Coloring Fast Without Learning Your Neighbors' Colors

Magnús M. Halldórsson
Reykjavik University, Iceland
mmh@ru.is

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

Yannic Maus
Technion – Israel Institute of Technology, Haifa, Israel
yannic.maus@cs.technion.ac.il

Alexandre Nolin
Reykjavik University, Iceland
alexandren@ru.is

Abstract

We give an improved randomized CONGEST algorithm for distance-2 coloring that uses Δ^2+1 colors and runs in $O(\log n)$ rounds, improving the recent $O(\log \Delta \cdot \log n)$-round algorithm in [Halldórsson, Kuhn, Maus; PODC '20]. We then improve the time complexity to $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases distributed graph coloring, distance 2 coloring, congestion

Digital Object Identifier 10.4230/LIPIcs.DISC.2020.39

Related Version Full version available under `https://arxiv.org/abs/2008.04303`.

Funding This project was supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 755839 and by the Icelandic Research Fund grant 174484.

1 Introduction

The distributed coloring problem is arguably the most intensively studied problem in the area of distributed graph algorithms and certainly also one of the most intensively studied problems in distributed computing more generally. The standard assumption is that the *coloring graph* – the graph on which we want to compute a coloring – is also the *communication network* – the graph forming the network topology. We explore in this paper the case when the latter is weaker than the former: the communication is constrained, and direct links are not available to all the "neighbors" that are to be colored differently.

The primary setting for this is the *distance-*2 *coloring* problem in the standard distributed CONGEST model. Given a graph $G=(V,E)$, in the *d2-coloring* problem on G, the objective is to assign a color x_v to each node $v \in V$ such that any two nodes u and v at distance at most 2 in G are assigned different colors $x_u \neq x_v$. Equivalently, d2-coloring asks for a coloring of the nodes of G such that for every $u \in V$, all the nodes in the set $\{u\} \cup N(u)$ (where $N(u)$ denotes the set of neighbors of u) are assigned distinct colors. Further note that d2-coloring on G is also equivalent to the usual vertex coloring problem on the graph G^2, where $V(G^2) = V$ and there is an edge $\{u,v\} \in E(G^2)$ whenever $d_G(u,v) \leq 2$.

The CONGEST model is a standard synchronous message passing model [32]. The graph on which we want to compute a coloring is also assumed to form the network topology. Each node $u \in V$ of the graph has a unique $O(\log n)$-bit identifier $\mathrm{ID}(u)$, where $n = |V|$ is the number of nodes of G. Time is divided into synchronous rounds and in each round, each node $u \in V$ of G can do some arbitrary internal computation, send a (potentially different) message to each of its neighbors $v \in N(u)$, and receive the messages sent by its neighbors in the current round. If the size of the messages is not restricted, the model is known as the LOCAL model [29, 32]. In the CONGEST model, it is further assumed that each message consists of at most $O(\log n)$ bits.

As our main result, we give an efficient $O(\log n)$-time randomized algorithm for d2-coloring G with at most $\Delta^2 + 1$ colors, where Δ is the maximum degree of G. This improves on a recent $O(\log \Delta \cdot \log n)$-time algorithm [23] and it matches the best known bound for ordinary distance-1 $(\Delta + 1)$-coloring in CONGEST as a function of n alone. We further explore more efficient algorithms when $\Delta \ll n$. Combining our main method with a range of powerful recent techniques, we obtain an algorithm that runs in time $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$.

Before discussing our results in more detail, we first discuss why we believe d2-coloring is interesting, what is known for the corresponding coloring problems on G and why it is challenging to transform CONGEST algorithms to color G into CONGEST algorithms for d2-coloring.

Wireless networking is a major motivation for distance-2 coloring, where nodes with a common neighbor should not simultaneously communicate to avoid a collision at the common neighbor [15, 31]. While the coloring is to be used for scheduling, the wireless channel need not be the medium for *computing* the coloring. With the advent of software-defined radio and hierarchical / heterogeneous networks, it is well motivated to consider coloring computation in a communication model more powerful than radio networks. Yet, asking for the different-message-to/from-all-neighbors feature of CONGEST may be hoping for too much. More generally, we view it as a major question in distributed graph algorithms whether one can relax the communication requirements for graph coloring. We ask:

How constrained can the communication structure be to allow for fast (logarithmic, sublogarithmic) distributed graph coloring computation?

Distributed d2-coloring is an interesting and important problem for several other reasons. The d2-coloring problem for example also occurs naturally when single-round randomized algorithms are derandomized using the method of conditional expectation [21]. d2-coloring in CONGEST is further of special interest as it appears to lie at the edge of what is computable efficiently, i.e., in polylogarithmic time, while distance-3 coloring is even hard to verify [18].

Distance-k problems have not been addressed widely in a distributed setting, partly because distance-k communication can be simulated in k steps of the LOCAL model. In CONGEST, the situation changes drastically as simulating a single round of a distance-1 coloring algorithm can incur a factor $\Theta(\Delta^{k-1})$ overhead, i.e., even for $k = 2$, the overhead can be linear in Δ. Even the very simple algorithm where each node picks a random available color cannot be efficiently used for d2-coloring as it is in general not possible to keep track of the set of colors chosen by 2-hop neighbors in time $o(\Delta)$. Recently, Halldórsson, Kuhn and Maus [23] treated d2-coloring in CONGEST and gave a randomized algorithm using $\Delta^2 + 1$-colors in $O(\log \Delta \cdot \log n)$ rounds, as well as a deterministic algorithm using $(1 + \epsilon)\Delta^2$ colors in $poly(\log n)$ rounds. Our main approach builds heavily on their framework, while simplifying certain features and strengthening structural properties. Distributed graph optimization problems on G^2 (with CONGEST-communication in G) such as vertex cover and minimum dominating set have recently been studied in [5].

Distributed graph coloring. The standard variant of the distributed coloring problem on G asks for computing a vertex coloring with at most $\Delta+1$ colors, which is computed by a simple sequential greedy algorithm. The main focus in the literature on distributed coloring has been on the LOCAL model, where by now the problem is understood relatively well. The best randomized $(\Delta+1)$-coloring algorithm known in the LOCAL model, due to Chang, Li, and Pettie [14], runs in $\operatorname{poly}\log\log n$ rounds. The complexity given in [14] is $2^{O(\sqrt{\log\log n})}$, while the improvement to $\operatorname{poly}\log\log n$ immediately follows from the recent breakthrough work on deterministic *network decomposition* of Rozhoň and Ghaffari [33]. For a more detailed discussion of related work on distributed coloring, we refer to [8, 14, 27].

While most known distributed coloring algorithms were developed for the LOCAL model, many of them work directly in the CONGEST model, including those in [1, 30, 29, 26, 28, 7, 10, 6, 9, 27, 4]. Still, the best complexity known for coloring in CONGEST, as a function of n alone, is $O(\log n)$, which is achieved by the following very simple OneShotColoring algorithm: Initially all nodes are uncolored. The algorithm runs in synchronous phases, where in each phase, each still uncolored node v chooses a uniform random color among its available colors (i.e., among the colors that have not already been picked by a neighbor) and v keeps the color if no of its uncolored neighbors tries the same color at the same time [26, 11].

