Distributed Computing (2021) 34:239–258
https://doi.org/10.1007/s00446-021-00397-4

Improved distributed Δ-coloring

Mohsen Ghaffari[1] · Juho Hirvonen[2] · Fabian Kuhn[3] · Yannic Maus[4]

Received: 1 October 2018 / Accepted: 29 May 2021 / Published online: 9 July 2021

Abstract

We present a randomized distributed algorithm that computes a Δ-coloring in any non-complete graph with maximum degree $\Delta \geq 4$ in $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$ rounds, as well as a randomized algorithm that computes a Δ-coloring in $O((\log \log n)^2)$ rounds when $\Delta \in [3, O(1)]$. Both these algorithms improve on an $O(\log^3 n / \log \Delta)$-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 $\Omega(\log \log n)$ round lower bound of Brandt et al. (STOC'16).

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 $\Delta \geq 3$. 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.

The LOCAL model of distributed computing [25,32]. 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.

Partly supported by ERC Grant No. 336495 (ACDC) and Ulla Tuominen Foundation.

✉ Fabian Kuhn
kuhn@cs.uni-freiburg.de

Mohsen Ghaffari
ghaffari@inf.ethz.ch

Juho Hirvonen
juho.hirvonen@aalto.fi

Yannic Maus
yannic.maus@campus.technion.ac.il

[1] ETH Zurich, Zürich, Switzerland

[2] Aalto University, Espoo, Finland

[3] University of Freiburg, Freiburg im Breisgau, Germany

[4] Technion, Haifa, Israel

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 $\Delta + 1$ colors. Notice that in any graph, a $(\Delta + 1)$ coloring can be computed via a trivial sequential greedy method: Iterate through the vertices in an arbitrary order and a node picks a color that is not used by any of its at most Δ already colored neighbors. Hence, in a sense, distributed $\Delta + 1$ 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. [13], which provides a $2^{O(\sqrt{\log \log n})}$-round randomized algorithm for $(\Delta + 1)$-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 [10,11], 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 $(\Delta + 1)$-coloring. See the 1975 work of Lovász [26] 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 $(\Delta + 1)$-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., [29, Theorem 1.4] and [8, Theorem 14.4]), while $(\Delta+1)$-coloring is usually passed over as a triviality. (2) There is a sizable literature on sequential and also parallel (PRAM) algorithms for computing Δ-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 [27] 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 Δ-coloring is similar to the relation between the two problems of $\Delta(1+o(1))$-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 [13,23], which only very recently led to a $2^{O(\sqrt{\log\log n})}$-round algorithm [13].

Why should we care about Δ-coloring? Technical distributed aspects. A concrete way of pointing out the difference between the two problems of Δ-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 Δ-coloring: we cannot extend any partial Δ-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 algorithms for local graph problems developed over the past few years rely on the so-called *shattering* technique [6,13,17,20,22,23]. 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 $\mathsf{poly}(\log n)$. Then, one can solve these smaller connected components using deterministic algorithms for graphs of size $\mathsf{poly}(\log n)$. 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 Δ-coloring gives us one clean local problem that reaches outside this circle. In particular, it is not clear if one can do shattering for Δ-coloring, i.e., it is not clear whether there is a way of computing a partial Δ-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, Δ-coloring has an $\omega(\log^* n)$ lower bound, even for constant-degree graphs [9,12]. 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 $\Omega(\log\log n)$ rounds are required by any randomized Δ-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 $\Omega(\log\log n)$ randomized lower bound [9] and an $\Omega(\log n)$ deterministic lower bound [9,12], with matching randomized and deterministic upper bounds [22]. The other problem is Δ-coloring, which also has an $\Omega(\log\log n)$ randomized lower bound [9] and an $\Omega(\log n)$ deterministic lower bound [12]; however, finding matching upper bounds has remained mostly open.

State of the art for distributed Δ-coloring. Panconesi and Srinivasan [30,31] gave a randomized distributed algorithm for computing a Δ-coloring in $O(\log^3 n/\log\Delta)$ rounds. They also provided a deterministic variant of their algorithm with complexity $O(\Delta\log^2 n)$. Recently, Aboulker 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. [12] give an $O(\log\log n)$-round randomized algorithm for computing a Δ-coloring. This, combined with their deterministic lower bound $\Omega(\log n)$ [12], 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 Δ-coloring constant-degree graphs.

Theorem 1 *There exists a randomized distributed algorithm that, in any non-complete graph G with maximum degree $\Delta \geq 3$ computes a Δ-coloring in time $O\left(\sqrt{\Delta\log\Delta}\cdot\log^*\Delta\cdot\log^2\log n\right)$, w.h.p.*[1]

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

Corollary 1 *There exists a randomized distributed algorithm that, in any non-complete graph G with maximum degree $\Delta\geq 3$, computes a Δ-coloring of G in time $O((\log\log n)^2)$, w.h.p.*

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

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 [25,30]. The round complexity of Corollary 1 gets significantly closer to the $\Omega(\log \log n)$ round 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 [30,31].

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 [30,31]:

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

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

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

Note that Theorem 3 is only a logarithmic factor away from the $\Omega(\log_\Delta n)$ deterministic lower bound of [12] when $\Delta = O(1)$.

1.3 Our methods

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

Theorem 4 (Distributed Brooks' Theorem) *Let G be a graph that is not a clique with maximum degree $\Delta \geq 3$, and let G be Δ-colored except for one node v. Now G can be Δ-colored by recoloring the $(2 \log_{\Delta-1} n)$-neighborhood of v and keeping the color of all nodes outside this neighborhood unchanged.*

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 after everything else is colored, and layers $B_1, \ldots, B_s$ where B_i consists of the nodes in distance i to B_0. To Δ-color all layers one can iteratively color the layers in *reverse order* while always respecting the already fixed colors. To Δ-color layer B_i, $i \neq 0$ we solve list coloring on the graph $G[B_i]$. Lists are of size $(\deg_{G[B_i]} + 1)$ 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 3. There the base layer B_0 consists of the nodes of a ruling set of G with large enough distance between the nodes. The ruling property of B_0 implies that we only need few layers to cover the whole graph and due to their large distance the nodes in B_0 can be colored independently with Theorem 4.

Let H denote the *graph of remaining nodes* after the base layer and all remaining layers have been removed. 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. Sect. 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 (each T-node picks two of its neighbors (non adjacent) and colors them with the same color; this introduces slack at the T-node as it can always find a valid color after every other node of the graph is colored). Then we use those T-nodes as a new base layer and remove—again with the layering technique—all nodes that have a T-node close by. Due to the expansion we can show that the probability to remain after the slack placement is $1/n^c$ for constant Δ; for non constant Δ a node has to be much closer to a node with slack (than in the constant degree case) to be removed and we can only bound the probability to remain by $1/\mathsf{poly}(\Delta)$ for a suitable polynomial. However, then standard shattering techniques (cf. Lemma 13) show that only *small* connected components remain which we color with a similar layering technique.

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. The idea of putting nodes away to be colored in the end has already been used in the deterministic coloring algorithm in [4] where graphs with bounded arboricity are colored. However, we are not aware of any randomized algorithm that uses this technique.

1.4 Outline

Section 2 provides our core structural results for Δ-coloring and its proofs are most involved. In particular, we show that

- A partial Δ-coloring of a graph with a single uncolored vertex v can be completed to a Δ-coloring of all vertices by only recoloring the vertices in a $O(\log_\Delta n)$ neighborhood of v (Distributed Brooks Theorem, Theorem 4),
- A partial Δ-coloring of a graph with an uncolored connected component C can be completed to a Δ-coloring of the whole graph without changing the colors of already colored vertices if C is a so called degree choosable component,
- Graphs that do not contain small diameter degree choosable components expand quickly,
- The uncolored part of a graph without small diameter degree choosable components expands quickly even if we randomly place T-node's. (Sect. 2.3)

We recommend to skip Sect. 2.3 when reading the paper for the first time.

Section 3 introduces algorithmic preliminaries and state of the art results for problems such as network decomposition, $(\deg+1)$ list coloring and ruling sets that we use as subroutines in our algorithms. In Sect. 4 we use the Distributed Brooks Theorem to provide two deterministic algorithms for Δ-coloring. These algorithms already contain much of the high level structure of our randomized algorithms that we present in Sect. 5. We end in Sect. 6 with a conclusion.

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 [31] in Sect. 2.4.

2.1 Gallai-trees and degree choosability

Definition 1 (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 2 (Gallai-Trees) A graph is a *Gallai-tree* if each of its maximal 2-connected components is a clique or an odd cycle.

Gallai-trees are exactly those graphs which are not degree-choosable.

Theorem 5 [16,35] *A graph is not degree-choosable if and only if it is a Gallai-tree.*

Now, consider the problem of Δ-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 5. These 2-connected subgraphs are called *degree-choosable components* (see Figs. 1 and 2).

Definition 3 (Degree-Choosable Component) A node-induced subgraph is a *degree-choosable component (DCC)* if it is 2-connected and not a clique nor an odd cycle.

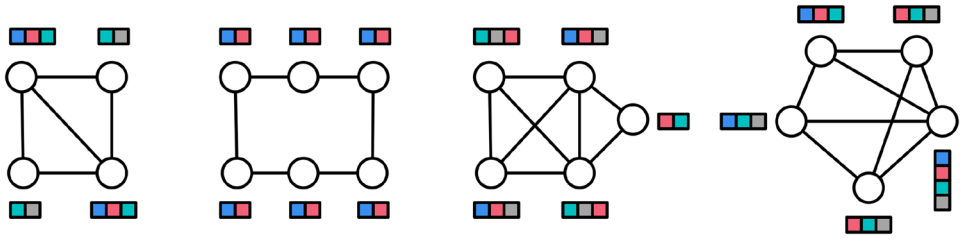
We often write *DCC* instead of degree choosable component and the usual graph notions can be extended to degree-choosable components. For example, the *diameter* of a degree-choosable component is the diameter of the node-induced subgraph. A connected graph is a *nice graph* if it is neither a path, a cycle, nor a clique [31]. Note that a degree choosable subgraph (DCC) can be Δ colored regardless of how other nodes outside the DCC are colored (see Fig. 2). All nice graphs are Δ-colorable and we assume that all graphs throughout the paper are nice graphs. A T-node is a node with two neighbors that have 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. For several (centralized) proofs of Brooks' Theorem and further work on Gallai trees and degree choosability we refer to [15]. We also want to point out that degree choosable components recently became important for the distributed coloring of sparse and planar graphs [1,14].

