ABSTRACT
We present a randomized distributed algorithm that computes a Δ-coloring in any non-complete graph with maximum degree Δ ≥ 4 in O(\log Δ + 2^{O(\sqrt{\log \log n})}) rounds, as well as a randomized algorithm that computes a Δ-coloring in O((\log \log n)^2) rounds when Δ ∈ [3, O(1)]. Both these algorithms improve on an O(\log^3 n / \log Δ)-round algorithm of Panconesi and Srinivasan [STOC’93], which has remained the state of the art for the past 25 years. Moreover, the latter algorithm gets (exponentially) closer to an O(\log log n) round lower bound of Brandt et al. [STOC’16].

CCS CONCEPTS
• Mathematics of computing → Graph coloring; • Theory of computation → Distributed algorithms;

KEYWORDS
distributed computing; distributed graph algorithms; graph coloring

1 INTRODUCTION AND RELATED WORK
This paper presents faster distributed algorithms, in the LOCAL model, for computing a Δ-coloring of any non-clique graph with maximum degree Δ ≥ 3. Moreover, we also provide certain structural results on the locality of the Δ-coloring problem. To formally present our results and put them in the context of the area, let us start with recalling the model.

Improved Distributed Delta-Coloring

Mohsen Ghaffari
ETH Zürich
Zürich
ghaffari@inf.ethz.ch

Fabian Kuhn†
University of Freiburg
Freiburg
kuhn@cs.uni-freiburg.de

Juho Hirvonen†
University of Freiburg
Freiburg
juho.hirvonen@cs.uni-freiburg.de

Yannic Maus†
University of Freiburg
Freiburg
yannic.maus@cs.uni-freiburg.de

The LOCAL Model of distributed computing [22, 28]. The graph is abstracted as an n-node network G = (V, E) with maximum degree at most Δ. Communications happen in synchronous rounds. Per round, each node can send one (unbounded size) message to each of its neighbors. At the end, each node should know its own part of the output, e.g., its own color.

1.1 Background and State of the Art
Graph coloring—assigning colors to the vertices of the graph such that no two adjacent vertices have the same color—has been a central problem in the study of distributed graph algorithms. We refer to the Distributed Graph Coloring book by Barenboim and Elkin [4].

Much of the focus in this area has been on computing a coloring with Δ + 1 colors. Notice that any graph has a (Δ + 1)-coloring, which can be computed via a trivial sequential greedy method. Hence, in a sense, distributed Δ + 1 coloring algorithms can all be viewed as attempts at parallelizing this greedy method. We are getting a better and better understanding of the complexity of this problem, see e.g., the very recent work of Chang et al. [12], which provides a 2^{O(\sqrt{\log \log n})}-round randomized algorithm for (Δ + 1)-coloring, and the references therein.

On the other hand, Δ-coloring is a problem of a very different nature. By a beautiful result of Brooks from 1941 [10], every connected graph admits a Δ coloring, unless it is exactly a complete graph or an odd cycle. The proof is of course far less trivial compared to that of (Δ + 1)-coloring. See the 1975 work of Lovász [23] for a simplified proof, which also supplies a polynomial-time centralized algorithm for computing a Δ-coloring.

Why should we care about Δ-coloring? General Aspects. One can argue that this single color of difference between Δ-coloring and (Δ + 1)-coloring is not relevant in practice. While that is probably true, we believe that there is a strong enough theoretical interest in investigating Δ-coloring. We view Δ-coloring as a clean and classic graph problem which reaches just outside the problems that we understand, and thus hopefully enables us to extend our understanding of the LOCAL model and to develop new algorithmic tools and techniques for it. The study of Δ-coloring has previously provided theoretical insight: (1) In the existential sense, Brooks’ theorem and proofs of it are widely studied and covered throughout graph theory textbooks (see e.g., [25, Theorem 1.4] and [8, Theorem
(\Delta + 1)-coloring, which is usually passed over as a triviality. (2) There is a sizable literature on sequential and also parallel (PRAM) algorithms for computing (\Delta) colorings. However, the sequential variant of (\Delta + 1)-coloring is again ignored as being a mere triviality. Moreover, the study of (\Delta + 1)-coloring in the PRAM model effectively stopped with the MIS algorithms of Luby [24] and Alon et al. [2], which led to an \(O(\log n)\)-round algorithm for (\Delta + 1)-coloring.

We also note that the relation between (\Delta + 1)-coloring and \(\Delta\)-coloring is similar to the relation between the two problems of (\Delta + 1 + 0(1))-coloring and (\Delta + 1)-coloring. One can argue that practically both are equally useful. However, the former can be solved easily in \(2^{O(\sqrt{\log \log n})}\) rounds using methods of Barenboim et al. [6], while there is still ongoing research on (\Delta + 1)-coloring [12, 21], which only very recently led to a \(2^{O(\sqrt{\log \log n})}\)-round algorithm [12].

Why should we care about \(\Delta\)-coloring? Technical Distributed Aspects. A concrete way of pointing out the difference between the two problems of \(\Delta\)-coloring and (\Delta + 1)-coloring is as follows: any partial coloring of vertices with \(\Delta + 1\) colors can be extended to a full coloring. However, this is not true for \(\Delta\)-coloring: we cannot extend any partial \(\Delta\)-coloring to a full coloring without changing the colors of some of the already colored vertices. This issue is one of the roots of our interest in understanding the complexity of this problem.

More concretely, many of the fast randomized algorithm for local graph problems developed over the past few years rely on the so-called shattering technique [6, 12, 14, 17, 20, 21]. In a rough sense, this method performs some randomized step which computes a partial solution such that the remaining part of the problem is made of several (disconnected) components, each of which is small, e.g., think of size \(\poly(\log n)\). Then, one can solve these smaller connected components using deterministic algorithms for graphs of size \(\poly(\log n)\). A crucial part here is that the partial solution is such that one can readily extend it to a full solution, in fact independently in each component, without needing to alter the already computed partial solution. The problem of \(\Delta\)-coloring gives us one clean local problem that reaches outside this circle. In particular, it is not clear if one can do shattering for \(\Delta\)-coloring, i.e., it is not clear whether there is a way of computing a partial \(\Delta\)-coloring such that the remaining components are small and they can be colored on their own without altering the already colored part.

Furthermore, in contrast to (\Delta + 1)-coloring, \(\Delta\)-coloring has an \(\omega(\log n)\) lower bound, even for constant-degree graphs [9, 11]. The nature of this problem is very different from (\Delta + 1)-coloring which can be computed in \(O(\log n)\) rounds in bounded degree graphs. Recently, in the context of lower bounds for the Lovász Local Lemma problem, Brandt et al. [9] proved that \(\Omega(\log n)\)-rounds are needed by any randomized \(\Delta\)-coloring algorithm, even in constant-degree graphs. These results led to two problems which exhibit an exponential separation between their randomized and deterministic complexity. Sinkless orientation has an \(\Omega(\log n)\) randomized lower bound [9] and an \(\Omega(\log n)\) deterministic lower bound [9, 11], with matching randomized and deterministic upper bounds [20]. The other problem is \(\Delta\)-coloring, which also has an \(\Omega(\log \log n)\) randomized lower bound [9] and an \(\Omega(\log n)\) deterministic lower bound [11]; however, finding matching upper bounds has remained mostly open.

State of the Art for distributed \(\Delta\)-coloring. Panconesi and Srinivasan gave a randomized distributed algorithm for computing a \(\Delta\)-coloring in \(O(\log^2 n / \log \Delta)\) rounds [26, 27]. They also provided a deterministic variant of their algorithm with complexity \(O(\Delta^2 \log n)\). Recently, Abouler et al. [1] gave a more general algorithm for \(d\)-list coloring graphs of maximum average degree \(d\) in time \(O(\Delta^4 \log^3 n)\). In the special case of trees of large enough maximum degree, Chang et al. [11] give an \(O(\log n)\)-round randomized algorithm for computing a \(\Delta\)-coloring. This, combined with their deterministic lower bound \(\Omega(\log n)\) [11], gives an exponential separation on trees. Our algorithms establish this separation in the general bounded-degree case.