The only known published algorithm in CONGEST with a better bound is due to Ghaffari [19], who obtains a $(\Delta+1)$-coloring in time $O(\log\Delta)+2^{O(\sqrt{\log\log n})}$. The second term is due to a network decomposition algorithm also introduced in [19]. Unlike for results in the LOCAL model, it is *not* directly possible to replace this decomposition with the recent construction in [33] to improve the dependence on n. The reason is that the complexity of the network decomposition construction of [33] grows at least linearly in the length of the node identifier bit strings. In the LOCAL model, it is possible to use a standard coloring algorithm of [29] to first map the IDs to $O(\log\log n)$-bit values that are unique up to a sufficient distance so that one can afterwards apply the algorithm of [33]. Subsequent to the publication of our results [20] improved upon the network decomposition algorithm from [33] (to deal with large IDs in the CONGEST model) and as a result obtains a $O(\log\Delta)+\operatorname{poly}\log\log n$ CONGEST algorithm for $(\Delta+1)$-coloring. Note that if we have graphs of size N and if IDs and colors can be represented with $\operatorname{poly}\log N$ bits, there is a recent *deterministic* $(deg+1)$-list coloring algorithm running in $\operatorname{poly}\log N$ time in CONGEST [4].

1.1 Contributions

We provide two efficient randomized CONGEST model algorithms to compute a d2-coloring of a given n-node graph $G=(V,E)$. If Δ is the maximum degree of G, the maximum degree of any node in G^2 is at most $\Delta+\Delta\cdot(\Delta-1)=\Delta^2$. As a natural analog to studying $(\Delta+1)$-coloring on G, we study the problem of computing a d2-coloring with Δ^2+1 colors.

▶ **Theorem 1.1.** *There is a randomized CONGEST algorithm that d2-colors a graph with Δ^2+1 colors in $O(\log n)$ rounds, with high probability.*

The algorithm is given in Sec. 2, with the key ideas and challenges outlined at the start of the section. Our second algorithm is more efficient if $\Delta \ll n$.

▶ **Theorem 1.2.** *There is a randomized CONGEST algorithm that d2-colors a graph with Δ^2+1 colors in $O(\log\Delta)+2^{O(\sqrt{\log\log n})}$ rounds, with high probability.*

Theorem 1.2 relies on the network decomposition algorithm of [22] that can compute a suitable network decomposition of G^2 despite a large ID space in $2^{O(\sqrt{\log\log n})}$ rounds. The unpublished result in [20] computes similar network decompositions despite a large ID space in $\text{poly}\log\log n$ rounds, but only for G and not for G^2. If the results of [20] can be extended to G^2, which in fact is likely, the runtime of Theorem 1.2 improves to $O(\log\Delta)+\text{poly}\log\log n$, and it then again – as at the time of submission of this manuscript – matches the complexity of ordinary distance-1 coloring in CONGEST. The key ideas of Theorem 1.2 appears in Sec. 3. Formal proofs of all statements as well as efficient implementations are discussed in detail in the full version [24].

2 Logarithmic Time Randomized Algorithm

We give randomized CONGEST algorithms that form a d2-coloring using Δ^2+1 colors. We first introduce notation that we use frequently throughout the proofs in this section.

Notation. The *palette* of available colors is $\{0, 1, 2, \ldots, \Delta^2\}$. The neighbors in G of a node are called *immediate neighbors*, while the neighbors in G^2 are *d2-neighbors*. For a (sub)graph K, let $N_K(v)$ denote the set of neighbors of v in K, and let $K[v] = K[N_K(v)]$ denote the subgraph induced by these neighbors. A node is *live* or *uncolored* until it becomes *colored*. An edge in G^2 corresponds to a 2-path (path of length 2) in G; thus, G^2 can have parallel edges.

A node has *slack* q if the number of colors of d2-neighbors plus the number of live d2-neighbors is Δ^2+1-q. In other words, a node has slack q if its palette size is an additive q larger than the number of its uncolored $d2$-neighbors.

An event holds *w.h.p.* (with high probability), if for any $c > 0$, we can choose the constants involved so that the event holds with probability $1-O(n^{-c})$.

2.1 Overview

Our algorithm builds on the approach of [23], which we first summarize. The simple *informed* color guessing approach – each node tries a random color not used by its d2-neighbors – fails because the nodes do not have the bandwidth to learn those colors. A simple *uninformed* approach – trying any random color – works fine if there is sufficient slack, either because the palette is strictly larger than the degree, or in the beginning when few neighbors have been colored. In this case, even trying a uniformly random color is successful with constant probability. If the node has a *sparse* neighborhood then in the very first round, many pairs of d2-neighbors will conveniently adopt the same color, as proved by Elkin, Pettie and Su [17], creating the needed slack. We are then left to deal with denser neighborhoods, of varying average non-degree.

The key idea of [23] is to have the colored nodes "help" the *live* nodes by checking random colors on their neighborhoods. This provides a probabilistic filter that helps reduce the load of the live nodes. It turns out that this alone is not sufficient due to *false negatives*: the helper may reject good colors because it has neighbors with those colors. The solution is for the helper to also query one of its neighbor w, and forward its color if w is not a d2-neighbor of the live node. It is shown that one of these forms of advice is good with constant probability, but could only argue that for those live nodes with a sparsity in a given range considered. This meant that the round complexity of the method had an extra $O(\log\Delta)$ factor for ranging through the different sparsity levels, on top of the $\log n$ factor for finishing off all nodes of that sparsity.

The main technical ingredient behind our $O(\log n)$-round algorithm is the adaptation and extension of the *almost-clique decomposition* (ACD) method initially proposed by Harris, Schneider and Su [25] for the LOCAL model and expanded by Assadi, Lee and Khanna

[2] for streaming and massively parallel settings. The nodes are partitioned into a set of sparse nodes – which can be handled by uninformed guesses – and low-diameter clusters of dense nodes. The ACD achieves the same aims as the *similarity graphs* of [23] that guide the querying and ensure effective filtering, but attain some additional crucial properties such as near-regular high degree. Our extension to ACD is to ensure that all nodes outside clusters have a low degree into the cores of the clusters, strengthening the divide between inside and outside. The decomposition additionally simplifies the technical arguments, including load balancing and probabilistic independence. The key property that we then obtain is that in each iteration, every live node (with at least logarithmic size palette) becomes colored with constant probability. That makes even faster algorithms possible, as we show in the next section. To finish off the nodes with a palette of at most logarithmic size, we apply a second method of [23] black-box, which *learns* the palette of the live nodes and then performs informed color guessing.

2.2 Algorithm Description

We now outline our algorithm, followed by details on the implementation.

Each live node v repeatedly *tries* a suggested color, which means to first *validate* it and then *contest* it. Validating a color means sending it to all immediate neighbors, who then report back if they or any of their neighbors had already adopted that color. Contesting a validated color means proposing it to intermediate neighbors, who report back if any other node also proposes it. If all answers are negative, then v adopts the color.

In what follows, let $\epsilon = 1/60$, $c_0 = 48e^4/\epsilon^2$, and c_3 be a constant to be determined. Also, c_2 is a sufficiently large constant needed for concentration.