2.2 Graphs with no small degree-choosable components

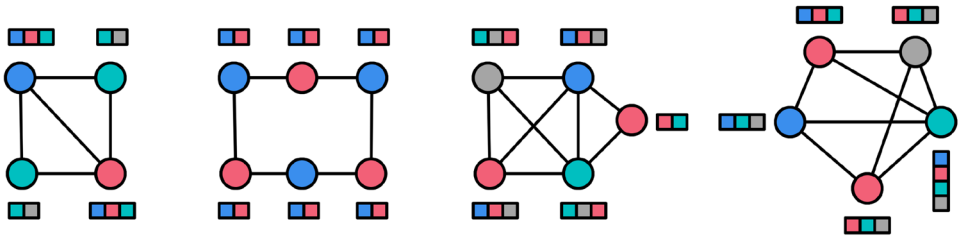
In this section we study graphs with no small degree-choosable components. Our goal is to prove that if such graphs are locally regular (and thus not easy to color locally), then these graphs must expand. Our general tool is to count the number of nodes in breadth-first search trees inside these graphs. Given a BFS tree $\mathrm{BFS}(v)$ 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 and 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.

Lemma 1 (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 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'



(a) Four examples of degree choosable components that can be degree list colored.



(b) A valid coloring of the degree choosable components where every node has a color from its list.

Fig. 1 Degree list coloring of degree choosable graphs

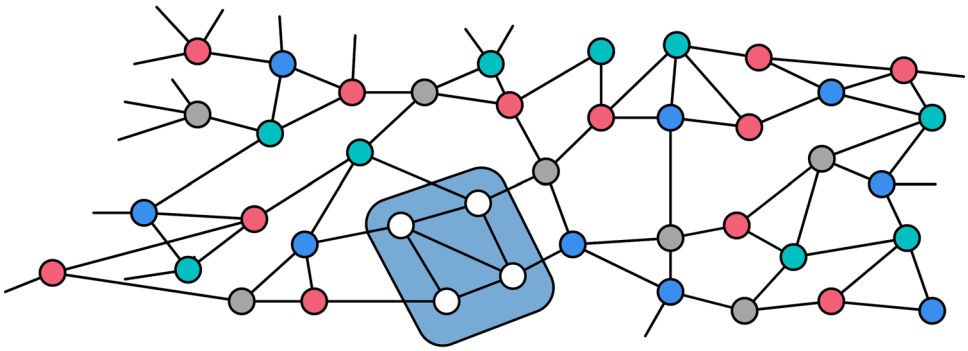


Fig. 2 If a degree choosable component (DCC) appears in a graph and all nodes around it are colored then this induces a degree list coloring problem for the degree choosable component. By the definition of a DCC one can always find a valid coloring of the DCC

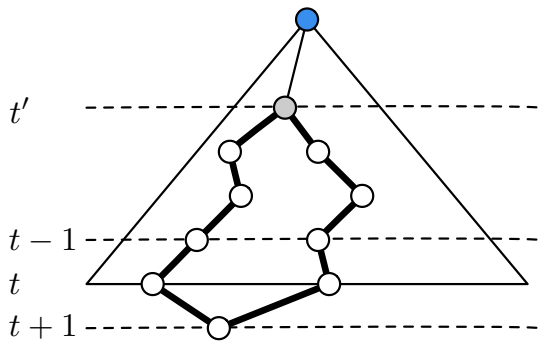


Fig. 3 Uniqueness of the BFS-tree. Assuming a unique BFS-tree up to level t, assume that a node on the next level $t+1$ can be reached via two distinct nodes u, v on level t. This creates an even cycle via the last common ancestor of u and v on some level $t' < t$. The cycle does not induce a clique since nodes of the cycle are on at least three different levels of the BFS-tree

be the least common ancestor of u and u'. Then there is the even cycle $\{w, u'\}$, $P_{u',v'}$, $P_{v',u}$, $\{u, w\}$ that does not induce a clique as the nodes u, w, u' and v' lie on three different levels, a contradiction. □

The following simple but useful result shows that the neighborhood of a node decomposes into cliques if there are no small degree-choosable components.

Lemma 2 *Let G be a graph with no degree-choosable components of radius* 1. *Then for every $v \in V(G)$ the connected components of $G[\Gamma(v)]$ are cliques.*

Proof Let u_1, u_2 and w be neighbors of v with $\{u_1, u_2\} \in E$. If $\{u_2, w\} \in E$ we also have that $\{u_1, w\} \in E$ as otherwise the graph induced by v, u_1, u_2 and w would be a degree choosable component of radius one, a contradiction. □

Note that all neighbors of any node v with degree Δ have to induce more than one clique as otherwise v and its neighbors would form a $(\Delta+1)$-clique and the graph does not admit a Δ-coloring.

For a node u in a BFS tree let $N_+(u)$ denote the set of children of u in the BFS tree and let $d(u) = |N_+(u)|$ be the number of children in the tree.

Lemma 3 (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 parent. Then $d(u) + d(u') \geq \min\{\deg(u), \deg(u')\}$ holds.

Proof Assume that the node u is on some level $j \leq r-1$ of the BFS tree. Then u' is 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 1) $u \neq v$ 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 at least one neighbor in $N(u)$ that is on level $j+1$ and let u'' be such a neighbor of u'. We show that u' does not have a neighbor on its level that is not connected to u. For contradiction, assume that u' also has a neighbor w on level $j+1$ that is not a child of u. Let v' denote the least common ancestor of u' and w in the BFS tree. The subgraph induced by the union of $\{u', w\} \cup \{u', u''\} \cup \{u, u''\} \cup P(v', w) \cup P(v', u)$ 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 $d(u) + d(u') \geq \deg(u') \geq \alpha$.

Now assume that u' has no neighbor in $N(u)$ on level $j+1$. 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 least common ancestor of w and w'. Then the union of edges $P(v, w) \cup \{w, u'\} \cup \{u', w'\} \cup P(w', v)$ 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 by w. We first prove the following subclaim. □

Subclaim: 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 least common ancestor of w and w'. Then

$$\{u', w'\}, P_{w'v'}, P_{v'w}, \{w, u\}, \{u, u'\}$$

form 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' and v' 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. □

Due to the subclaim 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 $d(u)$ 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. Lemma 3 implies that the BFS tree in the graph G (without the marking process) expands exponentially in $\Delta - 1$ with every two hops (see the proof of Lemma 8 for more details).

2.3 Exponential expansion after the marking process

Next, we define a randomized marking process and show that the BFS tree of unmarked nodes either contains a T-node or expands exponentially as well.

Marking process. In our algorithms we apply the following marking process. Each node v *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 v picks two non-adjacent neighbors and colors them with color *one*. We call these neighbors *marked* and say v *marked* them. In this case node v becomes a T-node. Lemmas 5, 7 show that if we apply the marking process, the graph of the unmarked nodes expands regardless of the randomness in the marking process or there was a DCC. The proof is based on the previous Lemma 3. Due to the backoff distance b marked nodes cannot exist too close to each other if they are not neighbors of the same T-node. Thus, for every node that is blocked from expanding due to marked nodes, there are many other nodes that are not blocked on that level of the BFS tree. These expansion results are used in the randomized algorithms for Theorems 1, 3; in particular we use the two main results of this section Lemmas 5 and 7 in Sect. 5.2 (to bound the failure probability in a shattering type argument) and Sect. 5.4 (to bound the failure probability in a union bound type argument).

We begin with a useful lemma on the structure of BFS trees in graphs without small degree choosable components. Note that the lemma can be applied to G itself but also to the graph that one obtains by removing marked nodes from G. Lemma 2 shows that the connected components of the neighborhood of a node are cliques; the next lemma shows related but even stronger statements about the graph induced by the children in an BFS tree in such graphs. Let w be a

node of a BFS tree. We say that a maximal clique of children of w in the BFS tree is of the *odd cycle type* if it has one node that has a neighbor on the same level of the BFS tree which is not connected to w. We say a degree choosable component is *small* if it is of radius at most r.

Lemma 4 *Let G be a graph with maximum degree $\Delta \geq 3$ such that G does not contain any DCCs of radius at most r and consider an induced subgraph H of G and a depth-r BFS tree in H. Then the following hold:*

1. *If u is a node in the BFS tree that has at least two neighbors on the same level of the BFS tree, then $\Delta > 3$ and u together with all of its neighbors on the same level of the BFS tree form a clique and all these nodes have the same parent as u.*
2. *If a node u in the BFS tree is adjacent to a child of its parent w, then every neighbor of u on the same level of the BFS tree is a child of w.*
3. *Any clique of the odd cycle type consists of at most one node. If w is a node on some level $t < r$ of the BFS tree, then among its children there is at most one clique of the odd cycle type.*

Proof We separately prove the three claims.

1. Let u_1 and u_2 be two neighbors on the same level of the BFS tree and let w be the parent of u. Further, let v' be the least common ancestor of u, u_1 and u_2. If any of the edges $\{u_1, u_2\}$, $\{w, u_1\}$, and $\{w, u_2\}$ does not exist then the graph induced by the edges $\{w, u\}, \{u, u_1\}$, and $\{u, u_2\}$ and the paths $P_{v',w}$, P_{v',u_1} and P_{v',u_2} implies a small DCC that is neither a clique nor an odd cycle, a contradiction.
2. Assume that u_2 is a second neighbor on the same level of the BFS tree and let v' be the least common ancestor of u, u_1 and u_2. Assume that u_2 is not a neighbor of w. Then the nodes u, u_1, u_2 and the paths P_{v',u_1}, P_{v',u_2} induce a small DCC, a contradiction.
3. The first claim follows as (2) implies that a clique with more than two nodes cannot have a neighbor with a different parent on the same level of the BFS tree.
   To prove the second claim, assume for contradiction that there are two odd cycle type cliques (on level $t+1$ of the BFS tree) with parent w. Because of the first claim, both odd cycle type cliques consist of a single node that we denote by v_i and v_j, respectively. Let u_i (u_j) be v_i's (v_j's) neighbor on level $t+1$ that exists by the definition of a clique of the odd cycle type. Then v_i, v_j, u_i, u_j and the respective path to their least common ancestor form a DCC of radius at most r.

□

We now show, that the BFS tree around a node after the marking process has been applied expands exponentially in $\Delta - 2$ with every two hops if we do not encounter a T-node.

Lemma 5 *Let $G = (V, E)$ be a graph with maximum degree $\Delta \geq 4$ and such that G does not contain any DCCs of radius at most r, for an even r. Let H be a graph obtained from G by applying the marking process to G with $b = 6$ and removing all marked nodes, and let $v \in V(H)$ be a node in H. If $\deg_G(u) = \Delta$ for each $u \in N_r(v)$, then the r-hop BFS tree around node $v \in V$ has at least $(\Delta - 2)^{r/2}$ nodes on level r that are reachable from v through paths of lengths r consisting of unmarked nodes or this depth-r BFS tree contains a T-node reachable from v through a path of unmarked nodes.*

Proof Consider the BFS tree in H around an arbitrary node v. By Lemma 1 (applied to H) this BFS tree is unique and due to Lemma 3 (applied to H) we have $d(u) + d(u') \geq \min\{\deg_H(u), \deg_H(u')\}$ for any node u of the BFS tree and its child u'. If we encounter a node with two marked neighbors in G during this process the lemma holds as the node is a T-node. Thus we assume that any encountered node has at most one marked neighbor.

We perform an induction on the depth of the BFS tree. The root v has at least $\Delta - 1$ unmarked neighbors. Let $B_t(v)$ denote the set of nodes at distance t from v in the BFS tree in H and let $u \neq v$ be a node in $B_t(v)$. The proof is divided into two cases based on the number of children of u in the BFS tree.