1.2 Our Results

Our first result is tailored to \(\Delta\)-coloring constant-degree graphs.

Theorem 1. There is a randomized distributed algorithm that, in any non-complete graph with maximum degree \(\Delta \geq 3\), computes a \(\Delta\)-coloring in \(O(\sqrt{\Delta \log \Delta \cdot \log^* \Delta \cdot \log \log n})\) rounds, w.h.p.

Theorem 1 immediately implies an \(O(\log \log n)^2\) round algorithm for constant-degree graphs.

Corollary 2. There is a randomized distributed algorithm that, in any non-complete graph \(G\) with maximum degree \(\Delta \in [3, O(1)]\), computes a \(\Delta\)-coloring of \(G\) in \(O((\log \log n)^2)\) rounds, w.h.p.

We comment that the condition of \(\Delta \geq 3\) is necessary as \(2\)-coloring graphs with \(\Delta = 2\) needs \(\Omega(n)\) rounds, even if possible, e.g., in the case of an even cycle [22, 26]. The round complexity of Corollary 2 gets significantly closer to the \(O(\log \log n)\) lower bound of Brandt et al. [9]. Even in constant-degree graphs, the previous best known bound was the \(O(\log^2 n)\)-round algorithm of Panconesi and Srinivasan [26, 27].

Our second result applies to all graphs with \(\Delta \geq 4\) and improves on the \(O(\log^3 n / \log \Delta)\) round complexity of Panconesi and Srinivasan [26, 27]:

Theorem 3. There is a randomized distributed algorithm that, in any non-complete graph \(G = (V, E)\) with maximum degree \(\Delta \geq 4\), computes a \(\Delta\)-coloring in \(O(\log \Delta) + 2^{O(\sqrt{\log \log n})}\) rounds, w.h.p.

We also improve the deterministic complexity of \(\Delta\)-coloring for graphs with \(\Delta = 2^{\sqrt{\log n}}\).

Theorem 4 (Deterministic \(\Delta\)-coloring). Non-clique graphs of maximum degree \(\Delta \geq 3\) can deterministically be \(\Delta\)-colored in \(O(\sqrt{\Delta \cdot \log^{-3/2} \Delta \cdot \log^2 \Delta \cdot \log^2 n})\) rounds.

Note that Theorem 4 is close to the \(\Omega(\log \Delta)\) deterministic lower bound of [11] when \(\Delta = O(1)\).

\[\text{As standard, we use the phrase with high probability (w.h.p.) to indicate that an event happens with probability } 1 - 1/n^c \text{ for a desirably large constant } c \geq 2.\]
1.3 Our Methods

Our algorithms are based on a structural result that essentially says that either a graph is easy to $\Delta$-color locally, or it expands locally. This also yields a new proof of the distributed Brooks’ Theorem by Panconesi and Srinivasan.

**Theorem 5 (Distributed Brooks’ Theorem).** Let $G$ be a graph that is not a clique with maximum degree $\Delta \geq 3$, and let $G$ be $\Delta$-colored except for one node $v$. Now $G$ can be $\Delta$-colored by fixing the coloring inside the $(2\log_{\Delta-1} n)$-neighborhood of $v$.

Our algorithms are based on a layering technique. In this technique we carefully choose a base layer $B_0 \subseteq V$ that is easy to color, and layers $B_1 , \ldots , B_s$ where $B_i$ consists of the nodes in distance $i$ to $B_0$. To $\Delta$-color all layers one can iteratively color the layers in reverse order while always respecting the already fixed colors. To $\Delta$-color layer $B_i$, $i \neq 0$ we solve list coloring on the graph $G[B_i]$. Lists are of size $(\deg + 1)$ as each node has an uncolored neighbor on a lower index layer. At the end layer $B_0$ is (usually) colored with different techniques.

The best way to understand the technique is the algorithm for Theorem 4. There the base layer $B_0$ consists of the nodes of a ruling set of $G$ with large enough distance between nodes. The ruling property of $B_0$ implies that we only need few layers to color the whole graph and due to their large distance the nodes in $B_0$ can be colored independently with Theorem 5.

In our algorithms we find such layers and remove them from the graph. Let $H$ denote the graph of remaining nodes. In the algorithm explained above $H$ is empty. In our randomized algorithms the layers do not always cover the whole graph and thus $H$ might not be empty. However, the base layer is chosen such that our structural results (cf. Section 2.2) show that $H$ expands: In particular, we identify all small node-induced subgraphs that are colorable regardless of colors outside the subgraphs, compute a ruling set of these subgraphs and put their nodes in the base layer $B_0$. Then the remaining graph $H$ does not have any small subgraphs that are easy to color and our structural results show that $H$ has to expand. We leverage the expansion by randomly placing ‘slack’ in the graph, i.e., so called $T$-nodes and by removing – again with the layering technique – all nodes that find slack close by. Due to the expansion we can show that the probability to remain after the slack placement is $1/n^2$ for constant $\Delta$, and $1/poly(\Delta)$ for a suitable polynomial. In the second case standard shattering techniques (cf. Lemma 18) show that only small connected components remain which we color with a similar layering technique.

In the end we color the removed layers in reverse order via $(\deg + 1)$ list colorings as explained in the layering technique. The base layer $B_0$ can be colored fast by definition as it consists of independent small and easy colorable subgraphs.

We emphasize that — to the best of our knowledge — our shattering is different from all previous shattering algorithms. Previous shattering algorithms compute a partial solution to shatter the graph into small unsolved components which are then solved to complete the partial solution. Here, the nodes in the small components are the last nodes to compute their output. Our algorithms shatter in a fundamentally different way. We shatter the graph by removing nodes from it. The nodes in remaining components are the first to compute their output. Only afterwards we add the removed nodes to the graph and let them compute their output last.

All missing proofs can be found in the full version [18].

2 GRAPH COLORABILITY AND STRUCTURAL RESULTS

In this section we study structural properties of graphs that are not degree-list colorable, at least locally. We will show several structural results about such graphs, which essentially tell that these graphs must expand exponentially. This will lead to a simplified proof of the ‘distributed’ Brooks’ theorem due to Panconesi and Srinivasan [27] in Section 2.3.

2.1 Definitions

**Definition 6 (degree-choosability).** A graph $G$ is degree-choosable, if for every assignment of lists $L$, such that $|L(v)| \geq \deg(v)$ for all $v$, there exists a proper coloring of $G$ with colors from $L$.

**Definition 7 (Gallai-trees).** A graph is a Gallai-tree if all of its maximal 2-connected components are either cliques or odd cycles.

It is known that Gallai-trees are exactly those graphs which are not degree-choosable.

**Theorem 8 ([13, 30]).** A graph is not degree-choosable if and only if it is a Gallai-tree.

Now, consider the problem of $\Delta$-coloring. Assume that we color the graph partially but leave a 2-connected subgraph that is neither a clique nor an odd cycle uncolored. Then the coloring can be completed in this subgraph due to Theorem 8. These 2-connected subgraphs are called degree-choosable components.

**Definition 9 (degree-choosable component).** A node-induced subgraph is a degree-choosable component if it is 2-connected and not a clique nor an odd cycle.

We often write DCC instead of degree choosable component and the usual definitions for graphs can be extended to degree-choosable components. For example, the diameter of a degree-choosable component is the diameter of the node-induced subgraph.

A connected graph is a nice graph if it is neither a path, nor a cycle, nor a clique [27]. All nice graphs are $\Delta$-colorable and we assume that all graphs throughout the paper are nice graphs. A $T$-node is a node with two neighbors with the same color. In a partially colored graph, node $u$ is a $T$-node of $v$ if $u$ is a $T$-node and there is an uncolored path from $u$ to $v$.