Algorithm D2-COLOR

```
If $\Delta^2 \geq c_2 \log n$ then
  1. Compute an almost-clique decomposition.
  2. repeat $c_0 \log n$ times:
       Each live node picks a random color and tries it.
  3. repeat $c_3 \log n$ times
       REDUCE-PHASE()
  4. LEARNPALETTE()
FINISHCOLORING()
```

We will discuss and analyze Steps 1–3 of the above algorithm in detail in the following. The remaining steps, LEARNPALETTE() and FINISHCOLORING(), are from [23]. In LEARNPALETTE(), each live node *learns the palette* of still available colors by cooperatively tallying the colors of d2-neighbors. In FINISHCOLORING(), the maximum degree is sufficiently small so that we can efficiently simulate the classic algorithm of informed color guessing to color the remaining live nodes.

The first step of the algorithm is to compute a decomposition of the nodes into: a) a set of nodes inducing a subgraph (in G^2) that is sufficiently sparse and b) a disjoint collection of almost-cliques (also in G^2). In the following, each of the almost-cliques is called a component C and we use two graphs H and $\hat{H}$ (closely related to the ones in [23]) that both essentially consist of all the components (i.e., all the almost-cliques) and all the G^2-edges connecting two nodes in the same component. Also computed within each component is a spanning tree for a fast aggregation. The exact definitions of the decomposition and of the graphs H and $\hat{H}$ appear in Subsection 2.3.

We next detail the steps of Reduce-Phase(), which is the core piece of our algorithm.

Algorithm Reduce-Phase()

1. Each live node randomly decides to be *active* with probability $1/8$. All other nodes are *inactive*.
2. Compute ϕ, the number of active live nodes in the component C, and distribute it to the nodes of C.
3. Each inactive node $u \in C$ computes ϕ_u, the number of 2-paths to active nodes (by asking its immediate neighbors of their active immediate neighbors). u flips a biased coin: with probability $\min(1, \phi_u/(4\phi))$ it picks one of the ϕ_u paths uniformly at random, while with probability $\max(0, 1 - \phi_u/(4\phi))$, u stops the execution of this iteration. Let v denote the active node at the other end of the path chosen. u verifies that it has only one 2-path to u (by inquiring to its immediate neighbors), and otherwise stops execution of this iteration.
4. u picks a random color $\hat{c}$ different from its own. If that color is not used by any of its $\hat{H}$-neighbors, then u sends the color to v as a proposal, assigning it a uniformly random priority. v tries the proposed color of highest priority (if any).
5. u sends query (v, u) along a random 2-path to an inactive $\hat{H}$-neighbor w, and assigns it a random priority.
6. Upon receipt of a query, node w selects the highest priority query (v, u), checks if v is a d2-neighbor, and if v and w are not d2-neighbors, it sends its color $c(w)$ to v (through u).
7. The active node v tries a color chosen uniformly random among the received proposed colors from Step 6 (if any).

A colored node assists an active node v in two ways: a) guesses and validates a random color for v to try, and b) sees if a random d2-neighbor is also a d2-neighbor of v. This is a probabilistic filter that reduces the workload of the active nodes. The key idea is that one of these forms of assistance is likely to be successful.

Complexity. We discuss the almost-clique decomposition in the next subsection and show how to implement it in $O(\log n)$ rounds, w.h.p. The second step clearly takes $\Theta(\log n)$ rounds. The procedure Reduce-Phase takes 24 rounds, or 8 (Step 2), 2 (Step 3), 2 (Step 4), 2 (Step 5), 6 (Step 6), and 4 (Step 7, including the notification of a new color).

Outline of this section: In Sec. 2.3 we describe the almost-clique decomposition and derive key structural properties of dense subgraphs, and in Sec. 2.4 we prove the correctness of the algorithms, i.e., we show that any node is colored w.h.p. after $O(\log n)$ rounds.

2.3 Almost-Clique Decomposition

We next define the notion of local sparsity and the almost-clique decomposition that we use in our paper. The first definition is a slight adaptation of a similar definition in [17].

▶ **Definition 2.1.** *A node v is ζ-*sparse *(or* has sparsity *ζ) if $G^2[v]$ contains $\binom{\Delta^2}{2} - \Delta^2 \cdot \zeta$ (distinct) edges.*

Sparsity is a rational number that indicates how many edges are missing from $G^2[v]$, compared with the densest case (when v's d2-neighborhood is a Δ^2-clique). If no pairs of d2-neighbors of v are adjacent, then $\zeta = (\Delta^2 - 1)/2$, while if $G^2[v]$ forms a Δ^2-clique, then $\zeta = 0$. Elkin, Pettie and Su [17] formalized the connection between sparsity and slack that appears after trying one uniformly random color.

▶ **Proposition 2.2** ([17], Lemma 3.1). *Let v be a vertex of sparsity ζ and let Z be the slack of v after trying a single random color. Then,* $\Pr[Z \leq \zeta/(4e^3)] \leq e^{-\Omega(\zeta)}$.

We require the constant c_2 to be such that if $\zeta \geq c_2 \log n$, then the contrapositive of Prop. 2.2 yields that $Z \geq \zeta/(4e^3)$, w.h.p.

Decomposition. We adapt the almost-clique decomposition of [2] (building on [25]) for the distance-2 setting in CONGEST and endow it with an additional property.

▶ **Definition 2.3.** *Assume $\epsilon \leq 1/60$. Nodes u and v are* ϵ-similar *if they share at least $(1-\epsilon)\Delta^2$ common d2-neighbors. An* almost-clique decomposition *(ACD) with parameter ϵ is a collection of sets $V_*, \hat{C}_1, \hat{C}_2, \ldots, \hat{C}_k$ that cover V and where the $\hat{C}_i$ are disjoint. Denote $C_i = \hat{C}_i \setminus V_*$, for $i = 1, \ldots, k$. The decomposition satisfies the following properties:*

1. *The nodes in V_* have sparsity at least $\epsilon^2\Delta^2/4$.*
2. *For any $i \in [k]$, C_i and $\hat{C}_i$ satisfy:*
   - **a.** $|C_i| \geq (1-2\epsilon)\Delta^2$.
   - **b.** *The nodes in $\hat{C}_i$ are mutually 10ϵ-similar.*
   - **c.** *Each $v \in \hat{C}_i$ has at most $28\epsilon\Delta^2$ d2-non-neighbors in $\hat{C}_i$ (i.e., $|\hat{C}_i \setminus N_{\hat{C}_i}(v)| \leq 28\epsilon\Delta^2$).*
   - **d.** *Each $v \in \hat{C}_i$ has at least $(1-10\epsilon)\Delta^2$ d2-neighbors in C_i.*
   - **e.** *Each $v \in C_i$ is ϵ-dissimilar to every node outside $\hat{C}_i$.*

We refer to each C_i as a *component* and $\hat{C}_i$ as an *extended component.* The properties imply additional ones: Each extended component is of size at most $(1+28\epsilon)\Delta^2$; and any two nodes in an extended component are within two hops (in G^2). The additional property we need that is not in the formulations of [2] or [25] is Property 2(e).

Let H denote the subgraph of G^2 induced by the components $C_1, \ldots, C_k$, i.e., $H = \cup_i G^2[C_i] = (V \setminus V_*, E_H)$ where E_H consists of the pairs of d2-neighbors within the same component. Similarly, let $\hat{H} = \cup_i G^2[\hat{C}_i]$. We consider H, $\hat{H}$ and G^2 to be *simple* graphs, ignoring multiple 2-paths between the same pair of nodes.

