

Distributed Degree Splitting, Edge Coloring, and Orientations *

Mohsen Ghaffari
MIT
ghaffari@mit.edu

Hsin-Hao Su
MIT
hsinhao@mit.edu

Abstract

We study a family of closely-related distributed graph problems, which we call *degree splitting*, where roughly speaking the objective is to partition (or orient) the edges such that each node’s degree is split almost uniformly. Our findings lead to answers for a number of problems, a sampling of which includes:

- We present a poly $\log n$ round deterministic algorithm for $(2\Delta - 1) \cdot (1 + o(1))$ -edge-coloring, where Δ denotes the maximum degree. Modulo the $1 + o(1)$ factor, this settles one of the long-standing open problems of the area from the 1990’s (see e.g. Panconesi and Srinivasan [PODC’92]). Indeed, a weaker requirement of $(2\Delta - 1) \cdot \text{poly log } \Delta$ -edge-coloring in poly $\log n$ rounds was asked for in the 4th open question in the *Distributed Graph Coloring* book by Barenboim and Elkin.
- We show that *sinkless orientation*—i.e., orienting edges such that each node has at least one outgoing edge—on Δ -regular graphs can be solved in $O(\log_{\Delta} \log n)$ rounds randomized and in $O(\log_{\Delta} n)$ rounds deterministically. These prove the corresponding lower bounds by Brandt et al. [STOC’16] and Chang, Kopelowitz, and Pettie [FOCS’16] to be tight. Moreover, these show that sinkless orientation exhibits an exponential separation between its randomized and deterministic complexities, akin to the results of Chang et al. for Δ -coloring Δ -regular trees.
- We present a randomized $O(\log^4 n)$ round algorithm for orienting a -arboricity graphs with maximum out-degree $a(1 + \varepsilon)$. This can be also turned into a decomposition into $a(1 + \varepsilon)$ forests when $a = \Omega(\log n)$ and into $a(1 + \varepsilon)$ pseudo-forests when $a = o(\log n)$. Obtaining an efficient distributed decomposition into less than $2a$ forests was stated as the 10th open problem in the book by Barenboim and Elkin.

*This research was supported by NSF Award Numbers CCF-1461559, CCF-0939370, CCF-1217506, and NSF BIO-1445983. Furthermore, the first author was also supported in part by a Simons award (no. 318723).

1 Introduction & Related Work

Graph symmetry breaking problems form one of the central subareas of distributed algorithms, and they have received extensive attention over the last three decades. See the book by Barenboim and Elkin [BE13] for an instructive survey. In this paper, we revisit some of the classical problems of this area, as well as some newer ones which have received attention only recently. The common denominator of the problems we consider is that they all revolve around a seemingly rudimentary edge symmetry-breaking task, which we refer to as *degree-splitting*, and discuss shortly. Degree splitting turns out to capture a core challenge in several problems, and the techniques we develop for it allow us to address a number of seemingly independent problems.

Throughout, we work with the standard distributed model called LOCAL, due to Linial [Lin92]: The network is abstracted as a graph $G = (V, E)$. There is one processor on each vertex, which initially knows only its neighbors. Per round, each processor can send one message of unbounded size to each of its neighbors.

1.1 Degree Splitting and Edge Coloring We start our discussion of degree splitting with the classical edge-coloring problem, as these two problems have a simple and clear relation. One basic version of degree splitting is as follows:

Undirected Degree Splitting: Given a graph with maximum degree Δ , color each edge red or blue such that each node has at most $\Delta(1 + \varepsilon)/2$ edges in each color, for a small $\varepsilon \geq 0$.

To understand the connection to edge-coloring, imagine an ideal world—though not always feasible—where everything is fair and one could always have a perfect split, i.e., with $\varepsilon = 0$. Then, by a recursion of depth $\log \Delta$ of degree-splittings, we would get to a setting where edges are colored in $2^{\log \Delta} = \Delta$ colors, and each node has at most 1 edge of each color. That is a proper Δ edge-coloring.

But that is too good to be true! Not every graph admits a Δ -edge-coloring¹, and not every graph admits a perfect degree-split. Take K_3 for instance. But almost perfect splits exist, with $\varepsilon = 2/\Delta$. Here is one way of getting them: Add a dummy node and connect it to all odd degree nodes; then take an Eulerian tour and color its edges red and blue in alternating order. The same guarantee also follows from the Discrepancy Theory result of Beck and Fiala [BF81]—which is in fact far more general².

We present distributed algorithms for computing almost perfect degree-splits. As Eulerian tours cannot be computed distributedly, our methods are vastly different from the above. As formalized in [Theorem 3.3](#), we present a randomized algorithm that achieves an almost-perfect split with per-color degree at most $\Delta/2 + 1$ in $\text{poly}(\Delta \log n)$ rounds. However, as far as we know, this result itself does not lead to an improvement in any of the well-studied distributed problems. Instead, as formalized in [Theorem 3.1](#), we present a deterministic algorithm that in $\text{poly} \log n$ rounds produces a degree-split with $\varepsilon = 1/\log^c n$, for a desirably large constant $c \geq 2$, in graphs of max-degree $\Delta = \Omega(\text{poly} \log n)$. This immediately leads to an improvement for edge-coloring:

THEOREM 1.1. *There is a deterministic $\text{poly} \log n$ -round algorithm for $(2\Delta - 1)(1 + o(1))$ -edge-coloring.*

As Panconesi and Rizzi [PR01] state, four key problems of the area from the 1990’s were to find $\text{poly} \log n$ -round deterministic algorithms for Maximal Independent Set, $(\Delta + 1)$ -vertex-coloring, $(2\Delta - 1)$ -edge-coloring, and Maximal Matching. To this day, only the Maximal Matching problem is resolved, due to a breakthrough of Hanckowiak, Karonski, and Panconesi [HKP98, HKP01]. [Theorem 1.1](#) almost settles the edge-coloring problem, modulo the $(1 + o(1))$ factor. The previously best-known number of required colors were $\Delta \cdot 2^{O(\frac{\log \Delta}{\log \log \Delta})}$, due to Barenboim and Elkin [BE11], and $O(\Delta \log n)$, due to Czygrinow et al. [CHK01]. We in fact present a simpler proof of the former via degree splittings in [Section 3.1](#). As stated in the 4th open problem of the *Distributed Graph Coloring* book by Barenboim and Elkin [BE13, Section 11], even $\Delta \cdot \text{poly}(\log \Delta)$ -edge-coloring in $\text{poly} \log n$ rounds remained open.

In [Section 3.5](#), we explain that mixing [Theorem 1.1](#) with some other ideas also leads to a fast randomized $(2\Delta - 1)(2 + o(1))$ -edge-coloring in $O(\text{poly}(\log \log n))$ rounds.

¹Although, a $(\Delta + 1)$ -edge-coloring is guaranteed to always exist, by Vizing’s theorem [Viz64].

²It follows from Beck-Fiala’s theorem that hypergraphs of rank t —i.e., where each hyperedge has at most t vertices—admit a red-blue hyperedge-coloring where each node has at most $\Delta/2 + t - 1$ edges in each color.

1.2 Degree Splitting and Edge Orientations We also consider the following natural variant of degree-splitting:

Directed Degree Splitting: Given a graph with maximum degree Δ , orient each edge such that each node has in-degree and out-degree at most $(1 + \varepsilon)\Delta/2$, for a small $\varepsilon \geq 0$.

We note that in directed splitting, even in graphs which admit perfect splits, computing them might be unavoidably time-consuming—e.g., perfectly splitting an n -node cycle would require $\Omega(n)$ rounds.

The directed degree splitting relates closely to Eulerian Orientations, as the latter requires an orientation such that each node has the same in-degree and out-degree. We note that on bipartite graphs (with the bi-partition given), the directed and undirected degree-splitting problems are equivalent. However, we are not aware of a formal reduction between them in the general case.

Despite that, we use more or less the same methodology as that of our undirected degree splitting to find almost perfect directed splits: As formalized in [Corollary 4.1](#), we get an existentially best possible guarantee of per-node out-degree and in-degree at most $\Delta/2 + 1$ in $\text{poly}(\Delta \log n)$ rounds randomized. More importantly, as formalized in [Theorem 4.1](#), we get a deterministic algorithm that computes a degree splitting with $\varepsilon = 1/\log^c n$, for a desirably large constant $c \geq 2$, in graphs of max-degree $\Delta = \Omega(\text{poly} \log n)$ in $\text{poly} \log n$ rounds.

Sinkless Orientation and Its Refinements: One related problem which has received attention recently is *sinkless orientation*, where the objective is to orient edges of a Δ -regular graph such that each node has out-degree at least 1. Note that this is clearly a much weaker requirement than that of the directed degree splitting problem. Recently, Brandt et al. [BFH⁺16] gave an elegant $\Omega(\log_\Delta \log n)$ round lower bound for this problem, which was then extended by Chang et al. [CKP16] to an $\Omega(\log_\Delta n)$ lower bound for deterministic algorithms. For this weaker orientation problem, we can achieve much better round complexities, which match the respective lower bounds.

THEOREM 1.2. *There is a randomized $O(\log_\Delta \log n)$ -round algorithm which solves sinkless orientation in all Δ -regular graphs, for $\Delta \geq 3$, and in fact all graphs of minimum degree at least Δ , with high probability. Moreover, the same problem has a deterministic $O(\log_\Delta n)$ -round algorithm.*

We can in fact guarantee a more balanced split in almost the same running time: We show how to find

an orientation with per-node in-degree and out-degree at most $5\Delta/6$ —that is, a directed degree split with $\varepsilon = 2/3$ —in $O(\log \log n)$ rounds randomized and $O(\log n)$ rounds deterministically.

Brandt et al. [BFH⁺16] used sinkless orientation to prove an $\Omega(\log_\Delta \log n)$ round lower bound on LOCAL-algorithms for Lovasz Local Lemma (LLL). Since LLL can provide much finer degree splits, studying these stronger degree-splits might expose higher lower bounds for LLL. Moreover, Chang et al. [CKP16] recently presented the first exponential separation between randomized and deterministic distributed complexities, by showing that Δ -vertex-coloring Δ -regular trees requires $\Theta(\log_\Delta \log n)$ rounds randomized and $\Theta(\log_\Delta n)$ rounds deterministically. [Theorem 1.2](#), in conjunction with the aforementioned lower bounds, exhibits the same exponential separation on sinkless orientation.

Low Out-Degree Orientations and Nash-Williams Decompositions: An alternative way of viewing the directed degree splitting problem is as follows: it asks for an orientation that achieves a maximum out-degree within a $1 + \varepsilon$ factor of what is necessary, given the maximum degree. In this regard, one can ask for a stronger guarantee: find an orientation with maximum out-degree $a(1 + \varepsilon)$, where a is the arboricity of the graph. Clearly any orientation of any graph with arboricity a admits an orientation with out-degree at least a . Our methods allow us to achieve such an approximation:

THEOREM 1.3. *There is a randomized $O(\text{poly } \log n/\varepsilon)$ -round algorithm which orients a -arboricity graphs with maximum out-degree at most $a(1 + \varepsilon)$. This is equivalent with a decomposition into $a(1 + \varepsilon)$ pseudo-forests. If $a = \Omega(\log n)$, we can turn this into a decomposition into $a(1 + \varepsilon)$ forests.*

See [Theorem 4.2](#) and [Lemma 4.8](#) for the formal statements. We note that efficient distributed orientation with out-degree less than $2a$ had remained open. The best previously known results were as follows: an orientation with out-degree at most $2a$ in $O(a \log n)$ rounds and an orientation with out-degree at most $(2 + \varepsilon)a$ in $O(\log n)$ rounds, both due to Barenboim and Elkin [BE10]. The same authors state the closely-related problem of efficient distributed decomposition into less than $2a$ forests as the 10th open problem in their book [BE13, Section 11].

1.3 Other Related Work **Related Work on Degree Splitting:** Degree splitting was first considered by Israeli and Shiloach [IS86] in the parallel algorithms model—a.k.a. PRAM—as a subroutine for computing a maximal matching. Their method for computing the split relies on finding an Eulerian cycle of the graph

and 2-coloring its edges in alternating order. We note that a number of other work in the PRAM model, e.g., [KS87], also used degree splittings but all relying on Eulerian cycles. Unfortunately, Eulerian cycles cannot be computed using distributed algorithms (with sublinear complexity). Hence, these methods cannot extend to our setting. Our approach is quite different, and it is morally much closer to the classic ideas of augmenting paths and blocking-flows in the maximum flow problem [EK72, Din].