2.2 Structure of Graphs with no Small Degree-Choosable Components

In this section we study graphs with no small degree-choosable components. Our goal is to prove that if such graphs are locally regular (and thus not easy to color locally), then these graphs must expand. Our general tool is to count the number of nodes in breadth-first search trees inside these graphs. Given a BFS tree BFS$(v)$ rooted at node $v$ of a graph $G$, we denote by $B_t(v)$ the set of nodes at distance $t$ from $v$ in this tree.
Lemma 10 (Unique BFS tree). Let $G$ be a graph with no degree-choosable components of radius $r$ or less. The depth-$r$ BFS tree rooted at an arbitrary $v \in V(G)$ is unique. In particular, any node $u \neq v$ on level $t \leq r$ has exactly one edge to the nodes on level $t - 1$ of the BFS tree.

Proof of Lemma 10. It is immediate that the zero and depth one BFS trees are unique. For larger depth consider the following proof by contradiction. For $t < r$ assume that $u$ and $u'$ are two nodes on the $t$-th level of the BFS tree that connect to the same node $w \notin B_{t-1} \cup B_t$, i.e., we assume that the next level of the BFS tree cannot be built uniquely. Then $w$ is on level $t + 1$. Let $v'$ be the least common ancestor of $u$ and $u'$. Then there is the even cycle $(w, u'), \text{path}(u', v'), \text{path}(v', u), [u, w]$ that does not induce a clique as the nodes $u, w, u'$ lie on three different levels, a contradiction. \hfill \Box

For a node $u$ in a BFS tree let $d(u)$ denote the number of children of $u$ in the BFS tree.

Lemma 11 (BFS expansion lemma). Let $G$ be a graph without any DCC of radius at most $r$ and BFS$(v)$ the unique depth-$r$ BFS tree rooted at some node $v \in V(G)$. Let $u'$ be a node of BFS$(v)$ with $\deg(u') \geq 3$ and $u$ its immediate ancestor. Then $d(u) + d(u') \geq \min(\deg(u), \deg(u'))$ holds.

For two nodes $u, w$ of a BFS tree let $P(u, w)$ denote the unique path from $u$ to $w$ in the BFS tree.

Proof of Lemma 11. Let $u$ be on some level $j \leq r - 1$ of the BFS tree and $u'$ on level $j + 1 \leq r$. If $u = v$ the statement holds trivially as $d(u) = \deg(u)$. So assume that $u \neq v$. Due to the uniqueness of the BFS tree (cf. Lemma 10) $u \neq v$ has a single neighbor on level $j - 1$ and $\deg(u) - 1$ neighbors on level $j$ and $j + 1$. Similarly $u$ is the only neighbor of $u'$ on level $j$. For two nodes $u, u'$ of the BFS tree let $P(u, u')$ denote the unique path in the BFS tree between $u$ and $u'$. Let $\alpha = \min(\deg(u), \deg(u'))$. The result holds if $d(u) \geq \alpha$. We consider two cases for $d(u) < \alpha$.

Case $d(u) = \alpha = 1$: Assume that $u'$ has one neighbor $u''$ in $N(u)$. We show that $u''$ does not have a neighbor on its level that is not connected to $u$. So assume that $u''$ also has a neighbor $w$ on level $j + 1$ that is not a child of $u$. Let $v'$ denote the last common ancestor of $u'$ and $w$ in the BFS tree. The subgraph induced by the union of $\{u', w\} \cup \{u', u''\} \cup \{u, u''\} \cup \{u, w\} \cup P(v', w) \cup P(v', u)$ is a DCC of radius at most $r$, a contradiction. Thus if $u'$ has one neighbor in $N(u)$ it has at most $\alpha - 2$ neighbors on its own level and we have $d(u') \geq \deg(u') - 1 - (\alpha - 2) = \deg(u') - \alpha + 1$. This implies $d(u) + d(u') \geq \alpha$.

Now assume that $u'$ has no neighbor in $N(u)$. We show that it can have at most one neighbor on its own level. Assume that it has two neighbors $w, w'$ on its own level. Let $v'$ denote the last common ancestor of $w$ and $w'$. Then the union of edges $P(v, w) \cup \{w, u''\} \cup \{w', u''\} \cup P(w', v)$ forms an even cycle that does not induce a clique, a contradiction. In this case we have $d(u') \geq \deg(u') - 2 \geq 1$ where the last inequality holds due to $\deg(u') \geq 3$. This implies $d(u) + d(u') \geq \alpha$.

Case $d(u) < \alpha - 1$: Because $d(u) < \deg(u) - 1$ node $u$ must have a neighbor on level $j$ that we denote with $w$. We first prove the following subclaim.

Claim: Node $u'$ has no neighbor on level $j + 1$ which is not connected to $u$.

Proof. Assume by contradiction that such a neighbor denoted with $w'$ exists. Note that the edge $(w, w')$ does not exist because otherwise the nodes $u, u', w'$ and $w$ would form a 4-cycle which does not induce a clique as $(u, w') \notin E$.

Let $v'$ be the last common ancestor of $u$, $w$, $u'$ and $w'$. Then $(u', w'), P(w', v'), P(v', w), [u, w, u']$ forms an even cycle. The path $P_{w', v'}$ neither goes through $w$ nor through $u$ as the edges $(w, w')$ and $(u, w')$ do not exist. Thus the even cycle actually is a proper cycle and does not collapse. Furthermore it does not induce a clique as the nodes $u, u', w$ and $w'$ lie on three different levels of the BFS tree. Thus it induces a short even cycle, i.e., a DCC with radius at most $r$, a contradiction. \hfill \Box

Due to the claim all neighbors of $u'$ in $G$ that are on the same level as $u'$ in the BFS tree are also connected to $u$ in the BFS tree. As $u$ has at most $d(u)$ children $u'$ has at most $d(u) - 1$ neighbors on the same level and one neighbor in the level above. This implies $d(u') \geq \deg(u') - 1 - (d(u) - 1) = \deg(u') - d(u)$. That is, $d(u) + d(u') \geq \deg(u') \geq \alpha$.

Informally, this lemma holds because if there are too many edges inside the local neighborhood of a node, then these edges must create a degree-choosable component.

Marking process. In our algorithms we apply the following marking process. Each node selects itself independently and uniformly at random with some probability $p$. Then, if there is another selected node within distance $b$ (the backoff distance), the node unselects itself. Otherwise it picks two non-adjacent neighbors and colors them with color one. We call these neighbors marked. The selected node becomes a T-node. The following lemma shows that if we apply the marking process, the graph of the unmarked nodes expands deterministically.

Lemma 12. Let $\Delta \geq 4$ and $G = (V, E)$ be a graph such that $G$ does not contain any DCCs of radius at most $r$, for an even $r$. Apply the marking process to $G$ with $b = 6$ and remove all marked nodes to obtain graph $H$. Consider an arbitrary node $v \in V(H)$. If $\Delta - 1 \leq \deg(u) \leq \Delta$ for each $u \in N_v(v)$, then the $r$-hop BFS tree around node $v \in V$ has at least $(\Delta - 2)^{b/2}$ nodes on level $r$.

The proof is based on the previous Lemma 11. Due to the backoff distance $b$ marked nodes cannot exist too close to each other if they do not form a T-node. Thus for every node on a level of the BFS tree that is blocked from expanding due to a marked node, there are many other nodes that are not blocked. We show that expansion also holds for $\Delta = 3$ if the backoff distance $b$ between selected nodes is increased to 12. These proofs appear in the full version.

2.3 A Simplified Proof for the Distributed Brooks’ Theorem

Panconesi and Srinivasan proved a distributed version of Brooks’ Theorem (cf. Theorem 5). The goal of this section is to provide a simplified proof of the result. We begin by observing that if a graph does not have any small degree-choosable components, it is locally
expanding. This result is easier to prove than Lemma 12 as it does not include the marking process.

Lemma 13. Let $G$ be a graph and $v \in V(G)$ be a node such that inside the $r$-radius neighborhood of $v$ there are no degree-choosable components and every node has degree $\Delta$. Then for each even $r$ there are at least $(\Delta - 1)^{r/2}$ nodes at distance $r$ from $v$.