▶ **Lemma 2.4.** *There is an $O(\log n)$-round CONGEST algorithm to form an almost-clique decomposition, for any fixed $\epsilon > 0$. Afterwards, each node knows its component ID.*

It is somewhat surprising that such a decomposition can be established efficiently in CONGEST. The key implementation ideas are in [2] for other models, which are essentially based on randomly sampling nodes. In the distance-2 setting we have the additional challenge of communication with one's d2-neighbors, but the key is to have both parties communicate only with the intermediate node that makes the deciding.

We strengthen the ACD-properties for dense nodes and show that they scale with the node sparsity. Note that dense nodes can have non-trivial sparsity and it is crucial in our argument to leverage the corresponding slack.

▶ **Lemma 2.5.** *Let $\epsilon \leq 1/30$. Let v be a node of sparsity ζ in an almost-clique C and extended component $\hat{C}$. Then,*

1. *v has at least $\Delta^2 - (2\zeta+1)/\epsilon$ $\hat{H}$-neighbors (in $\hat{C}$),*
2. *v has at most $|\hat{C} \setminus N_{G^2}(v)| \leq 3\zeta$ $\hat{H}$-non-neighbors, and*
3. *The number of edges in $\hat{H}[v]$ is at least $|E(\hat{H}[v])| \geq \binom{\Delta^2}{2} - (2/\epsilon+1)\zeta\Delta^2$.*

Proof. Recall that by the definition of sparsity, $G^2[v]$ has exactly $\Delta^2((\Delta^2-1)/2-\zeta)$ edges.

1. A d2-neighbor of v that is not ϵ-similar to v can share at most $(1-\epsilon)\Delta^2$ common d2-neighbors with v by ACD property 2(e). In other words, the d2-neighbors of v that are not $\hat{H}$-neighbors can have degree at most $(1-\epsilon)\Delta^2$ in $G^2[v]$. The number of edges in $G^2[v]$ is then at most

   $$\frac{1}{2}\left(|N_{\hat{H}}(v)|\Delta^2 + (|N_{G^2}(v)| - |N_{\hat{H}}(v)|)(1-\epsilon)\Delta^2\right) \le \frac{\Delta^2}{2}\left((1-\epsilon)\Delta^2 + \epsilon|N_{\hat{H}}(v)|\right) .$$

   Combining the two bounds on the number of edges in $G^2[v]$,

   $$\epsilon|N_{\hat{H}}(v)| \ge \Delta^2 - 1 - 2\zeta - (1-\epsilon)\Delta^2 = \epsilon\Delta^2 - 1 - 2\zeta .$$

   Namely, the number of $\hat{H}$-neighbors of v is at least $\Delta^2 - (2\zeta+1)/\epsilon$.
2. By sparsity, there are at most $(2\zeta+1)\Delta^2$ edges of $\hat{H}$ with exactly one endpoint in $N_{G^2}(v)$. Nodes in $\hat{C} \setminus N_{G^2}(v)$ share at least $(1-10\epsilon)\Delta^2$ d2-neighbors with v, by ACD property 2(b). Thus, there are at most $2\zeta\Delta^2/((1-10)\epsilon\Delta^2) = 2\zeta/(1-10\epsilon) \le 3\zeta$ nodes in $\hat{C}$ that are not d2-neighbors of v, using that $\epsilon \le 1/30$.
3. By **1** of this lemma, v has degree at least $\Delta^2 - q$ in $\hat{H}$, where $q = (2\zeta+1)/\epsilon$. The at most q nodes in $N_{G^2}(v) \setminus N_{\hat{H}}(v)$ have degree sum at most $q(\Delta^2 - q)$. Thus, the number of edges in $\hat{H}[v]$ is at least $\binom{\Delta^2}{2} - \zeta\Delta^2 - q\Delta^2$. ◀

2.4 Correctness

We prove that D2-COLOR correctly d2-colors G with Δ^2+1 colors in $O(\log n)$ rounds. We assume that the almost-clique decomposition and the graphs H and $\hat{H}$ have been correctly constructed, in the sense of Def. 2.3. Also, that nodes of sparsity $\zeta \ge c_2 \log n$ have slack at least $\zeta/(4e^3)$ as promised by Prop. 2.2. All statements in this section are conditioned on these events.

We first give a high-level proof which encapsulates the core of the technical argument in the following lemma, which is then proven in the upcoming subsubsection.

▶ **Lemma 2.6.** *There is an absolute constant c' such that the following holds. For a live node v in given iteration of* REDUCE-PHASE, *there is a subset $S \subseteq \psi_v$ of size at least $|\psi_v|/2$ such that each color in S has probability at least $1/(c'|\psi_v|)$ of being validated by v.*

We then easily dispose of the sparse nodes. Since they have slack linear in their degree, they get colored with constant probability in each round, simply by contesting a uniformly random color.

▶ **Lemma 2.7.** *Every node in V_* is colored after Step 2 of* D2-COLOR, *w.h.p.*

Proof. Let $v \in V_*$. By Def. 2.3(1), v has sparsity at least $\zeta \ge \epsilon^2\Delta^2/4$, and by Prop. 2.2, it has slack at least $c_{13} \doteq \epsilon^2/(16e^3)$, w.h.p. Furthermore, the probability that no d2-neighbor of v tries the same color in the same round is at least $(1-1/(\Delta^2+1))^{\Delta^2} \ge 1/e$, using that $(1-1/x)^{x-1} \ge 1/e$ for any $x > 1$. Thus, with probability at least c_{13}/e, v becomes colored in that round. Hence, the probability that it is not colored in all $c_0 \log n$ rounds is at most $(1-c_{13}/e)^{c_0 \log n} \le e^{-c_0 c_{13}/e \log n} \le n^{-c_0 c_{13}/e} = n^{-3}$, since $c_0 = 48e^4/\epsilon^2 = 3e/c_{13}$. ◀

▶ **Theorem 1.1.** *There is a randomized CONGEST algorithm that d2-colors a graph with Δ^2+1 colors in $O(\log n)$ rounds, with high probability.*

Proof. By Lemma 2.7, it suffices to focus on the dense nodes. We first claim that in each iteration, each live node v with palette size $\Omega(\log n)$ becomes colored with a constant non-zero probability.

Consider a given iteration and a live node v. With probability $1/8$, v is active. It has at most $|\psi_v|$ live neighbors and expected at most $|\psi_v|/8$ are active. By Markov's inequality, at most $|\psi_v|/4$ are active, with probability at least $1/2$. By Lemma 2.6, there is a subset $S \subseteq \psi_v$ of size at least $|\psi_v|/2$ such that each color in S has probability at least $1/(c'|\psi_v|)$ of being validated. Independent of what these active neighbors choose, there is then a subset of at least $|\psi_v|/2 - |\psi_v|/4 = |\psi_v|/4$ colors that are available to v, i.e., are not contested by d2-neighbors of v in that iteration. The probability that one of them is validated, and leading to a valid coloring of v, is then at least $c_* = \frac{1}{8} \cdot \frac{1}{2} \cdot \frac{|\psi_v|/4}{c'|\psi_v|} = \frac{1}{64c'}$, establishing the claim.

