Distributed Arboricity-Dependent Graph Coloring via All-to-All Communication

Mohsen Ghaffari
ETH Zurich, Switzerland
ghaffari@inf.ethz.ch

Ali Sayyadi
Sharif University of Technology, Iran
ali.sayyadi.98@gmail.com

Abstract

We present a constant-time randomized distributed algorithms in the congested clique model that computes an $O(\alpha)$-vertex-coloring, with high probability. Here, α denotes the arboricity of the graph, which is, roughly speaking, the edge-density of the densest subgraph. Congested clique is a well-studied model of synchronous message passing for distributed computing with all-to-all communication: per round each node can send one $O(\log n)$-bit message algorithm to each other node. Our $O(1)$-round algorithm settles the randomized round complexity of the $O(\alpha)$-coloring problem. We also explain that a similar method can provide a constant-time randomized algorithm for decomposing the graph into $O(\alpha)$ edge-disjoint forests, so long as $\alpha \leq n^{1-o(1)}$.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Distributed Computing, Message Passing Algorithms, Graph Coloring, Arboricity, Congested Clique Model, Randomized Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.142

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Funding The first author acknowledges the support of the Swiss National Foundation, under project number 200021_184735. The second author is grateful for the support of the Student Summer Research Fellowship (SSRF) program at the Computer Science department of ETH Zurich; this project was initiated during an undergraduate internship in the summer of 2018.

1 Introduction and Related Work

This paper settles the randomized complexity of the graph coloring problem with a number of colors linear in its arboricity, in the congested clique model of distributed computing: we show that constant time suffices. Before formally stating our result, let us start by reviewing the model and the problem statement, as well as state of the art.

1.1 Background: Model, Problem Statement, and State of the Art

The Congested Clique Model. We work with the congested clique model of distributed computing, which was introduced by Lotker et al. [18] and has received extensive attention over the past seven years[1]. There are n processors in the system. We are also given an n-node graph $G = (V, E)$, which has one node corresponding to each processor. This graph

[1] There are many paper, and it is well-beyond the scope of this paper to survey them all. As a prominent example, see the beautiful line of developments for the minimum spanning tree problem [18, 12, 11, 14], which also settled its complexity to be $O(1)$ rounds [14].

can abstract various things. For instance, it might be abstracting the dependencies in a *contention resolution* setting, as follows: processors that are adjacent in the graph are dependent and cannot perform their tasks simultaneously, e.g., they should not transmit their wireless messages simultaneously or they should not read/write to the same memory location simultaneously, or many other similar settings of accessing shared resources. We assume that the graph G is known to the processors in a distributed fashion: each processor/node knows its own neighboring processors/nodes in G. In the congested clique model, we assume that processors can communicate in synchronous rounds, in an all-to-all fashion. Per round, each processor can send one $O(\log n)$-bit message to each other processor. At the end, each processor should know its own part of the output, e.g., the color of its own node.

A discussion about the model, in contrast with other distributed models. Two other well-studied, and in fact more classic, models of distributed computing are CONGEST and LOCAL [23]. In these models, the communication network among the processors is the same as the graph G for which they want to solve the graph problem. In the CONGEST model, per round, each node can send one $O(\log n)$ bit message to each of its neighbors (in the graph, which abstract both the problem and the communication network). The LOCAL model is the variant where there is no upper bound on the message sizes.

These models CONGEST and LOCAL are perfectly suitable for settings such as computer networks, where we want to solve a graph problem about the communication network. This was in fact the original motive of these models. Moreover, there has been a host of beautiful and powerful algorithmic and impossibility results developed in the context of these two models. For some other applications, these models may be less suitable. In particular, when the graph problem is a logical one – as in the abstraction used in the above above example to capture contentions among processor – such a coincidence of communication graph and problem graph may look strange. This is perhaps a partial explanation for the surge of recent interest in distributed algorithms in the congested clique model. Some other reasons are more theoretical and have to do with setting the limitation of locality aside, see the introduction of [11] for a more extensive discussion of this aspect.

The congested clique model is more permissive than CONGEST and we can easily simulate CONGEST-model algorithms in congested clique. But that would not be enough for us, as there is no constant-algorithm for our problem in CONGEST. In fact, in the CONGEST and LOCAL models, the problem has an $\Omega(\log n)$ lower bound [17]. We build on the techniques developed in the CONGEST and LOCAL model, but we also present and use several other algorithmic ideas which allow us to go much further and solve the problem in constant time.

Problem Statement – Arboricity-Dependent Graph Coloring. The graph coloring problem is a well-studied problem with a wide range of applications. It asks for coloring the vertices with few colors in a way that no two adjacent nodes have the same color. For instance, in the context of the contention resolution example mentioned above, this would allow the processors to perform their tasks in a few time slots, without any two dependent processors working concurrently.

Our objective is to compute a coloring whose number of colors is $O(\alpha)$, where α denotes the *arboricity* of the graph. We recall that arboricity is a measure of sparsity of the graph. For a graph $G = (V, E)$, its arboricity is defined as the maximum edge-density $\lceil \frac{E(V')}{|V'|-1} \rceil$ among all induced subgraphs on vertices $V' \subseteq V$ with $|V'| > 1$. Alternatively, by a beautiful result of Nash-Williams[19], an equivalent formulation can be given as the minimum number of edge-disjoint forests into which we can decompose the edges of G. We assume that α is given as an input parameter and the input graph has arboricity at most α.

Any graph with arboricity α admits a coloring with 2α colors and this bound is sharp. For the former, note that we can arrange vertices as a_1 to a_n such that each v_i has at most $2\alpha - 1$ neighbors v_j with a higher index $j > i$. For the latter, a graph consisting of disjoint cliques, each having 2α vertices, gives the sharpness example.

Following [3, 5], we set out target to be obtaining a coloring close to this existentially optimal bound; ideally we would want a coloring with $(2+\epsilon)\alpha$ colors for a small constant $\epsilon > 0$, but we relax it further to $O(\alpha)$ colors, for this paper. Note that $\alpha \leq \Delta$, where Δ denotes the maximum degree of the graph. Hence, coloring with $O(\alpha)$ colors also directly gives a coloring with $O(\Delta)$ colors. However, α can be much smaller than Δ; take for instance a star graph where $\Delta = n - 1$ and $\alpha = 1$. For many graph problems in practice, α is indeed much smaller than Δ: even though they might have a few nodes of high degree, typically they do not contain very dense subgraph, at least not as dense as average degree of Δ.