Now we can use previous lemmas to show that the uncolored node in the statement of Theorem 5 can fix its color as it sees a degree-choosable component or a node of degree $< \Delta$ inside its $O(\log \Delta \cdot n)$-neighborhood.

Lemma 14. Let $G$ be a graph with maximum degree $\Delta \geq 3$. The $(2\log \Delta - 1)\cdot n$-neighborhood of any node contains a degree-choosable component or it contains a node of degree $< \Delta$.

Proof. Fix a node $v \in V(G)$ and assume that its $r = 2\log \Delta - 1 \cdot n$ neighborhood does not contain a degree-choosable component and that nodes in this neighborhood have degree $\Delta$. By Lemma 13, the BFS tree has $|B_r(v)| \geq (\Delta - 1)^{r/2} \cdot n$ nodes. Therefore the BFS tree cannot expand, and there is an edge to a lower level of BFS($v$) from $B_r(v)$, or there is a node of degree $< \Delta$ in $B_r(v)$.

Proof of Theorem 5. Let $c$ denote the partial coloring $G$, with $c(v) = \perp$. We say that $v$ has a token. We can always do the following operation: let $u$ be an arbitrary neighbor of $v$. If $u$ does not have a free color, that is, all of its $\Delta$ neighbors have $\Delta$ different colors, then we can move the token to $u$ and color the node $v$ with color $c(u)$. If $u$ has a free color, it can choose that color and the token is eliminated. Now, if the $(2\log \Delta - 1)\cdot n$-neighborhood of $v$ contains a node of smaller degree, we can move the token to that node, and it is guaranteed to have a free color. Now assume that no such node exists. By Lemma 14, there exists a degree-choosable component in the $(2\log \Delta - 1)\cdot n$-neighborhood of $v$. Let $u$ be one of the closest nodes to $v$ in the degree-choosable component $B$. We move the token from $v$ to $u$ by the shortest path. Next we uncolor the whole of $B$. By definition there exists a $\Delta$-coloring of $B$ compatible with the existing coloring in the rest of the graph.

Remark 15. Theorem 5 implies an SLOCAL($O(\log \Delta \cdot n)$) algorithm (cf. [19] for the model).

3 DETERMINISTIC $\Delta$-COLORING (PROOF OF THEOREM 4)

In this section we present our deterministic $\Delta$-coloring algorithm, exemplifying our layering technique.

Layering technique. In the layering technique there is a carefully chosen base layer $B_0$ that is easy to color and layers $B_1, \ldots, B_k$, where $B_i$ consists of the nodes in distance $i$ to $B_0$. This is particularly helpful for $\Delta$-coloring as we can $\Delta$-color the layers in reverse order while respecting the colored neighbors in layers with a larger index. To $\Delta$-color layer $B_i$, $i \neq 0$ we need to solve a $(\deg + 1)$ list coloring on the graph $G[B_i]$: A node $v \in B_i$ builds its list by removing the colors of neighbors in $B_{i-1} \cup \ldots \cup B_k$ from the set $[1, \ldots, \Delta]$. The size of this list is at least $\deg c(u) + 1$ as $v$ has one neighbor in layer $B_{i-1}$. Then layer $B_0$ is colored after all other layers with different techniques as $\Delta$-coloring $B_0$ while respecting already colored neighbors might not be a deg + 1 list coloring instance. To make sure that we can still $\Delta$-color $B_0$ efficiently (we might have to recolor previously colored nodes) it has to be chosen carefully.

This section uses the technique in a simple setting. We continue with list coloring results. In the $(\deg + 1)$ list coloring problem each node $v$ has a list $L(v)$ of available colors with $|L(v)| \geq \deg(v) + 1$. The objective is to properly color the graph such that each node picks a color from its list.

Theorem 16 ([15] + [5]). There is a deterministic distributed $(\deg + 1)$ list coloring algorithm with complexity $O(\sqrt{\Delta} \log \Delta \cdot \log^* \Delta)$ given a $O(\Delta^2)$ coloring of the graph.

Theorem 17 (List Coloring [17]). There is a randomized distributed algorithm that solves the $(\deg + 1)$-list coloring problem in $O(\log \Delta + 2O(\sqrt{\log \log n}))$ rounds.

We also use ruling set algorithms several times. A $(\alpha, \beta)$ ruling set of a graph $G$ is a subset $M \subseteq V(G)$ of the nodes such that dist$(v, M \setminus \{v\}) \geq \alpha$ for all $v \in M$ and dist$(v, M) \leq \beta$ for all $v \in V$. Table 1 summarizes the used distributed ruling set algorithms.

Table 1 summarizes the used distributed ruling set algorithms. Usually, when computing a ruling set it comes with a so called ruling forest. See [3, Definition 3] for a definition of ruling forests.

The following algorithm improves the deterministic complexity of distributed $\Delta$-coloring. Its proof is based on the layering technique with $O(\log^2 \Delta)$ layers and Theorem 5. The layer $B_0$ is chosen to be a ruling set such that Theorem 5 can be applied to color the nodes in $B_0$ independently. The runtime is dominated by the $O(\log^2 \Delta)$ iterations of list coloring due to the layering technique.

Algorithm. First, color all nodes of $G$ with $O(\Delta^2)$ colors with Linear's algorithm [22]. These colors are only used for symmetry breaking when applying list coloring algorithms and do not coincide with the desired $\Delta$-coloring. Let $R = 4 \log \Delta - 1 \leq 7 \log n / \log \Delta$ and $z = 4 \cdot R^2$.

1. (Build layer $B_0$) Compute a $(R, z)$ ruling forest of $G$ with part (2) of Table 1. Add all nodes of the ruling set to layer $B_0$.
2. (Remove layers $B_0, \ldots, B_z$) Define layers $B_1, \ldots, B_z$ where $v \in B_i$ if the distance of $v$ to $B_0$ is $i$. Remove all layers from the graph.
3. (Color layers $B_2, \ldots, B_k$) Add the layers $B_2, \ldots, B_k$ to the graph one by one: When adding layer $B_i$ color the nodes of $B_i$ such that $G_{f_i} = G[B_i]$ is validly $\Delta$-colored. Step $i = 2, \ldots, 1$ is a deg + 1 list coloring instance on $G_i = G[B_i]$ because a node $v \in B_i$ has an uncolored neighbor in $B_{i-1}$. We use Theorem 16 to solve each list coloring instance.
4. (Color layer $B_0$) Use Theorem 5 to independently color the nodes in $B_0$ through recoloring nodes within distance at most $2 \log \Delta - 1 \leq R/2$.

Proof of Theorem 4. By the definition of a ruling set every node of $G$ is in distance at most $z$ from its root in the ruling forest. Thus every node is contained in the $z + 1$ layers and is colored.

We formally show that coloring each layer is an instance of deg + 1 list coloring in the graph $G_i$. Assume that we are in step $i$ and want to color the nodes of $B_i$ such that $G_{f_i}$ is validly $\Delta$-colored. Pick a node $v \in B_i$. The list of available colors of $v$ is
The third step takes \( \Delta \) set of well-separated nodes and color two of their neighbors with randomly inserting \( H \) graph has been colored. These are removed from the graph, along with each node has an uncolored neighbor in a lower layer. The runtime of the first step and the second step is

\[
O(R^2 \cdot \sqrt{\Delta} + R \cdot \log^s n + z + \log^s n) = O(R^2 \cdot \sqrt{\Delta}).
\]

The third step takes \( O(\sqrt{\Delta} \log \Delta \log^s \Delta) \) rounds for each of the \( z = O(R^2) = O(\log^2 n) \) iterations. The fourth step takes \( O(R) \) rounds. In total the runtime is dominated by the third step.