Applying a Chernoff bound to the above claim, after $5/c_* \cdot \log n$ iterations of Reduce-Phase, it holds with probability at least $1 - 1/n^3$ that all nodes are either colored or have palette size $O(\log n)$ (in which case they have $O(\log n)$ uncolored d2-neighbors). The coloring is then completed by the two algorithms of [23], both running in $O(\log n)$ rounds. ◀

2.4.1 Proof of Lemma 2.6

We prove our main result in two parts, given in Lemmas 2.11 and 2.12, distinguishing between the two forms of making progress: based on Step 4 or Steps 5-7 of Reduce-Phase.

▶ **Definition 2.8.** *An inactive node is v-*decent *(or just* decent*) if it has at most 4ϕ 2-paths in its almost-clique C to active nodes (in C)* and *has exactly one 2-path to v.*

The distinction between 2-paths and d2-neighbor relations is the rationale for the *decent* definition. Those nodes with lots of paths to active nodes can cause much congestion with poor proposals, while being of limited use to those H-neighbors to which they have few paths.

▶ **Lemma 2.9.** *Let v be a live node and w be a node, both in $\hat{C}$. Then, v and w have at least $\Delta^2/4$ common d2-neighbors in C that are v-decent.*

Proof. The two nodes v and w are 10ϵ-similar, by Def. 2.3(2b). Since v has at least $(1-10\epsilon)\Delta^2$ distinct d2-neighbors in C (by Def. 2.3(2d)), they share at least $(1 - 20\epsilon)\Delta^2 \geq 2\Delta^2/3$ d2-neighbors in C (using that $\epsilon \leq 1/60$). This also means that there are at most $10\epsilon\Delta^2 \leq \Delta^2/6$ nodes in C with multiple 2-paths to v. Also, since there are at most $\phi\Delta^2$ total number of 2-paths to the ϕ live nodes in C, there are at most $\Delta^2/4$ nodes with 4ϕ or more 2-paths to live nodes. Hence, there are at least $2\Delta^2/3 - \Delta^2/6 - \Delta^2/4 = \Delta^2/4$ common d2-neighbors of v and w in C that are decent, i.e., have at most 4ϕ 2-paths to active nodes and exactly one 2-path to v. ◀

A color proposed to an active node v is *bad* if it is already assigned to a d2-neighbor of v. Namely, it is bad if it is a "false positive".

▶ **Lemma 2.10.** *The expected number of bad proposals generated in Step 4 for v is at most $(1/\epsilon + 1)\zeta/\phi$.*

Proof. Let u be an inactive H-neighbor of v. Let Y_u be the event that u picks v in Step 2, and note that $\Pr[Y_P] \leq 1/(4\phi)$. Let X_u be the event that u generates a bad proposal for v in Step 4. That event occurs when u's randomly chosen color is used by a node in S_u, where $S_u = N_{G^2}(v) \setminus N_{\hat{H}}[u]$ is the set of d2-neighbors of v that are not $\hat{H}$-neighbors of u (nor u itself).

The number of such colors is at most $|S_u| = |N_{G^2}(v) \setminus N_{\hat{H}}[u]| \leq (\Delta^2 - 1) - |N_{\hat{H}}(u) \cap N_{\hat{H}}(v)|$. There are at most Δ^2 colors to choose from – all except the one on u – so

$$\Pr[X_u | Y_u] \leq \frac{|S_u|}{\Delta^2} \leq \frac{(\Delta^2 - 1) - |N_{\hat{H}}(u) \cap N_{\hat{H}}(v)|}{\Delta^2} \ .$$

By applying Lemma 2.5(3), we have that

$$\sum_{u \in N_{\hat{H}}(v)} |N_{\hat{H}}(u) \cap N_{\hat{H}}(v)| = 2|E(\hat{H}[v])| \geq \Delta^2(\Delta^2 - 1) - (4/\epsilon + 2)\zeta\Delta^2 \ .$$

Combining the two bounds, letting I denote the set of inactive H-neighbors of v, we get that

$$\sum_{u \in I} \Pr[X_u | Y_u] \leq \sum_{u \in N_{\hat{H}}(v)} \Pr[X_u | Y_u] \leq (4/\epsilon + 2)\zeta \ .$$

Hence, the expected number of bad proposals generated for v is

$$\sum_{u \in I} \Pr[X_u \cap Y_u] = \sum_{u \in I} \Pr[Y_u] \cdot \Pr[X_u | Y_u] \leq \frac{1}{4\phi} \sum_{u \in I} \Pr[X_u] \leq \frac{(4/\epsilon + 4)\zeta}{4\phi} \ . \qquad ◀$$

Let ψ_v denote the set of colors in v's palette before a given round, i.e., the set of colors that have not already been taken by its d2-neighbors. Let $\overline{\psi_v}$ be the set of colors in v's palette that appear on nodes in $\hat{C}$. These colors must then appear only on non-$\hat{H}$-neighbors of v.

▶ **Lemma 2.11.** *Suppose $|\psi_v| \geq 2|\overline{\psi_v}|$ and $|\psi_v| = \Omega(\log n)$. Then, there is an absolute constant c such that each color in $\psi_v \setminus \overline{\psi_v}$ has probability at least $1/(c|\psi_v|)$ of being validated and contested by v in Step 4.*

Proof. Let $\hat{\psi} = \psi_v \setminus \overline{\psi_v}$. Any color from $\hat{\psi}$ that is guessed in Step 4 (by some H-neighbor u of v) becomes a *good* proposal to v (i.e., one that would pass validation). Let A (B) denote the expected number of good (bad) proposals to v, respectively. Let q be a color in $\hat{\psi}$ and let A_q be the expected number of proposals of q to v. We shall show that A_q is large, for colors in $\hat{\psi}$, and thus A is large in comparison to B. We then show that A_q is also large relative to the total number of proposals, $A + B$.

The probability that a decent H-neighbor u chooses to help v is $1/(4\phi)$, and the probability that it guesses q is $1/\Delta^2$. By Lemma 2.9, v has at least $\Delta^2/4$ decent H-neighbors. Summing up, $A_q \geq \sum_u 1/(4\phi) \cdot 1/\Delta^2 \geq 1/(16\phi)$, and $A \geq \sum_{q \in \hat{\psi}} A_q \geq |\hat{\psi}_v|/(16\phi) \geq |\psi_v|/(32\phi)$. By Lemma 2.10, $B \leq (1/\epsilon+1)\zeta/\phi$ and by Prop. 2.2, $|\psi_v| \geq \zeta/(4e^3)$. Thus, $B \leq (128e^3(1/\epsilon+1))A$. We can also bound A from above, summing over the at most Δ^2 H-neighbors and all the colors in v's palette:

$$A \leq \sum_{q' \in \psi_v} \sum_{u \in N_H(v)} \frac{1}{4\phi\Delta^2} = \frac{|\psi_v|}{4\phi} \leq 4|\psi_v| A_q \ .$$

By Markov's inequality, the probability that at most $2(A+B)$ proposals are generated for v is at least $1/2$. The probability that a proposal of q is chosen for validation is then at least

$$\frac{A_q}{4(A+B)} \geq \frac{A/(4|\psi_v|)}{4(1 + 128e^3(1/\epsilon + 1))A} = \frac{1}{16(1 + 128e^3(1/\epsilon + 1))|\psi_v|} \ . \qquad ◀$$

▶ **Lemma 2.12.** *Suppose $|\psi_v| < 2|\overline{\psi_v}|$. Then, there is an absolute constant c such that each color in $\overline{\psi_v}$ has probability at least $1/(c|\psi_v|)$ of being validated and contested by v in Step 7.*

Proof sketch. For success in Step 7, only colors of nodes in $\hat{C}$ that are not d2-neighbors of v count (and by Lemma 2.5(2), there are at most 3ζ such nodes). The proof (in the full version) requires on one hand to lower bound the probability of a given such node is contacted on behalf of v. The technically more involved task is to show that such a query to w will survive the competition, both at w and at v. ◀

Lemma 2.6 follows from Lemmas 2.11 and 2.12.