However, there is an ingenious line of work that comes close to distributedly computing degree splits. Inspired by the parallel maximal matching algorithm of Israeli and Shiloach [IS86], Hanckowiak, Karonski, and Panconesi [HKP98, HKP01] used a clever relaxation of the degree splitting to distributedly compute a maximal matching in $\text{poly } \log n$ rounds deterministically. Their relaxation allows a small but non-negligible fraction $\delta > 0$ of nodes to have an unfair split, even possibly having all of their edges in one color. This relaxation is indeed essential to their method. While for Maximal Matching this relaxation is good enough, it becomes too costly for the other problems, e.g., the number of required colors in edge-coloring blows up by an $O(\log n)$ factor, as Czygrinow et al. [CHK01] show. In this regard, one can view our degree-splitting results as a *qualitative* improvement on those of [HKP98, HKP01], as we do not need the relaxation of admitting some unbalanced nodes. However, we pay for this refinement and our round-complexity ends up being a higher $\text{poly } \log n$.

Related Work on Edge Coloring: Edge coloring is one of the classical distributed problems and it has been studied extensively over the years. There is a clear dichotomy between randomized and deterministic algorithms for this problem, and we review the results in these two categories.

First, we review the deterministic results. Panconesi and Rizzi provide an $O(\Delta + \log^* n)$ algorithm for $(2\Delta - 1)$ -edge-coloring. This complexity was recently improved by Barenboim [Bar15] to $O(\Delta^{3/4} \log \Delta + \log^* n)$ and subsequently to $O(\Delta^{1/2} \log^{5/2} \Delta + \log^* n)$ by Fraigniaud, Heinrich, and Kosowski [FHK16]. Both these results work indeed for the harder problem of $(\Delta + 1)$ -vertex coloring. However, these complexities can be much larger than $\text{poly } \log n$. The best known number of required colors for $\text{poly } \log n$ -round algorithms remained at $\Delta \cdot 2^{O(\frac{\log \Delta}{\log \log \Delta})}$, due to Barenboim and Elkin [BE11], and $O(\Delta \log n)$, due to Czygrinow et al. [CHK01]. See also [BE13, Chapter 8 & Chapter 11.1].

Now, we review the randomized results. The classical $O(\log n)$ round randomized Maximal Independent Set algorithm of Luby [Lub86] leads to a randomized $O(\log n)$ -round $(2\Delta - 1)$ -edge coloring. This round

complexity was improved to $O(\log \Delta) + e^{O(\sqrt{\log \log n})}$ by Barenboim et al. [BEP16]. This was further improved to just $2^{O(\sqrt{\log \log n})}$ by Elkin, Pettie, and Su [EPS15]. In the randomized world, there are also algorithms for finding colorings with smaller number of colors. Panconesi and Srinivasan [PS97] give the first such result. Dubhashi, Grable and Panconesi [DGP98] later improve this to $(1 + \varepsilon)\Delta$ -edge-coloring in $O(\log n)$ rounds, when $\Delta = \Omega(\log^{1+\Omega(1)} n)$. This was later refined and extended to graphs of degree $\Delta \geq C_0$, for a constant C_0 depending on ε . The final work in this track is by Elkin, Pettie, and Su [EPS15] which improves the complexity to $O(\log^* \Delta \cdot \max\{1, \frac{\log n}{\Delta^{1-o(1)}}\})$.

Related Work on Orientations: Distributed low out-degree orientation of low-arboricity graphs was first studied by Barenboim et al. [BE10] and the same results have been used in a few subsequent works. This orientation was then turned into a forest decomposition which subsequently lead to sublinear-time algorithms for maximal independent set, vertex coloring, edge coloring and maximal matching in graphs of low arboricity. See [BE13, Chapter 4 & Chapter 11.3]. Sinkless orientation was recently introduced by Brandt et al. [BFH⁺16] and studied also by Chang et al. [CKP16].

1.4 Our General Method In a Nutshell Our general method follows a natural idea and it is inspired by classical concepts from the maximum flow problem. Although, as we soon see, to have an efficient algorithm, particularly an efficient distributed algorithm and especially a deterministic one, various aspects require novel techniques.

Let us consider directed degree splitting as our running example in discussing the methodology. Consider an arbitrary orientation. In this orientation, some nodes might have an out-degree (much) larger than in-degree and some nodes might have a larger in-degree than out-degree. Virtually, we can think of out-degree as the commodity of our flow. This means that the first group have *excess flow* and the second group have *flow deficiency*. Naturally, we wish to transfer some flow from the first group to the second to even things out. For instance, if we find a directed path from the first group to the second—i.e. what we usually call an *augmenting path*—we can flip the direction of its edges, improving the degrees on the two endpoints, but keeping them unchanged in the middle nodes. We would continue doing this until each node has about the same out-degree and in-degree, at which point we have found our almost-perfect degree split.

Finding an efficient distributed algorithm following this idea necessitates a number of considerations and novel techniques. First, we need the augmenting paths to be short, as otherwise we cannot even find them dis-

tributedly. That issue is not hard, because as we will see imperfect splits have relatively short augmenting paths. Second, in a fast distributed algorithm, we cannot afford to fix imbalanced nodes by using augmenting paths one by one; we instead need to have many “disjoint” short augmenting paths. Third, we need to find them fast distributedly. These second and third issues are much more crucial. We will show that imperfect splits in fact have large sets of “disjoint” augmenting-paths. However, finding such a set distributedly, and especially doing it deterministically, will require quite some effort. We leave those discussions to the technical sections.

Finally, we note that in some problems, we desire much faster solutions, e.g., an $O(\log_\Delta \log n)$ round complexity in sinkless orientation. In these cases, our general methodology provides some algorithm but not quite matching the lower bound. There, we deviate slightly from this flow augmenting mentality and use some other ideas to optimize the complexity.

2 Sinkless Orientation

In this section, we present a simple $O(\log_\Delta \log n)$ round randomized algorithm for sinkless orientation in Δ -regular graphs. This matches the $\Omega(\log_\Delta \log n)$ lower bound presented by Brandt et al. [BFH⁺16]. As a component of this result, we also present an $O(\log_\Delta n)$ round deterministic algorithm for the same problem, which itself proves the corresponding $\Omega(\log_\Delta n)$ lower bound of Chang et al. [CKP16] to be tight. These results prove [Theorem 1.2](#). We first present an algorithm that works assuming $\Delta > 500$. The extension to all cases with $\Delta \geq 3$, to cases with irregularity, and also to a setting where we desire a more balanced split of in-degree and out-degree, are described in [Section 2.2](#).

2.1 Sinkless Orientation for Δ -regular graphs with $\Delta > 500$ Notice that if $\Delta = \Omega(\log n)$, orienting each edge randomly ensures that all nodes have at least $\Theta(\Delta)$ outgoing edges, with high probability, which is a sinkless orientation. The far more interesting case of the problem is when $\Delta = O(\log n)$, and this is the focus of this section.

The algorithm is composed of two parts: a randomized part, which *shatters* the graph thus leaving only small components of polylog n size, and then a deterministic part, which takes care of these remaining small components. The random part will take $O(1)$ rounds and then the deterministic part will solve these remaining polylog n -size components in $O(\log_\Delta \log n)$ rounds, hence leading to the overall randomized complexity of $O(\log_\Delta \log n)$. The deterministic part itself is a full solution for n -node graphs with complexity $O(\log_\Delta n)$. Next, we present these two parts.

2.1.1 The Randomized Part of the Algorithm

(Pre-Shattering) The randomized algorithm is quite simple and it works as follow. Before presenting the algorithm, we note that in the course of the algorithm, we will allow *half-edges*, which are edges with only one endpoint (which needs the edge to be outgoing).

Algorithm 1 Randomized Orientation

Mark each edge with probability $\frac{1}{4}$.
 For each marked edge, orient it randomly with probability $1/2$ for each direction.
 For each node v , mark v as a *bad node* of the following types according to these rules:

- Type I. If v has more than $\Delta/2$ marked edges incident to it.
- Type II. If v is not Type I but it has at least one neighbor of Type I.
- Type III. If v is not Type I or Type II but it has no outgoing marked edges.

Unmark all the edges incident to Type I nodes.
 Orient unmarked edges which both of their endpoints are good nodes arbitrarily.
 Consider unmarked edges with exactly one good endpoint as a half-edge only attached to the bad-node.
 Run the deterministic algorithm on the components induced by the bad nodes and their edges or half-edges.

FACT 2.1. *After Algorithm 1, each node is incident to at least $\Delta/2$ unmarked edges.*

Proof. Node of type I get all their Δ edges unmarked. Each other node has at most $\Delta/2$ marked edges incident to it, by definition.

FACT 2.2. *After Algorithm 1, if the unmarked edges are oriented in a way such that all bad nodes have at least one outgoing unmarked edge, then the orientation is sinkless.*

Proof. If a node is bad, then by the assumption it will have at least one outgoing unmarked edge. Otherwise, it must have at least one outgoing marked edge by the definition of Type III.

In Section 2.2.1, we show the deterministic algorithm can be used orient the unmarked edges if the number of unmarked edges incident to each node is between $\Delta/2$ and Δ . In the following we show that with high probability, each connected component induced by bad nodes has size $O(\Delta^2 \log n) = \text{poly } \log n$. Therefore, the deterministic algorithm solves these remaining components, where each remaining node has at least $\Delta/2$ edges or half-edges, in $O(\log_{\Delta} \log n)$ rounds.

Let $T_I(v)$ and $T_{III}(v)$ denote the events that v is Type I or Type III, respectively.

LEMMA 2.1. *For any $v \in G$, $\Pr(T_I(v)) \leq \exp(-\Delta/12)$.*

Proof. Let X denote the number of edges incident to v . We have $\mathbb{E}[X] = \Delta/4$. By a Chernoff Bound, $\Pr(X > \Delta/2) \leq \exp(-\Delta/12)$.

LEMMA 2.2. *For any $v \in G$, $\Pr(T_{III}(v)) \leq \exp(-\Delta/8)$.*

Proof. The probability that an edge is marked and oriented toward v is $1/8$. Therefore, the probability that no edges are marked and oriented toward v is $(1 - 1/8)^\Delta \leq \exp(-\Delta/8)$.

Let $\text{dist}(u, v)$ denote the distance between u and v in G . If $\text{dist}(u, v) \geq 2$, then it is clearly that the event $T_I(u)$ (or $T_{III}(u)$) and $T_I(v)$ (or $T_{III}(v)$) are independent. Let V' be the set of nodes that are Type I or Type III. Define $\text{dist}(X, v) = \min_{u \in X} \text{dist}(u, v)$. Let $E_{2,4} = \{uv \mid 2 \leq \text{dist}(u, v) \leq 4\}$ denote the set of edges whose endpoints have distance between 2 and 4. Let $N_k(u) = \{x \mid \text{dist}(x, u) \leq k\}$ be the set of nodes within distance k to u .

LEMMA 2.3. *Let C be a connected component induced by the bad nodes. Then, there exists $S \subseteq V' \cap C$ such that $|S| \geq |C|/\Delta^2$ and $(S, E_{2,4})$ is connected.*

Proof. We will construct S step by step. First notice that $V' \cap C$ must be non-empty, since a Type II node must be adjacent to a Type I node, which must be in $V' \cap C$. Let $u \in V' \cap C$. Initially, Let $S = \{u\}$. Now we will show how to extend S by adding one node $z \in (V' \cap C) \setminus S$ into it provided that $|S| < |C|/\Delta^2$, and z is connected to some node in S with an edge in $E_{2,4}$.

Suppose that $|S| < |C|/\Delta^2$, then there exists a node w in $C \setminus S$ such that $\text{dist}(S, w) = 3$, since the 2-neighborhood of S can only span at most $|S|(1 + \Delta + \Delta \cdot (\Delta - 1)) \leq |S|\Delta^2$ nodes and C is connected. If w is Type I or Type III, then $w \in V' \cap C$ and we can add w to S . Otherwise, it is Type II, which implies it has a neighbor z of Type I. We must have $2 \leq \text{dist}(u, z) \leq 4$. Thus, we can add z to S .

Therefore, given a connected component C induced the bad nodes, we can find a tree T in $(V' \cap C, E_{2,4})$ such that $|T| \geq |C|/\Delta^2$. Next we show that any sufficient large tree are not likely to occur, which implies no big bad components exist.

LEMMA 2.4. *The probability that any tree T with $T \subseteq (V', E_{2,4})$ and $|T| = \Omega(\log n)$ exists is at most $1/\text{poly}(n)$.*

Proof. Let T be any tree such that $T \subseteq (V', E_{2,4})$ and $|T| = t$. The probability that a node in T is marked as Type I and Type III (thus in V') is

at most $\exp(-\Delta/12)$ by [Lemma 2.1](#) and [Lemma 2.2](#). Therefore, the probability that T occurs is at most $\exp(-t \cdot \Delta/12)$, since the events $T_I(u)$ (or $T_{III}(u)$) are independent among the nodes $u \in T$. The total possible number of such trees is at most $4^t n (\Delta^4)^{t-1}$. By union bounding the possible trees, the probability that any tree in $(V', E_{2,4})$ of size t occur is at most $n \cdot (4\Delta^4 \cdot \exp(-\Delta/12))^t$. For $\Delta \geq 500$, this is at most $n \cdot e^{-t}$. Thus, the probability that any tree in $(V', E_{2,4})$ of size at least $10 \log n$ exists is at most $(1/n^9) \cdot \sum_{i=0}^{\infty} e^{-i} = 1/\text{poly}(n)$.