1. $d(u) = 0$. *Claim: u has a marked neighbor.* As $u \neq v$ it has a parent w in the BFS-tree. If u had no marked neighbor it would have $\Delta - 1 \geq 2$ neighbors on level t of the BFS tree and its parent w on level $t-1$. By Lemma 4 u, w and these neighbors would form a clique of size $\Delta + 1$, a contradiction.
   Instead u has a marked neighbor and only $k = \Delta - 2$ neighbors on level t of the BFS tree. Let $u_1, \ldots, u_k$ be these neighbors. As $k \geq 2$ Lemma 4 implies that all these neighbors form a clique and are neighbors of w as well.
   Claim: None of the nodes $u_1, \ldots, u_k$ has a marked neighbor. Let v_T be the T-node with the marked neighbors v_{M_1} and v_{M_2} such that v_{M_1} is neighbor of u. Let $j \in \{1, \ldots, k\}$. As the distance between two T−*nodes* is at least $b = 6$ node u_j cannot have a marked neighbor other than v_{M_1} or v_{M_2}. Any u_j is not a neighbor of v_{M_2} as then $v_T, v_{M_1}, v_{M_2}, u, u_j, w$ induces a small DCC (the edge $\{v_{M_1}, v_{M_2}\}$ does not exist). If w is not a neighbor of v_{M_1} then none of the nodes u_j, $j = 1, \ldots, k$ can have v_{M_1} as a neighbor because otherwise w, v_{M_1}, u, u_j would induce a small DCC. If w is a neighbor of v_{M_1} then v_{M_1} is on the same level of the BFS tree as u and the other nodes

and, by Lemma 4, they all form a $(\Delta+1)$ clique with w, a contradiction.

Due to the claim each u_j, $j = 1, \ldots, k$ does not have a marked neighbor. We deduce that $d(u_j) = 1$ holds for $j = 1 \ldots, k$ as u_j only has $\Delta - 2$ neighbors on level t, i.e., $u, u_1, \ldots, u_{j-1}, u_{j+1}, \ldots u_k$, (any further neighbor on this level would also be part of the clique, cf. Lemma 4) and w is the only neighbor of u_j on level $t-1$. Let u'_j denote the single unmarked neighbor of u_j on level $t+1$ of the BFS tree.

Claim: Node u'_j cannot have a marked neighbor. With the notation from before u has the marked neighbor v_{M_1}. Node u'_j cannot have a marked neighbor that was created by a T-node other than v_T due to $b = 6$. If u'_j was connected to u_{M_2} the nodes $v_T, v_{M_1}, v_{M_2}, u, u_j, u'_j$ would form a small DCC. If u'_j was connected to u_{M_1} the nodes u, u_{M_1}, u, j, u'_j would form a small DCC.

We have $\deg_H(u_j) = \Delta$ and $\deg_H(u'_j) = \Delta$ and $d(u_j) = 1$ for all $j = 1, \ldots, k$. Now, we apply Lemma 3 to each u_j and each of their children u'_j and obtain that $d(u'_j) \geq \Delta - d(u_j) = \Delta - 1$, i.e., u'_j has $\Delta - 1$ unmarked neighbors on level $t+2$.

Therefore these nodes (u and its neighbors on the same level of the BFS tree) contribute a total of at least $(\Delta - 1)(\Delta - 2)$ nodes to the BFS tree at level $t+2$.

2. $d(u) \geq 1$. There are two subcases based on whether the children of u form a clique of size $\Delta - 1$ or not; here we consider the topology of the children of u including the children of u that potentially are marked.

   First assume that they do form a clique of size $\Delta - 1$. Each node in the clique has $\Delta - 2$ nodes on the same level of the BFS tree, one parent and one further unique neighbor (on the next level) that potentially is marked. Due to the uniqueness of the BFS tree (cf. Lemma 1) all these further neighbors are distinct. At most one of the nodes of the clique can have its subtree blocked because it is marked itself or because its unique child is marked. In any case there are at least $\Delta - 2$ nodes in the clique that each have one distinct unmarked child on level $t+2$ of the BFS tree.

   Now, assume that the children of u do not form a $(\Delta-1)$-clique (this paragraph is still considering the topology including children of u that potentially are marked). Instead, let $C_1, C_2, \ldots, C_k$ form the maximal cliques formed by the children of u (cf. Lemma 2). Recall that a clique is of the odd cycle type if it is of size one and closes an odd cycle, i.e., it has 1 edge to another node on the same level that is not a neighbor of its parent u. Also recall that if $|C_i| \geq 2$, there are no edges from C_i to other nodes on the same level of the BFS tree (cf. Lemma 4).

   Let C_i be a clique that is not of the odd cycle type. Then the only edges of node of the clique C_i to nodes on level $t+1$ are the $|C_i| - 1$ edges inside C_i. Furthermore, each vertex of the clique has one edge to level t (to u). Thus we obtain that any u' in such a clique has $d(u') \geq \deg_H(u') - (|C_i| - 1) - 1 \geq \Delta - 1 - (|C_i| - 1) - 1 = \Delta - |C_i| - 1$. Thus any such clique contributes

$$\sum_{u' \in C_i} d(u') \geq |C_i|(\Delta - |C_i| - 1)$$

   unmarked nodes on level $t+2$ of the BFS tree. This value is always greater than $\Delta - 2$. Thus as soon as there is a clique that is not of the odd cycle type there are at least $\Delta - 2$ unmarked nodes on level $t+2$ of the BFS tree. Now, assume that all cliques are of the odd cycle type. Then, there is actually just one clique due to Lemma 4, (3). Thus $d(u) = 1$ holds. Let u' be the only node of the single clique of odd cycle type. Lemma 3 implies that $d(u') \geq \min\{\deg_H(u'), d_H(u)\} - d(u) \geq \Delta - 2$, i.e., u' has at least $\Delta - 2$ unmarked children on level $t+2$ of the BFS tree.

To bound the total number of nodes in $B_{t+2}(v)$ let x denote the number of nodes $u \in B_t(v)$ of the first type, that is, with $d(u) = 0$. The analysis for those nodes contributes $(\Delta - 1)(\Delta - 2)$ nodes on level $t+2$ of the BFS tree but also uses the other $(\Delta - 2)$ nodes on level t of the BFS tree that form a clique with u. Thus there are at least $|B_t(v)| - x(\Delta - 1)$ nodes $\tilde{u} \in B_t(v)$ of the second type, that is, nodes with $d(\tilde{u}) \geq 1$, for which we can independently count their contribution of at least $(\Delta - 2)$ nodes on level $t+2$ per node of the second type. Thus the total number of nodes on level $t+2$ can be bounded by

$$\begin{aligned} |B_{t+2}(v)| &\geq x(\Delta-1)(\Delta-2) \\ &\quad + (|B_t(v)| - x(\Delta-1))(\Delta-2) \\ &= (\Delta-2)|B_t(v)|. \end{aligned}$$

□

If $\Delta = 3$ an expansion that is exponential in $\Delta - 2 = 1$ is not useful and we cannot guarantee that we expand by a factor of $\Delta - 1$ every two hops after the marking process has been applied. Instead we prove that we expand by a factor of $\Delta - 1$ every six hops or encounter a T-node. We first need a simple but useful lemma.

Lemma 6 *Let $\Delta \geq 3$ and consider a depth-r BFS tree with root v in some graph without DCCs of radius at most r. Then the following hold:*

1. *Any node of the BFS tree at distance at most $r-1$ from the root and with degree Δ in G has at least one child in the BFS tree.*

2. *Let u be a node of the BFS tree and $k + dist(v, u) \leq r$. If all nodes of the k-hop subtree rooted at u have degree Δ and are unmarked then the subtree contains at least $(\Delta - 1)^{\lfloor k/2 \rfloor}$ unmarked nodes on level k and all of them are reachable from u through paths of length at most k consisting of unmarked nodes.*

Proof 1. This is trivially satisfied for the root of the BFS tree. Let u be a node that is not the root of the BFS tree and let w be its parent in the tree. To prove the claim we only need to show that u cannot have $\Delta - 1$ neighbors on the same level of the BFS tree. For contradiction, assume that it has neighbors $u_1, \ldots, u_{\Delta-1}$ on the same level of the tree. Due to Lemma 4 all these nodes must be also connected to w which implies that $u, w, u_1, \ldots, u_{\Delta-1}$ form a $(\Delta + 1)$-clique, a contradiction.

2. If $u = v$ the result follows with Lemma 5. If $u \neq v$ let $B_t(u)$ be the nodes on level t of the BFS tree rooted at u. Due to (1) we have $1 \leq d(w) \leq \Delta - 1$ for all nodes the BFS tree rooted at u. The statement holds trivially for $k = 0$ and we show the result by induction on k. Now consider some level k. For each node $w \in B_k(u)$, we get via Lemma 3 (we can apply the lemma as there are no marked nodes) that the number of descendants of w in $B_{k+2}(u)$ is at least

$$\sum_{w' \in N_+(w)} d(w') \geq d(w)(\Delta - d(w)) \geq \Delta - 1.$$

Since each node has a unique ancestor in the BFS tree, we get that
$|B_{k+2}(v)| \geq (\Delta - 1)|B_k(v)|$. □

Lemma 7 *Let $\Delta = 3$ and $G = (V, E)$ be a graph such that G does not contain any DCCs of radius at most r, for an r divisible by 6. Apply the marking process to G with $b = 15$. Let $v \in V$ be an arbitrary unmarked node. If $\deg_G(u) = \Delta$ for each $u \in N_r(v)$ then the r-hop connected component of unmarked nodes around v contains a T-node or at least $4^{r/6}$ nodes.*

Proof We consider the tree $\mathfrak{T}$ around v that one obtains by cutting the BFS tree around v in G whenever a marked node is encountered. Let $\mathfrak{T}_t(v)$ denote the nodes on level t of $\mathfrak{T}$. We show by induction on t that $\mathfrak{T}_t(v)$ contains at least $4^{t/6}$ nodes if we do not encounter a T-node (note that the marked nodes that make a node of $\mathfrak{T}_t(v)$ a T-node are not part of $\mathfrak{T}_t(v)$). If we encounter a node with two marked neighbors during the induction the lemma holds as it forms a T-node. Thus we assume that any encountered node has at most one marked neighbor.

Base case: There are at least 4 *unmarked nodes on level 6 of* $\mathfrak{T}$. Due to $b = 15$ at most two of the 5-hop subtrees rooted at the children of v are cut off due to a marked node. The remaining 5-hop subtree rooted at the third child cannot be cut off due to a marked node. Thus it contributes at least $(\Delta - 1)^2 = 4$ (unmarked) nodes on level 6 of the tree $\mathfrak{T}$ due to Lemma 6, (2).

Inductive step. Consider the set $\mathfrak{T}_t(v)$ for some $t \geq 6$. We split the proof in three cases: for each node $u \in \mathfrak{T}_t(v)$ either $d(u) = 2$, $d(u) = 1$, or $d(u) = 0$ holds where $d(u)$ denotes the number of children of u in the tree $\mathfrak{T}$.