3 Sub-Logarithmic Distance-2 Coloring

In this section, we extend the algorithm of Sec. 2 and combine it with the *graph shattering* technique [11, 12], which has been used extensively in recent years to get sub-logarithmic-time distributed algorithms for a large number of graph problems (mostly in the LOCAL model). By using this technique in our setting, we prove the following theorem.

Theorem 1.2 (restated). *There is a randomized CONGEST algorithm that d2-colors a graph with Δ^2+1 colors in $O(\log \Delta) + ND_2(\log n) \cdot \operatorname{poly}\log\log n$ rounds, with high probability.*

Here $ND_2(\log n)$ is the sum of $d \cdot c \cdot x$ and the time to that one needs to compute a distance-2 CONGEST-routable network decomposition (with weak cluster diameter d, c color classes and routing parameter x) on subgraphs of size $\operatorname{poly}\log n$ with node identifiers from a space of size $\operatorname{poly} n$ (see Definition 3.3 for the formal definition of such a decomposition).

▶ Remark 3.1. The current state of the art for $ND_2(\log n)$ is $2^{O(\sqrt{\log\log n})}$ [22]. However, the complexity for distance-1 network decompositions that can deal with a large identifier space was improved subsequent to the submission of this manuscript to $\operatorname{poly}\log\log n$ rounds [20]. Before the publication of [20] the complexity in Theorem 1.2 for distance-2 coloring matched the state of the art for distance-1 $(\Delta+1)$-coloring [19]. As the achievements of [20] improve the complexity for distance-1 coloring from $O(\log \Delta) + 2^{O(\sqrt{\log\log n})}$ to $O(\log \Delta) + \operatorname{poly}\log\log n$ there currently is a gap between the complexities of distance-1 and distance-2 coloring. If [20] (or an alternative approach) extends to distance-2 decompositions, and such an extension is very likely, it will match again. In the remaining part of the writeup we use the best known upper bound of $ND_2(\log n) = 2^{O(\sqrt{\log\log n})}$.

From a very high-level point of view, the rough idea of graph shattering applied to our problem is as follows. The algorithm of Sec. 2 consists of $O(\log n)$ individual $O(1)$-round steps, where in each step, each live node gets colored with constant probability. Thus, very roughly, if we just run the algorithm for $O(\log \Delta)$ steps, each node remains uncolored with probability at most $1/\operatorname{poly}(\Delta)$. Further, if nodes succeeded sufficiently independently, after $O(\log \Delta)$ rounds, each node would only have $O(\log n)$ uncolored neighbors. By combining these two properties, one can hope that after $O(\log\log n)$ more rounds, all the remaining live nodes induce components (in G^2) of size at most $\operatorname{polylog} n$. By adapting techniques developed in [11] to our G^2-coloring algorithm, we will show that this indeed (almost) is the case. We call this part of the algorithm, where we reduce the original problem to a problem on components of $\operatorname{polylog} n$ size, the *preshattering phase* of our algorithm.

The remaining problem that we need to solve on the components of size $\operatorname{polylog} n$ is a list coloring problem. Because these problems for each component are on much smaller graphs, they can be solved efficiently by using the best known deterministic algorithm. For the specific setting, where we have small components, but each node still has an ID from the original large ID space, the best known deterministic CONGEST algorithm (that can tolerate such a large ID space and works for G^2) can be obtained by combining a network decomposition algorithm of Ghaffari and Portmann [22] with a recent deterministic CONGEST coloring algorithm of Bamberger, Kuhn, and Maus [4]. It requires $2^{O(\sqrt{\log N})} = 2^{O(\sqrt{\log\log n})}$ time, where $N = \text{polylog}\, n$ is the maximum component size. We call this second phase of solving the remaining list coloring instances on the components the *postshattering phase*.

While the general outline of the algorithm is relatively standard and largely follows the ideas of the distance-1 coloring algorithm for the LOCAL model in [11], there are various challenges that we have to cope with in order to apply the idea in the CONGEST model and to the d2-coloring problem. In [11, 19], the algorithm for the preshattering phase has every live node try a uniformly random color from its current list of available colors repeatedly, which we cannot do in the d2-coloring setting as it is not possible for a live node to learn its list of available colors. We would therefore like to show that the much more involved randomized algorithm of Sec. 2 also has the same shattering properties as the basic "choose-a-random-available-color" algorithm. Unfortunately, this is not obvious and we use a multi-stage algorithm to prove what we need. Greatly simplified, we do the following. We first show that $O(\log\Delta)$ rounds of an adaptation of the algorithm of Sec. 2 suffice to (essentially) reduce the maximum degree of the subgraph of G^2 induced by the live nodes to $O(\log n)$. At this point, it is possible for each live node to learn a sufficiently large list of available colors in $O(\log\Delta)$ rounds and we can now indeed run the basic preshattering algorithm of [11] to reduce the problem to a problem on polylog n-size components.

For the post-shattering phase, while we only have components of $\text{poly}\log n$ size, the input to the problem is still large because each node still has an ID of size $O(\log n)$ bits and because each node has a color list consisting of up to $O(\log n)$ colors from a range of size $O(\Delta^2)$. In order to have an efficient CONGEST algorithm for the problem, we have to reduce both the ID space and the color space of the remaining components. It is sufficient to obtain new node IDs that are unique up to distance $\text{poly}\log\log n$. We can obtain such IDs with $O(\log\log n)$ bits by first applying the network decomposition algorithm of [22] and then assigning unique labels in each cluster. For reducing the color space, we show that in each cluster of the network decomposition, we can efficiently (and deterministically) find a renaming of the colors such that for every node v, all colors in v's list are mapped to distinct new colors and such that the colors are from a space of size $\text{poly}\log n$. For each of the steps, the implementation in G^2 rather than in G adds some additional complications. In the following, we give a detailed overview over all the steps of our algorithm.

3.1 Preshattering: Algorithm Overview and Key Ideas

If $\log n = O(\log\Delta)$, we use the $O(\log n)$-time algorithm of Sec. 2. If $\Delta \le \log n \cdot \text{poly}\log\log n$ we essentially simulate the preshattering algorithm of [11] for G on G^2 and combine it with our postshattering algorithm from Section 3.2. In all other cases, that is, if $\Delta = 2^{o(\log n)} \cap \tilde{\Omega}(\log n)$, we perform the following steps. They can be implemented in $O(\log\Delta) + \text{poly}\log\log n$ rounds (except for the postshattering phase which takes $2^{O(\sqrt{\log\log n})}$ rounds).