COROLLARY 2.1. *The probability that any connected component induced by the bad nodes has size of $\Omega(\Delta^2 \log n)$ is at most $1/\text{poly}(n)$.*

Proof. If there exists a bad connected component C with size $\Omega(\Delta^2 \log n)$, then by [Lemma 2.3](#), there exists $S \subset V' \cap C$ such that $|S| = \Omega(\log n)$ and $(S, E_{2,4})$ is connected. Therefore, a tree T in $(V', E_{2,4})$ occurs with $|T| = \Omega(\log n)$, which happens with probability $1/\text{poly}(n)$ by [Lemma 2.4](#).

2.2 Generalization to Irregular Graphs with Min-Degree $d \geq 3$, and Refinements In the previous subsection, we presented an $O(\log_{\Delta} \log n)$ sinkless orientation algorithm for Δ -regular graphs with $\Delta > 500$. Here, we extend the result to a irregular graphs with min-degree $d \geq 3$, with round complexity becoming $O(\log_d \log n)$. We also show how to achieve a more refined directed degree split in almost the same running time.

First let us deal with irregular graphs with min-degree $d > 500$.

LEMMA 2.5. *There is a randomized algorithm that computes a sinkless orientation of graphs of minimum degree $d \geq 500$ in $O(\log_d \log n)$ rounds.*

Proof. We transform graph G into a d -regular structure H , which is essentially a graph but allowing edges with only one endpoint, which we call half-edges. For each node $v \in G$ with degree $d' > d$, remove v and instead add $\lfloor d'/d \rfloor$ copy-nodes, assign d of edges of v to each of these copy-nodes, and mark the remaining edges. We do not need those marked edges to be oriented outwards from v (or its copy-nodes). If an edge is marked by both of its end-points, drop it. Otherwise, think of it simply as a *half-edge*, having only one endpoint which may wish to have this edge outgoing. Now, the graph is transformed into a new structure where each node is incident on exactly d edges or half-edges. A sinkless orientation of this structure can be compute using [Lemma 2.8](#).

We now explain how to extend the algorithm to cases where min-degree is $d \in [3, 500]$.

LEMMA 2.6. *There is a randomized algorithm that computes a sinkless orientation of graphs of minimum degree $d \in [3, 500]$ in $O(\log \log n)$ rounds.*

Proof. Let c be a small constant such that $(d-1)^{c/2} > 500$. First, we find an orientation for all edges which are in cycles of length up to $3c$, in $O(1)$ rounds. This can be done easily using the method of [Section 2.2.1](#). This already takes care of giving an outgoing edge to nodes which are incident on these edges. We next handle the rest of the nodes.

By means of the method of the previous paragraph, we can assume without loss of generality that the graph is d -regular, albeit possibly having half-edges. Note that this step cannot introduce a cycle of length less than $3c$. Now, compute a Maximal c -Independent Set S , on the graph while ignoring the half-edges. This can be done in $O(d^c + \log^* n) = O(\log^* n)$ rounds using standard algorithms. Then, cluster nodes by letting each node join the cluster of the closest node in S , while breaking ties arbitrarily. Since we have no cycle of length less than $3c$, each cluster is a tree of depth at least $c/2$ and with at least $(d-1)^{c/2} > 500$ edges connecting to other clusters, and each two clusters are connected by at most 1 edge. We now think of contracting each cluster into one node. Communications on this contracted graph can be simulated with a constant running time overhead as each cluster has constant diameter. Since each two clusters are connected with at most 1 edge and as each cluster has at least 500 edges connecting to other clusters, the graph after this contraction is a simple graph with each node incident on at least 500 edges or half-edges. We can now orient this graph using the method of [Lemma 2.5](#) in $O(\log \log n)$ rounds. At the end, each contracted cluster has at least one outgoing edge. We can then orient the edges inside the cluster towards this outgoing edge, hence ensuring that all other nodes of the cluster also have out-degree at least 1.

LEMMA 2.7. *There is a randomized algorithm that, for a desirably small constant $\delta > 0$ and sufficiently large constant C , computes an orientation of Δ -regular graphs with $\Delta \geq C$ in $O(\log_{\Delta} \log n)$ rounds which guarantees a lower bound of $(1/6 + \delta)\Delta$ on the in-degree and out-degree of each node, with high probability³.*

Proof. We here simply sketch the necessary changes for obtaining this result. In the randomized algorithm of [Section 2.1.1](#), mark each edge with probability $1/3 - \delta$, for a small constant δ , and redefine type I bad nodes as those with more than $\Delta/3$ marked edges. Also, redefine

³We have not tried to optimize the constants or to extend the result to irregular graphs. We believe that both should be possible without too much more effort.

type III bad nodes to be those which are not type I or type II but still have less than $(1/6 - \delta)\Delta$ incoming marked edges or less than $(1/6 - \delta)\Delta$ outgoing marked edges. It is easy to go over the analysis of this algorithm and see that bad components will induce components of size at most $\text{poly log } n$, with high probability. Moreover, nodes that are not bad (and thus also not bad type III) already have at least $(1/6 - \delta)\Delta$ incoming edges and at least $(1/6 - \delta)\Delta$ outgoing edges.

Now we turn to the deterministic algorithm that is to be run on these bad nodes, each of which is incident on at least $2\Delta/3$ unmarked edges. Now, replace each of these bad nodes v with $\lceil \Delta/6 \rceil$ copy-nodes, and assign 4 edges of v to each of its copies. Leave the remaining edges as half-edges, connected only to the other endpoint. Now we are dealing with a graph H where each node is incident on 4 edges or half-edges. We will compute a sinkless and sourceless orientation of H , hence ensuring that each node of G has at least $(1/6 - \delta)\Delta$ incoming edges and at least $(1/6 - \delta)\Delta$ outgoing edges.

First, compute a sinkless orientation of H using the deterministic algorithm of Section 2.2.1. It is easy to see that in this orientation, all nodes have at least one outgoing edge and at least one incoming edge, except for long nodes which are at maximal distance from short nodes. These long nodes then have only an outgoing edge, but the rest of their edges were oriented arbitrarily. We fix these arbitrary orientations to give these long-nodes also at least one outgoing edge, hence making the overall orientation of H sinkless and sourceless. Let us call those long nodes at maximal distance from short nodes *leaves*. Each leaf v has one of its edges, one of those that go closer to short nodes, oriented outwards. If v has any other edge to a non-leaf node, orient that edge inwards, hence giving v also an incoming edge and solving its case. Each remaining leaf v has $4 - 1 = 3$ edges, which are either half-edges, or they connect to non-leaf nodes. Take the graph induced by the remaining leaves and these remaining edges, orient it sinkless by repeating the deterministic orientation algorithm, and then flip all of these edges. That ensures each of these remaining leaves to also have at least one incoming edge, hence giving us the desired sinkless and sourceless orientation of H .

2.2.1 The Deterministic Part of the Algorithm (Post-Shattering) Consider an N -node graph where each node is incident on at least $d \geq 3$ edges or half-edges. We explain how to find a sinkless orientation of this graph in $O(\log_d N)$ rounds deterministically. Note that when plugging this subroutine in the algorithm of the previous section, we will have $N = \text{poly log } n$ and $d \geq \Delta/2$. Hence, this deterministic piece would work in $O(\log_\Delta \log n)$ rounds.

The Deterministic Algorithm: Orient half-edges outwards from their single endpoint. For the edges, do as follows: Uniquely identify cycles by appending the ids of the related edges. For each cycle, define its preferred orientation by taking the smallest id edge from the lower id node to the higher id node, and then following this direction through the whole cycle. Call a cycle short if it has at most $2 \log_{d-1} N + 1$ edges. Call an edge *short* if it is in at least one short cycle. First, orient each short edge e consistent with the preferred orientation of the smallest id short cycle that contains e . Call a node *short* if it is incident on at least one short edge, or on a half-edge, and *long* otherwise. Then, for each long node v , let u be one neighbor of v who is closer to short nodes (compared to v). Then orient this edge as $v \rightarrow u$.

LEMMA 2.8. *For any $d \geq 3$, this deterministic algorithm works in $O(\log_d N)$ rounds on any N -node graph where each node is incident on at least d edges or half-edges, and it orients these edges or half-edges such that each node has out-degree at least 1.*

Proof. Regarding the time complexity, notice that each half-edge is oriented immediately and each short edge can find its orientation in $O(\log_{d-1} N) = O(\log_d N)$ rounds. Hence, in $O(\log_{d-1} N)$ rounds, we have all short edges oriented. We argue that orienting edges for long nodes can also be done in $O(\log_{d-1} N)$ rounds because each such node has distance at most $O(\log_{d-1} N)$ to some short node. More concretely, we argue that each node has either a half-edge or a short cycle within its $\log_{d-1} N$ neighborhood. This is true because otherwise, the BFS tree of depth $\log_{d-1} N$ rooted at this node would have minimum degree d and depth $\log_{d-1} N$. Such a tree necessarily has more than N nodes, which would be a contradiction.

We next argue that each node has out-degree at least 1. The argument for long nodes is easy as they are oriented towards the short cycles. The argument for nodes incident on half-edges is also trivial. The key part is to argue that in the orientation of short edges, despite the fact that different short edges act according to possibly different short cycles, each short node has out-degree at least 1. For that, consider a short node v , consider all the short edges incident on it, and let C be the smallest id short cycle which contains at least one of these edges. Suppose that C has edges $e_1 = (v, u_1)$ and $e_2 = (v, u_2)$ incident on v . If e_1 is oriented as $v \rightarrow u_1$, we are done. Suppose e_1 is oriented as $u_1 \rightarrow v$. We claim that then it must be the case that e_2 is oriented as $v \rightarrow u_2$, which would finish the proof. This claim is true because the only reason for e_2 to be oriented in the opposite direction is if e_2 is a part of short cycle C' which has an id smaller than that of C . However, that would be in contradiction with the choice of C .

3 Edge-Coloring via Undirected Degree Splitting

In this section, we explain a method for $(2\Delta - 1) \cdot (1 + o(1))$ -edge coloring graphs with maximum degree Δ , based on degree splitting. As a formalized restatement of [Theorem 1.1](#), we get:

THEOREM 3.1. *There is a deterministic distributed algorithm that computes a $(2 + \epsilon)\Delta$ -edge coloring of any graph with maximum degree Δ , in $O(\log^{11} n / \epsilon^3)$ rounds.*

We note that, coarser degree splittings can also be used to obtain much simpler algorithms for edge coloring. Particularly, in [Section 3.1](#), we explain a much simpler method for edge-coloring that matches the bounds of Barenboim and Elkin [[BE11](#)]. However, to get [Theorem 3.1](#), we need to have an almost perfect split, particularly the loss in each degree-splitting iteration should be at most a $1 + 1/\text{poly } \log n$. This would allow us to say that even after $\log n$ iterations, the overall loss is negligible. In [Section 3.2](#) and [Section 3.3](#), we explain how to achieve this fine degree splitting. We first explain in [Section 3.2](#) how to split graphs of maximum degree at most $\text{poly } \log n$ into two spanning subgraphs, each with maximum degree almost half the previous maximum degree. Then, in [Section 3.3](#), we explain how to lift this solution to graphs of higher degree, and how to use that to obtain the claimed edge-coloring result. In [Section 3.4](#), we give a randomized algorithm that is capable of achieving the best possible split. Finally, in [Section 3.5](#), we show how to combine the graph shattering technique and our deterministic algorithm to obtain a faster randomized edge-coloring algorithm.

3.1 Edge-Coloring via Coarse-grained Degree Splitting In this subsection, we explain a deterministic distributed edge-coloring algorithm based on a trivial and crude degree splitting. Interestingly, this simple approach already matches the state of art for very fast algorithms. More concretely, it provides a considerably simpler method for edge-coloring that matches the bounds of Barenboim and Elkin [[BE11](#)]. See also [[BE13](#), Chapter 8].

The Algorithm: Let x be a free parameter. The algorithm is recursive, we explain one level of recursion. Consider graph G , and suppose its maximum degree is Δ . If $\Delta = O(x)$, compute and output a $(2\Delta - 1)$ edge coloring of G in $O(x + \log^* n)$ rounds, using the classical algorithm of Panconesi and Rizzi [[PR01](#)]. Suppose $\Delta = \Omega(x)$. Let each node split itself into $\lceil \Delta/x \rceil$ copy-nodes, and partition its edges between these copy-nodes such that each copy-node is incident on at most x edges. Call this new graph H . Note that H has maximum degree at most x . Use the algorithm of Panconesi and Rizzi [[PR01](#)] to find a $(2x - 1)$ edge coloring of H , in

$O(x + \log^* n)$ rounds. This coloring provides a $(2x - 1)$ coloring of edges of G such that in each color class, there are at most $\lceil \Delta/x \rceil$ edges incident on each node. That is, we have partitioned G into $2x - 1$ graphs G_1, \dots, G_{2x-1} , each with per-node degree at most $\lceil \Delta/x \rceil$. Now recursively run the procedure on each of these subgraphs.