4 RANDOMIZED \( \Delta \)-COLORING (PROOFS OF THEOREM 1 AND 3)

In this section we present our algorithm. The algorithm is split into two slightly different versions based on \( \Delta \): one version can handle any \( \Delta \geq 4 \) and the other any \( 3 \leq \Delta \leq O(1) \). We refer to these two versions as the large-\( \Delta \) version and the small-\( \Delta \) version. In this section we present the algorithms of Theorems 1 and 3 and their proofs.

Both variants share the same basic structure. We decompose the graph into layers \( B_0, \ldots, B_s, C_0, \ldots, C_2r \) (and in some cases also layers \( D_0, \ldots, D_a \)) of nodes such that all nodes are either colored or are in one of the layers. Then, the layers are iteratively colored in the reverse order that they were built. Coloring a single layer requires solving a (deg + 1)-list coloring instance since we will guarantee that each node has an uncolored neighbor in a lower layer.

In Phase I we build layers \( B_0, \ldots, B_s \); we identify the dense parts of the graph—the parts which are easy to color after the rest of the graph has been colored. These are removed from the graph, along with the nodes around them, to be colored later. Let \( H \) denote the remaining graph.

In Phase II we extract layers \( C_0, \ldots, C_{2r} \) from \( H \): The first phase guarantees that \( H \) does not contain any dense parts, and therefore the remaining graph must expand. We take advantage of this by randomly inserting slack into the graph. This means that we pick a set of well-separated nodes and color two of their neighbors with the same color: these nodes now effectively have decreased their degree and will be easy to color later. We again remove the nodes with slack along with the nodes around them to be colored later. Due to the expansion of \( H \) we can prove that the probability of each node to remain after this process is small.

Actually, in the small-\( \Delta \) case we prove by union bound that with high probability no node remains after this process. In the large-\( \Delta \) case we show that the graph formed by the remaining nodes has shattered: remaining connected components are of small size and can be colored efficiently with a similar layering technique using layers \( D_0, \ldots, D_a \) (cf. Phase III).

In Phase III we color the layers \( C_0, \ldots, C_{2r} \) in reverse order. Then, in Phase IV, we color the layers \( B_1, \ldots, B_s \) in the reverse order. By definition \( B_0 \) consists of (dense) parts that are easy to color if the remaining graph is colored, actually \( B_0 \) consists of DCCs and we can color the components of \( B_0 \) at the very end.

4.1 The Randomized \( \Delta \)-Coloring Algorithms

First, we remove all degree-choosable components of radius \( r \) or less from the graph. This implies that the graph must expand locally (Lemma 14). The two versions differ in the radius \( r \): in the small version we choose \( r = O(\log \log n) \) and in the large version \( r = O(1) \). Let \( b = 6 \) and \( p = \Delta^{-b} \).

First, color all nodes of \( G \) with \( O(\Delta^2) \) colors with Linial’s algorithm [22]. These colors are only used for symmetry breaking when applying list coloring algorithms and do in no way coincide with the desired \( \Delta \)-coloring.

I Removing Degree Choosable Components with Small Radius

(a) Each node that is contained in at least one degree choosable subgraph with radius at most \( r \) selects one such subgraph. Let \( G_{DCC} \) be the virtual graph that has a node for each selected degree choosable subgraph, and two subgraphs in \( V(G_{DCC}) \) are connected in \( G_{DCC} \) by an edge if they share a vertex or if they are connected by an edge in \( G \). The graph \( G_{DCC} \) has at most \( n \) nodes, maximum degree at most \( \Delta^{r+1} \leq \Delta^r \), and one round of a distributed algorithm in it can be simulated in \( O(r) \) rounds in \( G \).

Runtime: \( O(r) \).

(b) (Build layer \( B_0 \)) Find a \( (2, \beta) \) ruling set \( M \) of \( G_{DCC} \) with \( \beta = 6 \cdot r \). Add all nodes of \( O \) that are in a component of \( M \) to the base layer \( B_0 \).

Runtime(\( n \)): Use the Table 1, (4) in time

\[
O\left(\log^{1/\beta} \Delta + 2^{O(\sqrt{\log \log n})}\right).
\]

Runtime(\( n, \Delta \)): Use Table 1, (1) in time

\[
O(\beta \cdot \Delta^{2r/\beta} + \log^s n) = O(\Delta^{1/3} + \log^s n).
\]

Runtime for small \( \Delta \): Use Table 1, (3) in time

\[
O(r \cdot (\log \log n)) = O(\log^2 \log n).
\]

(c) (Remove layers \( B_1, \ldots, B_s \)) For \( s = \beta \cdot (r + 1) \) define layers \( B_1, \ldots, B_s \). Layer \( B_i \) consists of the nodes of \( G \) that have the closest node with regard to distance in \( B_0 \) in distance exactly \( i \). Remove all layers \( B_0, \ldots, B_s \) from the graph.

Runtime for large \( \Delta \): \( O(\beta \cdot r) = O(1) \).

Runtime for small \( \Delta \): \( O(\log \log n) \).
Note that (besides potentially some other nodes) in particular all nodes that are in a degree choosable component with radius at most \( r \) are removed from the graph after phase Ic.

II Shattering of the Remaining Graph

d) (Random \( \ell \)-node creation) Consider the remaining graph

\[ H = G \setminus \left( \bigcup_{i=0}^{s} B_i \right). \]

Each node of \( H \) becomes selected independently with probability \( p \). Then, if there is another selected node within distance \( h \), both become unselected. If not, the selected node picks a random pair of non-adjacent neighbors and colors them with color one. We call these neighbors marked.

Runtime: \( O(1) \).

e) (Remove layers \( C_0, \ldots, C_{2r} \)) We call a node happy if it has an uncolored path to a \( \ell \)-node in its \( r \)-neighborhood. By this definition we assign each happy node to its closest \( \ell \)-node in its \( r \)-neighborhood.

We define the boundary of graph \( H \) as the set of nodes with degree less than \( \Delta \) in \( H \). Nodes that are colored and have distance at most \( \ell \) steps away from the boundary now remove their color and each such node is assigned to its closest boundary node, breaking ties using identifiers. Now it might happen that a node \( v \) that is \( \ell \leq \ell' < 2\ell \) steps away from the boundary was assigned to a node \( w \) in the first round of assigning and that \( w \) is at most \( r - 1 \) steps away from the boundary. Due to the uncoloring \( w \) might not be a \( \ell \)-node anymore. However, \( w \) is assigned to a node \( w' \) on the boundary. Then there is an uncolored path of length at most \( 2r \) from \( v \) to \( w' \) through \( w \) and we assign \( v \) to \( w' \) as well.

Define layers \( C_0, \ldots, C_{2r} \), where \( C_i \) consists of nodes of \( H \) that are at distance \( i \) from their respective assigned node. The layer \( C_0 \) consists of \( \ell \)-nodes, and all nodes that have degree less than \( \Delta \) in \( H \). Remove the layers \( C_0, \ldots, C_{2r} \) and the marked nodes from the graph.

In Section 4.4 we show that in the algorithm for small \( \Delta \), all nodes are removed after this phase with high probability. Hence this algorithm proceeds directly to Phase IIIg.

Runtime for large \( \Delta \): \( O(r) = O(1) \).

Runtime for small \( \Delta \): \( O(r) = O(\log \log n) \).

(f) (Color Small Components) Consider the remaining graph \( L = H \setminus (C_0 \cup \ldots \cup C_{2r} \cup C') \) where \( C' \) are the marked nodes. In Section 4.2 we show that the probability for a node of \( H \) to remain in \( L \) is small and then the standard shattering technique (cf., e.g., [7] or Lemma 18) implies that \( L \) consists of small connected components of size at most \( N := \text{poly} \; \Delta \cdot \log \Delta \).