Case $d(u) = 2$. As u has a parent and two children, say u_1 and u_2, in $\mathfrak{T}$ it does not have a marked neighbor. Due to $b = 15$ there can be at most two marked nodes in the 6-hop subtree rooted at u and both stem from the same T-node. If the 5-hop subtree of any of u's children has no marked nodes it contributes at least 4 unmarked nodes to $\mathfrak{T}_{t+6}$ due to Lemma 6, (2). If both 5-hop subtrees rooted at u_1 and u_2 have a marked node, say u_{M_1} and u_{M_2}, then they stem from the same T-node v_T. Then $\{v_T, v_{M_1}\}, \{v_T, v_{M_2}\}, P_{u,v_{M_1}}, P_{u,v_{M_2}}$ must induce an odd cycle as otherwise it induces a small DCC, in particular, the edge $\{u_1, u_2\}$ does not exist. Thus u_1 has a neighbor u_1' that lies not on the path $P_{u,v_{M_1}}$ and the 4-hop subtree rooted at u_1' does not contain a marked node. Further, note that the 4-hop subtree rooted at u_1' is part of the subtree rooted at u as u_1' must be a child of u_1 in $\mathfrak{T}$: Assume u_1' is not a child of u_1, that is it is on level $t+1$ of the BFS tree. Then $\{v_T, v_{M_1}\}, \{v_T, v_{M_2}\}, P_{u,v_{M_1}}, P_{u,v_{M_2}}$ and $P_{v',u_1'}, P_{v',u}$, $\{u_1, u_1'\}$ induce a small DCC where v' is the least common ancestor of u and u_1'. Hence Lemma 6, (2) implies that the 4-hop subtree rooted at u_1' contributes at least 4 unmarked nodes to $\mathfrak{T}_{t+6}$.

Case $d(u) = 1$. First consider the case that u has a marked neighbor: Let u' be the single child of u in $\mathfrak{T}$. If the 5-hop subtree rooted at u' does not contain a marked node it contributes at least 4 nodes to $\mathfrak{T}_{t+6}(v)$ due to Lemma 6, (2). So assume that it contains a marked node v_{M_2} and let v_{M_1} be the marked neighbor of u. Note that $v_{M_1} \neq v_{M_2}$ as the subtree rooted at u' does not contain v_{M_1}. Then u' cannot be a neighbor of v_{M_1} as otherwise $D := \{u, v_{M_1}\}, \{v_{M_1}, v_T\}$, $\{v_T, v_{M_2}\}$, and $P_{u',v_{M_2}}$ induces a small DCC. u' can also not have another neighbor u'' on level $t + 1$ of the BFS tree as then D together with $\{u', u''\}$, $P_{v',u}$, $P_{v',u''}$ induces a small DCC where v' is the least common ancestor of u and u''. Thus u' has a further child w that does not lie on the path $P_{u',v_{M_2}}$ and the 4-hop subtree rooted at w does not contain any marked nodes. Hence Lemma 6, (2) implies that this subtree contributes at least 4 nodes to $\mathfrak{T}_{t+6}$.

If u has no marked neighbor, the 6-hop subtree rooted at u may contain a marked node. If the subtree does not contain a marked node, it contributes at least $(\Delta - 1)^3 = 8$ nodes to $\mathfrak{T}_{t+6}$ due to Lemma 6, (2). If the subtree rooted at u contains a marked node let u' be the unique neighbor on level t that u must have. We show that the 6-hop subtree

rooted at u' does not contain a marked node: Let v_{M_1} be the marked node in the subtree of u created by T-node v_T. The subtree rooted at u' cannot contain v_{M_1} as otherwise $P_{u,v_{M_1}}$, $P_{u',v_{M_1}}$, $\{u, u'\}$, $P_{v',u}$, $P_{v',u'}$ induces a small DCC where v' is the least common ancestor of u and u'. The subtree also cannot contain a marked node that stems from any T-node other than v_T due to $b = 15$. Thus let v_{M_2} be the other marked node that v_T created. Node v_{M_2} cannot be a node in the 6-hop subtree rooted at u' because then $P_{u,v_{M_1}}, \{v_{M_1}, v_T\}$, $\{v_{M_2}, v_T\}$, $P_{u',v_{M_2}}, \{u, u'\}$, $P_{v',u}$, $P_{v',u'}$ induces a small DCC where v' is the least common ancestor of u and u'. Hence the 6-hop subtree rooted at u' does not contain any marked nodes and contributes 8 nodes to $\mathfrak{T}_{t+6}(v)$ due to Lemma 6, (2). Therefore u and u' together contribute at least 8 nodes to $\mathfrak{T}_{t+6}(v)$.

Case $d(u) = 0$. Node u must have one marked neighbor (on level $t + 1$), one parent and one unmarked neighbor u' on level t of the BFS tree. The 6-hop subtree rooted at u' does not contain a marked node: Assume that it does contain a marked node v_{M_2} and let v_{M_1} be the marked neighbor of u. First note that $v_{M_1} \neq v_{M_2}$ and let v_T be the T-node that marked both nodes. Let v' be the least common ancestor of u and u'. Then $P_{v',u}$, $P_{v',u'}$, $\{u, u'\}$, $\{u, v_{M_1}\}$, $\{v_{M_1}, v_T\}$, $\{v_T, v_{M_2}\}$, and $P_{v_{M_2},u'}$ induces a small DCC, a contradiction.

Then Lemma 6, (2) implies that the 6-hop subtree rooted at u' contributes at least $(\Delta - 1)^3 = 8$ unmarked nodes on level $t + 6$, that is, u and u' together contribute at least 8 nodes.

In every case we obtain at least 4 unmarked reachable nodes on level $\mathfrak{T}_{t+6}$ per node on level $\mathfrak{T}_t$, which proves the claim. □

2.4 A simplified proof for the distributed Brooks' theorem

Panconesi and Srinivasan proved a distributed version of Brooks' Theorem (cf. Theorem 4). 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 5 as it does not include the marking process.

Lemma 8 *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 Δ. Then for each even r there are at least $(\Delta - 1)^{r/2}$ nodes at distance r from v.*

Proof Consider the BFS tree around node v. The 1-hop neighborhood of v consists of Δ nodes that form disjoint cliques due to Lemma 2. As not all neighbors can form a single clique each such neighbor has at least one edge to level two of the BFS tree. This implies that $|B_2(v)| \geq \Delta \geq \Delta - 1$.

Now consider some level t. For each node $u \in B_t(v)$, we get via Lemma 3 that the number of descendants of u in $B_{t+2}(v)$ is at least

$$\sum_{u' \in N_+(u)} d(u') \geq d(u)(\Delta - d(u)) \geq \Delta - 1.$$

Since each node has a unique ancestor in the BFS tree, we get that

$$|B_{t+2}(v)| \geq (\Delta - 1)|B_t(v)| \qquad \square$$

Now we can use previous lemmas to show that the uncolored node in the statement of Theorem 4 can fix its color as it sees a degree-choosable component or a node of degree $< \Delta$ inside its $O(\log_\Delta n)$-neighborhood.

Lemma 9 *Let G be a graph with maximum degree $\Delta \geq 3$. The $(2\log_{\Delta-1} n)$-neighborhood of any node contains a degree-choosable component or it contains a node of degree smaller than Δ.*

Proof Fix a node $v \in V(G)$ and assume that its $r = 2\log_{\Delta-1} n$ neighborhood does not contain a degree-choosable component and that nodes in this neighborhood have degree Δ. By Lemma 8, the BFS tree has $|B_r(v)| \geq (\Delta - 1)^{r/2} \geq n$ 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 $B_r(v)$. □

Now we are ready for the proof of the distributed Brooks' theorem.

Proof of Theorem 4 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 v does not have a free color, that is, all of its Δ neighbors have Δ different colors, then we can move the token to u and color the node v with color $c(u)$. If v has a free color, it can choose that color and the token is eliminated. Now, if the $(2\log_{\Delta-1} n)$-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 9, there exists a degree-choosable component in the $(2\log_{\Delta-1} n)$-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 B. By definition there exists a Δ-coloring of B compatible with the existing coloring in the rest of the graph. □

Remark 1 Theorem 4 implies an $SLOCAL(O(\log_\Delta n))$ algorithm (see [21] for the model). This combined with [21, Theorem 1.11] immediately implies the existence of a randomized polylogarithmic round algorithm for Δ-coloring. Note that [31] explicitly gives such an algorithm and we provide a faster randomized algorithm in Theorem 1.

3 Algorithmic preliminaries and notation

Given a subset $C \subseteq V$ of nodes of a graph $G = (V, E)$, C has *weak diameter* d if $d_G(v, w) \leq d$ for all $v, w \in C$.

Definition 4 (Network Decomposition [3]) A weak $\big(d(n), c(n)\big)$-*network-decomposition* of an n-node graph $G = (V, E)$ is a partition of V into clusters such that each cluster has weak diameter at most $d(n)$ and the cluster graph is properly colored with colors $1, \ldots, c(n)$.

One can compute a decomposition with $d(n), c(n) = 2^{O(\sqrt{\log n})}$ in $2^{O(\sqrt{\log n})}$ rounds [30].

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 6 [18] + [5] *There is a deterministic distributed algorithm that solves the* $(\deg+1)$ *list coloring problem in time* $O\big(\sqrt{\Delta \log \Delta} \cdot \log^* \Delta\big)$ *given a* $O(\Delta^2)$ *coloring of the graph.*

By iterating through the color classes and greedily picking colors we obtain the following.

Theorem 7 [30] *Given a weak* $\big(d(n), c(n)\big)$-network-decomposition *one can solve the* $(\deg+1)$ *list coloring problem in time* $O(c(n) \cdot (d(n) + 1))$. *In particular* $(\deg+1)$ *list coloring can be solved in* $2^{O(\sqrt{\log n})}$ *rounds.*

Theorem 8 (List Coloring [20]) *There is a randomized distributed algorithm that solves the* $(\deg+1)$*-list coloring problem in* $O\big(\log \Delta + 2^{O(\sqrt{\log \log n})}\big)$ *rounds.*

For some graph G and an integer $R \geq 1$, we define $G^R = (V(G), \{\{u, v\} \mid \text{dist}_G(u, v) \leq R\})$. An (α, β) ruling set of a graph G is a subset $M \subseteq V(G)$ of the nodes such that $\text{dist}(v, M \setminus \{v\}) \geq \alpha$ for all $v \in M$ and $\text{dist}(v, M) \leq \beta$ for all $v \in V$. If $\alpha = 1$, we also omit the parameter speak of β*-ruling sets*. Usually, when computing a ruling set it comes with a so called *ruling forest*.

Definition 5 (Ruling Forest, [3]) Given a graph $G = (V, E)$ and a set $V' \subseteq V$, we say that a forest $F_r = (V_r, E_r)$ with $V_r \supseteq V'$ is an (α, β)*-ruling forest* with respect to V' if the following conditions holds:

1. The root of the trees, i.e., the connected components of F_r, are in V',
2. the distance in G between any two roots is $\geq \alpha$,
3. the depth of each tree in the forest is $\leq \beta$.