LEMMA 3.1. *The algorithm works in $O((x + \log^* n) \log \Delta / \log x)$ rounds and produces a $2^{1+\log \Delta / \log x} \Delta$ edge coloring.*

Proof. In each iteration, the maximum degree goes down by an x factor. Thus, $\log \Delta / \log x$ recursions suffice. Each recursion level takes $O(x + \log^* n)$ rounds, which means we use $O((x + \log^* n) \log \Delta / \log x)$ rounds in total. To bound the number of colors, let us consider the summation of the maximum degrees in different subgraphs. Since at the end each subgraph will be colored with about 2 factor of its max degree colors, modulo the 2 factor, this summation is an upper bound on the number of used colors. In each iteration, we lose at most a 2 factor in this summation, because we split a graph of maximum degree d into $2x - 1$ subgraphs each of maximum degree at most d/x . Hence, after $\log \Delta / \log x$ recursions, we use $2^{1+\log \Delta / \log x} \Delta$ colors.

The next corollary shows that by setting x appropriately, we can reconstruct the edge-coloring results of Barenboim and Elkin [[BE11](#)]. See also [[BE13](#), Theorem 8.14].

COROLLARY 3.1. *Consider a graph $G = (V, E)$, and let $\epsilon > 0$ be an arbitrarily small constant.*

- (1) *An $O(\Delta)$ -edge-coloring of G can be computed in $O(\Delta^\epsilon + \log^* n)$ time.*
- (2) *A $\Delta^{1+o(1)}$ -edge-coloring of G can be computed in $O((\log \Delta)^{1+\epsilon} + \log^* n \cdot \frac{\log \Delta}{\log \log \Delta})$ time.*
- (3) *An $O(\Delta^{1+\epsilon})$ -edge-coloring of G can be computed in $O(\log^* n \cdot \log \Delta)$ time.*

Proof. Respectively use $x = \Delta^\epsilon$, $x = \log^\epsilon \Delta$, or $x = 2^{1/\epsilon}$, in [Lemma 3.1](#).

3.2 Deterministic Undirected Degree Splitting for Low-Degree Graphs We say a red-blue edge coloring is t -balanced, if there are at most t red edges and at most t blue edges incident to each node. We show the following result:

LEMMA 3.2. *Given a graph G with maximum degree d , a $\lfloor (1 + \epsilon)d/2 \rfloor$ -balanced coloring can be obtained in $O((d^2 \log^5 n)/\epsilon)$ rounds provided that $(4 \log_{1.5} m)/d < \epsilon < 1$.*

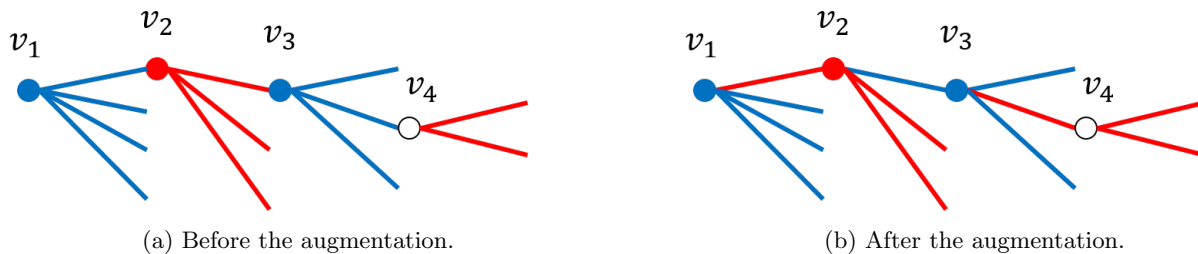


Figure 1: $t = 4$ and $v_1v_2v_3v_4$ forms an augmenting path.

Given a t -balanced coloring with $t > \lfloor (1 + \epsilon)d/2 \rfloor$, we show how to improve the coloring to a $(t - 1)$ -balanced coloring. Then we iterate t from d to $\lfloor (1 + \epsilon)d/2 \rfloor + 1$.

Label a node blue if it has t or $t - 1$ blue edges incident to it. Label a node red if it has t or $t - 1$ red edges incident to it. Note that a node can either be red or blue, but it cannot be both, since $t > \lfloor (1 + \epsilon)d/2 \rfloor$. A node is a *source* if it is incident to exactly t red edges or t blue edges. Let S be the set of all source nodes. An *alternating path* $v_1 \dots v_k$ is a path that satisfies the following:

1. $v_1 \in S$.
2. v_i and v_{i+1} alternate between red and blue for $1 \leq i \leq k - 2$.
3. Edge v_iv_{i+1} is colored the same with the label of v_i for $1 \leq i \leq k - 1$.

An *augmenting path* is an alternating path with an additional condition:

4. v_k is unlabeled or labeled the same as v_{k-1} .

Suppose there is an augmenting path that starts at v_1 . Without loss of generality, suppose that v_1 is blue, v_1v_2 is blue, v_2 is red, v_2v_3 is red, etc. By augmenting along the augmenting path, we recolor v_1v_2 with red, v_2v_3 with blue, etc. The number of red/blue edges incident to v_2, v_3, \dots, v_{k-1} remain the same after the augmentation. The number of blue edges incident to v_1 decreases by 1. If k is even, then $v_{k-1}v_k$ was blue. When we recolor $v_{k-1}v_k$ in red, the number of red edges incident to v_k increases by 1. Since v_k is not labelled red by (4.), after the increment, the number of red edges is still at most $t - 1$. Therefore, the number of source nodes decreases by 1. If k is odd, then a similar argument applies. See Figure 1 for an illustration.

Thus, given an augmenting path, we can decrease the number of source nodes by 1. By finding the augmenting paths repeatedly, we can eliminate all the sources, and so the graph becomes $(t - 1)$ -balanced. However, to reduce the number of sources efficiently, we

need to do multiple augmentations in parallel. Therefore, we find a set of *almost edge-disjoint augmenting paths*, as described below.

Finding Many Almost Edge-Disjoint Augmenting Paths:

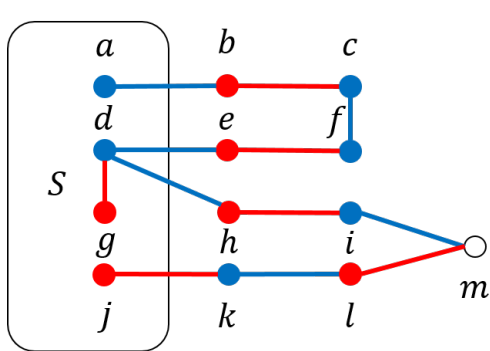
DEFINITION 3.1. (ORDERED DISJOINTNESS) We say two paths $v_1 \dots v_k$ and $v'_1 \dots v'_{k'}$ are *ordered disjoint* if there does not exist $1 \leq i < k$ and $1 \leq j < k'$ such that the ordered pair (v_i, v_{i+1}) and (v'_j, v'_{j+1}) are the same.

A set of almost edge-disjoint augmenting paths, $|S|$, is a set of augmenting paths with the following property (see Figure 2a for an example):

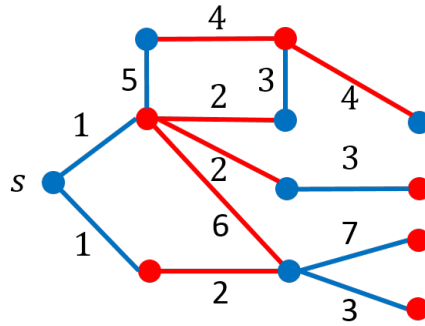
1. The first nodes of the augmenting paths in S are distinct.
2. For any two augmenting paths in S , they are ordered disjoint.

The second property actually characterizes the edge-disjointness property among the paths except on the last edge. Indeed, suppose that (v_i, v_{i+1}) and (v'_j, v'_{j+1}) denote the same edge v_iv_{i+1} (that is, $\{v_i, v_{i+1}\} = \{v'_j, v'_{j+1}\}$). Then v_i and v'_j must have been labeled the same color with the edge v_iv_{i+1} by definition of an augmenting path. Moreover, since $v_i \neq v'_j$, we must have $v_i = v'_{j+1}$ and $v'_j = v_{i+1}$. This implies that v_i and v_{i+1} are labeled the same color and so are v'_j and v'_{j+1} . Therefore, $v_1 \dots v_{i+1}$ and $v'_1 \dots v'_{j+1}$ must be augmenting paths, which implies that v_iv_{i+1} is their last edge.

The augmenting paths may share the same last node. Augmenting along more than one augmenting paths of the same last node may increase the red/blue degree of it to more than $t - 1$, creating new sources. However, if each such node accepts only one path that ends at it and we only augment the accepted paths, then the red/blue degrees can only increase to at most $t - 1$. Since we know that there can be at most d paths in $|S|$ that end at the same node, if we find $\Omega(|S|)$ augmenting paths, at least $1/d$ fraction of them can be augmented. Thus, the sources can be reduced to $(1 - \Omega(1/d)) \cdot |S|$.



(a) A set of almost edge-disjoint augmenting paths. S is the set of sources. The 4 augmenting paths are abc , def , ghm , and jkl . Note that cf is included in both abc and def , as their last edge.



(b) A pseudo-tree rooted at s with 4 leaves. The numbers on the edges denote their depth in the pseudo-tree. Each leaf corresponds to an alternating path from s to it.

Figure 2: A set of almost edge-disjoint augmenting path and a pseudo-tree.

In the rest of this subsection we show how to find $\Omega(|S|)$ almost edge-disjoint augmenting paths deterministically with the restriction that $(4 \log_{1.5} m)/d < \epsilon < 1$. In Section 3.4 we show how to find $\Omega(|S|)$ almost edge-disjoint augmenting paths randomly but without the restriction that $\epsilon d = \Omega(\log n)$.

Technical Overview: Corollary 3.2 in Section 3.4 shows that there exists a set of $|S|$ almost edge-disjoint augmenting paths of length $O(\log n/\epsilon)$. Moreover, the cardinality of any maximal set of almost edge-disjoint augmenting paths of length $O(\log n/\epsilon)$ is $|S|$. This implies in the sequential setting we can find such a set of paths greedily. Similar to the approach in [LPSP15] by Lokter et al., in Section 3.4, we show it can be found in the distributed setting randomly by simulating Luby’s maximal independent set algorithm on a super-graph. However, finding a maximal set of (almost) edge-disjoint augmenting paths deterministically is more challenging technically. Fortunately, in this problem, we have the property that if we build a search tree from a source to search for an augmenting path, the search tree expands very quickly. This property allows us to build (almost) edge-disjoint search trees from different sources such that they are still able to expand quickly. Note that our algorithm and analysis below is independent of that in Section 3.4.

The deterministic algorithm is described in Algorithm 2. The main idea is to grow a tree-like structure from each $s \in S$ simultaneously. We call these *pseudo-trees* (See Figure 2b for an example). In a pseudo-tree, each edge is associated with 0, 1, or 2 children edges, who are the adjacent edges. Each edge except those who are adjacent to s have exactly one parent edge. The edges without children are leaves. Also, the structure does not contain a cycle. That is, if we view each edge as a

node, then the pseudo-tree is a collection of rooted trees whose roots are the edges incident to s . Thus, there is an unique path from s to each leaf. We require each such path to be an alternating path.

When we grow the pseudo-trees T_s and $T_{s'}$ from different sources s and s' , we require that any alternating paths $v_1 \dots v_k$ from T_s and $v'_1 \dots v'_k$ from $T_{s'}$ are ordered disjoint. This ensures that if we extract an augmenting path from each of the pseudo-trees, they will be almost edge-disjoint.

To grow the pseudo-trees simultaneously, each source $s \in S$ maintain a set of tokens. Initially, there is only one token starting at each source $s \in S$. Then, they split as they travel. The edges traveled by the tokens from s form a pseudo-tree of s . In each step, the tokens at each node request for the edges to grow the pseudo-trees. Because of the ordered disjointness property, each edge uv can be assigned only once to the tokens at u and once to the tokens at v . In our algorithm, each request of the token will be granted either 1 edge or 2 edges. Those who got 1 edge will travel along the edge. Those who got 2 edges will split into two tokens. Then each edge will be traveled by one of the token. We will assign the edges properly so that a large fraction of the pseudo-trees will grow exponentially.

Define $L_1 = 1$ and $L_{i+1} = 2\lceil 3L_i/4 \rceil$. Algorithm 2 consists of multiple levels. At level i , an active source s maintains L_i tokens. The goal is for $\lceil 3L_i/4 \rceil$ tokens of s to split into two tokens, so the number of tokens becomes $L_{i+1} = 2\lceil 3L_i/4 \rceil$ (discard the rest). Once a token splits into two, they will pause until the next level. We will show that a large fraction of sources achieve the goal in each level.