Section 4.3 explains in detail how to color these small components. The core idea is that we can again handle the small components by constructing layers \( D_0, \ldots, D_a \) where \( a = O(\log^2 \log n) \). Besides some other nodes layer \( D_0 \) contains the nodes that have an uncolored neighbor in the layers \( C_0 \cup C_1 \cup \ldots \cup C_{2r} \), i.e., they just did not get removed because the closest \( T \)-node was a little bit too far away. One can show that layer \( D_0 \) contains at least one node of each small component. However, then all nodes of the component are in one of the layers, because, assuming that a node \( v \) of a small component does not see a node of the first few layers, the BFS tree of \( v \) within the component expands so fast (basically due to Lemma 12) that it sees the whole component in \( O(\log N = O(\log \log n) \) hops, a contradiction.

Runtime for small \( \Delta \): \( O(\sqrt{\Delta \log \Delta \log^* \Delta \cdot \log^2 \log n}) \) due to the \( O(\log^2 \log n) \) iterations of list coloring.

III Color Happy Nodes From the Shattering Process

(g) (Color layers \( C_{2r}, \ldots, C_0 \)) Assume that the remaining small components are colored with \( \Delta \) colors in Phase III. Go through the layers \( C_{2r}, \ldots, C_0 \) grown in step IIe in reverse order and \( \Delta \)-color them one at a time while respecting the colors of nodes that are already colored. Coloring layer \( C_i \) corresponds to a \((\deg' + 1)\)-list coloring instance on \( H[C_i] \), since for each \( i \), \( i \) each node has a neighbor at a lower level and the nodes in \( C_{2r} \) have two neighbors of the same color.

Runtime for large \( \Delta \): \( O(\log \log n \sqrt{\Delta \log \Delta \log^* \Delta}) \) with Theorem 17.

Runtime for small \( \Delta \): \( O(\sqrt{\Delta \log \Delta \log^* \Delta}) \) time with Theorem 16.

IV Color Degree Choosable Components

(h) (Color layers \( B_s, \ldots, B_1 \)) Go through the layers \( B_s, \ldots, B_1 \) grown in step Ic and color each layer with \( \Delta \) colors while respecting nodes colored previously. Coloring layer \( B_i \) forms a \((\deg' + 1)\)-list coloring instance on \( G[B_i] \), since each node has an uncolored neighbor in \( B_{i-1} \).

Runtime for large \( \Delta \): \( O(\log \log n \sqrt{\Delta \log \Delta \log^* \Delta}) \) with Theorem 17.

Runtime for small \( \Delta \): \( O(\log \log n \sqrt{\Delta \log \Delta \log^* \Delta}) \) time with Theorem 16.

(i) (Color layer \( B_0 \)) By definition, the components \( M \) selected in step Ib, i.e., the nodes in \( B_0 \), are \( \Delta \)-list colorable and of radius \( \leq r \). As there are no edges between different components in \( M \) by construction they can be handled independently. We find a coloring by brute forcing each component.

Runtime for large \( \Delta \): \( O(r) = O(1) \).

Runtime for small \( \Delta \): \( O(r) = O(\log \log n) \).

4.2 Shattering of the Remaining Graph (Phases IIId-IIIf)

In this section we show that the processing of phase IIId and IIIf produce a graph with remaining components of size \( O(\text{poly}(\Delta) \cdot \log n) \). In Section 4.3 we show how to color these small components fast. The nodes that are put into the layers in phase IIe are colored later in phase IIIg.

For a node \( v \) let \( E_v \) be the event that \( v \) is removed in the graph in phases IIId-IIIf. Let \( t \) be the radius such that the event \( E_v \) only depends on the random bits of nodes in radius \( t \) of \( v \). The standard shattering technique (cf. Lemma 18) shows that the connected components of non-removed nodes are small if the probability of \( E_v \) is upper bounded by \( 1/\text{poly}(\Delta) \) where the polynomial depends on the radius \( t \).
Lemma 18 (The Shattering Lemma, [14, 18]). Let $H = (V, E)$ be a graph with maximum degree $\Delta$. Consider a process which generates a random subset $B \subseteq V$ where $P(v \in B) \leq \Delta^{-c_1}$, for some constant $c_1 \geq 1$, and that the random variables $1(v \in B)$ depend only on the randomness of nodes within at most $c_2$ hops from $v$, for all $v \in V$, for some constant $c_2 \geq 1$. Moreover, let $Z = H[2c_2 + 1, 4c_2 + 2]$ be the graph which contains an edge between $u$ and $v$ if their distance in $H$ is between $2c_2 + 1$ and $4c_2 + 2$. Let $L = H[B]$. Then with probability at least $1 - n^{-c_1}$, for any constant $c_3$ satisfying $c_1 > c_3 + 4c_2 + 2$, we have the following three properties:

(P1) $Z[B]$ has no connected component $U$ with $|U| \geq \log_A n$.
(P2) Each connected component of $L$ has size at most $O(\log_A n \cdot \Delta^{c_2+c_3})$.
(P3) $L$ admits a $(\lambda, O((\log^{1/3} n \cdot \log^2 \log n))$ network decomposition, for any integer $\lambda \geq 1$, which can be computed by a randomized algorithm in $O(\lambda \log^{1/3} n \cdot 2^{O(\sqrt{\log \log n})})$ rounds, w.h.p.

(P4) For any integer $R \geq 1$ there is a randomized algorithm to compute a $\left(2^{O(\log \log n) \cdot R}, R \cdot 2^{O(\sqrt{\log \log n})}\right)$ network decomposition of $L^R$ in $O(R \cdot 2^{O(\sqrt{\log \log n})})$ rounds, w.h.p.

To show that the probability of $\overline{S_v}$ is small enough we show that the BFS tree of uncolored nodes around $v$ expands exponentially. Thus after $O(1)$ steps of expansion we see uncolored paths to enough nodes that independently form a $T$-node with probability $\Theta(p)$ and the probability that none of them actually is a $T$-node will be at most $1/poly(\Delta)$ for a sufficiently small polynomial.

Now, we upper bound the probability that a given node does not become happy after the shattering process. Due to Lemma 12 the BFS tree around a node expands deterministically even after the marking process which implies the next lemma.

Lemma 19. For every $0 < t \leq r$ and after the selection and marking process the $t$-neighborhood of every node $v$ contains a boundary node or a set of nodes $S_v$ with the following properties:

1. $|S_v| \leq (\Delta - 2)^{t/2} \cdot \Delta^{-6}$,
2. All nodes in $S_v$ are reachable through uncolored nodes from $v$,
3. For each $u \in S_v$ the probability that it is selected and creates a $T$-node that does not block the path to $v$ is at least $1/3 \cdot p(1 - p)^t$. The events are independent for distinct $u \in S_v$,
4. For each $u \in S_v$, the event that it forms a $T$-node of the above type only depends on the random bits of nodes in radius $t + 7$ around $v$.

Proof of Lemma 19. Let $v$ be a fixed node. Due to Lemma 12 the BFS tree around $v$ restricted to unmarked nodes contains at least $(\Delta - 2)^{t/2}$ nodes on level $t$. Let $A_v$ be the set of these nodes.

For each node $u \in A_v$, whose children in the BFS tree form a $\Delta - 1$ clique we remove $u$ from $A$ and add one of its children $u'$ in the BFS tree to $A_v$. As the child has the $\Delta - 2$ nodes of the clique on its own level and $u$ as parent it has only one child in the BFS tree. Thus the children of $u'$ in the BFS tree cannot form a $\Delta - 1$ clique. Furthermore $u'$ is distinct from all other nodes in $A_v$ as the BFS tree is unique.

Now, we greedily add nodes of $A_v$ to $S_v$. When we add a node $u \in A_v$ to $S_v$ we remove the nodes from $A_v$ that are in the $6$-neighborhood of $u$; these are at most $\Delta^6$ many. Thus the size of $S_v$ is at least $|A_v| \cdot \Delta^{-6} = (\Delta - 2)^{t/2} \cdot \Delta^{-6}$ and nodes in $S_v$ have pairwise distance at least $7$.

