edge-coloring algorithm on the 2^h subgraphs using disjoint palettes to obtain a (Δ_h+z)-edge-coloring for each. The total number of colors that are used is

$$\begin{aligned}
2^h\cdot(\Delta_h+z) &\le 2^h\cdot(1+\epsilon/4)\cdot\Delta_h\\
&\qquad \text{(since } \Delta_h \ge (1/2+\epsilon')^h\cdot\Delta \ge 8z/\epsilon\text{)}\\
&= 2^h\cdot(1+\epsilon/4)^2\cdot(1/2+2\epsilon')^h\cdot\Delta\\
&\qquad \text{(since } \Delta_h \le (1+\epsilon/4)\cdot(1/2+2\epsilon')^h\cdot\Delta\text{)}\\
&= (1+\epsilon/4)^2\cdot(1+4\epsilon')^h\cdot\Delta\\
&\le (1+\epsilon/4)^2\cdot(1+8h\epsilon')\cdot\Delta\\
&\qquad \text{(since } (1+a)^b \le 1+2ab \text{ for } 0\le ab\le 1\text{)}\\
&\le (1+\epsilon/4)^2\cdot(1+\epsilon/4)\cdot\Delta\\
&\qquad \text{(since } \epsilon' = \epsilon/(32\log_{4/3}\Delta) \le \epsilon/(32h)\text{)}\\
&\le (1+\epsilon)\Delta\\
&\qquad \text{(when } \epsilon < 1/8\text{)}\ .
\end{aligned}$$

Since $\Delta_h = O(z/\epsilon)$, the running time is $\tilde{O}(\log\Delta\cdot\epsilon'^{-1}\cdot\log\epsilon'^{-1}\cdot\log n) + O((1/\epsilon)^x(z+1)^x\log^y n)$. □

Faster algorithms. We provide more efficient algorithms that use fewer number of rounds in the full version of this paper [35]. Our technique is based on another type of fans that was introduced by Gabow et al. [16]. This allows us to reduce the number of rounds by a factor Δ^2 for general graphs and Δ^4 for bipartite graphs for the randomized and deterministic algorithms.

ACKNOWLEDGMENTS

We thank the anonymous referees for their helpful comments.

REFERENCES

[1] Noga Alon, László Babai, and Alon Itai. 1986. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms* 7, 4 (1986), 567–583.

[2] Eshrat Arjomandi. 1982. An Efficient Algorithm for Colouring the Edges of a Graph With (Δ + 1) Colours. *INFOR: Information Systems and Operational Research* 20, 2 (1982), 82–101.

[3] Leonid Barenboim. 2015. Deterministic (Δ + 1)-Coloring in Sublinear (in Δ) Time in Static, Dynamic and Faulty Networks. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. 345–354.

[4] Leonid Barenboim and Michael Elkin. 2010. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Computing* 22, 5-6 (2010), 363–379.

[5] Leonid Barenboim, Michael Elkin, and Uri Goldenberg. 2018. Locally-Iterative Distributed (Δ+ 1)-Coloring Below Szegedy-Vishwanathan Barrier, and Applications to Self-Stabilization and to Restricted-Bandwidth Models. In *the Proc. of ACM Symp. on Princ. of Dist. Comp. (PODC)*. 437–446.

[6] L. Barenboim, M. Elkin, and F. Kuhn. 2014. Distributed (Δ + 1)-Coloring in Linear (in Δ) Time. *SIAM J. Comput.* 43, 1 (2014), 72–95. https://doi.org/10.1137/12088848X arXiv:http://epubs.siam.org/doi/pdf/10.1137/12088848X

[7] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. 2016. The Locality of Distributed Symmetry Breaking. *Journal of the ACM (JACM)* 63, 3 (2016). Article 20.

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

[9] Y.-J. Chang, W. Li, and S. Pettie. 2018. An optimal distributed (Δ + 1)-coloring algorithm?. In *Proceedings 50th ACM Symposium on Theory of Computing (STOC)*. 445–456.

[10] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. 2014. Distributed algorithms for the lovász local lemma and graph coloring. In *the Proc. of ACM Symp. on Princ. of Dist. Comp. (PODC)*. ACM, 134–143.

[11] Ran Duan, Haoqing He, and Tianyi Zhang. 2019. Dynamic Edge Coloring with Improved Approximation. In *SODA*. SIAM, 1937–1945.