Each level has h steps. In each step j of level i , the active tokens are assigned either 1 or 2 edges. If a node

Algorithm 2 Finding Almost Edge-Disjoint Augmenting Paths(S)

Each source $s \in S$ create a token located at s .

for level $i = 1, 2, \dots, \log_{1.5} m$ **do**

For each node u , set its budget, $budget(u)$, to be $\lfloor (2t - d - 2) / \log_{1.5} m \rfloor$.

for step $j = 1, 2, \dots, h$, where $h = \lceil (16/3) \log_{1.5}^2 m / \epsilon \rceil$ **do**

Each active token requests for edge assignment at its current node.

For each node u , $\min(\#request, budget(u))$ requests will be granted two tokens, others one edge. Each red (blue) node assign unused, distinct red (blue) edges to tokens. Update $budget(u)$.

if a token has been assigned one edge **then**

Travel along the edge.

else

▷ The token has been assigned two edges.

Split into two tokens. Then each token take one of the edge and travel along it.

Deactivate the tokens.

For each source:

if a path from the source to one of its tokens forms an augmenting path **then**

Deactivate all the tokens from s and save the augmenting path.

else if it has less than L_{i+1} **then**

Mark s as **failure** and remove s from S and deactivate all the tokens.

else if it has at least L_{i+1} **then**

Discard the tokens so that it has exactly L_{i+1} tokens.

Set the L_{i+1} tokens to be active.

has more budget than the number of requesting tokens, then every token on it is given 2 edges. Otherwise, the node is allowed to assign $budget(u)$ tokens 2 edge; other tokens get 1 edge.

Now we bound the number of requests at u by $d - t + 1$. First note that the requests happen only at the labelled nodes, since if a token reaches an unlabelled node then an augmenting has been found. W.l.o.g. assume u is blue. There are two cases. If u has exactly $t - 1$ blue edges, then there is at most $d - t + 1$ red edges incident to it. If a token enters through one of the blue edges then the augmenting path must have been found and so the token will be deactivated. Therefore, those who are requesting must enter u through red edges. Thus, the number of requests is at most $d - t + 1$. If u has exactly t blue edges, then there is at most $d - t$ red edges incident to it. By the same argument, the number of requests is at most $d - t$. However, since $u \in S$, an additional token has been created at u initially. In any case, the number of requests at u is bounded $d - t + 1$.

The intuition of why each token can split without traveling too far is that because the number of edges that can be assigned is at least $t - 1$, the average number of edges that can be assigned to each request is $(t - 1) / (d - t + 1) = 1 + \Omega(\epsilon)$, since $t > \lfloor (1 + \epsilon)d / 2 \rfloor$. Therefore, intuitively, it seems possible for each pseudo-tree to grow by an $1 + \Omega(\epsilon)$ factor in each step. However, since $1 + \Omega(\epsilon)$ is not even an integer, it is not clear what it means to have each token to split into $1 + \Omega(\epsilon)$ tokens. Instead, our analysis shows that most tokens will split into two after $\tilde{O}(1/\epsilon)$ steps.

We use $budget(u)$ to control how many edges can a node assign in each level. Suppose that we assign 1 edge to all the $d - t + 1$ requests first, then the number of unused blue edges is at least $t - 1 - (d - t + 1) = 2t - d - 2$. These edges can be used to grant $2t - d - 2$ tokens 1 additional edge. We divide the budget across all the $\log_{1.5} m$ levels, so each level has $\lfloor (2t - d - 2) / \log_{1.5} m \rfloor$ budget. We say a token did not *successfully split*, if it has not been granted two edges during the h steps. We say a source *failed* if less than $3/4$ fraction of their tokens successfully split. Given the budget, we show that the number of failure sources in each level is bounded. If a source did not fail or did not found an augmenting path yet, then we say it is *active*.

LEMMA 3.3. *Suppose that $(4 \log_{1.5} m) / d < \epsilon < 1$. Let S_i denote the set of active sources at the beginning of level i and let F_i be the set of failure sources during level i . We have $|F_i| \leq |S_i| / (2 \log_{1.5} m)$.*

Proof. First, call a token *unlucky* if it has not been successfully split after the h steps. Otherwise, we say a token is *lucky* (either an augmenting path has been found or the token has successfully split.) Consider a bipartite multi-graph where the left nodes X denote the unlucky tokens and the right nodes Y denote the lucky tokens. Add an edge between an unlucky token x and a lucky token y , if y 's split prevents x to split on one of its h steps. Note that in case x traveled through the same node multiple times, multiple edges can be added between the same pair of nodes.

Observe that the degree of a left node is exactly $h \cdot \lfloor (2t - d - 2) / \log_{1.5} m \rfloor$. The number of edges in the bipartite graph is $h \cdot |X| \cdot \lfloor (2t - d - 2) / \log_{1.5} m \rfloor$. Also,

the degree of each right node is at most $d - t + 1$, as the number of requests at each node is at most $d - t + 1$. Therefore,

$$\begin{aligned}
|Y| &\geq h \cdot |X| \cdot \lfloor (2t - d - 2) / \log_{1.5} m \rfloor / (d - t + 1) \\
&\geq |X| \cdot \frac{h}{d} \cdot \left\lfloor \frac{ed - 2}{\log_{1.5} m} \right\rfloor \quad t \geq \lfloor (1 + \epsilon)d/2 \rfloor + 1 \\
&\geq |X| \cdot \frac{h}{d} \cdot \left(\frac{ed}{2 \log_{1.5} m} \right) \\
&\quad ed \geq 4 \log_{1.5} m \text{ and when } \log_{1.5} m \geq 2 \\
&\geq |X| \cdot (8/3) \log_{1.5} m \quad h \geq (16/3) \log_{1.5}^2 m / \epsilon
\end{aligned}$$

Therefore,

$$\begin{aligned}
|S_i| \cdot L_i &= |X| + |Y| \\
&\geq |X| \cdot (1 + (4/3) \log_{1.5} m) \\
&\geq |F_i| \cdot \left(\frac{3}{4} L_i \right) \cdot (1 + (4/3) \log_{1.5} m) \\
&\geq |F_i| \cdot \left(\frac{3}{4} L_i \right) \cdot \left(\frac{8}{3} \log_{1.5} m \right)
\end{aligned}$$

Thus, $|F_i| \leq |S_i| / (2 \log_{1.5} m)$.

LEMMA 3.4. *Let $A_i \subseteq S_i \setminus F_i$ denote the set of sources that successfully find an augmenting path during level i . $|\bigcup_i A_i| \geq |S|/2$.*

Proof. Notice that $S_{i+1} = S_i \setminus (A_i \cup F_i)$. First we claim that $|\bigcup_i F_i| \leq |S|/2$. Since $|F_i| \leq |S_i| / (2 \log_{1.5} m) \leq |S| / (2 \log_{1.5} m)$, we have $\sum_i |F_i| \leq |S|/2$. Now we show that every source $x \in S \setminus \bigcup_i F_i$ must be in one of A_i . Suppose to the contrary, there exists $x \in S$ such that $x \in S_{\log_{1.5} m} \setminus (A_{\log_{1.5} m} \cup F_{\log_{1.5} m})$. Consider the edges travel by the tokens of x . At level i , since it has successfully advanced to level $i+1$, it must have traveled at least L_{i+1} edges. The total number of edges traveled by the tokens of x is $\sum_{i=1}^{\log_{1.5} m} L_{i+1} > (3/2)^{\log_{1.5} m} = m$. A contradiction occurs.

In [Algorithm 2](#), level i takes $O(i \cdot h)$ rounds, since the length of the path spanned by the tokens from the source is $O(i \cdot h)$ and they have to communicate with the source at the end of the level. Therefore, the total number of rounds of [Algorithm 2](#) is $O(\sum_{i=1}^{\log_{1.5} m} i \cdot h) = O(h \log_{1.5}^2 m) = O((\log^4 n) / \epsilon)$.

Proof. [Proof of [Lemma 3.2](#)] Given a t -balanced coloring, we can improve it to a $(t-1)$ -balanced coloring by calling [Algorithm 2](#) repeatedly. By [Lemma 3.4](#), each invocation of [Algorithm 2](#) finds $|S|/2$ almost edge-disjoint augmenting paths from distinct sources. Of those $|S|/2$ augmenting paths, at least $1/d$ fraction will be accepted

and augmented, since there can be at most d paths ending with the same node. Therefore, the source reduces by a $1/(2d)$ fraction for each invocation. Since $|S| \leq n$, after $O(d \log n)$ invocations, we have obtained a $(t-1)$ -balanced coloring. If we iterate t from d to $\lfloor (1 + \epsilon)d/2 \rfloor - 1$, we obtain a $\lfloor (1 + \epsilon)d/2 \rfloor$ -balanced coloring. The total number of invocation of [Algorithm 2](#) is $O(d^2 \log n)$. Thus, the running time is $O((d^2 \log^5 n) / \epsilon)$.

3.3 Deterministic Undirected Degree Splitting for High-Degree Graphs

Suppose that the input is a graph with maximum degree Δ . In this case, if we apply [Lemma 3.2](#) directly, it takes $\tilde{O}(\Delta^2 / \epsilon)$ rounds to get a $\lfloor (1 + \epsilon)\Delta/2 \rfloor$ -balanced coloring. Here, we show a method which removes the dependency on Δ .

THEOREM 3.2. *Suppose that $\Delta \geq \lceil 32 \log_{1.5} m / \epsilon^2 \rceil$, then a $\lfloor (1 + \epsilon)\Delta/2 \rfloor$ -balanced coloring can be obtained in $O((\log^7 n) / \epsilon^3)$ rounds.*

Proof. Let $\epsilon' = \epsilon/2$ and $d = \lceil 4 \log_{1.5} m / \epsilon' \rceil$. Thus, we have $\Delta \geq 2d/\epsilon'$. First, obtain G' as follows: For each node u , split it into $\lceil \Delta/d \rceil$ copy-nodes and divide evenly the edges adjacent to u between the copy-nodes such that each copy-node, except possibly one, has degree d . Since $\epsilon'd \geq 4 \log_{1.5} m$, we can apply [Lemma 3.2](#) to get a $\lfloor (1 + \epsilon')d/2 \rfloor$ -balanced coloring in G' in $O((d' \log^5 n) / \epsilon') = O((\log^7 n) / \epsilon^3)$ rounds. Then, we merge the copy-nodes back. Then, for each node of G , the number of incident edges of each color is bounded by:

$$\begin{aligned}
\left\lfloor \frac{(1 + \epsilon')d}{2} \right\rfloor \cdot \left\lceil \frac{\Delta}{d} \right\rceil &\leq \frac{(1 + \epsilon')d}{2} \cdot \left(\frac{\Delta}{d} + 1 \right) \\
&\leq \frac{\Delta}{2} \cdot (1 + \epsilon') + d \\
&\leq \frac{\Delta}{2} \cdot \left(1 + \epsilon' + \frac{2d}{\Delta} \right) \leq \frac{\Delta}{2} \cdot (1 + \epsilon) \quad \Delta \geq 2d/\epsilon'
\end{aligned}$$

Since the number of colors must be an integer, it is a $\lfloor (1 + \epsilon)\Delta/2 \rfloor$ -balanced coloring.

Proof. [Proof of [Theorem 3.1](#)]

Let $\epsilon' = \epsilon / (2 \log_2 \Delta)$ and $\Delta_0 = \Delta$. Suppose that $\Delta_0 \leq \lceil 32 \log_{1.5} m / \epsilon'^2 \rceil$, we can use Panconesi and Rizzi's algorithm [[PR01](#)] that runs in $O(\Delta_0 + \log^* n)$ rounds to get $(2\Delta_0 - 1)$ -edge coloring. Otherwise, we apply [Theorem 3.2](#) to get a $\lfloor (1 + \epsilon')\Delta/2 \rfloor$ -balanced coloring. For the subgraph consists of red edges and the subgraph consists of blue edges, we recursively apply this procedure on both of them in parallel with a new maximum degree $\Delta_{i+1} = \lfloor (1 + \epsilon')\Delta_i/2 \rfloor$. Let t be the level where the recursion halts. That is, t is the smallest integer such that $\Delta_t \leq \lceil 32 \log_{1.5} m / \epsilon'^2 \rceil$. The recursion will stop at level t , where we will apply Panconesi and Rizzi's algorithm to get an $(2\Delta_t - 1)$ -edge coloring on

each subgraph. Since the number of subgraphs at level t is 2^t , the total number of color used is

$$\begin{aligned}
(2\Delta_t - 1) \cdot 2^t &\leq 2 \cdot (1 + \epsilon')^t \Delta \\
&\leq 2\Delta + 4t\epsilon' \Delta \\
(1 + x)^n &\leq 1 + 2nx \text{ for } 0 \leq nx \leq 1 \\
&\leq (1 + 2t\epsilon') \cdot 2\Delta \\
&\leq (1 + \epsilon) \cdot 2\Delta \quad t \leq \log_2 \Delta
\end{aligned}$$

We apply the balanced coloring procedure for $O(\log \Delta)$ rounds, each takes $O(\log^{10} n / \epsilon^3)$ rounds by [Theorem 3.2](#). At the last level, Panconesi and Rizzi's algorithm takes $O(\Delta_k + \log^* n) = O(\log n / \epsilon'^2 + \log^* n) = O(\log^3 n / \epsilon'^2)$ rounds. Therefore, the total number rounds is: $O(\log^{11} n / \epsilon^3)$.