State-of-the-Art – LOCAL and CONGEST. As it will become clear soon, there is a large body of work on distributed algorithms for graph coloring, and other related problems, and it is well beyond the scope of this paper to review them all. For a survey of the results prior to 2013 on algorithms in the LOCAL and CONGEST models, we refer the reader to the *Distributed Graph Coloring* book by Barenboim and Elkin [3]. We will just mention the best known result in randomized and deterministic settings:

In the LOCAL model, the best known randomized upper bound for $(\Delta + 1)$-coloring problem is $2^{O(\sqrt{\log \log n})}$ [7] and the best known deterministic upper bound for $(\Delta + 1)$-coloring is $2^{O(\sqrt{\log n})}$[20]. Only if we relax the number of colors to $\Delta^{1+\epsilon}$ for some constant $\epsilon > 0$, a polylogarithmic-time – and particularly $O(\log \Delta \log n)$-round – algorithm is known, thanks a breakthrough of Barenboim and Elkin [2]. For the CONGEST model, the best known randomized algorithm for $(\Delta+1)$-coloring runs in $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$ rounds [9]. If we parameterize the algorithm's complexity by Δ, the best known algorithm runs in $\tilde{O}(\sqrt{\Delta}) + O(\log^* n)$ rounds [4].

Regarding coloring dependent on α, Linial [17] showed that any coloring with $O(\alpha)$ colors, or even $poly(\alpha)$ colors, needs at least $\Omega(\log n)$ rounds, even in the LOCAL model. Barenboim and Elkin[1] gave a deterministic algorithm that gives a $\Theta(\alpha^2)$ coloring in $O(\log n)$ rounds, and an $O(\alpha^{1+\epsilon})$ coloring in $O(\log \alpha \log n)$ rounds. Ghaffari and Lymouri [10] gave a randomized algorithm that computes a $(2+\epsilon)\alpha$ colors, for constant $\epsilon > 0$, if $\alpha = \tilde{\Omega}(\log n)$, an $O(\alpha \log \alpha)$-coloring in $O(\log n)$ rounds, and an $O(\alpha)$-coloring running in $\tilde{O}(\log n)$ rounds.

State-of-the-Art – CONGESTED-CLIQUE. There has been a sequence of improvements, for coloring with a number of colors dependent on the maximum degree Δ, as we overview next: Hegeman and Pemmaraju [12] gave algorithms for $O(\Delta)$-coloring in the CONGESTED-CLIQUE model, which run in $O(1)$ rounds if $\Delta \geq \Theta(\log^4 n)$ and in $O(\log \log n)$ rounds otherwise. For $(\Delta+1)$ coloring problem, recently Parter [21] designed the first sublogarithmic-time $(\Delta+1)$ coloring algorithm for CONGESTED-CLIQUE, which runs in $O(\log \log \Delta \log^* \Delta)$ rounds. Parter and Su [22] improved the bound to $O(\log^* \Delta)$ round. Finally, the round complexity of $(\Delta+1)$-coloring was settled very recently $O(1)$, by a recent work of Chang, Fischer, Ghaffari, Uitto, and Zheng [6].

For coloring with the number of colors dependent on the arboricity α, we are aware only one prior work in the CONGESTED-CLIQUE: Barenboim and Khazanov[5] gave a deterministic $O(\alpha)$-coloring which runs in α^ϵ rounds, for a constant $\epsilon > 0$, and an $O(\alpha^{2+\epsilon})$-coloring, which runs in $O(\log^* n)$ rounds. For randomized algorithms, we are not aware of any faster algorithm than these deterministic ones, or the randomized algorithms of the CONGEST model (which need at least $\Omega(\log n)$ rounds).

1.2 Our Results

Coloring. We settle the randomized complexity of $O(\alpha)$-coloring in the congested clique model, by showing that constant rounds suffice.

▶ **Theorem 1.** *There is a randomized distributed algorithm that computes an $O(\alpha)$-coloring in constant rounds of the congested clique model, with high probability.*

Forest Decomposition. Our techniques also allow us to compute a decomposition of the graph into $O(\alpha)$ edge-disjoint forests in constant time, so long as $\alpha \leq n^{1-o(1)}$. The best known previous algorithm for this problem was by Barenboim and Khazanov [5] and it uses $O(\log \alpha)$ rounds. The statement of our result appears in the following theorem.

▶ **Theorem 2.** *There is a randomized distributed algorithm that computes a decomposition of the graph into $O(\alpha)$ edge-disjoint forests in constant rounds of the congested clique model, whenever $\alpha \leq n^{1-O(\frac{1}{\sqrt{\log n}})}$, with high probability.*

1.3 Technical Overview

On a high-level, our result makes use of a number of technical ingredients, and puts them together – along with several smaller ideas – in a way that results in the claimed constant-round algorithm for $O(\alpha)$ coloring. We start with a brief list of these technical ingredients:

1. The iterative peeling algorithm of Barenboim and Elkin [1], which partitions vertices into successive layers, along with an orientation from lower layers to higher layers, such that each node has out-degree at most $(2+\epsilon)\alpha$.
2. A topology-gathering procedure which allows us to exponentially speed up some LOCAL model algorithms, on sparse graphs and up to a small locality bound [16, 8].
3. An opportunistic topology-gathering idea which allows us to speed up some LOCAL model algorithms to just $O(1)$ rounds of the CONGESTED-CLIQUE model. Variants of this idea have been used by [14, 6].
4. Some ad-hoc but simple random sampling ideas which allow us to use a much sparser subgraph to perform an approximate peeling about the original graph.
5. A simple random partition idea which allows us to break the case of graphs with large arboricity to several graphs with smaller arboricity, or sometimes graphs that are almost small aboricity, in the sense that they have some number of extra edges.
6. A simple randomized coloring algorithm where per round each node chooses a random color from its palette and keeps it, if none of the neighbors (or sometimes neighbors in certain subsets) chose that color. Variants of this simple idea are standard in distributed coloring algorithms, perhaps the first appearance was in the work of [13].
7. A routing method of Lenzen [15], which by now has become a staple in CONGESTED-CLIQUE algorithms. It is worth noting that the topology gathering ingredients mentioned in items 2 and 3 above also make use of this routing method.