Almost Clique Decomposition

1. Compute the ACD exactly in $O(\log\Delta)$ rounds by hashing IDs to $O(\log\Delta)$ bits.
   Guarantee: Nodes know whether they are sparse/dense. Further each dense node knows an identifier of its almost clique.

Color Sparse Nodes

2. Every node (dense or sparse) tries a uniformly random color for $O(\log\Delta)$ rounds.
   Guarantee: All nodes have slack proportional to their sparsity and at most $O(\log n)$ live sparse neighbors.

3. Sparse nodes try $O(\log n)$ random colors simultaneously. In total, trying $O(\log n)$ colors requires sending/receiving $O(\log \Delta \cdot \log n)$ bits to immediate neighbors, which can be sent in $O(\log \Delta)$ rounds (by packing $O(\log n)$ bits in each message).
   Core idea: Each color you try has a constant probability to not be tried by anyone else nor adopted by a neighbor.
   Guarantee: All sparse nodes are colored, w.h.p.

Only dense (intermediate degree) nodes execute the remaining steps.

Degree Reduction of Uncolored Graph

4. Perform $O(\log \Delta)$ iterations of Reduce-Phase.
   Guarantee: Uncolored nodes either have low uncolored degree (at most $\tilde{\Delta}$), or are connected to at most $\tilde{\Delta}$ other high uncolored degree nodes, where $\tilde{\Delta} = O(\log n)$.
5. Estimate uncolored degree with $\Theta(\log n)$ precision.
   Guarantee: Uncolored nodes know whether they have low uncolored degree or not.

Let U^{lo} and U^{hi} be the sets of low and high uncolored degree vertices. All the steps afterwards first take place on U^{lo}, then on U^{hi}.

6. Try $\Theta(\log n)$ color proposals that arrive through parallel Reduce-Phases.
   Core idea: Compressing the messages communicated in a Reduce-Phase into $O(\log \Delta)$ bits. Argue a bound of $O(\log \Delta)$ on the congestion of each edge.
   Guarantee: Nodes with slack $\Omega(\log^2 n)$ become colored, w.h.p. All remaining live nodes then have sparsity $O(\log^2 n)$ (needed for Step 7 and 9).

Shattering Into Small Connected Uncolored Components

7. **Learn your list:** Expand on the method LearnPalette of [23] to have each live node learn a list of at least $d(v) + 1$ available colors from its palette. If the node has sparsity $O(\log n)$, we learn the exact list using LearnPalette as is. Otherwise, we randomly try colors not used in the almost-clique to learn enough available colors.
   Core idea: The bottleneck of the method is sending $O(\log n)$ colors over a single link, i.e., $O(\log n \log \Delta)$ bits. By compressing messages this can be done in $O(\log \Delta)$ rounds.
8. **Shattering:** Perform $O(\log \tilde{\Delta}) = O(\log \log n)$ informed color tries (OneShotColoring).
   Guarantee: Uncolored vertices induce $\text{poly}(\tilde{\Delta}) \log n = \text{poly} \log n$ sized components in G^2, and uncolored vertices know a palette that exceeds their degree.
9. **Add Steiner Nodes:** Add all vertices that link live nodes in different almost cliques. Inside each almost clique, learn all live neighbors IDs through ID-renaming, pick one intermediate node as Steiner node per pair of uncolored nodes in the almost clique.
   Guarantee: $G^2[U]$ connected components are G-connected and of size $N = \text{poly} \log n$.

Postshattering: Before the process, uncolored dense nodes U form small connected components and each node has a palette of size that exceeds its degree. Further, with the Steiner nodes connected components of $G^2[U]$ are G-connected and have $N = \text{poly} \log n$ size. This is enough to apply Lemma 3.2 in Section 3.2 and list color the remaining components in $2^{O(\sqrt{\log N})} = 2^{O(\sqrt{\log \log n})}$ rounds.

Intuition for Correctness and Implementation of the Preshattering Phase: The shattering framework with informed color trials is well established, but we apply it here in an unusual setting where the nodes do not know their palette. Instead, we argue that each live node becomes colored in each iteration with constant probability (bounded away from 0). More strongly, we show that half of the colors of its palette have good probability of becoming the node's color in each round, and this holds independent of what its neighbors do (as long as the unlikely event of too many of them are activated does not happen). Then we show that these conditions are sufficient to leave us with two disjoint subgraphs of live nodes, both of logarithmic degree, which we handle sequentially (we first execute all steps after Step 5 including the postshattering phase for the one subgraph and then for the other subgraph). After conducting additional $O(\log \Delta)$ informed color trials, the uncolored vertices induce polylogarithmic size components. The rest of the coloring can then be completed in the post-shattering phase. The idea of producing two subgraphs of small degree already appeared in [11], but it is significantly easier to show that they cover all uncolored vertices if one can perform informed color trials.

Several further technical complications arise that do not occur for ordinary graph coloring: determining which of the two subgraphs the live node should join; adding Steiner nodes to make the components connected in G (not just in G^2); and learning enough of the palette before the post-shattering phase, even when the palette might be large. All of these steps, however, are implementable within the $O(\log \Delta)$ time bound, with techniques of modest novelty. The key idea for their efficient implementation is to compress the communication so that multiple messages fit in a single CONGEST message. Color values use $\log \Delta$ bits, but we also compress node identifiers into $O(\log \Delta)$ bits, either through hashing or renumbering within a component. This allows us to speed up communication-heavy parts: $O(\log n \cdot \log \Delta)$ bits per edge can be sent in $O(\log \Delta)$ rounds.

All of the above is for dense nodes, for which we have the structure of the almost-clique decomposition to guide us. For sparse nodes, we can use simple uniformed color guessing, first with individual colors and then with parallel color guesses, to finish them off early.

3.2 Postshattering: Algorithm Overview and Key Ideas

The high level idea is to compute a network decomposition $\mathcal{D}$ on each connected component of uncolored vertices to split the components into small diameter clusters. Afterwards, we use the deterministic $(deg + 1)$-list coloring algorithm from [4] on each cluster (iterating through the clusters in an order that is given by $\mathcal{D}$). To obtain an efficient algorithm, we need a network decomposition with two features: a) it handles distance-2 relations, and b) it handles large node identifiers (in comparison with the component sizes). The latter is not handled by the new poly $\log n$ result of Rozhoň and Ghaffari [33]. Hence, we cannot currently reduce the dependence on n in the time complexity to poly $\log\log n$. Instead, the construction of Portmann and Ghaffari [22] handles both of these features. The downside is the resulting time complexity of $2^{O(\sqrt{\log\log n})}$. Further, the runtime of the list-coloring algorithm in [4] depends on the size of the colorspace, and we equip our algorithm with methods to reduce the colorspace before we apply [4].