3.4 Randomized Undirected Degree Splitting

In this subsection, we give a randomized distributed algorithm for obtaining a $\lceil (1 + \epsilon)\Delta/2 \rceil$ -balanced coloring where $0 < \epsilon < 1$ in $O(\Delta^2 \log^4 n / \epsilon^2)$ rounds. Note that this allows one to obtain a $\lceil (\Delta + 1)/2 \rceil$ -balanced coloring in $O(\text{poly}(\log n, \Delta))$ rounds. Also note that the change from floor in the previous sections to ceiling is necessary. Consider when $\Delta = 2$ and $\epsilon = 1/2$. Obtaining a 1-balanced coloring is impossible in an odd cycle. In the previous sections, the restriction that $\epsilon\Delta = \Omega(\log n)$ avoided this problem.

THEOREM 3.3. *Given a graph G with maximum degree Δ , a $\lceil (1 + \epsilon)\Delta/2 \rceil$ -balanced coloring can be obtained in $O(\Delta^2 \log^4 n / \epsilon^2)$ for $0 < \epsilon < 1$.*

We use the same approach with that in the previous sections. Given a t -balanced coloring, we show how to improve the coloring to a $(t - 1)$ -balanced coloring. Then we iterate t from Δ to $\lceil (1 + \epsilon)\Delta/2 \rceil + 1$. The only difference is that we show how to find a set of almost edge-disjoint augmenting paths of size $\Omega(|S|)$ in $O(\text{poly}(\log n, \epsilon^{-1}))$ rounds *without* the restriction that $\epsilon\Delta = \Omega(\log n)$.

The following lemma can be used to show that the size of any maximal set of almost edge-disjoint short augmenting paths is $|S|$.

LEMMA 3.5. *Let \mathcal{P}' be a set of almost edge-disjoint augmenting paths. If s is not a source of an augmenting path in \mathcal{P}' , then an augmenting path P from s of length at most $O(\log n / \epsilon)$ exists and $\mathcal{P}' \cup P$ is also a set of almost edge-disjoint augmenting paths.*

Proof. Grow a tree from s such that the path from s to the leaves are alternating paths. In each step, each leaf grows by adding all the edges that have the same color with it. Three cases may occur when a leaf tries to add an edge that does not intersect with any paths in \mathcal{P}' .

If the endpoint of an edge is an unvisited node (i.e. the node is not in the tree) with the opposite color, then it will be added to the tree. If the endpoint of an edge is a node with the same color or an unlabelled node, then an augmenting path is found. Otherwise, if the endpoint is a visited node, we will ignore it.

There are two cases when a leaf u tries to add an edge uv that intersects with some path $Q \in \mathcal{P}'$. If u comes before v in Q , then we will not add v to the tree and ignore it. If v comes before u in Q , then it must be the case that u and v are labeled the same color. Therefore, uv is the last edge of Q and an augmenting path P has been found from s . Therefore, any augmenting path found during this process must be almost edge-disjoint from \mathcal{P} .

Suppose that an augmenting path has not been found and T' is the tree after growing T by one level. We show that $|T'| \geq (1 + \epsilon)|T|$. First, note that each node $u \in T$ is not the last node of any path Q in \mathcal{P} . Let $G' = G \setminus \mathcal{P}$ be the graph obtained by deleting all the edges in \mathcal{P} from G . Let $\text{indeg}_H(u)$ denote the number of incident edges to u with the opposite color in a subgraph H . Let $\text{outdeg}_H(u)$ denote the number of incident edges to u with the same color in H . We claim that for any $u \in T$, $\text{outdeg}_{G'}(u) - \text{indeg}_{G'}(u) \geq 2t - 2 - d$.

Note that since u is a labelled node, we have $\text{outdeg}_G(u) \geq t - 1$, $\text{indeg}_G(u) \leq d - t + 1$ and so $\text{outdeg}_G(u) - \text{indeg}_G(u) \geq 2t - 2 - d$. Suppose that u is not the first node of any paths in \mathcal{P} , then both $\text{indeg}_G(u)$ and $\text{outdeg}_G(u)$ decreases by 1 when we delete the path from G . This implies $\text{outdeg}_{G'}(u) - \text{indeg}_{G'}(u) \geq 2t - 2 - d$. On the other hand, if u is the first node of some path in \mathcal{P} , then u must be a source. In this case, deleting the path decreases $\text{outdeg}_G(u)$ by 1. Also, since u is a source, $\text{outdeg}_G(u) - \text{indeg}_G(u) \geq 2t - d$. Therefore, $\text{outdeg}_{G'}(u) - \text{indeg}_{G'}(u) \geq \text{outdeg}_G(u) - 1 - \text{indeg}_G(u) \geq 2t - 1 - d$.

For each edge uv where $u, v \in T$, it must be the case that u and v are colored differently. Otherwise, an augmenting path would have been found. Therefore, there must be at least $\sum_{u \in T} (\text{outdeg}_{G'}(u) - \text{indeg}_{G'}(u)) \geq |T| \cdot (2t - 2 - d) \geq |T| \cdot (2(\lceil (1 + \epsilon)d/2 \rceil + 1) - 2 - d) \geq \epsilon|T| \cdot d$ edges going outside of T . Since the maximum degree is d , the number of nodes added must be at least $\epsilon|T|$. Therefore, $|T'| \geq (1 + \epsilon)|T|$. After $O((\log n)/\epsilon)$ levels, the tree would grow to contain more than n nodes. Therefore, an augmenting path must have been found before this happens.

COROLLARY 3.2. *The size of any maximal set of almost edge-disjoint augmenting paths of length $O(\log n / \epsilon)$ is $|S|$.*

Proof. If \mathcal{P} is a maximal set of almost edge-disjoint augmenting paths of length $O(\log n / \epsilon)$ with cardinality less than $|S|$, then there is a source $s \in S$ that does

not appear in \mathcal{P} . We can apply [Lemma 3.5](#) to add an augmenting path of length $O(\log n/\epsilon)$ to \mathcal{P} without violating the maximality condition.

LEMMA 3.6. *A maximal set of almost edge-disjoint augmenting paths of length at most l can be found in $O(l^2 \log n)$ rounds.*

Proof. We construct a super-graph \mathcal{G} where each node in \mathcal{G} denotes an augmenting path of length at most l . Two nodes P_1 and P_2 are connected if they are not ordered disjoint or if they share the same source. A maximal independent set (MIS) in \mathcal{G} corresponds to a maximal set of almost edge-disjoint augmenting paths of length at most l in G .

To simulate the computation on \mathcal{G} , we let each source $s \in S$ to be responsible for the nodes whose corresponding augmenting path start at s . Then, one round of communication in \mathcal{G} can be simulated in $O(l)$ rounds. The total number of nodes in \mathcal{G} is $O(n^l)$. Therefore, Luby's MIS algorithm takes $O(\log(n^l)) = O(l \log n)$ rounds in \mathcal{G} . Since each round in \mathcal{G} can be simulated in $O(l)$ rounds, the number of rounds needed to simulate Luby's algorithm is $O(l^2 \log n)$.

By setting $l = O(\log n/\epsilon)$, we can find $|S|$ almost edge-disjoint augmenting paths in $O(\log^3 n/\epsilon^2)$ rounds. Then, at least $1/\Delta$ fraction of the paths can be augmented, since there are at most Δ augmenting paths could end at the same node and one of them will be augmented. Therefore, since the number of sources is at most n , after $O(\Delta \log n)$ iterations, all the sources are saturated. Also, since we iterate t from Δ to $\lceil (1 + \epsilon)\Delta/2 \rceil$, the total number of rounds is $O(\Delta^2 \log^4 n/\epsilon^2)$.

3.5 Randomized Edge Coloring In this subsection, we show how to obtain a faster randomized edge-coloring algorithm using $(4 + \epsilon)\Delta$ colors, in $\text{poly}(\log \log n)$ rounds. This is by combining the graph shattering technique with our deterministic algorithm. For $\Delta \in [\text{poly}(\log \log n), \frac{\log n}{\text{poly}(\log \log n)}]$, this is faster than the $O(\sqrt{\Delta} \log \Delta)$ -round $O(\Delta)$ -edge-coloring that follows from the work of Barenboim [[Bar15](#)], and the $O(\log^* \Delta \cdot \max\{1, \frac{\log n}{\Delta^{1-o(1)}}\})$ -round $((1 + \epsilon)\Delta)$ -edge-coloring result of Elkin, Pettie, and Su [[EPS15](#)].

THEOREM 3.4. *Given a graph G and $0 < \epsilon < 1$, a $(4 + \epsilon)\Delta$ -edge coloring can be obtained in $O((\log^{11} \log n)/\epsilon^3)$ rounds.*

Proof. First we will assume that $\Delta = O(\log^2 n)$. Since for $\Delta = \Omega(\log^2 n)$, by using Elkin et al.'s algorithm [[EPS15](#), Theorem 2.1], a $(1 + o(1))\Delta$ -edge-coloring can be obtained in $O(\log^* n)$ rounds. Let $\epsilon' = \epsilon/4$. We will

divide the $(4 + \epsilon)\Delta$ colors into two sets C_1 and C_2 with an equal size, so each set consists of $2(1 + \epsilon')\Delta$ colors.

Pre-shattering: Let $x = \epsilon'^2 \Delta / (18 \log \Delta)$. First we partition the edges randomly to form subgraphs G_1, G_2, \dots, G_x . The expected degree of each node u in G_i is at most $\Delta/x = (18 \log \Delta)/\epsilon'^2$. By Chernoff Bound, $\Pr(\deg_{G_i}(u) \geq (1 + \epsilon')\Delta/x) \leq e^{-\epsilon'^2(18 \log \Delta)/(3\epsilon'^2)} \leq (1/\Delta)^6$. For $v \in G$, we mark v as a Type I node if there exists $1 \leq i \leq x$, such that $\deg_{G_i}(u) \geq (1 + \epsilon')\Delta/x$. By an union bound over $1 \leq i \leq x$, the probability that v is Type I is at most $x \cdot (1/\Delta)^6 \leq 1/\Delta^5$.

If $u \in G$ is a not Type I node but it is adjacent to a Type I node, then it is a Type II node. Type I nodes and Type II nodes are the bad nodes. Let B be the set of bad nodes and V' be the set of Type I nodes.

Post-shattering: First note that every subgraph $G_i[V \setminus V']$ has maximum degree bounded by $(1 + \epsilon')\Delta/x$. We divide the colors in C_1 evenly into $C_{i1}, C_{i2}, \dots, C_{i_x}$ so that each has size $2(1 + \epsilon')\Delta/x$. We will run Panconesi and Rizzi's algorithm [[PR01](#)] to get a $2(1 + \epsilon')\Delta/x$ -edge coloring in $O(\Delta/x + \log^* n) = O((\log \Delta)/\epsilon'^2 + \log^* n) = O((\log \log n)/\epsilon'^2)$ rounds on each $G_i[V \setminus V']$ in parallel with the color set C_{i_i} .

Now the uncolored edges must be the ones that are adjacent to V' . Since $B = N(V') \cup V'$, all the uncolored edges must be in $G[B]$. Similar to the analysis of sinkless orientation in [Section 2](#), we will show that each component in $G[B]$ has their size bounded by $\text{poly}(\log n)$. Then, we will apply our deterministic algorithm on each component with the color set C_2 .

Let $\text{dist}(u, v)$ denote the distance between u and v in G . If $\text{dist}(u, v) \geq 2$, then it is clearly that the event u becomes Type I and the event v becomes Type I are independent. Let $E_{2,4} = \{uv \mid 2 \leq \text{dist}(u, v) \leq 4\}$ denote the set of edges whose endpoints have distance between 2 and 4.

LEMMA 3.7. *Let C be a connected components in $G[B]$. Then, there exists $S \subseteq V' \cap C$ such that $|S| \geq |C|/\Delta^2$ and $(S, E_{2,4})$ is connected.*

Proof. The proof is exactly the same with that of [Lemma 2.3](#) except that now we are omitting Type III nodes.