From a bird's eye viewpoint, our algorithm works as follows: we use the random partitioning of item (5) to break the graph into several graphs of much lower arboricity, plus some additional edges. We use the oppurtunistic topology-gathering of item (3) to learn some sufficent local topology in each of these subgraphs, so that we can locally simulate a non-constant number of iterations of the peeling algorithm of item (1) and then apply a suitable variant of the randomized coloring algorithm of item (6). This will be only a partial coloring, but the number of nodes that remain uncolored will be considerably smaller. Then, we can apply the sampling idea of item (5) and gather the sampled sparse graph, using the routing method of (7), so that we can solve the problem in a centralized fashion, and make even more progress in reducing the number of nodes that remain uncolored. This number at the end will be so small that we can gather all the induced edges by the remaining nodes, using the routing of item (7), and solve the problem in a centralized way. There are several subtleties in this rough outline; we leave the details to the technical sections.

Roadmap. We start with a warm up in Section 2 where we present an $O(\log\log\log n)$ round algorithm for forest-decomposition. Of course, that $O(\log\log\log n)$ complexity is much higher than our claimed bound of $O(1)$ round complexity, but we can introduce a number of the key algorithmic ideas that we use in the context of this simpler but slower algorithm. Then, in Section 3, we present our $O(1)$-round algorithm for $O(\alpha)$-coloring.

2 Warm up: Forest Decomposition in $O(\log\log\log n)$ rounds

In this section, as a warm up, we describe an algorithm for $(2+O(\epsilon))\alpha$ forest decomposition for graphs with $\alpha \le n^{1-O(\frac{(\log\log n)^2}{\log n})}$, in $O(\log\log\log n)$ rounds of the congested clique model, for any desirably small constant $\epsilon > 0$. This allows us to introduce some of the basics of our end-results, in a simpler setting.

Studying $\alpha = O(\log n)$ Suffices. Before starting our forest-decomposition algorithm, we comment that almost without loss of generality, we can assume that $\alpha = O(\log n/\epsilon^2)$. The reason is as follows: suppose that $\alpha = \Omega(\log n)$. Partition the edges into $k = \epsilon^2\alpha/(10\log n)$ parts $E_1, \ldots, E_k$, randomly. Then, as we show in a simple lemma below, we can see that each group is a subgraph with arboricity $\alpha' = (1+\epsilon)\alpha/k \le \Theta(\log n)$, with high probability. We describe an algorithm below, which when applied to each of these subgraphs, it decomposes each such subgraph into $(2+\epsilon)\alpha'$ edge-disjoint forests. Thus, overall, the graph is decomposed into $k\cdot(2+\epsilon)\alpha' = (2+\epsilon)\alpha$ forests. We will remark at the end of this section why we have sufficient communication capacity to perform this algorithm on all the subgraphs in parallel.

▶ **Lemma 3.** *If we randomly partition edges of a graph with arboricity $\alpha = \Omega(\log n/\epsilon^2)$ into $k = \epsilon^2\alpha/(10\log n)$ parts $E_1, \ldots, E_k$, then each part is a subgraph with arboricity at most $\alpha' = (1+\epsilon)\alpha/k$, with high probability.*

Proof. We focus on one part E_i. Consider an arbitrary subset of vertices $V' \le V$, where $|V'| \ge 2$. The expected number of edges of V' that would be put in E_i at most $E(V')/k \le (|V|'-1)\alpha/k$ edges. By a basic Chernoff bound, the probability that there are more than $(|V|'-1)\alpha/k(1+\epsilon)$ edges is at most $2exp(-(|V|'-1)\alpha/(3k)) \le exp(-(|V'|\log n)$. Now, we can use a union bound over all possible subset of size $|V'|$, for which there are at most $\binom{n}{|V'|} \ll 2^{|V'|\log n}$ possibilities, and conclude that the edge density of each of them is at most α', with probability $1-exp(-\Theta(|V'|\log n)) \ge 1-1/poly(n)$. Finally, a union bound over all the n possibilities for the value $|V'|$ and at most $\alpha \le n$ possibilities for index i, in part E_i, concludes the proof. ◀

We will next describe our forest decomposition algorithm for graphs with arboricity $\alpha = O(\log n)$. The algorithm we present in this section will provide a stronger structure, called *H-partition* by Barenboim and Elkin [5], defined as follows: The set V of nodes will be partitioned into $L = O(\log n)$ disjoint sets $V_1, V_2, \ldots, V_L$, which we will call *layers* of the H-partition, such that for each $i \in \{1, 2, \ldots, L\}$, each node $v \in V_i$ has at most $(2+\epsilon)\alpha$ neighbors in $\cup_{j=i}^{L} V_j$.

Given such an H-partition, one can easily obtain a forest-decomposition, i.e., decompose the graph into edge-disjoint forests $F_1, \ldots F_K$, where $K = (2+\epsilon)\alpha$, as follows: (A) orient each edge $\{v, u\}$ between two disinct sets V_i and V_j where $i \neq j$ from the lower index one to the higher indexed one, i.e., from V_i to V_j iff $i < j$. (B) orient each edge $\{v, u\}$ where both endpoints are in the same layer V_i from the lower ID node to the higher ID node. Then, clearly each node has out-degree at most K. Each node v puts each of its outgoing edges in a distinct one of F_1 to F_k. Hence, each F_i is a directed acyclic graph with out-degree at most one. Thus, if we ignore the directions, F_i is a forest. See section 5.1 of the book by Barenboim and Elkin[3], for an elaborate proof.

Our H-partition Algorithm. We build our $O(\log\log\log n)$-round H-partition (and therefore forest decomposition) algorithm in two steps. First, in Section 2.1, we present a slower algorithm that computes H-partition in $O(\log\log n)$ rounds, with high probability. Then, in Section 2.2, we explain a topology-gathering idea (stated in Lemma 9), which allows us to improve this round complexity to $O(\log\log\log n)$, thus proving the following theorem.