Note that by definition the forest F_r spans its vertex set V_r. The following lemma summarizes the known distributed ruling set algorithms that we use.

Lemma 10 *For any integers* $k, \beta \geq 1$ *there are the following ruling set algorithms.*

1. $(2, \beta)$ *in time* $O(\beta\Delta^{2/\beta} + \log^* n)$ [34]
2. $(k, k^2\beta)$ *in time* $O(k^2 \cdot \beta\Delta^{2/\beta} + k \cdot \log^* n)$ [34] + [7]
3. $(2, O(\log \log n))$ *in time* $O(\log \log n)$ [19] + [34]
4. $(2, \beta)$ *in time* $O(\log^{1/\beta} \Delta) + 2^{O(\sqrt{\log \log n})}$ [20]
5. $(2, 1)$ *in time* $2^{O(\sqrt{\log n})}$ [3,30]

Algorithms (1), (2), and (5) are deterministic, algorithms (3) and (4) are randomized.

Proof We only provide a proof for the deterministic $(k, k^2\beta)$ algorithm which is alike the words in [7, Section 1.1]. Consider G^{k-1} and note that the maximum degree of G^{k-1} can be upper bounded by Δ^k. Then apply Lemma 10 (1) on G^{k-1} with $\beta' = k\beta$. Note that each step of the algorithm can be executed in k steps in G leading to a running time of

$$O(k \cdot (\beta'\Delta^{2k/\beta'} + \log^* n)) = O(k^2 \cdot \beta\Delta^{2/\beta} + k \cdot \log^* n).$$

□

4 Deterministic Δ-coloring (Theorem 3)

In this section we present our deterministic Δ-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 $B_1, \ldots, B_s$ where B_i consists of the nodes in distance i to B_0. This is particularly helpful for Δ-coloring as we can Δ-color the layers in reverse order while respecting the colored neighbors in layers with a larger index. To Δ-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 $B_{i+1} \cup \ldots \cup B_s$ from the set $\{1, \ldots, \Delta\}$. The size of this list is at least $\deg_{G[B_i]}(v) + 1$ as v has one neighbor in layer B_{i-1}. Then layer B_0 is colored after all other layers with different techniques as Δ-coloring B_0 while respecting already colored neighbors might not be a $\deg+1$ list coloring instance. To make sure that we can still Δ-color B_0 efficiently (we might have to recolor previously colored nodes) it has to be chosen carefully. Also consult Fig. 4 for an illustration of the technique.

The deterministic algorithm in this section uses the layering technique in a simple setting with $O(\log^2 n)$ layers. The layer B_0 is chosen to be a ruling set in which nodes have large distances such that Theorem 4 can be applied to color the nodes in B_0 independently and in parallel. The total running time is dominated by the $O(\log^2 n)$ iterations of list coloring due to the layering technique.

Algorithm. First, color all nodes of G with $O(\Delta^2)$ colors with Linial's algorithm [25]. These colors are only used for symmetry breaking when applying list coloring algorithms

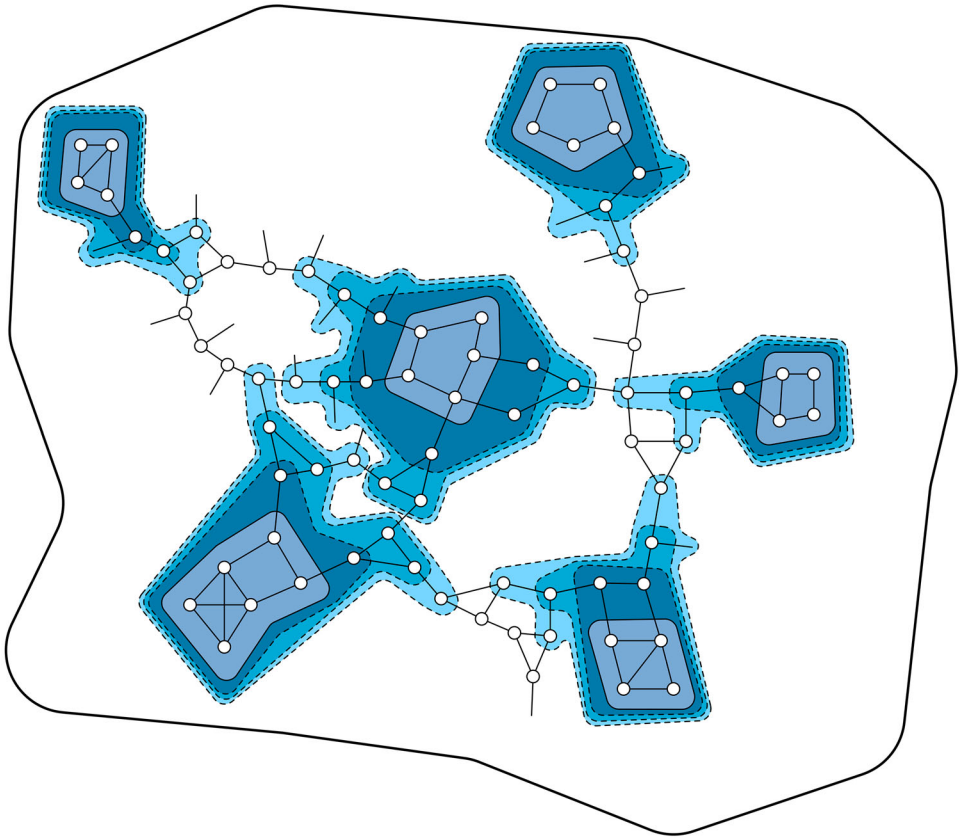


Fig. 4 The layering technique: First, the base layer is removed, then neighbors of the base layer are removed, then their neighbors and so on. Attention! The base layer in the illustrated example is build as in our main algorithm in Sect. 5 and consists of degree choosable components. To illustrate the technique in the setting of Sect. 4 simply assume that each component of the base layer consists of a single node. This illustration only shows one connected component of the base layer and in general the illustrated layer building starts at several places of the graph (wherever nodes are in the base layer) at the same time

and do not coincide with the desired Δ-coloring. Let $R := 4\log_{\Delta-1} n + 1 \leq 7\log n/\log\Delta$ and $z = 4 \cdot R^2$.

1. (*Build layer* B_0) Compute a (R, z) ruling forest of G with Lemma 10, (2). Add all nodes of the ruling set to layer $B_0 \subseteq V$.
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* $B_z, \ldots, B_1$) Add the layers $B_z, \ldots, B_1$ to the graph one by one: When adding layer B_i color the nodes of B_i such that $G_{\uparrow i} = G[\bigcup_{j=i}^{z} B_j]$ is validly Δ-colored. Step $i = z, \ldots, 1$ 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 6 to solve each list coloring instance.
4. (*Color layer* B_0) Use Theorem 4 to independently color the nodes in B_0 through recoloring nodes within distance at most $2\log_{\Delta-1} n < R/2$.

Proof of Theorem 3 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 $G_{\uparrow i}$ is validly Δ-colored. Pick a node $v \in B_i$. The list of available colors of v is $\{1, \ldots, \Delta\} \setminus F_v$ where F_v is the set of colors that have already been chosen by v's colored neighbors in $G_{\uparrow i}$. The size of F_v is at most $\deg_{G_{\uparrow i}}(v) - \deg_{G_i}(v)$. The degree $\deg_{G_{\uparrow i}}(v)$ is upper bounded by $\Delta - 1$ because at least one of v's neighbors in G is contained in B_{i-1}. Thus the list of available colors of v has size at least $\Delta - |F_v| \geq \Delta - (\deg_{G_{\uparrow i}}(v) - \deg_{G_i}(v)) \geq \deg_{G_i}(v) + 1$.

The running time of the first and second step is $O\big(R^2 \cdot \sqrt{\Delta} + R \cdot \log^* n + z + \log^* n\big) = O\big(R^2 \cdot \sqrt{\Delta}\big)$. The third step takes $O(\sqrt{\Delta \log \Delta} \log^* \Delta)$ rounds for each of the $z = O(R^2) = O(\log^2_\Delta n)$ iterations. The fourth step takes $O(R)$ rounds. In total, the running time is dominated by the third step. □

The following theorem appeared as [31, Theorem 5]. Our techniques can be used to give an alternative proof.

Theorem 9 ([31], rephrased, reproved) *Nice graphs can be Δ-colored deterministically in the distributed model of computation in $2^{O(\sqrt{\log n})}$ rounds.*

Algorithm.

1. (*Build layer* B_0) Define R as $R = 4\log_{\Delta-1} n + 1 \leq 7\log n/\log\Delta$ and compute a $\big(2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})}\big)$ net-

work decomposition of G^R with [30]. Then compute an $(R, R+1)$ ruling set with the help of the network decomposition in $2^{O(\sqrt{\log n})}$ rounds. Let B_0 be the nodes in the ruling set.

2. (*Remove layers* $B_0, \ldots, B_z$) Define layers $B_1, \ldots, B_z$ where $z = R+1$ and $v \in B_i$ if the distance of v to B_0 is i. Remove all layers from the graph.
3. (*Color layers* $B_z, \ldots, B_1$) Add the layers $B_z, \ldots, B_1$ to the graph one by one: When adding layer B_i color the nodes of B_i such that $G_{\uparrow i} = G[\bigcup_{j=i}^{z} B_j]$ is validly Δ-colored. Step $i = z, \ldots, 1$ 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}. Use the network decomposition to solve the list colorings (note that a network decomposition of G^R is also a network decomposition of G with an R factor increase in the diameter of the clusters).
4. (*Color layer* B_0) Use Theorem 4 to independently color the nodes in B_0 through recoloring nodes in distance at most $2\log_{\Delta-1} n < R/2$.

Proof The proof of correctness is along similar lines as the proof of Theorem 3. The running time of the third step dominates and is $O\left(z \cdot 2^{O(\sqrt{\log n})}\right) = 2^{O(\sqrt{\log n})}$. □

5 Randomized Δ-coloring (Theorems 1 and 2)

The randomized algorithm is split into two slightly different versions based on Δ: one version can handle any $\Delta \geq 4$ and the other any $3 \leq \Delta = O(1)$. We refer to these two versions as the *large*-Δ version and the *small*-Δ version. In this section we present the algorithms of Theorems 1, 2 and their proofs. We recommend to read the algorithms in a top-down manner beginning with the captions of the respective parts.

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 $D_0, \ldots, D_\alpha$) 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.[2] 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: Phase I 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-Δ case we prove by union bound that with high probability no node remains after this process. In the large-Δ 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 $D_0, \ldots, D_\alpha$ (cf. Phase (6) in Sect. 5.1).

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 independent DCCs and we can color the components of B_0 at the very end.

5.1 The randomized Δ-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 9). 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 $b = 6$ in the large version and $b = 15$ in the small version. Set $p = \Delta^{-b}$.

First, color all nodes of G with $O(\Delta^2)$ colors with Linial's algorithm [25]. These colors are only used for symmetry breaking when applying deterministic list coloring algorithms and they in no way coincide with the desired Δ-coloring.