We now compute the probability that a node $u \in S_v$ is selected and creates a $T$-node that does not block the path to $v$. To ensure that the path to $v$ is not blocked we (1) condition on the event that certain nodes in the BFS tree around $v$ are not uncolored (through the usage of Lemma 12) and (2) we ensure that none of the two nodes that $u$ colors is the single neighbor $u'$ of $u$ that lies on the unique path in the BFS tree to $v$. Node $u$ is selected with probability $p$ and stays selected if no neighbor in its 6-neighborhood is selected, i.e., at least with probability $(1 - p)^4$. As $u$ does not have a $\Delta - 1$ clique on the next level of the BFS tree there are at least two non adjacent neighbors $u_1$ and $u_2$ of $u$ that are distinct from $u'$. So the probability that $u$ does not mark $u'$ is at least $1/3$.

In this whole process we expanded for $t$ steps to obtain the set $A_v$. The set $S_v$ contains nodes in distance at most $t + 1$ from $v$ and we use that nodes in distance 6 to nodes in $S_v$ are not selected, i.e., the probabilities only depend on the $t + 7$ radius of $v$. The event whether distinct nodes in $S_v$ can generate $T$-nodes are independent as they have pairwise distance at least 7.

For any node $v$ Lemma 19 provides a large set of independent nodes that have uncolored paths to $v$. Thus we can upper bound the probability that a node remains after the shattering process.

Lemma 20 (Shattering Probability). Let $\Delta \geq 4$. There is an $r = O(1)$ such that every node finds an uncolored path of length at most $r - 7$ to a $T$-node with probability at least $1 - \left(\frac{1}{\Delta}\right)^{4r+3}$ using only the randomness in its $t$ neighborhood. The constant $r$ is independent from the graph.

Proof. Let $v$ be a node in $H$. Apply Lemma 19 with $t = r - 7$ and obtain a set $S_v$ in which each node independently forms a $T$-node that is reachable from $v$ through an uncolored path with probability $1/3 \cdot p(1 - p)^t$. The probability that $v$ remains after phase IIc is upper bounded by

$$\left(1 - \frac{1}{3}p(1 - p)^t\right)^{|S_v|} \leq e^{-\frac{|S_v|}{3}p(1 - p)^t} \leq e^{-\left(\frac{\Delta - 2}{\Delta}\right)^{t/2} \cdot \Delta^{-6}} \leq \Delta^{-4t - 32}$$

where the $\left(\frac{\Delta - 2}{\Delta}\right)^{t/2}$ is satisfied if the exponent $-12^{-1}(\Delta - 2)^{t/2} \cdot \Delta^{-6}$ is smaller than $-(4t + 32) \cdot \ln \Delta$ which holds for some $t = O(1)$ and implies an $r = O(1)$ that is independent from $v$ and the graph.

Lemma 20 and the standard shattering technique (cf. Lemma 18) imply that the graph $L$ that remains after phase IIc consists of connected components of size at most poly $\Delta \cdot \log_A n$. Section 4.3 explains in detail how these components can be $\Delta$-colored while respecting the nodes colored with color one in phase IIb. The algorithm for the small components relies on a layering technique with $O(\log \log n)$ layers; after removing these layers no node is left.

4.3 Shattering: Coloring Small Remaining Components

We now explain how one can solve the small components that are left after the shattering process. Let $C$ be a small component with size at most $N := \text{poly}(\Delta) \cdot \log_A n$. Call a node in $C$ free if it has
degree < Δ or at least one neighbor outside of C that is not colored with the first color after the shattering process. We color the nodes of C with the following algorithm where \( R = 2 \log_{\log n} N + 1 = O(\log \log n) \). The algorithm is explained from the view of a single component.

1. Each free node selects itself. Further, each node that is contained in at least one DCC with radius at most R selects one of these subgraphs. Let \( C_{DCC} \) be the virtual graph that has a node for each selected node and degree choosable subgraph. Any two subgraphs (or nodes) of \( C_{DCC} \) are connected in \( C_{DCC} \) if they share a vertex or are connected by an edge in G. The maximum degree of \( C_{DCC} \) is \( \min\{N, O(\Delta^0(R)) \} \) and it has at most \( |C| = N \) nodes. One round of an algorithm on \( C_{DCC} \) can be executed in \( O(R) \) steps in G.

2. Find a \((2, \gamma)\) ruling set \( M' \) of \( C_{DCC} \) where \( \gamma = O(R) \) such that \( \Delta(C_{DCC})^{(2)}/\gamma \leq \Delta^{(2)}/2 \).

**Runtime(n):** We use with Lemma 18 (P4) to compute a \( (2O(\log \log n), 4R \cdot 2O(\log \log n)) \) network decomposition of \( L^4R \). Then each node assigns its color in this network decomposition to its corresponding selected node in \( C_{DCC} \). This yields a \( (2O(\log \log n), 2O(\log \log n)) \) network decomposition of \( C_{DCC} \). Then iterate through the colors of the network decomposition to compute the ruling set in time \( O\left(R \cdot 2O(\log \log n)\right) = 2O(\log \log n) \).

**Lemma 21.** If \( D_0 \) is not empty each node of the component is in one of the layers.

**Proof.** The layers \( D_0, \ldots, D_{\gamma - (R + 1)} \) contain all free nodes, all nodes that are in a DCC with radius at most R and all nodes that have degree smaller \( \Delta \). The layers \( D_0, \ldots, D_{\gamma - (R + 1)} + R \) additionally contain the nodes that have such a DCC or such a node in distance at most R. To show that all nodes are removed we assume that there is a node \( v \in C \) that is in none of the layers. In particular it does not have a DCC or a free node in distance R, all nodes in its R-neighborhood have degree \( \Delta \) or \( \Delta - 1 \). As the R-neighborhood of \( v \) does not contain a free node it can only hit the boundary of C at colored nodes, i.e., its R-neighborhood can be obtained from the marking process as described in Section 2.2. Thus we can apply Lemma 12 and obtain that the BFS tree around \( v \) and within the component expands and contains at least \((\Delta - 2)^{R/2} > N\) nodes, a contradiction.

**Lemma 22.** \( D_0 \) is not empty.

**Proof.** Assume that \( D_0 \) is empty. Let \( v \) be an arbitrary node of C. Its R-neighborhood neither contains a DCC of radius at most R nor a free node and all its nodes have degree \( \Delta \) or \( \Delta - 1 \). As the R-neighborhood of \( v \) does not contain a free node it can only hit the boundary of C at colored nodes, i.e., its R-neighborhood can be obtained from the marking process as described in Section 2.2. Thus we can apply Lemma 12 and obtain that the BFS tree around \( v \) and within the component expands and contains at least \((\Delta - 2)^{R/2} > N\) nodes, a contradiction.

The above algorithm provides the following lemma.

**Lemma 23.** Let \( \Delta \geq 4 \). Then the small components can, w.h.p., be \( \Delta \)-colored in time

\[
\min\{\gamma^{O(\log \log n)}, O(\log^2 n \cdot \sqrt{\Delta \log \Delta \Delta^*(\Delta)} \}.
\]

**Proof.** Lemmas 21 and 22 imply that each node is colored. The proof that coloring a single layer in phase 4 is a a deg + 1 list coloring instance is along similar lines as in the proof of Theorem 4. The components and free nodes in \( D_0 \) can be colored independently because the stem from the independent set \( M' \). In both variants the runtime is dominated by phase 4 step which implies the result.

**Remark 24.** The algorithm to solve the small components only uses randomization if it computes the network decomposition (Lemma 18).

### 4.4 Global Success After Marking Process for Small \( \Delta \)

We will assume that locally all nodes in the remainder graph \( H \) have degree \( \Delta = O(1) \), since nodes close to the boundary of \( H \) automatically become happy.

Lemma 12 and its strengthened version imply that for \( \Delta = O(1) \) we can choose an \( r = O(\log \log n) \) such that for an arbitrarily large constant \( c \), we have \( |B_r(v)| \geq c \log_n n \) after the marking process.

**Lemma 25.** Let \( u \) be a node such that there is an unmarked, unselected path from \( u \) to \( v \). Then \( u \) or its child \( u' \) becomes a T-node of \( v \) with a constant probability.