▶ **Theorem 4.** *There is a randomized distributed algorithm that for any n-node graph with arboricity $\alpha = O(\log n)$ computes H-partition with parameter $(2+\epsilon)\alpha$ in $O(\log\log\log n)$ rounds with high probability.*

2.1 H-partition in $O(\log\log n)$

We now present our slow $O(\log\log n)$ round algorithm for H-partition. This algorithm relies on two key ingredients: (I) an iterative peeling algorithm of Barenboim and Elkin [5], which on its own would need $R = \Theta(\log n)$ rounds to build the H-partition. (II) A random edge sampling idea which will allow us to compress the last $R - O(\log\log n) = O(\log n)$ rounds of the peeling algorithm into just $O(1)$ rounds of the congested clique.

High-Level Outline. First by running $O(\log\log n)$ iterations of the peeling algorithm of Barenboim and Elkin [5], we reduce the number of nodes to $O(\frac{n}{\log^2 n})$. Then, we use an randomized edge sampling process which allows us to generate a graph with just $O(n)$ edges, using which we can perform the rest of the peeling rounds. Since this graph has just $O(n)$ edges, we can deliver it to a single node, using Lenzen's routing method [15], and thus perform all these remaining peeling rounds in a centralized fashion there, without having to use more rounds of communication in the congested clique model.

Part 1 – Slow Iterative Peeling. First, we perform $O(\log\log n/\epsilon)$ iterations. In iteration i, we remove all nodes whose degree in the remaining graph is at most $(2+\epsilon)\alpha$ and we put them in layer i. Then, we proceed to the next iteration. A more formal description can be found in Algorithm 1.

▶ **Lemma 5.** *After $2\log\log n \cdot \frac{2}{\epsilon}$ iterations of peeling, at most $\frac{n}{\log^2 n}$ nodes remain.*

Proof. After each iteration, the number of remaining nodes reduces by a factor of $\frac{2}{2+\epsilon}$. This is because any remaining graph has arboricity at most α, which means it has average degree at most 2α, and thus only $\frac{2}{2+\epsilon}$ fraction of its nodes can have degree higher than $(2+\epsilon)\alpha$ and remain for the next iteration. Hence, after $2\log\log n \cdot \frac{2}{\epsilon}$ iterations, the number of remaining nodes is at most $(\frac{2}{2+\epsilon})^{\lceil 2\log\log n \cdot \frac{2}{\epsilon}\rceil} n \leq \frac{n}{\log^2 n}$. ◀

Algorithm 1 Computing a partial H-partition for a graph G with arboricity α.

```
1:  procedure H-PARTITION(($\alpha, \epsilon$))
2:      An algorithm for each vertex v of G:
3:      $i = 1$
4:      while $i \leq \lceil \frac{4}{\epsilon} \log\log n \rceil$ do
5:          if v is active and has at most $(2+\epsilon)\alpha$ active neighbors then
6:              make v inactive
7:              add v to $H_i$
8:              send the message "inactive" and "v joined $H_i$" to all the neighbors
9:          for each received "inactive" message do
10:             mark the sender neighbor as inactive
11:         $i = i + 1$
```

Part 2 – Speeding up Peeling via Random Edge Sampling. Next we focus on the remaining nodes and we find an H-partition of them in constant rounds of the congested clique. Let N denote the number of remaining nodes, and notice that we know $N \leq \frac{n}{\log^2 n}$. Consider $O(\log n)$ independent sampling process where in each, each edge of the graph induced by the remaining nodes is sampled with probability $p = min(\frac{1000 \log n}{\epsilon^2 a}, 1)$. Let G_i be the graph with sampled edges in i^{th} process. We will show in Lemma 6 that the total number of sampled edges is $O(n)$, with high probability. Thus, using Lenzen's routing method [15], we can collect all sampled edges in one node in $O(1)$ rounds. Then, we devise an iterative processing algorithm to compute H-partition of the remaining nodes, by processing only these sampled edges (instead of working on the base graph). We note that this iterative process will happen in a centralized fashion in the node that holds all the sampled edges, and thus it needs no communication. At the end, we can report to each node of the graph the peeling iteration in which it was removed. The peeling algorithm based on the randomly sampled edges is described in Algorithm 2. We note that for each iteration, this algorithm uses a different randomly sampled subgraph G_i, and these samplings are performed independent of each other. We will show that even though we are working on a randomly sampled subgraph, our peeling process will still produce a correct H-partition, in the following senses: On one hand, in Lemma 7, we show that in each iteration, all nodes of degree less than $(2 + \frac{\epsilon}{2})\alpha$ in the remaining graph will be removed, with high probability. Hence, we will be done after $O(\log n)$ iterations of peeling. On the other hand, in Lemma 8, we show that any node that gets removed in an iteration has degree at most $(2 + 2\epsilon)\alpha$, with high probability, among the nodes that had remained for that iteration. Hence, each node has at most $(2+2\epsilon)\alpha$ neighbors in its own layer or the future layers. We next present these lemmas and their proofs.

We first show that the number of sampled edges is small enough to allow us to deliver them to one node, via Lenzen's routing.

▶ **Lemma 6.** *The total number of sampled edges is at most $O(n)$, with high probability.*

Proof. The number of remaining nodes is $N \leq \frac{n}{\log^2 n}$. We have $O(\log n)$ independent sampling process, where in each process, each edge is sampled with probability $p = min(\frac{1000 \log n}{\epsilon^2 a}, 1)$. Hence, the expected number of sampled edges is $O(n)$. Moreover, since the samplings are independent (across different edges and different runs of the process), we can apply the Chernoff bound and conclude that with high probability, at most $O(n)$ edges are sampled. ◀

Now, we argue that in each iteration of peeling on the sampled subgraph, we will remove all nodes of relatively small degree in the base graph (among remaining nodes).