I Removing Degree Choosable Components and Layers around them

(1) Each node that is contained in at least one degree choosable subgraph with radius at most r selects one such subgraph. Let $\mathcal{G}_{DCC}$ be the virtual graph that has a node for each selected degree choosable subgraph, and two subgraphs in $V(\mathcal{G}_{DCC})$ are connected in $\mathcal{G}_{DCC}$ by an edge if they share a vertex or if they are connected by an edge in G. The graph $\mathcal{G}_{DCC}$ has at most n nodes as every node adds at most one degree choosable compo-

[2] *Dense* in the sense that the part has a small DCCs that by definition can be colored easily after everything else is colored. The term *dense* is inspired as the expansion results in Sect. 2 show that parts without DCCs are in some sense *sparse*.

nent, maximum degree at most $\Delta^{2r+1} \leq \Delta^{3r}$, and one round of a distributed algorithm in it can be simulated in $O(r)$ rounds in G.
Runtime for large Δ: $O(r) = O(1)$.
Runtime for small Δ: $O(r) = O(\log\log n)$.

(2) (*Build layer* B_0) Find a $(2, \beta)$ ruling set M of $\mathcal{G}_{DCC}$ with $\beta = 6 \cdot r$. Add each node $v \in V(G)$ that is contained in a DCC $C \in M$ to the base layer B_0.
Runtime for large Δ: With Lemma 10, (4) $O\big(\log^{1/\beta}\Delta + 2^{O(\sqrt{\log\log n})}\big)$.
Runtime for small Δ: With Lemma 10, (3) $O(r \cdot (O(\log\log n))) = O(\log^2\log n)$.

(3) (*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 distance i (measured in G) from a node in B_0. Remove all layers $B_0, \ldots, B_s$ from the graph.
Runtime for large Δ: $O(s) = O(\beta \cdot r) = O(1)$.
Runtime for small Δ: $O(s) = O(\beta \cdot r) = O(\log^2\log n)$.

Note that (besides potentially some other nodes) all nodes that are in a degree choosable component with radius at most r are removed from the graph after phase (3).

II Shattering of the Remaining Graph

(4) (*Random T-node creation*) Consider the remaining graph

$$H = G \setminus \Big(\bigcup_{i=0}^{s} B_i\Big) .$$

Each node of H becomes *selected* independently with probability p. Then, if there is another selected node within distance b, both become unselected. If not, the selected node picks a random pair of non-adjacent neigbors and colors them with color *one*. We call these neighbors *marked*.
Runtime: $O(1)$.

(5) (*Remove layers* $C_0, \ldots, C_{2r}$) We call a node *happy* if it has an uncolored path to a T-node in its r-neighborhood. By this definition we assign each happy node to its closest T-node in its r-neighborhood.
We define the *boundary* of graph H as the set of nodes with degree less than Δ in H. Nodes that are colored and have distance at most r steps away from the boundary now remove their color and each such node is assigned to its closest boundary node, breaking ties using identifiers. It might happen that a node v that is $r \leq \ell < 2r$ steps away from the boundary was assigned to a node w that is at most $r-1$ steps away from the boundary. Due to the uncoloring w might not be a T-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 the nodes of H that are at distance i from their respective assigned node. The layer C_0 consists of T-nodes, and all nodes that have degree $< \Delta$ in H. Remove the layers $C_0, \ldots, C_{2r}$ and the marked nodes from the graph.
In Sect. 5.4 we show that in the algorithm for small Δ, all nodes are removed after this phase with high probability. Hence this algorithm proceeds directly to Phase (7).
Runtime for large Δ: $O(r) = O(1)$.
Runtime for small Δ: $O(r) = O(\log\log n)$.

(6) (*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 Sect. 5.2 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 13) implies that L consists of small connected components of size at most $N := \mathsf{poly}\,\Delta \cdot \log_\Delta n$.
Section 5.3 explains in detail how to color these small components if $\Delta \geq 4$. The core idea is to again handle the small components by constructing layers $D_0, \ldots, D_\alpha$ where $\alpha = O(\log^2\log n)$. 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 5) that it sees the whole component in $O(\log_\Delta N) = O(\log\log n)$ hops, a contradiction.
Section 5.4 shows that for $\Delta = 3$ this step can be omitted and L does not contain any vertices.
Runtime for large Δ: $2^{O(\sqrt{\log\log n})}$ via Lemma 16.
Runtime for small Δ: $O\left(\sqrt{\Delta\log\Delta}\log^*\Delta \cdot \log^2\log n\right)$ via Lemma 16

III Color Happy Nodes From the Shattering Process

(7) (*Color layers* $C_{2r}, \ldots, C_0$) Assume that the remaining small components are colored with Δ colors in Phase (6). Go through the layers $C_{2r}, \ldots, C_0$ grown in step (5) in reverse order and Δ-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 $i = 2r, \ldots, 1$ each node has a neighbor at a lower level and the nodes in C_0 have two neighbors of the same color.

Runtime for large Δ: $O(\log\Delta + 2^{O(\sqrt{\log\log n})})$ with using Lemma 8 $2r = O(1)$ times.
Runtime for small Δ: $O(\log\log n(\sqrt{\Delta\log\Delta}\cdot\log^*\Delta))$ with using Theorem 6 $2r$ times.

IV Color Degree Choosable Components and Layers around them

(8) (*Color layers* $B_s,\ldots,B_1$) Go through the layers $B_s,\ldots,B_1$ grown in step (3) and color each layer with Δ 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 Δ: $O(\log\Delta + 2^{O(\sqrt{\log\log n})})$ with using Lemma 8 $s = O(1)$ times.
Runtime for small Δ: $O(\log\log n(\sqrt{\Delta\log\Delta}\cdot\log^*\Delta))$ with using Theorem 6 s times.

(9) (*Color layer* B_0) Each connected component of layer B_0 corresponds to one DCC in M (selected in step (2)). Thus each connected component of B_0 is Δ-list colorable and of radius $\leq r$. We find a coloring by brute forcing each component independently.
Runtime for large Δ: $O(r) = O(1)$.
Runtime for small Δ: $O(r) = O(\log\log n)$.

Proof of Theorem 1 The runtime follows from summing up the runtimes for small Δ of all phases and is dominated by the runtime of phases (7) and (8) in which we need to solve $O(r+s) = O(\log^2\log n)$ list coloring instances in $O(\sqrt{\Delta\log\Delta}\log^*\Delta)$ rounds each. Solving the small components in phase (6) also has a significant contribution and can be done in $O(\sqrt{\Delta\log\Delta}\log^*\Delta\cdot\log^2\log n)$ rounds via Lemma 16 if $\Delta\geq 4$. If $\Delta = 3$ then Sect. 5.4 shows that phase (6) can be omitted as L is the empty graph. The ruling set in phase (2) can be found in $O(\log^2\log n)$ rounds. All other phases take at most $O(r+s) = O(\log^2\log n)$ rounds.

All nodes are colored at the end because any nodes that is in none of the layers

$$B_0,\ldots,B_S,C_0,\ldots,C_{2r},C'$$

is contained in a 'small component' and colored in phase (6), w.h.p. In Sects. 5.2 and 5.3 we prove that this is indeed the case for $\Delta\geq 4$. For $\Delta = 3$ we show in Sect. 5.4 that the aforementioned layers already contain all vertices of the graph w.h.p, that is, w.h.p. the graph L defined in phase (6) does not have any vertices. □

Proof of Theorem 2 The runtime follows from summing up the runtimes for large Δ of all phases and is dominated by the runtime of phases (7) and (8) in which we need to solve $O(r+s) = O(1)$ list coloring instances in $O(\log\Delta) + 2^{O(\sqrt{\log\log n})}$ rounds. Solving the small components in phase (6) also has a significant contribution and can be done in $2^{O(\sqrt{\log\log n})}$ rounds via Lemma 16. The ruling set in phase (2) can be found in $O(\log\Delta)+2^{O(\sqrt{\log\log n})}$ rounds. All other phases take $O(r) = O(1)$ rounds.

All nodes are colored at the end because any nodes that is in none of the layers

$$B_0,\ldots,B_S,C_0,\ldots,C_{2r},C'$$

is contained in a 'small component' and colored in phase (6), w.h.p. In Sects. 5.2 and 5.3 we prove that this is indeed the case. □

5.2 Shattering of the remaining graph (phases (4)–(6))

In this section we show that the process of phase (4) and (5) produces a graph with remaining components of size $O(\mathsf{poly}(\Delta)\cdot\log n)$. In Section 5.3 we show how to color these small components fast. The nodes that are put into the layers in phase (5) are colored later in phase (7).

For a node v let $\mathcal{E}_v$ be the event that v is removed in the graph in phases (4)–(5). Let t be the radius such that the event $\mathcal{E}_v$ only depends on the random bits of nodes in radius t of v. The standard shattering technique (cf. Lemma 13) shows that the connected components of non-removed nodes are *small* if the probability of $\overline{\mathcal{E}_v}$ is upper bounded by $1/\mathsf{poly}(\Delta)$ where the polynomial depends on the radius t.

To show that the probability of $\overline{\mathcal{E}_v}$ 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 $1/\mathsf{poly}(\Delta)$ 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 5 the BFS tree around a node expands deterministically even after the marking process which implies the next lemma.

Lemma 11 *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. $|S_v| \geq (\Delta-2)^{t/2}\cdot\Delta^{-6}$,
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 $1/3\cdot p(1-p)^{\Delta^6}$. The events are independent for distinct $u\in S_v$,*

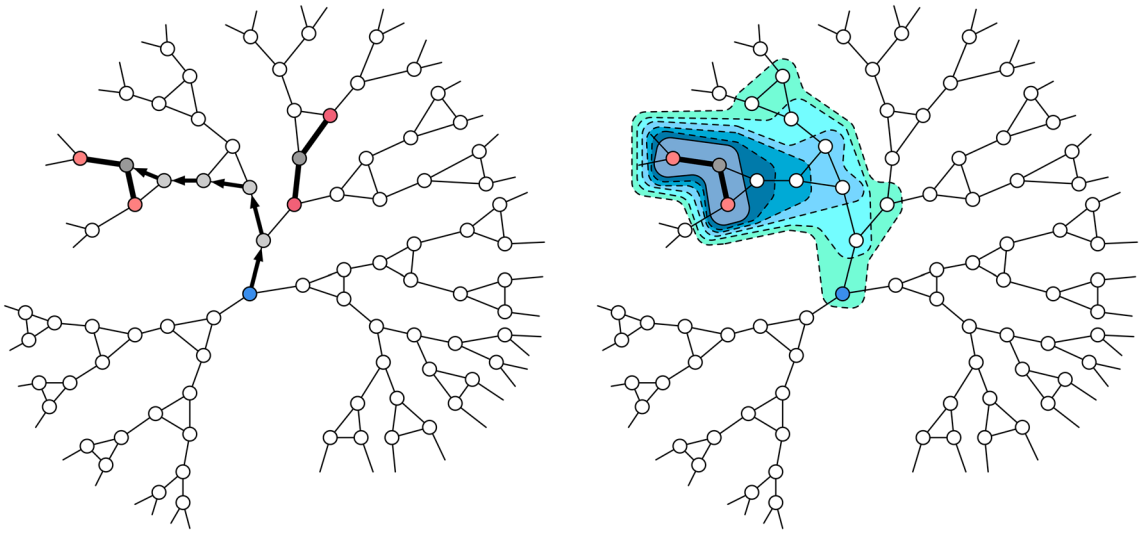


Fig. 5 The creation of T-nodes in Phase (4). Note that the top right T-node is not reachable from the blue node in the center through a path of unmarked nodes. To be removed in the layer creation around the T-nodes it is essential that a T-node is reachable through an unmarked path. The image is also helpful to get an understanding of how the locally Gallai tree like graph H looks like (a tree of cliques and odd cycles). However, note that certain edges are left out to simplify the illustration. In the right hand side illustration you can see how the T-node is removed from the layer creation around the T-node because it has a short path of unmarked nodes to the T-node node.