Note that the following analysis is done for \( b = 12 \). For \( \Delta \geq 4 \) we could equally well use \( b = 6 \) to optimize the constants.

**Proof Sketch.** A node \( u \) can become a T-node to node \( v \) if it has two non-adjacent neighbors that are also not on the path from \( u \) to \( v \). Assume first that this is the case.

The children of \( u \) form at least two distinct cliques (including single nodes). There are \( \Delta - 1 \) pairs that include the parent of \( u \) and at least \( \Delta - 2 \) pairs that do not. Therefore if \( u \) can become a T-node of \( v \),
and if it is selected becomes one with probability at least $1/3$. Node $u$ succeeds with probability at least \( p' = (p/3)(1-p)^\Delta \leq \Theta(1) \), since $\Delta = O(1)$. Now if the children of $u$ form a clique, $u$ cannot become a $T$-node of $v$. Its children must have a successor in the BFS tree, and therefore can become a $T$-node for $v$. The events that $u$ or a child of $u$ succeed are not independent but are disjoint, so the claim holds.

Note that the event in Lemma 25 depends on randomness at distance at most 13.

**Lemma 26.** The marking process generates a $T$-node for every node of the remainder graph $H$ with high probability.

**Proof.** Consider an arbitrary node $v \in V(H)$. By Lemma 12 and its strengthened version, for any $\epsilon > 0$ we can choose $\epsilon = \Theta((\log \log n)$ such that at distance $t$ from the root of any BFS tree, there are at least $c_p/\Delta^3 \ln n$ nodes such that their path to $u$ is unmarked and unselected. From this set find a set $S$ of nodes as in Lemma 25, of size $c_p/\Delta^3 \ln n$ such that they are at pairwise distance of at least 13 and can each produce a $T$-node for $v$ with probability $p'$. The events that each $u \in S$ becomes a $T$-node for $v$ are independent due to the pairwise distance of the nodes. Therefore no node of $S$ becomes a $T$-node of $v$ with probability at most 
\[
(1-p')^c/\Delta^3 \ln n \leq e^{-\epsilon} \ln n \leq n^{-c} .
\]

With a union bound over all nodes, all nodes of $H$ are happy with probability at least $1 - 1/n^{c-1}$.

**REFERENCES**

[1] Pierre Aboulker, Marthe Bonamy, Nicolas Bousquet, and Louis Esperet. 2018. Coloring in sparse graphs with fewer colors. In Proc. 37th ACM Symposium on Principles of Distributed Computing (PODC 2018). To appear.

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

[3] B. Awerbuch, A.V. Goldberg, M. Luby, and S.A. Plotkin. 1989. Network Decomposition and Locality in Distributed Computation. In Proc. 30th Symposium on Foundations of Computer Science (FOCS 1989). 364–369.

[4] Leonid Barenboim and Michael Elkin. 2013. Distributed graph coloring: Fundamentals and recent developments. *Synthesis Lectures on Distributed Computing Theory* 4, 1 (2013), 1–171.

[5] Leonid Barenboim, Michael Elkin, and Uri Goldenberg. 2018. Locally-Iterative Distributed ($\Delta+1$)-Coloring below Szegedy-Vishwanathan Barrier, and Applications to Self-Stabilization and to Restricted-Bandwidth Models. In Proc. 37th ACM Symposium on Principles of Distributed Computing (PODC 2018).

[6] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2012. The Locality of Distributed Symmetry Breaking. In Proc. 33rd Symposium on Foundations of Computer Science (FOCS 2012). 321–330.

[7] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2016. The Locality of Distributed Symmetry Breaking. *Journal of the ACM* 63, 3, Article 20 (2016), 45 pages. https://doi.org/10.1145/2903137

[8] J.A. Bondy and U. Murty. 1976. *Graph Theory with Applications*. Elsevier.

[9] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiainen, Joel Rybicki, Jukka Suomela, and Jara Uitto. 2016. A lower bound for the distributed Lovász local lemma. In Proc. 48th ACM Symposium on Theory of Computing (STOC 2016). ACM, 479–488.

[10] Rowland Leonard Brooks. 2009. On colouring the nodes of a network. In *Classic papers in combinatorics*. Springer, 118–121.

[11] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. 2016. An exponential separation between randomized and deterministic complexity in the local model. In Proc. 57th Symposium on Foundations of Computer Science (FOCS 2016). IEEE, 615–624.

[12] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. 2018. An Optimal Distributed ($\Delta+1$)-Coloring Algorithm?. In Proc. 59th ACM Symposium on Theory of Computing (STOC 2018). To appear.

[13] Paul Erdős, Arthur Rubin, and Herbert Taylor. 1979. Choosability in Graphs. In Proc. West Coast Conference on Combinatorics, Graph Theory, and Computing, Vol. 26. Congressus Numerantium, 125–157.

[14] Manuscha Fischer and Mohsen Ghaffari. 2017. Sublogarithmic Distributed Algorithms for Lovász Local Lemma, and the Complexity Hierarchy. In Proc. 31st International Symposium on Distributed Computing (DISC 2017) (LIPIcs), Vol. 94. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 18–1:18–16.

[15] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. 2016. Local Conflict Coloring. In Proc. 57th Symposium on Foundations of Computer Science (FOCS 2016). 625–634.

[16] Beat Gfeller and Elias Vicari. 2007. A Randomized Distributed Algorithm for the Maximal Independent Set Problem in Growth-bounded Graphs. In Proc. 26th ACM Symposium on Principles of Distributed Computing (PODC 2007). ACM, New York, NY, USA, 53–60. https://doi.org/10.1145/1281100.1281111

[17] Mohsen Ghaffari. 2016. An Improved Distributed Algorithm for Maximal Independent Set. In Proc. 27th ACM-SIAM Symposium on Discrete Algorithms (SODA 2016). 270–277. https://doi.org/10.1137/1.9781611974331.ch20

[18] Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, and Yannic Maus. 2018. Improved Distributed $\Delta$-Coloring. CoRR abs/1803.03248 (2018).

[19] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. 2017. On the complexity of local distributed graph problems. In Proc. 49th ACM Symposium on Theory of Computing (STOC 2017). ACM, 784–797.

[20] Mohsen Ghaffari and Hsin-Hao Su. 2017. Distributed degree splitting, edge coloring, and orientations. In Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017). Society for Industrial and Applied Mathematics, 2505–2523.

[21] David G Harris, Johannes Schneider, and Hsin-Hao Su. 2016. Distributed ($\Delta+1$)-coloring in sublogarithmic rounds. In Proc. 48th ACM Symposium on Theory of Computing (STOC 2016). ACM, 465–478.

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

[23] László Lovász. 1975. Three short proofs in graph theory. *Journal of Combinatorial Theory, Series B* 19, 3 (1975), 269–271.

[24] Michael Luby. 1986. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.* 15, 4 (1986), 1056–1065.

[25] Michael Molloy and Bruce Reed. 2013. *Graph colouring and the probabilistic method*. Vol. 23. Springer Science & Business Media.

[26] Alessandro Panconesi and Aravind Srinivasan. 1992. Improved distributed algorithms for coloring and network decomposition problems. In Proc. 24th ACM Symposium on Theory of Computing (STOC 1992). ACM, 581–592.

[27] Alessandro Panconesi and Aravind Srinivasan. 1995. The local nature of $\Delta$-coloring, and its algorithmic applications. *Combinatorica* 15, 2 (1995), 255–280. https://doi.org/10.1007/BF01200759

[28] David Peleg. 2000. Distributed computing. *SIAM Monographs on discrete mathematics and applications* 5 (2000).

[29] Johannes Schneider, Michael Elkin, and Roger Wattenhofer. 2013. Symmetry breaking depending on the chromatic number or the neighborhood growth. *Theoretical Computer Science* 509 (2013), 40–50.

[30] Vadim Vizing. 1976. Vertex coloring with given colors. *Metody Diskret. Anal.* (1976), 293–310.