Algorithm 2 Computing H-partition for the remaining nodes by processing the sampled subgraphs G_i for $i \in 1, \dots, \Theta(\log n)$.

```
1: procedure Processing Sampled graphs($G_i$)
2:     $i = 1$
3:     while $i \leq \Theta(\log n)$ do
4:         for for each vertex v of $G_i$ do
5:             if v has at most $(2+\epsilon)\alpha p$ neighbors in $G_i$ then
6:                 add v to $H_{i+T}$
                                   ▷ T=$\Theta(\log\log n)$ is the number of iterations in Algorithm 1
7:                 delete v from all $G_k$ for $k \in \{i+1, ..., \Theta(\log n)\}$
8:         $i = i + 1$
```

▶ **Lemma 7.** *In each iteration of working with the sampled subgraphs, all nodes of degree less than $(2 + \frac{\epsilon}{2})\alpha$ in the remaining base graph will be removed, with high probability.*

Proof. Consider iteration i. Let G' be the subgraph of the base graph induced by the nodes that remain in iteration i. Consider a remaining node v and suppose that its degree $d_{G'}(v) \leq (2 + \frac{\epsilon}{2})\alpha$. We argue that such a node will be remove in this iteration, with high probability. Then, in the sample subgraph for process i , each of these edges is sampled with probability p. Thus, the expected number of the sampled edges of v is $(2 + \frac{\epsilon}{2})\alpha p$. Since the sampling for this process is independent of the randomness of the previous processes, and since different edges are sampled independently, we can apply a Chernoff bound and conclude that the number of sampled edges of v is at most $(2+\epsilon)\alpha p$, with high probability. Hence, node v gets removed in this iteration. ◀

As mentioned before, Lemma 7 allows us to argue that after each iteration the number of nodes reduces by a factor of $\frac{2}{2+\frac{\epsilon}{2}}$ with high probability. Thus, after $O(\log n)$ iterations, the number of nodes that did not find their proper partition is below 1, i.e., no node remains. We also need to ensure that the peeling performs the correct thing in that any node that gets removed in some iteration has degree at most $(2 + 2\epsilon)\alpha$ in the base graph, among the remaining nodes.

▶ **Lemma 8.** *In each iteration of working with the sampled subgraphs, any node that gets removed has degree at most $(2 + 2\epsilon)\alpha$ in the base graph among the nodes that remained for that iteration, with high probability.*

Proof. Consider iteration i. Let G' be the subgraph of the base graph induced by the nodes that remain in iteration i. Consider a remaining node v and suppose that its degree $d_{G'}(v) \geq (2+2\epsilon)\alpha$. We argue that such a node will be not be removed in this iteration, with high probability. In the sample subgraph for process i , each of these edges is sampled with probability p. Thus, the expected number of the sampled edges of v is at least $(2 + 2\epsilon)\alpha p$. Since the sampling for this process is independent of the randomness of the previous processes, and since different edges are sampled independently, we can apply a Chernoff bound and conclude that the number of sampled edges of v is strictly exceeds $(2 + \epsilon)\alpha p$, with high probability. Hence, node v does not get removed in this iteration. ◀

To conclude, as argued above Lemma 7 implies that we are done in $O(\log n)$ iterations of peeling, and Lemma 8 shows that each node that gets removed in each iteration of peeling has degree at most $(2 + 2\epsilon)\alpha$ among the nodes that had remained for that iteration. That is, the nodes peeled in different iterations gives a partitioning of the nodes so that each node in a given layer has at most $(2+2\epsilon)\alpha$ neighbors in that layer or the future ones. This gives us an H-partition with out-degree $(2+2\epsilon)\alpha$.

2.2 H-partition in $O(\log\log\log n)$ rounds

In this section, we explain how to speed up the $O(\log\log n)$-round algorithm described in the previous subsection to run in $O(\log\log\log n)$ rounds.

Recall that our previous algorithm had two main part, a slow $O(\log\log n)$ round for iterative peeling, and a fast $O(1)$ round algorithm where all the other peelings happen in constant rounds of the congested clique, by processing a sparse randomly sampled subgraph of the remaining graph. Here, we focus on the first part and improve its complexity to $O(\log\log\log n)$-round, using a topology-gathering idea.

Notice that the first part of the algorithm did not need any all-to-all communication. It was simply iterations of peeling in the graph. Hence, if a node v knew all of the topology within its R hops, it could simulate this peeling process for R rounds. We can learn this R-hop topology much faster than R rounds in the congested clique model, thanks to the all-to-all communication, but subject to some constraint: it is only possible if the topology to be learned is relatively small. In particular, we next discuss a special case of such a topology learning on graphs where the maximum degree is small. Later, we will discuss how to incorporate this maximum degree limitation into our peeling algorithm.

First, we start with a generic topology-gathering lemma, for small-degree graphs. Variants of such a statement have been used in prior work, see [16, 8], for instance.

▶ **Lemma 9.** *Suppose that F is graph with n nodes and maximum degree at most $\log^{100} n$. Then we can make each node $v \in F$ learns its R-hop neighborhood in F for $R = O(\log\log n)$, in $O(\log R) = O(\log\log\log n)$ rounds in congested clique.*

Proof. We use an induction to prove that for any $k \leq O(\log\log\log n)$, in $O(k)$ rounds, we can make the nodes learn the graph $F' = F^{2^k}$, which is formed by connecting each two nodes that are within distance 2^k. After that, each node can directly receive all the edges of all of its F' neighbors and thus know the whole topology within its 2^k-hop neighborhood, using Lenzen's routing[15]. The base case of induction where $k = 1$ is trivial, as that is knowing the graph F itself. Suppose that assume that $O(k)$ rounds have passed and each node already knows its neighbors in the graph F^{2^k}. Now, the number of neighbors in this graph is at most $(\log^{100} n)^{(2^k)} \leq 2^{O((\log\log n)^2)} \ll \sqrt{n}$. Make each node send the names of each of its neighbors in F^{2^k} to each of its neighbors in F^{2^k}. That is at most n messages. Hence, this information can be routing in $O(1)$ round using Lenzen's routing. Then, each node knows its neighbors of neighbors in F^{2^k}, which means it can know it knows all of its neighbors in $F^{2\cdot 2^k} = F^{2^{k+1}}$. ◀

Applying Topology-Gathering to Speed Up Peeling. If we could apply the above topology gathering to our $O(\log\log n)$ round peeling algorithm, we would compress it to $O(\log\log\log n)$ rounds. However, this is not immediately possible. Our original graph may have nodes of very large degree, much larger than $poly(\log n)$, despite its small arboricity $\alpha = O(\log n)$. This means we cannot directly apply the topology gathering idea of the above lemma. However, fortunately, our graph cannot have too many nodes of large degree (exactly because of its small arboricity). This allows us to freeze those few high degree nodes, and still use a peeling process on the rest of the graph to reduce the number of remaining nodes (including the frozen ones) to just $O(\frac{n}{\log^2 n})$.