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 For a fixed node and due to Lemma 5 the BFS tree around v restricted to unmarked nodes contains at least $(\Delta-2)^{t/2}$ nodes on level t or we encounter a T-node. 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_v 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 Δ^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 5) 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 $(1-p)^{\Delta^6}$. 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 11 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 12 (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 $1-\left(\frac{1}{\Delta}\right)^{4r+4}$ using only the randomness in its t neighborhood. The constant r is independent from the graph (including its size).*

Proof Let v be a node in H. Apply Lemma 11 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 $1/3 \cdot p(1-p)^{\Delta^6}$. The probability that v remains after phase (5) is upper bounded by

$$\left(1-\frac{1}{3}p(1-p)^{\Delta^6}\right)^{|S_v|} \leq e^{-\frac{|S_v|}{3}p(1-p)^{\Delta^6}}$$

$$= e^{-\frac{(\Delta-2)^{t/2}\cdot\Delta^{-12}}{12}}$$
$$\overset{(*)}{\leq} \Delta^{-4t-32},$$

where $(*)$ is satisfied if the exponent $-(\Delta-2)^{t/2}\cdot\Delta^{-12}/12$ 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, in particular r can be chosen independently from the graph size n. □

The following lemma is the most important result of the standard shattering technique.

Lemma 13 (The Shattering Lemma, [17], cf. [7]) *Let $H = (V, E)$ be a graph with maximum degree Δ. 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 iff their distance in H is between $2c_2+1$ and $4c_2+2$. Let $L = H[B]$. Then with probability at least $1 - n^{-c_3}$, 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 $|U| \geq \log_\Delta n$.*
(P2) *Each connected component of L has size at most $O(\log_\Delta n \cdot \Delta^{2c_2\Delta})$.*
(P3) *L admits a $(\lambda, O(\log^{1/\lambda} n \cdot \log^2 \log n))$ network decomposition, for any integer $\lambda \geq 1$, which can be computed by a randomized algorithm in $O\big(\lambda \log^{1/\lambda} n \cdot 2^{O(\sqrt{\log\log n})}\big)$ rounds, w.h.p.*
(P4) *for any $R \geq 1$ there is a randomized algorithm to compute a $\big(2^{O(\sqrt{\log\log n})}, R \cdot 2^{O(\sqrt{\log\log n})}\big)$ network decomposition of L^R in $O(R \cdot 2^{O(\sqrt{\log\log n})})$ rounds, w.h.p.*

Proof $(P1)$–$(P3)$ are proven in [17]. The proof of $(P4)$ is along similar lines as the proof of $(P3)$ in [17] and we only provide a sketch here: First one computes a ruling set M with parameters $\big(2c_2 + 1, \Theta(\log\log n)\big)$ on L^R with the randomized algorithm Lemma 10, (3). Similar to the arguments in [7, Section 3.2, Step 3/4] this ruling set has, if restricted to a single connected component of L, at most $\log_\Delta n$ nodes. Now, we assign each node of the connected components to the closest ruling set node and form a cluster graph. Two clusters in this cluster graph are connected if they have two nodes that are neighbors in the original network. On this cluster graph and for each component in parallel we perform the deterministic network decomposition algorithm from [30] to compute a $\big(2^{O(\sqrt{\log N})}, 2^{O(\sqrt{\log N})}\big)$ for each cluster graph where $N = O(\log_\Delta n)$ is an upper bound on the size of each cluster graph. The runtime of the network decomposition depends on the size of the id space of the nodes and [7, Remark 3.5] explains how to compute a new id space for each cluster graph. As one round on the cluster graph can be executed in $O\big(R \cdot \log\log n\big)$ rounds in H the runtime of this step is $R \cdot 2^{O(\sqrt{\log N})} = R \cdot 2^{O(\sqrt{\log\log n})}$. To obtain a network decomposition of L^R we add each non ruling set node of B to the cluster of its closest ruling set node. This increases the diameter of each cluster by at most a factor $\Theta(\log\log n)$. □

Remark 2 The computation of the single network decomposition in $(P3)$ (or in $(P4)$) only uses randomness for the ruling set computation in the first step. In contrast to the deterministic network decomposition algorithm that is computed on each component separately and in parallel this randomized step is not performed on each component separately but on the whole graph. In particular its runtime and failure probability depend on n where n is the size of the original graph. Furthermore, the ruling set algorithm does not require that the components of size N are also equipped with an ID space of size $\mathsf{poly}\, N$, but works with the ID space of the original graph. The same holds for the network decompositions and ruling sets that are computed to color the small components (cf. Section 5.3).

Lemma 12, 13 imply that the graph L that remains after phase (5) consists of connected components of size at most $\mathsf{poly}\,\Delta\cdot\log_\Delta n$. Section 5.3 explains in detail how these components can be Δ-colored while respecting the nodes colored with color *one* in phase (4).

5.3 Shattering: coloring small remaining components (phase (6))

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 $N := \mathsf{poly}(\Delta)\cdot\log_\Delta n$. Call a node in C *free* if it has degree $< \Delta$ 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 $R = 2\log_{\Delta-2} N + 1 = O(\log\log n)$. 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 $\mathcal{C}_{DCC}$ be the virtual graph that has a node for each selected node and degree choosable subgraph. Any two subgraphs (or nodes) of $\mathcal{C}_{DCC}$ are connected in $\mathcal{C}_{DCC}$ if they share a vertex or are connected by an edge in G. The maximum degree of $\mathcal{C}_{DCC}$ is $\min\{N, O(\Delta^{O(R)})\}$ and it has at most $|C| = N$ nodes. One round of an algorithm on $\mathcal{C}_{DCC}$ can be executed in $O(R)$ steps in G.

(2) Find a $(2,\gamma)$ ruling set M' of $\mathcal{C}_{DCC}$ where $\gamma = O(R)$ such that $\Delta(\mathcal{C}_{DCC})^{2/\gamma} \leq \Delta^{1/2}$.
Runtime for large Δ: Compute a $\left(2^{O(\sqrt{\log\log n})}, 4R \cdot 2^{O(\sqrt{\log\log n})}\right)$ network decomposition of L^{4R} with Lemma 13 (P4). Then each node assigns its color in this network decomposition to its corresponding selected node in $\mathcal{C}_{DCC}$. This results in a $\left(2^{O(\sqrt{\log\log n})}, 4R \cdot 2^{O(\sqrt{\log\log n})}\right) = \left(2^{O(\sqrt{\log\log n})}, 2^{O(\sqrt{\log\log n})}\right)$ network decomposition of $\mathcal{C}_{DCC}$. Then iterate through the colors of the network decomposition to compute the ruling set in time $O\left(R \cdot 2^{O(\sqrt{\log\log n})}\right) = 2^{O(\sqrt{\log\log n})}$.
Runtime for small Δ: Use Lemma 10, (1) in time $O\left(R \cdot \gamma \cdot \Delta(\mathcal{C}_{DCC})^{2/\gamma} + \log^* n\right) = O\left(\log^2\log n \cdot \sqrt{\Delta}\right)$.

(3) For $i = 0, \ldots, \gamma\cdot(R+1)+R$ define layers D_i where D_i consists of the nodes that are at distance i to the closest node that is contained in a component in M'.
Runtime: $O(R^2) = O(\log^2\log n)$.

(4) We color the layers in order $i = \gamma\cdot(R+1)+R, \ldots, 1$; each layer is a deg $+1$ list coloring instance. There are R^2+2R layers and we obtain the following runtimes.
Runtime for large Δ: In time $O((R^2+2R)\cdot 2^{O(\sqrt{\log\log n})}) = 2^{O(\sqrt{\log\log n})}$ via computing a single network decomposition for C with Lemma 13, (P3).
Runtime for small Δ: If we first use Linial's algorithm to compute a $O(\Delta^2)$ coloring the running time is $O\left((R^2+2R)\cdot\sqrt{\Delta\log\Delta}\cdot\log^*\Delta\right) = O\left(\log^2\log n\sqrt{\Delta\log\Delta}\cdot\log^*\Delta\right)$ with Theorem 6.

(5) Now, we color the nodes that are in D_0. Each DCC is brute-forced independently in time $O(R)$. Each free node in D_0 can be colored in a single time unit as it has one uncolored neighbor outside the component it has a free color.
Runtime: $O(R) = O(\log\log n)$.

Lemma 14 *If D_0 is not empty each node of the component is in one of the layers.*

Proof The layers $D_0, \ldots, D_{\gamma\cdot(R+1)}$ contain all free nodes, all nodes that are in a DCC with radius at most R and all nodes that have degree smaller Δ. The layers $D_0, \ldots, D_{\gamma\cdot(R+1)+R}$ 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 Δ 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 5 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 15 *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 Δ 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 Sect. 2.2. Thus we can apply Lemma 5 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 as in the worst case the whole component is a DCC (it cannot be an odd cycle due to $\Delta \geq 4$ and not a $(\Delta-1)$-clique; if it was a $(\Delta-1)$ clique and D_0 is empty all nodes have to be neighbors of the same marked node (due to $b=6$) which implies a Δ-clique, a contradiction). □

The runtimes of the above algorithm provide the following lemma.

Lemma 16 *Let $\Delta \geq 4$. Then the small components can, w.h.p., be Δ-colored in time*

$$\min\left\{2^{O(\sqrt{\log\log n})}, O\left(\log^2\log n\cdot\sqrt{\Delta\log\Delta}\log^*\Delta\right)\right\}.$$

Proof Lemma 14, 15 imply that each node becomes 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 3. The components and free nodes in D_0 can be colored independently because they stem from the independent set M'. In both variants the runtime is dominated by phase (4) step which implies the result. □

Remark 3 The algorithm to solve the small components only uses randomization to compute the network decomposition (Lemma 13 and Remark 2).

5.4 Global success after marking process for small Δ (no phase(6))

In this section we show that a vertex $v \in V(H)$ is contained in $C_0, \ldots, C_{2r}, C'$ w.h.p. if $\Delta = O(1)$. As nodes which have an uncolored path of length $\leq 2r$ to a vertex of degree $< \Delta$ will be contained in one of the layers $C_0, \ldots, C_{2r}$ we assume throughout this section that v and all vertices reachable from v through uncolored path of length at most r have degree Δ in H.