[12] Devdatt Dubhashi, David A Grable, and Alessandro Panconesi. 1998. Near-optimal, distributed edge colouring via the nibble method. *Theoretical Computer Science* 203, 2 (1998), 225–251.

[13] Michael Elkin, Seth Pettie, and Hsin-Hao Su. 2015. $(2\Delta-1)$-edge-coloring is much easier than maximal matching in the distributed setting. In *Symp. on Discrete Algorithms (SODA)*. SIAM, 355–370.

[14] Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. 2017. Deterministic Distributed Edge-Coloring via Hypergraph Maximal Matching. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*. 180–191.

[15] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. 2016. Local Conflict Coloring. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*. to appear, arXiv:1511.01287.

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

[17] Shashidhar Gandham, Milind Dawande, and Ravi Prakash. 2008. Link scheduling in wireless sensor networks: Distributed edge-coloring revisited. *J. Parallel and Distrib. Comput.* 68, 8 (2008), 1122 – 1134.

[18] Mohsen Ghaffari, David Harris, and Fabian Kuhn. 2018. On Derandomizing Local Distributed Algorithms. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*. 662–673.

[19] Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, Yannic Maus, Jukka Suomela, and Jara Uitto. 2017. Improved Distributed Degree Splitting and Edge Coloring. In *Proc. of the Int'l Symp. on Dist. Comp. (DISC)*. 1–15.

[20] Mohsen Ghaffari, Fabian Kuhn, Yannic Maus, and Jara Uitto. 2018. Deterministic Distributed Edge-coloring with Fewer Colors. In *Proc. 50th ACM Symposium on Theory of Computing (STOC 2018)*. 418–430.

[21] Mohsen Ghaffari and Hsin-Hao Su. 2017. Distributed Degree Splitting, Edge Coloring, and Orientations. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2505–2523.

[22] A. V. Goldberg and S. A. Plotkin. 1987. Parallel $(\Delta+1)$-coloring of constant-degree graphs. *Inform. Process. Lett.* 25, 4 (1987), 241 – 245.

[23] A. V. Goldberg, S. A. Plotkin, and G. E. Shannon. 1988. Parallel Symmetry-Breaking in Sparse Graphs. *SIAM J. Discrete Math.* 1, 4 (1988), 434–446.

[24] David Harris, Johannes Schneider, and Hsin-Hao Su. 2018. Distributed $(\Delta+1)$-Coloring in Sublogarithmic Rounds. *J. ACM* 65, 4 (2018). Article 19.

[25] Öjvind Johansson. 1999. Simple Distributed $\Delta+1$-coloring of Graphs. *Inf. Process. Lett.* 70 (1999).

[26] Fabian Kuhn and Roger Wattenhofer. 2006. On the Complexity of Distributed Graph Coloring. In *Symp. on Principles of Distributed Computing (PODC)*.

[27] Nati Linial. 1992. Locality in Distributed Graph Algorithms. *SIAM J. Comput.* 21, 1 (1992), 193–201.

[28] Michael Luby. 1986. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing* 15, 4 (1986), 1036–1053.

[29] J. Misra and David Gries. 1992. A Constructive Proof of Vizing's Theorem. *Inf. Process. Lett.* 41, 3 (March 1992), 131–133.

[30] Alessandro Panconesi and Romeo Rizzi. 2001. Some simple distributed algorithms for sparse networks. *Distributed computing* 14, 2 (2001), 97–100.

[31] Alessandro Panconesi and Aravind Srinivasan. 1992. Improved distributed algorithms for coloring and network decomposition problems. In *Proc. of the Symp. on Theory of Comp. (STOC)*. ACM, 581–592.

[32] Alessandro Panconesi and Aravind Srinivasan. 1997. Randomized Distributed Edge Coloring via an Extension of the Chernoff–Hoeffding Bounds. *SIAM J. Comput.* 26, 2 (1997), 350–368.

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

[34] S. Ramanathan. 1999. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks* 5, 2 (1999), 81–94.

[35] Hsin-Hao Su and Hoa T. Vu. 2019. Towards the Locality of Vizing's Theorem. *CoRR* abs/1901.00479 (2019). arXiv:1901.00479 http://arxiv.org/abs/1901.00479

[36] V. G. Vizing. 1964. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz* 3 (1964), 25–30.