In particular, freeze each node whose degree is greater than $\log^{100} n$. Notice that since the graph has arboricity $\alpha = O(\log n)$, as we justified this assumption at the beginning of this section and in Lemma 3, the number of frozen nodes is at most $\frac{2n\alpha}{\log^{100}(n)} \leq \frac{2n}{\log^{99} n}$. Now, we apply $O(\log\log n)$ iterations of the peeling algorithm on the remaining nodes, but sped up to $O(\log\log\log n)$ rounds of the congested clique model, using the topology-gathering approach of Lemma 9. The number of nodes that remain from this peeling is $O(n/\log^2 n)$. Hence, even including the at most $\frac{2n}{\log^{99} n}$ frozen nodes, the number of remaining nodes is $O(n/\log^2 n)$. That means, we can now apply the algorithm of the second part, described in the previous subsection, to finish all of the peeling of the H-partition in $O(1)$ rounds of the congested clique (via working on the sampled subgraphs).

Running the Algorithm in Different Parts with Arboricity $O(\log n)$

At the beginning of the section, we explained that we partition graphs with arboricity higher than $\Theta(\log n)$ into many $\alpha/\Theta(\log n)$ subgraphs, each with arboricity $\Theta(\log n)$, and we then decompose each subgraph separately, using the procedure described above. It remains to explain why we can run the algorithm on all these subgraphs simultaneously. Notice that the local communications performed for peeling are in the edges of the subgraph, and therefore those can be performed in parallel. When performing a local topology gathering, each node needs to send or receive $2^{O((\log\log n)^2)}$ bits of information. Hence, even over all the $\alpha/\Theta(\log n)$ subgraphs, the total information that each each node needs to send or receive is $O(n)$, as we have assumed $\alpha \leq n^{1-O(\frac{(\log\log n)^2}{\log n})}$. After that, for the final step of the algorithm, we just need to bring a graph of size $O(n)$ to some leaders node and solve the problem of remaining nodes there, for each subgraph. It suffices to use different such leaders for different subgraphs.

3 Constant-Time Coloring

In this section, we provide our proof of Theorem 1, thus showing a randomized distributed algorithm that colors any graph of arboricity at most α using $O(\alpha)$ colors, w.h.p.

Proof of Theorem 1. The proof has several steps. We start with a high-level outline: we will first argue that it suffices to work with graphs of arboricity at most $O(\log n)$, because higher-arboricity graphs can be broken to this case easily, via randomness. Second, we set aside nodes of degree higher than α^{10}, to color them only later using some other fresh $O(\alpha)$ colors. Then, we explain how we partition any graph with arboricity $\alpha = O(\log n)$ to β/α subgraphs, each being a graph with arboricity almost $\beta = poly(\log\alpha)$ plus a small number of extra edges. Then, we explain how we color almost the entirety of each of these subgraphs, using different colors for different subgraphs. For this partial coloring, we will first explain a slow algorithm in the CONGEST model and then discuss how to simulate it in $O(1)$ rounds of the CONGESTED-CLIQUE model. The nodes that remain uncolored will be so few that we can gather the topology induced by them and color them using some 2α extra colors. We next explain these steps separated by paragraph titles, to reflect the above outline.

Studying $\alpha = O(\log n)$ Suffices. We focus on the case where $\alpha = O(\log n)$. Coloring of graphs with $\alpha = \Omega(\log n)$ arboricity can be transformed easily to this $O(\log n)$-arboricity case: place each node in a randomly chosen one of $k = \alpha/\Theta(\log n)$ parts. Each part will induce a subgraph with arboricity $\Theta(\log n)$, with high probability. The reason is that the original graph with arboricity α admits an orientation with out-degree 2α. Fix such an orientation. We do not need to know this orientation but use it for our analysis, only. When we randomly partition nodes, we expect each node's out-degree in its own part to be $2\alpha/k = \Theta(\log n)$. Hence, by applying a Chernoff bound, each node's outdegree in this imagined orientation is at most $3\alpha/k$. Thus, each of the subgraphs admits an orientation with out-degree at most $3\alpha/k$, which means it has arboricity at most $3\alpha/k$.

We color each of these parts using $\Theta(\log n)$ separate colors, all in parallel. For each part, we apply the algorithm for graphs with arboricity at most $O(\log n)$, which we describe next.

Setting aside nodes with degree $\Omega(\alpha^{10})$. As a preparation step, we set aside all nodes which have degree at least α^{10}. We note that a graph of arboricity α can have at most $2n/\alpha^9$ such nodes, simply because it has at most $n\alpha$ edges. This number of left-over nodes is small enough that allows us to deal with coloring these nodes later, using some fresh colors.

Partitioning to lower-arboricity subgraphs, plus few extra edges. We now have a graph $G = (V, E)$ with at most n nodes, arboricity $\alpha = O(\log n)$, and maximum degree at most α^{10}. We randomly partition V into $k = \frac{\alpha}{\log^3 \alpha}$ parts $V_1, V_2, \ldots, V_k$, by placing each node in a random part V_i. In Claim 10, we show that, with high probability, the subgraph $G[V_i]$ induced by the nodes in each part V_i is a graph with arboricity $(2+\epsilon)\log^3 \alpha$, plus at most $n/(\alpha)^{10}$ extra edges, for an arbitrarily small constant $\epsilon > 0$. We partially color each part $G[V_i]$ separately, using $O(\log^3 \alpha)$ different colors. This partial coloring will leave some $O(n/\alpha^5)$ nodes uncolored per part. In fact, the partial coloring will be done in two steps, one leaves at most $O(n/\log^3 \alpha)$ uncolored nodes, and the second reduces the number of uncolored nodes to $O(n/\alpha^5)$. At the very end, we will then gather the subgraph induced by all these remaining nodes over all parts – which has at most $k \cdot O(n/\alpha^5) \cdot \alpha = O(n)$ edges – in one node and color them using 2α extra colors.

▷ Claim 10. The subgraph $G[V_i]$ induced by each part V_i is a graph with arboricity $(2+\epsilon)\log^3 \alpha$, plus at most $n/(\alpha)^{10}$ extra edges, for an arbitrarily small constant $\epsilon > 0$.