Lemmas 5 and 7 imply that for $\Delta = O(1)$ and $b = 15$ we can choose an $r = \Theta(\log\log n)$ such that for an arbitrarily large constant c, we have $|B_r(v)| \geq c\log_\Delta n$ after the marking process.

Lemma 17 *Let u be a node such that there is an unmarked, unselected path from u to v. Then u or its child u' become a T-node of v with a constant probability.*

Note that the following analysis is done for $b = 15$. For $\Delta \geq 4$ we could equally well use $b = 6$ to optimize the constants.

Proof Let u be a node such that there is an unmarked, unselected path from v to u. In the following we consider the 2-hop neighborhood of u including marked nodes and naturally extend the BFS-tree from v to u for 2 hops to u's 2-hop neighborhood. We make a case distinction depending on how the children of u in this BFS tree are connected.

Case 1, the children of u do not form a clique: The children of u form at least two distinct cliques (including single nodes). Among all pairs of non-adjacent neighbors of u there are at most $\Delta - 1$ pairs that include the parent of u and at least $\Delta - 2$ pairs that do not include the parent of u. Therefore, if u is selected and does not back off due to another node being selected in distance at most b, it actually chooses a non-adjacent pair of neighbors that does not contain its parent, i.e., that does not block the uncolored path to v, with probability at least $(\Delta - 2)/(\Delta - 2 + \Delta - 1) \geq 1/3$. Thus node u becomes a T-node for v with probability at least $p' = (p/3)(1-p)^{\Delta^{15}} = \Theta(1)$, since $\Delta = O(1)$.

Case 2, the children of u form a clique: In this case node u cannot become a T-node of v as it does not have two non-adjacent neighbors that do not block the path to v. However, u's children must have a successor in the BFS tree, and therefore can become a T-node of v. We show that each child of u becomes a T-node of v with constant probability. Let u' be one child of u and u'' the unique child of u'. All children of u form a clique and are connected to u'. If u' is selected and does not back off the only pairs of non adjacent neighbors that u' can select is u'' and u, or u'' and one of u's children. In neither case the uncolored path to v is blocked. With the same reasoning as before u' becomes a T-node of v with constant probability.

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 17 depends on randomness at distance at most 16.

Lemma 18 *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 Lemmas 5 and 7, for any $c > 0$ we can choose $r = O(\log \log n)$ such that at distance t from the root of any BFS tree, there are at least $c/p'\Delta^{16} \ln n$ nodes such that their path to u is unmarked and unselected. From this set find a set S of nodes as in Lemma 17, of size $c/p' \ln n$ with pairwise distance of at least 16 and they can each produce a T-node for v with constant probability p'.

The events that each $u \in S$ become a T-node of v are independent due to the pairwise distance of the nodes. Thus no node of S becomes a T-node of v with probability at most

$$(1-p')^{c/p' \ln n} \leq e^{-c \ln n} \leq n^{-c}.$$

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

6 Conclusion

We have provided several structural results for the Δ-coloring (Sect. 2) that hopefully will be of use for future algorithmic improvements to the problem. For constant degree graphs we provided a deterministic algorithm with $O(\log^2 n)$ round complexity (Theorem 3) and a $O(\log \log n)$ round randomized algorithm (Theorem 1) . The respective lower bounds are $\Omega(\log n)$ and $\Omega(\log \log n)$ and despite only a polynomial difference between upper and lower bounds it remains an intriguing open question whether the true complexity of the problem is at the lower or the higher end.

After our submission, in a breakthrough result, Rozhoň and Ghaffari [33] found a polylogarithmic deterministic time algorithm to compute $(\mathsf{poly}\log n, \mathsf{poly}\log n)$ network decompositions. As one of many implications the runtime of our deterministic Δ-coloring algorithm (for unbounded degree) drops to $\mathsf{poly}\log n$ and our randomized algorithm for non-clique graphs with (unbounded) maximum degree $\Delta \geq 4$ drops to $O(\log \Delta) + \mathsf{poly}\log\log n$ (Theorem 2). Further, [24] provides an improved list coloring algorithm. Combining it with the techniques in our paper [24] gives a $\log^3 n \cdot 2^{O(\sqrt{\log \Delta})}$ round algorithm for Δ-coloring. Also [28] provides improved list coloring algorithms, i.e., it shaves of the $O(\log^* \Delta)$ term in Theorem 6 if the color space is at most exponential in the maximum degree of the uncolored graph. As a result the $\log^* \Delta$ term can also be removed in the runtime of Theorems 1 and 3.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Aboulker, P., Bonamy, M., Bousquet, N., Esperet, L.: Distributed coloring in sparse graphs with fewer colors. In: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, UK, 23–27, 2018, pp. 419–425 (2018)
2. Alon, N., Babai, L., Itai, A.: A fast and simple randomized parallel algorithm for the maximal independent set problem. J. Algorithms **7**(4), 567–583 (1986)
3. Awerbuch, B., Goldberg, A.V., Luby, M., Plotkin, S.A.: Network decomposition and locality in distributed computation. In: Proceedings of 30th Symposium on Foundations of Computer Science (FOCS 1989), pp. 364–369 (1989)
4. Barenboim, L., Elkin, M.: Distributed graph coloring: fundamentals and recent developments. Synth. Lect. Distrib. Comput. Theory **4**(1), 1–171 (2013)
5. Barenboim, L., Elkin, M., Goldenberg, U.: Locally-iterative distributed $(\Delta+1)$-coloring below szegedy-vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. In: Proceedings of 37th ACM Symposium on Principles of Distributed Computing (PODC 2018) (to appear) (2018)
6. Barenboim, L., Elkin, M., Pettie, S., Schneider, J.: The locality of distributed symmetry breaking. In: Proceedings of 53rd Symposium on Foundations of Computer Science (FOCS 2012), pp. 321–330 (2012)
7. Barenboim, L., Elkin, M., Pettie, S., Schneider, J.: The locality of distributed symmetry breaking. J. ACM **63**(3), 201–2045 (2016)
8. Bondy, J.A., Murty, U.: Graph Theory with Applications. Elsevier (1976)
9. Brandt, S., Fischer, O., Hirvonen, J., Keller, B., Lempiäinen, T., Rybicki, J., Suomela, J., Uitto, J.: A lower bound for the distributed Lovász local lemma. In: Proceedings of 48th ACM Symposium on Theory of Computing (STOC 2016), pp. 479–488. ACM (2016)
10. Brooks, R.L.: On colouring the nodes of a network. Math. Proc. Camb. Philos. Soc. **37**(2), 194–197 (1941)
11. Leonard Brooks, R.: On colouring the nodes of a network. In: Classic Papers in Combinatorics, pp. 118–121. Springer (2009)
12. Chang, Y.-J., Kopelowitz, T., Pettie, S.: An exponential separation between randomized and deterministic complexity in the local model. In: Proceedings of 57th Symposium on Foundations of Computer Science (FOCS 2016), pp. 615–624. IEEE (2016)
13. Chang, Y.-J., Li, W., Pettie, S.: An optimal distributed $(\Delta+1)$-coloring algorithm? In: Proceedings 50th ACM Symposium on Theory of Computing (STOC 2018) (to appear) (2018)
14. Chechik, S., Mukhtar, D.: Optimal distributed coloring algorithms for planar graphs in the local model. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), SODA'19, pp. 787–804 (2019)
15. Cranston, D.W., Rabern, L.: Brooks' theorem and beyond. J. Graph Theory **80**(3), 199–225 (2015)
16. Erdős, P., Rubin, A., Taylor, H.: Choosability in graphs. In: Proceedings of West Coast Conference on Combinatorics, Graph Theory and Computing, vol. 26, pp. 125–157. Congressus Numerantium (1979)
17. Fischer, M., Ghaffari, M.: Sublogarithmic distributed algorithms for Lovász local lemma, and the complexity hierarchy. In: Proceedings of the International Symposium on Distributed Computing (DISC), pp. 18:1–18:16 (2017)
18. Fraigniaud, P., Heinrich, M., Kosowski, A.: Local conflict coloring. In: Proceedings of 57th Symposium on Foundations of Computer Science (FOCS 2016), pp. 625–634 (2016)
19. Gfeller, B., Vicari, E.: A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. In: Proceedings of 26th ACM Symposium on Principles of Distributed Computing (PODC 2007), pp. 53–60, New York, NY, USA (2007). ACM
20. Ghaffari, M.: An improved distributed algorithm for maximal independent set. In: Proceedings of 27th ACM-SIAM Symposium on Discrete Algorithms (SODA 2016), pp. 270–277 (2016)
21. Ghaffari, M., Kuhn, F., Maus, Y.: On the complexity of local distributed graph problems. In: Proceedings of 49th ACM Symposium on Theory of Computing (STOC 2017), pp. 784–797. ACM (2017)
22. Ghaffari, M., Su, H.-H.: Distributed degree splitting, edge coloring, and orientations. In: Proceedings of 28th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017), pp. 2505–2523. Society for Industrial and Applied Mathematics (2017)
23. Harris, D.G., Schneider, J., Su, H.-H.: Distributed $(\Delta+1)$-coloring in sublogarithmic rounds. In: Proceedings of 48th ACM Symposium on Theory of Computing (STOC 2016) pp. 465–478. ACM (2016)
24. Kuhn, F.: Faster deterministic distributed coloring through recursive list coloring. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1244–1259 (2020)
25. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**(1), 193–201 (1992)
26. Lovász, L.: Three short proofs in graph theory. J. Combin. Theory Ser. B **19**(3), 269–271 (1975)
27. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM J. Comput. **15**(4), 1036–1053 (1986)
28. Maus, Y., Tonoyan, T.: Local conflict coloring revisited: linial for lists. In: Proceedings of the International Symposium on Distributed Computing (DISC), pp. 16:1–16:18 (2020)
29. Molloy, M., Reed, B.: Graph Colouring and the Probabilistic Method, vol. 23. Springer (2013)
30. Panconesi, A., Srinivasan, A.: Improved distributed algorithms for coloring and network decomposition problems. In: Proceedings of 24th ACM Symposium on Theory of Computing (STOC 1992), pp. 581–592. ACM (1992)
31. Panconesi, A., Srinivasan, A.: The local nature of Δ-coloring and its algorithmic applications. Combinatorica **15**(2), 255–280 (1995)
32. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM (2000)
33. Rozhoň, V., Ghaffari, M.: Polylogarithmic-time deterministic network decomposition and distributed derandomization. In: Proceedings of the ACM Symposium on Theory of Computing (STOC), pp. 350–363 (2020)
34. Schneider, J., Elkin, M., Wattenhofer, R.: Symmetry breaking depending on the chromatic number or the neighborhood growth. Theoret. Comput. Sci. **509**, 40–50 (2013)
35. Vizing, V.: Vextex coloring with given colors. Metody Diskretn. Anal. **29**, 3–10 (1976)