LEMMA 3.8. *The probability that any tree T with $T \subseteq (V', E_{2,4})$ and $|T| = \Omega(\log n)$ exists is at most $1/\text{poly}(n)$.*

Proof. Let T be a tree such that $T \subseteq (V', E_{2,4})$ and $|T| = t$. The probability that a node in T is marked as Type I is at most $1/\Delta^5$. Therefore, the probability that T occurs is at most $1/\Delta^{5t}$, since the events each node in T becomes Type I are independent. The total possible number of such trees is at most $4^t n (\Delta^4)^{t-1}$. By

an union bound over all possible trees, the probability that any tree in $(V', E_{2,4})$ of size t occur is at most $n \cdot (4\Delta^4 \cdot (1/\Delta^5))^t$. For $\Delta \geq 4e$, this is at most $n \cdot e^{-t}$. By summing over $t \geq 10 \log n$, the probability that any tree in $(V', E_{2,4})$ of size at least $10 \log n$ is at least $(1/n^9) \cdot \sum_{i=0}^{\infty} e^{-i} = 1/\text{poly}(n)$.

Therefore, if C is a connected component in $G[B]$ of size $\Omega(\Delta^2 \log n)$. Then by Lemma 3.7, there exists a set of vertices $S \subseteq V' \cap C$ with $|S| = \Omega(\log n)$ and $(S, E_{2,4})$ is connected. Take a spanning tree T of the graph $(S, E_{2,4})$. Since $|T| = \Omega(\log n)$, by Lemma 3.8, the probability that $|T|$ exists is at most $1/\text{poly}(n)$. Therefore, we conclude that the probability $|C| = \Omega(\Delta^2 \log n)$ is at most $1/\text{poly}(n)$.

Since $\Delta = O(\log^2 n)$, each component has size at most $O(\log^3 n)$. Now we will run our deterministic $(2 + \epsilon')\Delta$ -edge coloring using colors in C_2 on each of the component. By Theorem 3.1, the running time is $O((\log^{11} \log n)/\epsilon^3)$.

4 Directed Degree Splitting

In this section, we consider the problem of orienting the edges such that the out-degree and in-degree of each node are both upper bounded by D , for any given $D \geq \lceil (\Delta + 1)/2 \rceil$. The following lemma allows us to focus on only the out-degree side of the problem, and then extend it to both out-degree and in-degree:

LEMMA 4.1. *Let $D \geq \lceil (\Delta + 1)/2 \rceil$. Let G be a graph with an arbitrary orientation. Suppose that A is a distributed algorithm that orients G into G' such that the out-degree of each node is at most D in T rounds with the following property: For each u , $\text{outdeg}_{G'}(u) \geq \min(\text{outdeg}_G(u), D)$. Then in $O(T)$ rounds, G can be oriented such that both the in-degree and the out-degree of each node is at most D .*

Proof. First run A on G to obtain an orientation with out-degree at most D , say the resulting graph is G_1 . Then, we reverse each edge in G_1 to obtain G_2 . G_2 is a graph such that the in-degree of each node is at most D . Now, run A on G_2 to obtain G_3 . If the in-degree of a node u in G_2 is at most D , then

$$\begin{aligned} \text{indeg}_{G_3}(u) &= \text{deg}(u) - \text{outdeg}_{G_3}(u) \\ &\leq \text{deg}(u) - \min(\text{outdeg}_{G_2}(u), D) \\ &\quad \text{by the property of } A \\ &\leq \text{deg}(u) - \min(\text{deg}(u) - D, D) \\ &\quad \text{indeg}_{G_2}(u) \leq D \\ &\leq \text{deg}(u) - (\text{deg}(u) - D) = D \\ &\quad D > \Delta/2 \text{ and } \text{deg}(u) \leq \Delta \end{aligned}$$

Therefore, the in-degree of every node is still at most D in G_3 .

By Lemma 4.1, it suffices to develop algorithms that orient the graph such that each node's out-degree is bounded by D , with the stated additional property. Our augmentation-based approach satisfies this property, because we only decrease the out-degree in nodes with out-degree at least $D + 1$.

For deterministic algorithms, we will use the same approach as in Section 3. Recall that in Section 3 we assumed that a t -balanced coloring is given, and we showed how to improve it to a $(t - 1)$ -balanced coloring. Similarly, here we assume that we have an orientation with the out-degree of each node upper bounded by t . We use the same approach to improve it to an orientation with the out-degree of each node upper bounded by $t - 1$. The outer-loop will iterate t from Δ to $D + 1$.

The definition of an augmenting path is much more straightforward here. An augmenting path is just a directed path that starts from a node with out-degree equals to t and ends at a node with outdegree at most $t - 2$. We augment along a path by reversing the orientation of each edge. After we augmented along an augmenting path, the out-degree of the first node decreased by 1 and the out-degree of the last node increased by 1. All the other nodes remain to have the same out-degree. Since $t > D$, we will not decrease the out-degree of a node with out-degree at most D . This satisfies the property stated in Lemma 4.1.

Let S be the source nodes, which are the nodes with out-degree t . Similar to Section 3, we will find a set of $\Omega(|S|)$ edge-disjoint augmenting paths from distinct sources in S here. However, the singularity that appeared in Section 3 where the last edges of augmenting paths in a set of almost edge-disjoint augmenting paths may overlap do not appear here. We just need to find a set of edge-disjoint augmenting path. By repeating the same arguments as in Section 3, we can obtain the analogues of Lemma 3.2 and Theorem 3.2, which we state without proofs.

LEMMA 4.2. *Given a graph G with maximum degree d , an orientation where the out-degree and the in-degree of each node is at most $\lfloor (1 + \epsilon)d/2 \rfloor$ can be obtained in $O((d^2 \log^5 n)/\epsilon)$ rounds provided that $(4 \log_{1.5} m)/d < \epsilon < 1$.*

THEOREM 4.1. *Suppose that $\Delta \geq \lceil 32 \log_{1.5} m/\epsilon^2 \rceil$, then an orientation where the out-degree and the in-degree of each node is at most $\lfloor (1 + \epsilon)\Delta/2 \rfloor$ can be obtained in $O((\log^7 n)/\epsilon^3)$ rounds.*

4.1 Randomized Directed Degree Splitting and Graphs with Bounded Arboricity In this subsection, we show how to obtain an orientation with the out-degree of each node bounded by $\lceil (1 + \epsilon)a \rceil$ for any $0 < \epsilon < 1$ in graphs with arboricity bounded by a .

THEOREM 4.2. *There is a randomized distributed algorithm that in $O(\log^4 n/\epsilon^3)$ rounds, produces an orientation of a -arboricity graphs with per-node out-degree at most $\lceil(1+\epsilon)a\rceil$, for any $0 < \epsilon < 1$.*

Notice that by setting $\epsilon = 1/a$, we get an orientation with per-node out-degree at most $a + 1$ in $O(a^3 \log^4 n)$ rounds. Moreover, notice that each graph with maximum degree Δ has arboricity $a \leq \Delta/2$. Hence, the above theorem already supplies an orientation with per-node out-degree at most $\lceil(1+\epsilon)\Delta/2\rceil$, in $O(\log^4 n/\epsilon^3)$ rounds, which can again be made a much finer orientation with out-degree at most $\lceil(\Delta+1)/2\rceil$, in $O(\Delta^3 \log^4 n)$ rounds, by setting $\epsilon = 1/\Delta$. Moreover, our algorithm is again augmentation-based and it satisfies the properties in [Lemma 4.1](#), which can be used to satisfy the requirement on both the in-degree and the out-degree of each node.

COROLLARY 4.1. *Given a graph G with maximum degree Δ , an orientation where the in-degree and the out-degree of each node is bounded by $\lceil(1+\epsilon)\Delta/2\rceil$ can be obtained in $O(\log^4 n/\epsilon^3)$ for $0 < \epsilon < 1$.*

Note that in comparison with our randomized algorithm stated in [Theorem 3.3](#) of [Section 3.4](#), we have removed the dependency on Δ in the running time. This is because in [Section 3.4](#) there is a singularity on the augmenting paths who intersects on the last edge. Here, the structure of the augmenting paths is the same with the flow networks. This allows us to apply the known techniques such as the arguments of blocking-flows.

We now explain our method for achieving [Theorem 4.2](#). Let G_0 be a directed graph obtained by orienting the original graph arbitrarily. Let $D = \lceil(1+\epsilon)a\rceil$. Define an augmenting path to be a directed path starting from a node with out-degree at least $D + 1$ and ends at a node with out-degree at most $D - 1$. By augmenting along an augmenting path, we flip all the edges to the reverse direction. After the augmentation, the out-degree of the starting node decreased by 1 and the out-degree of the ending node increased by 1.

Define G'_0 to be a directed graph by adding a source node s and a sink node t to G_0 . Also, add $\text{outdeg}_{G_0}(u) - D$ edges from s to every node with degree at least $D + 1$ and add $D - \text{outdeg}_{G_0}(u)$ edges from every node with degree at most $D - 1$ to t . Now we will do multiple augmentations on G'_0 to obtain $G'_1 \dots G'_l$. Define G_i to be $G'_i \setminus \{s, t\}$. First note that $\text{dist}_{G'_0}(s, t) \geq 3$. In step i , we will find a maximal set of edge-disjoint paths of length $3 + i$ from s to t in G'_i and augment along them to obtain G'_{i+1} . The standard blocking-flow type argument shows that the distance from s to t increases after the augmentation.

LEMMA 4.3. $\text{dist}_{G'_i}(s, t) \geq 3 + i$ for $0 \leq i \leq l$.

Proof. We will show by induction that $\text{dist}_{G'_i}(s, t) \geq 3 + i$. When $i = 0$, it is true that $\text{dist}_{G'_0}(s, t) \geq 3$. Suppose that it is true that $\text{dist}_{G'_i}(s, t) \geq 3 + i$. Consider G'_{i+1} . Let \mathcal{P} be a maximal set of paths of length $3 + i$ from s to t in G'_i . If \mathcal{P} is non-empty, then all paths in it must have length $3 + i$ and so they are the shortest paths.

First we show that $\text{dist}_{G'_{i+1}}(x, t) \geq \text{dist}_{G'_i}(x, t)$ for every x by induction on $\text{dist}_{G'_i}(x, t)$. For a node x with $\text{dist}_{G'_i}(x, t) = 1$, it is obviously true that $\text{dist}_{G'_{i+1}}(x, t) \geq 1$ since $x \neq t$. Suppose that it is true that $\text{dist}_{G'_{i+1}}(x, t) \geq \text{dist}_{G'_i}(x, t)$ for all $\text{dist}_{G'_i}(x, t) < k$. For a node x with $\text{dist}_{G'_i}(x, t) = k$, suppose that there is a path P of length at most $k - 1$ from x to t in G'_{i+1} . Then it must be the case that P intersects with some path in \mathcal{P} . Let uv be the first edge in P that has a non-empty intersection with a path (say, Q) in \mathcal{P} . Without loss of generality, assume that $\text{dist}_{G'_i}(u, t) = \text{dist}_{G'_i}(v, t) + 1$. We have:

$$(4.1) \quad \text{dist}_Q(s, u) + \text{dist}_Q(v, t) = 3 + (i - 1)$$

$$(4.2) \quad \text{dist}_P(x, v) + \text{dist}_P(u, t) = k - 2$$

If $\text{dist}_P(x, v) + \text{dist}_Q(v, t) \leq k - 1$, then it implies that $\text{dist}_{G'_i}(x, t) \leq k - 1$ and a contradiction occurs. Otherwise, $\text{dist}_P(x, v) + \text{dist}_Q(v, t) \geq k - 2$, we have

$$\begin{aligned} \text{dist}_Q(s, u) + \text{dist}_Q(u, t) &= \text{dist}_Q(s, u) + \text{dist}_{G'_i}(u, t) \\ &\leq \text{dist}_Q(s, u) + \text{dist}_{G'_{i+1}}(u, t) \\ &\quad \text{by induction hypothesis} \\ &\leq \text{dist}_Q(s, u) + \text{dist}_P(u, t) \\ &\leq 3 + (i - 1). \end{aligned}$$

by (4.1) and (4.2)

This contradicts with that $\text{dist}_Q(s, t) = 3 + i$. Therefore, we have $\text{dist}_{G'_{i+1}}(x, t) \geq \text{dist}_{G'_i}(x, t)$ for all x . Now suppose to the contrary that a path P' from s to t with length at most $3 + i$ exists in G'_{i+1} . If P' does not intersect with any paths in \mathcal{P} , then it must have length d and so it must have been included in \mathcal{P} . Otherwise, P' must intersect with some edge of the paths in \mathcal{P} . Suppose that $u'v'$ is the first edge P' has a non-empty intersection with and $u'v' \in Q' \in \mathcal{P}$. Without loss of generality, assume that $\text{dist}_{G'_i}(u', t) = \text{dist}_{G'_{i+1}}(v', t) + 1$. We have:

$$(4.3) \quad \text{dist}_{Q'}(s, u') + \text{dist}_{Q'}(v', t) = 3 + (i - 1)$$

$$(4.4) \quad \text{dist}_{P'}(s, v') + \text{dist}_{P'}(u', t) \leq 3 + i$$

If $\text{dist}_{P'}(s, v') + \text{dist}_{Q'}(v', t) \leq 3 + (i - 1)$, then $\text{dist}_{G'_i}(s, t) \leq 3 + (i - 1)$, a contradiction occurs. Oth-

erwise, we have $\text{dist}_{P'}(s, v') + \text{dist}_{Q'}(v', t) \geq 3 + i$ and

$$\begin{aligned} & \text{dist}_{Q'}(s, u') + \text{dist}_{Q'}(u', t) \\ &= \text{dist}_{Q'}(s, u') + \text{dist}_{G'_i}(u', t) \\ &\leq \text{dist}_{Q'}(s, u') + \text{dist}_{G'_{i+1}}(u', t) \\ &\leq \text{dist}_{Q'}(s, u') + \text{dist}_{P'}(u', t) \\ &\leq 3 + (i - 1) \end{aligned} \quad \text{by (4.3) and (4.4)}$$

This contradicts with that $\text{dist}_{Q'}(s, t) = 3 + i$. Therefore, we must have $\text{dist}_{G'_{i+1}}(s, t) \geq 3 + (i + 1)$

LEMMA 4.4. *Let S_i denote the set of node with out-degree at least $D + 1$ in G_i and T_i denote the set of node with out-degree at most $D - 1$ in G_i . We have $S_{i+1} \subseteq S_i$ and $T_{i+1} \subseteq T_i$.*

Proof. Let \mathcal{P} be the set of shortest path from s to t in G'_i . Each $P \in \mathcal{P}$ must contain exactly one node in S_i as the second node, since P is a shortest path. Similarly, P must contain exactly one node in T_i as the second to the last node. Therefore, augmenting along P can only decrease the out-degree of the node in S_i by 1 and increase the out-degree of T_i by 1 in G_i . The out-degrees of other nodes remain the same. Therefore, it is impossible to create new nodes with out-degree at least $D + 1$ or new nodes with out-degree at most $D - 1$.