Proof of Claim 10. Consider a hypothetical orientation of the graph with out-degree 2α and call a node v bad if in the subgraph $G[V_i]$ to which v belongs, its outdegree exceeds $2\alpha/k(1+\epsilon) = (2+\epsilon)\log^3 \alpha$. Notice that the probability of a node being bad is exponentially small in its expected outdegree, i.e., it is at most $exp(-\Theta(\alpha/k)) \leq exp(-\log^2 \alpha)$. Hence, the expected number of nodes that are bad is at most $nexp(-\log^2 \alpha)$. We would like to say that, with high probability, the number of bad nodes is at most $n \cdot exp(-\log^2 \alpha)$. We cannot directly apply the Chernoff bound, because the events of different nodes being bad are not independent of each other. However, the events are independent for any two nodes that do not share a common neighbor. Since we are now on a graph with maximum degree at most α^{10}, as we have set aside nodes of higher degree, we can infer that the even of each node being bad depends on the events of at most $d = \alpha^{20}$ many other nodes – all those within 2 hops. Hence, we can apply the extension of Chernoff to the setting with a bounded degree of dependency[24]. In particular, we get that the probability that the number of bad nodes exceeds $\mu = n \cdot exp(-\log^2 \alpha)$ by more than a constant factor is at most $\Theta(d) \cdot exp(-\Theta(\mu/d)) = \Theta(\alpha^{10}) \cdot exp(-\Theta(n \cdot exp(-\log^2 \alpha)/\alpha^{20})) \ll 1/poly(n)$. Hence, with high probability, the number of bad nodes is less than $\Theta(n \cdot exp(-\log^2 \alpha))$. A node that is bad introduced at most α^5 bad edges, in $G[V_i]$, because it is incident on at most α^{10} edges. Thus, except for at most $n \cdot exp(-\log^2 \alpha) \cdot \alpha^{10} \ll n/(\alpha)^{10}$ edges incident to bad nodes, all other edges can be oriented with out-degree $(2+\epsilon)\log^3 \alpha$. Hence, the subgraph is graph with arboricity $\beta = (2+\epsilon)\log^3 \alpha = O((\log\log n)^3)$, plus at most $n/(\alpha)^{10}$ extra edges. ◁

Outline for First Partial Coloring of Each Part. We now focus on one subgraph $H = G[V_i]$ which is a subgraph with arboricity $\beta = (2+\epsilon)\log^3 \alpha = O((\log\log n)^3)$, plus at most $n/(\alpha)^{10}$ extra edges. We next describe an algorithm $\mathcal{A}$ that runs in $O(\log^2 \beta)$ rounds of the CONGEST model and colors all but $O(n/\beta)$ many of the nodes of H, using $O(\beta)$ colors. Then, we explain how we simulate $\mathcal{A}$ in just $O(1)$ rounds of the CONGEST-CLIQUE model.

A CONGEST Algorithm for Partial Coloring of One Part. We focus on one part $H = G[V_i]$. First, we put aside all nodes of degree greater than β^2. The number of such nodes is $O(n/\beta)$. Then, we attempt to color most of the rest, in $O(\log^2 \beta)$ rounds, as follows. First, for $\ell = 40 \log \beta$ iterations, in each iteration j, we remove nodes of degree at most 2.1β and put them in *layer* L_j. Since the graph has arboricity β, in each iteration at least a constant fraction of nodes get removed. Thus, We can show that the number of remaining nodes after $40 \log \beta$ iterations is $O(n/(\beta)^5)$. Then, we attempt to color the nodes in layers L_ℓ, $L_{\ell-1}$,... L_1 using 2.2β colors. We spend one phase to color each layer layer, as follows.

We now describe each phase, which is simply $10 \log \beta$ repetitions of a simple randomized coloring attempt. In each round of the j^{th} phase, each node of layer L_j picks a color at random from 2.2β colors and keeps it if it is different than the colors of its neighbors in layers $L_{j'}$ for $j' \geq j$. A node that did not choose such a color remains uncolored in this round. After $10 \log \beta$ rounds, all nodes of L_j that remain uncolored are removed. Per round, each node gets colored with at least $0.1\beta/2.2\beta = 0.02$, regardless of the choices of its other up to 2.1β neighbors. Hence, the probability of a node remaining uncolored after $10 \log \beta$ rounds is at most β^5. Moreover, the events of different nodes remaining uncolored is independent, because in the above argument we only relied on the randomness in the color choices of the node itself. We can conclude that, with high probability, the number of these nodes in L_j is $O(n/(\beta)^2)$. We then proceed to the next phase. Over all the phases, the number of nodes in L_j that remain uncolored is at most $O(n \log \beta/(\beta)^2)$. Moreover, the number of nodes that were not in any of the layers is at most $O(n/(\beta)^2)$. Hence, overall, this process leaves at most $O(n/\beta)$ nodes of H uncolored.

A small final note about algorithm $\mathcal{A}$ is that each node uses at most $O(\log^2 \beta) \ll O(\log n)$ bits of randomness, $O(\log \beta)$ many bits in each round where it picks a random color from $\{1, 2, \ldots, 2.2\beta\}$. We will use this fact about the amount of randomness, later, when we speed up the algorithm in the CONGESTED-CLIQUE model.

CONGESTED-CLIQUE Algorithm for Partial Coloring of One Part. We focus on one part $H = G[V_i]$ and show how we mimic the algorithm described above, in just $O(1)$ rounds of CONGESTED-CLIQUE. As before, we put aside all nodes of degree greater than β^2. We next try to simulate $\mathcal{A}$, using a randomized information gathering approach.

We now use an opportunistic information gathering to compress the round complexity in CONGESTED-CLIQUE to $O(1)$. Concretely, we make each remaining node v send a message to all other nodes, where the message contains the name of v, its degree, and the at most $O(\log n)$-bits of randomness that v uses in algorithm $\mathcal{A}$. Besides the above, each v also sends each of its edges to each other node, but only randomly: each edge is sent to each node with probability $1/\beta^2$, all independent of each other. Notice that the total number of edges that are to be sent to each node is at most $O(n) \cdot \beta^2/\beta^2 = O(n)$, with high probability.