LEMMA 4.5. *No augmenting path of length at most l exists in G_l .*

Proof. By Lemma 4.3, we have $\text{dist}_{G'_l}(s, t) \geq l + 3$. Suppose that P is an augmenting path with length at most l in G_l . Let x and y be the starting node and the ending node of P . Since the out-degree of x in G_l is at least $D + 1$, by Lemma 4.4, the out-degree of x in G_0 is also at least $D + 1$. This implies $\text{outdeg}_{G_0}(x) - D$ edges have been added from s to x . Since the out-degree of x is the same in G'_0, \dots, G'_l and the out-degree of x in G_l is at least $D + 1$, it must be the case that there is at least one edge going from s to x in G'_l . Similarly, it must be the case that there is at least one edge going from y to t in G'_l . This implies P can be extended to a direct path sPt of length $l + 2$ in G'_l , which contradicts with the fact $\text{dist}_{G'_l}(s, t) \geq l + 3$.

Therefore, no augmenting path of length at most l exists in G_l . However, by setting $l = \Omega((\log n)/\epsilon)$, this contradicts with the following lemma.

LEMMA 4.6. *Let G be a directed graph. Let $D = \lceil (1 + \epsilon)a \rceil$, where a is the arboricity. Suppose that the out-degree of u is at least $D + 1$, then an augmenting path from u of length $O((\log n)/\epsilon)$ exists.*

Proof. Let B_i be the set of all nodes reachable from u by directed paths of length i . We show by induction

that $|B_i| \geq (1 + \epsilon)^{i-1}$, unless B_i includes a node of out-degree less than t in which case, we have found an augmenting path of length i . The base case $i = 0$ is trivial as $B_0 = \{u\}$. For the inductive step, suppose that all nodes in B_i have out-degree at least $D \geq (1 + \epsilon)a$. Then, B_i is incident on at least $|B_i| \cdot D$ outgoing edges. Since all these edges have both their endpoints in B_{i+1} , and as B_{i+1} has at most $a|B_{i+1}|$ edges by definition of arboricity, we get that $|B_{i+1}| \geq \frac{D}{a}|B_i| \geq (1 + \epsilon) \cdot |B_i| \geq (1 + \epsilon) \cdot (1 + \epsilon)^{i-1} = (1 + \epsilon)^i$. Now, since this growth cannot continue beyond $h = \log_{1+\epsilon} n = O(\frac{\log n}{\epsilon})$ hops, as that would exhaust the graph, we get that there must be a node of out-degree less than D within h hops, i.e., an augmenting path of length at most $h = O(\frac{\log n}{\epsilon})$.

LEMMA 4.7. *Suppose the current graph is G_i . Then, G_{i+1} can be computed in $O(i^2 \log n)$ rounds.*

Proof. Recall that G'_{i+1} is obtained by augmenting along a maximal set of edge-disjoint paths from s to t of length $2 + i$ in G'_i . We consider the supergraph \mathcal{G} , where each node denotes a path from s to t in G'_i of length $2 + i$. An edge is added between two nodes, if the corresponding paths intersects at some edges. Then, a maximal independent set (MIS) in \mathcal{G} corresponds to a maximal set of edge-disjoint paths from s to t of length i in G'_i .

We will show how to simulate the computation of the MIS in \mathcal{G} when the underlying network is G_i . Let S_i denote the set of nodes with out-degree at least $D + 1$ in G_i . Each node $x \in S_i$ is responsible for the paths from s to t of length $2 + i$ whose second node is x (the first node is s). Therefore, each node in \mathcal{G} is taken care by some node $x \in S_i$. One round of communication between an edge in \mathcal{G} can be simulated in $O(i)$ rounds in G_i , since the length of the paths is $2 + i$. Since the number of paths of length $2 + i$ is at most n^{2+i} , Luby's MIS algorithm takes $O(\log n^{2+i}) = O(i \log n)$ rounds. It takes $O(i)$ rounds to simulate a round in \mathcal{G} . Therefore, the running time is $O(i^2 \log n)$.

The total running time is $\sum_{i=1}^l O(i^2 \log n) = O(\log^4 n / \epsilon^3)$.

Turning Low Out-Degree Orientations to Forest Decomposition:

Above, we explained a method for obtaining an orientation with out-degree $a(1 + \epsilon)$. This is immediately a decomposition of the edges into $a(1 + \epsilon)$ pseudo-forests. Recall that a pseudo-forest is a graph where each connected component is a pseudo-tree, that is a tree with the exception of having at most one more edge, which creates one cycle. If we let each node number its at most $a(1 + \epsilon)$ out-going edges by numbers $1, 2, \dots, a(1 + \epsilon)$ uniquely, then the edges of each number form a pseudo-forest, as they form a graph with per-node out-degree at most 1. For practically all

the distributed applications that we are aware of, graphs with out-degree 1 are as good as trees. However, from an aesthetic viewpoint, having a decomposition into actual forests would be much nicer. We next explain a method for decomposing into forests in high-arboricity graphs. Indeed, the decomposition will have an extra property which might be quite useful in the distributed context:

LEMMA 4.8. *There is a randomized $O(\log n)$ -round algorithm that for graphs of arboricity $a = \Omega(\log n/\varepsilon^2)$, transforms orientations with out-degree at most $a(1+\varepsilon)$ to an edge-decomposition into $a(1+8\varepsilon)$ forests, with high probability. Moreover, except for $O(\varepsilon)$ fractions of the forests, each connected component in the other forests is merely a star-graph, that is, a tree with diameter 2.*

Proof. We first randomly build $a(1+\varepsilon)$ primary forests such that each node v has at most $3\varepsilon a$ of its outgoing edges not put in these forests. We then put these leftover edges into $7\varepsilon a$ additional forests, hence getting a decomposition into $a(1+8\varepsilon)$ forests. The connected components of the primary forests will be stars.

First, notice that some nodes might have out-degree less than $a(1+\varepsilon)$. For simplicity, we first remove this imperfection, by giving each node v with out-degree d_v exactly $a(1+\varepsilon)d_v$ outgoing edges that go to dummy nodes. These dummy nodes are just simulated by v and no other actual node will need to interact with them. Now, each real node has out-degree $a(1+\varepsilon)$. Once we have the forest decomposition, we will drop these edges going to the dummy nodes.

Call each real node v active in each of the primary $a(1+\varepsilon)$ forests with probability $q = \frac{1-\varepsilon}{1+\varepsilon}$ independently. Given that $a = \Omega(\log n/\varepsilon^2)$, by a Chernoff bound, the number of active forests in each node is a number in $[a(1-2\varepsilon), a]$, with high probability.

Now, we find the outgoing edges of each node in each of its active forests. For each forest i and each node v , we will find an outgoing edge of v that goes to a neighbor u who is not active in forest i (or a dummy neighbor u). We claim that node v can find a collection of such edges, one for each of its active forests, with high probability, given that $a = \Omega(\log n)$. The argument is as follows: we will find these edges for the active forests of v greedily, and one by one. In each step, when looking for an edge for forest i , there are at least $a\varepsilon$ outgoing edges of v remaining. This is because v has $a(1+\varepsilon)$ outgoing edges and we only find edges for at most a active forests. Now each of the endpoints of these remaining outgoing edges is active in the current forest i with probability $q = \frac{1-\varepsilon}{1+\varepsilon}$. Hence, with probability at least $1 - (\frac{1-\varepsilon}{1+\varepsilon})^{a\varepsilon} \geq 1 - 1/\text{poly}(n)$, at least one of these outgoing neighbors is not active in forest i . We assign the outgoing edge to that neighbor to forest i . Also, notice that this process can be done in just 1 round,

by each node informing its incoming neighbors of its active forests, and then each node v picking its outgoing edges for its active forests. At the end, we have created $a(1+\varepsilon)$ forests, where indeed each connected component is a star.

What remains for each node is at most $3\varepsilon a$ outgoing edges. These can be put in $7\varepsilon a$ additional forests in $O(\log n)$ rounds, by a method of Barenboim and Elkin [BE10]. Hence, we have our desired decomposition into $a(1+8\varepsilon)$ forests.

References

- [Bar15] Leonid Barenboim. Deterministic $(\Delta + 1)$ -coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, pages 345–354, New York, NY, USA, 2015. ACM.
- [BE10] Leonid Barenboim and Michael Elkin. Sublogarithmic distributed mis algorithm for sparse graphs using nash-williams decomposition. *Distributed Computing*, 22(5-6):363–379, 2010.
- [BE11] Leonid Barenboim and Michael Elkin. Distributed deterministic edge coloring using bounded neighborhood independence. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, pages 129–138, 2011.
- [BE13] Leonid Barenboim and Michael Elkin. Distributed graph coloring: Fundamentals and recent developments. *Synthesis Lectures on Distributed Computing Theory*, 4(1):1–171, 2013.
- [BEP16] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *J. ACM*, 63(3):20:1–20:45, 2016.
- [BF81] József Beck and Tibor Fiala. integer-making theorems. *Discrete Applied Mathematics*, 3(1):1–8, 1981.
- [BFH⁺16] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed lovász local lemma. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 479–488, 2016.
- [CHK01] Andrzej Czygrinow, M Hańćkowiak, and M Karoński. Distributed $O(\Delta \log n)$ -edge-coloring algorithm. In *European Symposium on Algorithms*, pages 345–355. Springer, 2001.
- [CKP16] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the local model. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, page to appear, 2016.
- [DGP98] Devdatt Dubhashi, David A Grable, and Alessandro Panconesi. Near-optimal, distributed edge colouring via the nibble method. *Theoretical Computer Science*, 203(2):225–251, 1998.
- [Din] EA Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation, soviet math. doll. 11 (5), 1277-1280,(1970). *English translation by RF. Rinehart.*

- [EK72] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.
- [EPS15] Michael Elkin, Seth Pettie, and Hsin-Hao Su. $(2\Delta - 1)$ -edge-coloring is much easier than maximal matching in the distributed setting. In *Symp. on Discrete Algorithms (SODA)*, pages 355–370. SIAM, 2015.
- [FHK16] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, pages to appear, arXiv:1511.01287, 2016.
- [HKP98] Michal Hanckowiak, Michal Karonski, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. In *Pro. of ACM-SIAM Symp. on Disc. Alg. (SODA)*, pages 219–225, 1998.
- [HKP01] Michal Hanckowiak, Michal Karonski, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. *SIAM Journal on Discrete Mathematics*, 15(1):41–57, 2001.
- [IS86] Amos Israeli and Yossi Shiloach. An improved parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):57–60, 1986.
- [KS87] Howard J Karloff and David B Shmoys. Efficient parallel algorithms for edge coloring problems. *Journal of Algorithms*, 8(1):39–52, 1987.
- [Lin92] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.
- [LPSP15] Z. Lotker, B. Patt-Shamir, and S. Pettie. Improved distributed approximate matching. *J. ACM*, 62, 2015. Article 38.
- [Lub86] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.
- [PR01] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed computing*, 14(2):97–100, 2001.
- [PS97] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.
- [Viz64] Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz*, 3(7):25–30, 1964.