We say a node u successfully received the *relevant ball* of node v – with respect to algorithm $\mathcal{A}$ – if node u received all the edges incident on all nodes w within distance $\Theta(\log^2 \beta)$ of v. The number of such edges is at most $\beta^{\Theta(\log^2 \beta)} \leq 2^{\Theta(\log^3 \beta)}$. The probability of each of them being sent to u is $1/\beta^2$. Thus, the probability that they are all sent to u is $(1/\beta^2)^{2^{\Theta(\log^3 \beta)}} \geq 2^{-2\log\beta \cdot 2^{\Theta(\log^3 \beta)}} \geq 2^{-2^{\Theta(\log^3 \beta)}} \gg 2^{-\sqrt{\log n}}$, given that $\log \beta = \Theta(\log\log\log n)$. Since each node u successfully receives the ball of v with probability at least $2^{-\sqrt{\log n}}$, and as there are n possibilities for u, with high probability, there is at least one node u that successfully receives the ball of v. Notice that node u realizes this successful event as it knows the degree of each node and it can distinguish whether it has received all of the edges or not. Once u has received the relevant ball of node v, node u can simulate the CONGEST-model algorithm $\mathcal{A}$ and inform node v about the status of node v at the end of $\mathcal{A}$. This status is simply whether v is colored or not, and if yes, with which color.

Notice the small subtlety that there might be many nodes like u that receive the ball. However, since they all use the same randomness to simulate the behavior of each node in this ball (the randomness received from that node), they all get the same output for v.

Second Step of Partial Coloring. Finally, we go back to those $O(n/\beta)$ nodes that remain uncolored and we color enough of them that the number of uncolored nodes reduces to $O(n/\alpha^5)$. Given that H is a graph with arboricity at most β plus some $O(n/\alpha^{10})$ edges, the number of edges induced by the remaining $O(n/\beta)$ nodes is at most $O(n)$. That means we can move all these edges to one node . Then, we run a process similar to algorithm $\mathcal{A}$ on these remaining nodes, but in a centralized fashion. In particular, in $O(\log \alpha)$ iterations, we remove nodes of degree at most 2.1β. Then, we color these removed nodes greedily from the last layer to the beginning, using 2.2β extra colors. The only nodes that are not colored are those that are not in any of these $O(\log \alpha)$ layers. Since each layer of peeling and coloring removes at least a $0.1/2.2$ fraction of nodes (similar to the previous arguments), after $O(\log \alpha)$ such iterations, the number of nodes that remain uncolored is at most $O(n/\alpha^5)$ many.

Final Step, Coloring the Remaining Nodes. Finally, we handle these remaining nodes of different parts. In each of the parts $G[V_i]$, we have $O(n/\alpha^5)$ remaining nodes. Moreover, we had some $O(n/\alpha^9)$ nodes that we set aside at the very beginning, because they had degree greater than α^{10}. We collect the subgraph induced by all remaining nodes into one node, which has no more than $O(n/\alpha^4)$ edges, and we color them using 2α new colors. ◀

References

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

2 Leonid Barenboim and Michael Elkin. Deterministic distributed vertex coloring in polylogarithmic time. *Journal of the ACM (JACM)*, 58(5):23, 2011.

3 Leonid Barenboim and Michael Elkin. Distributed Graph Coloring: Fundamentals and Recent Developments. *Synthesis Lectures on Distributed Computing Theory*, 4(1):1–171, 2013.

4 Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-Iterative Distributed ($\Delta+$ 1):-Coloring below Szegedy-Vishwanathan Barrier, and Applications to Self-Stabilization and to Restricted-Bandwidth Models. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 437–446. ACM, 2018.

5 Leonid Barenboim and Victor Khazanov. Distributed Symmetry-Breaking Algorithms for Congested Cliques. In *International Computer Science Symposium in Russia*, pages 41–52. Springer, 2018.

6 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The Complexity of $(\Delta+1)$ Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation. *Manuscript*, 2019.

7 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An Optimal Distributed $(\Delta+1)$-coloring Algorithm? In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 445–456, 2018.

8 Mohsen Ghaffari. Distributed MIS via all-to-all communication. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 141–149. ACM, 2017.

9 Mohsen Ghaffari. Distributed Maximal Independent Set using Small Messages. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 805–820. SIAM, 2019.

10 Mohsen Ghaffari and Christiana Lymouri. Simple and Near-Optimal Distributed Coloring for Sparse Graphs. In *31st International Symposium on Distributed Computing (DISC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

11 Mohsen Ghaffari and Merav Parter. MST in log-star rounds of congested clique. In *Proc. of the Symp. on Princ. of Dist. Comp. (PODC)*, pages 19–28, 2016.

12 James W. Hegeman and Sriram V. Pemmaraju. Lessons from the Congested Clique Applied to MapReduce. *Theor. Comput. Sci.*, 608(P3):268–281, December 2015.

13 Öjvind Johansson. Simple distributed $\Delta+$ 1-coloring of graphs. *Information Processing Letters*, 70(5):229–232, 1999.

14 Tomasz Jurdziński and Krzysztof Nowicki. MST in O (1) rounds of congested clique. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2620–2632. SIAM, 2018.

15 Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 42–50. ACM, 2013.

16 Christoph Lenzen and Roger Wattenhofer. Brief announcement: Exponential speed-up of local algorithms using non-local communication. In *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 295–296. ACM, 2010.

17 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing (SICOMP)*, 21(1):193–201, 1992.

18 Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-weight spanning tree construction in O (log log n) communication rounds. *SIAM Journal on Computing*, 35(1):120–131, 2005.

19 C St JA Nash-Williams. Decomposition of finite graphs into forests. *Journal of the London Mathematical Society*, 1(1):12–12, 1964.

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

21 Merav Parter. (Δ+1) coloring in the congested clique model. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, 2018.

22 Merav Parter and Hsin-Hao Su. $(\Delta + 1)$ coloring in $O(\log^* \Delta)$ congested-clique rounds. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, 2018.

23 David Peleg. *Distributed Computing: A Locality-sensitive Approach.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

24 Sriram V Pemmaraju. Equitable coloring extends Chernoff-Hoeffding bounds. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 285–296. Springer, 2001.