Preconditions: We are given an n-vertex graph G with maximum degree Δ and a partial $d2$-coloring $\phi : V \to [\Delta^2] \cup \{\bot\}$. Let $U = \{\phi^{-1}(\bot)\} \subseteq V$ be the uncolored vertices. Further, we are given a subset $S \subseteq V$ and $\hat{\Delta} = O(\log n)$ such that:

- Each node $u \in U$ has at most $\hat{\Delta}$ $d2$-neighbors in U.
  This immediately implies that each node in V has at most $\hat{\Delta}$ U-neighbors in G.
- $d2$-connected components of $G^2[U]$ have size $\text{poly}(\hat{\Delta}) \log n = \text{poly} \log n$.
- For any $u \in U$ and any of its $d2$-neighbor $u' \in U$ there exists some $s \in S$ such that s is neighbor of u and u'.
- Let $K = G[U \cup S] \setminus E(G[S])$ the subgraph of G induced by $U \cup S$ without edges between vertices in S. The connected components of K have size at most $N = \text{poly} \log n$. And,
- Each vertex $u \in U$ is equipped with a list L_u of colors that are not used in its $d2$-neighborhood. The size of $|L_u| \leq L \leq O(\log n) \leq N$.

▶ **Lemma 3.2** (Postshattering)**.** *There is a deterministic CONGEST algorithm on communication network G that, under the above assumptions, list colors the nodes in U such that $d2$-neighbors pick distinct colors. The runtime of the algorithm is $2^{O(\sqrt{\log\log n})}$ rounds.*
Lemma 3.2 uses two subroutines from previous work. First, a network decomposition algorithm that works for G^k and does not rely on a small IDspace.

▶ **Definition 3.3** (Network Decomposition, x-CONGEST-routable [3])**.** *A weak $(d(n), c(n))$-*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 $1, \ldots, c(n)$. If the decomposition is equipped with a routing backbone such that one can simulate one round of communication within clusters of G^k in $k \cdot x$ rounds of communication on G (if only clusters of one color class communicate at the same time) the decomposition is called* x-CONGEST-routable*.*

As we only use network decomposition in a blackbox manner we do not detail on the additional backbone structure for routing and refer to [22] which proves the next theorem.

▶ **Theorem 3.4** (Network Decomposition of G^k, [22])**.** *There is a deterministic distributed algorithm that in any N-node network G, which has S-bit identifiers and supports $O(S)$-bit messages for some arbitrary S, computes an x-CONGEST-routable $(g(N); g(N))$-network decomposition of G^k in $k \cdot g(N) \cdot \log^* S$ rounds, for any k and $g(N) = x = 2^{O(\sqrt{\log N})}$.*

Second, a CONGEST algorithm that can list-color graphs efficiently if their diameter, the maximum degree and the color space size are small.

▶ **Theorem 3.5** (Diameter List Coloring, [4])**.** *There is a deterministic CONGEST algorithm that given a list-coloring instance $G = (V, E)$ with color space $[C]$, lists $L(v) \subseteq [C]$ for which $|L(v)| \geq \deg(v) + 1$ holds for all $v \in V$ and an initial m-coloring of G, list-colors all nodes in $O\big(D \cdot \log N \cdot \log C \cdot (\log \Delta + \log m + \log\log C)\big)$ rounds.*
The message size of the algorithm is $O(\log C + \log m + \log \Delta)$.

We will need additional reasoning to execute the algorithm of Theorem 3.5 on parts of $G^2[U]$ while the communication network is G; for that it is essential that we reduce the color space. The core steps of the postshattering phase are as follows.

1. **Network decomposition:** Compute a distance-2 network decomposition $\mathcal{D}$ of connected components in graph K using the algorithm of Theorem 3.4 (or an alternative algorithm).
2. **ID space reduction:** Assign new IDs to vertices in U that are unique within each cluster of $\mathcal{D}$. The size of the IDspace is bounded by the cluster size and by N.
3. **Colorspace reduction:** Within each cluster $\mathcal{C}$ deterministically compute a colorspace reduction $f_C : [\Delta^2] \to \text{poly}\, N$. f is a *colorspace reduction* for the cluster $\mathcal{C}$ if it injectively maps each color list L_u for $u \in \mathcal{C}$.
   Core Idea: A random hash function (from a suitable space of hash functions), in expectation, fails for few vertices of the cluster. We derandomize the process of picking such a random hash function with the method of conditional expectation, similar to [13, 16, 4].
4. **Final $(deg + 1)$-list coloring:** Iterate through the color classes of the network decomposition $\mathcal{D}$ and run the $(deg + 1)$-list coloring algorithm of Theorem 3.5 on each cluster. Care is needed when refining the lists, i.e., when deleting colors of $d2$-neighbors of previously colored clusters.

Formal proofs about the correctness and an efficient implementation of Steps 1–4 are discussed in detail in the full version [24].

References

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

2 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta+1)$ vertex coloring. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 767–786, 2019.

3 Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.

4 Philipp Bamberger, Fabian Kuhn, and Yannic Maus. Efficient deterministic distributed coloring with small bandwidth. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, page 243–252, 2020.

5 Reuven Bar-Yehuda, Keren Censor-Hillel, Yannic Maus, Shreyas Pai, and Sriram V. Pemmaraju. Distributed approximation on power graphs. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, page 501–510, 2020.

6 Leonid Barenboim. Deterministic $(\Delta + 1)$-coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In *Proc. 34th Symp. on Principles of Distributed Computing (PODC)*, pages 345–354, 2015.

7 Leonid Barenboim and Michael Elkin. Deterministic distributed vertex coloring in polylogarithmic time. In *Proc. 29th Symp. on Principles of Distributed Computing (PODC)*, 2010.

8 Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool Publishers, 2013.

9 Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-iterative distributed $(\Delta + 1)$-coloring below Szegedy-Vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. In *Proc. 37th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 437–446, 2018.

10 Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. *SIAM J. on Computing*, 43(1):72–95, 2015.

11 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. In *Proc. 53th Symp. on Foundations of Computer Science (FOCS)*, 2012.

12 József Beck. An algorithmic approach to the Lovaśz local lemma. *Random Structures & Algorithms*, 2:343–365, 1991.

13 Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. Derandomizing local distributed algorithms under bandwidth restrictions. In *Proc. 31st Symp. on Distributed Computing (DISC)*, pages 11:1–11:16, 2017.

14 Yi Jun Chang, Wenzheng Li, and Seth Pettie. An optimal distributed $(\Delta + 1)$-coloring algorithm? In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 445–456, 2018.

15 Imrich Chlamtac and Shay Kutten. A spatial-reuse TDMA/FDMA for mobile multi-hop radio networks. In *Proc. 4th IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 389–394, 1985.

16 Janosch Deurer, Fabian Kuhn, and Yannic Maus. Deterministic distributed dominating set approximation in the CONGEST model. In *Proc. 38th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 94–103, 2019.

17 Michael Elkin, Seth Pettie, and Hsin-Hao Su. $(2\Delta - 1)$-edge-coloring is much easier than maximal matching in the distributed setting. In *Proc. of 26th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 355–370, 2015.

18 Pierre Fraigniaud, Magnús M. Halldórsson, and Alexandre Nolin. Distributed testing of distance-k colorings. In *Proc. 27th Coll. on Structural Information and Communication Complexity (SIROCCO)*, pages 275–290, 2020.

19 Mohsen Ghaffari. Distributed maximal independent set using small messages. In *Proc. 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 805–820, 2019.

20 Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. Improved deterministic network decomposition, 2020. `arXiv:2007.08253`.

21 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *Proc. 59th Symp. on Foundations of Computer Science (FOCS)*, pages 662–673, 2018.

22 Mohsen Ghaffari and Julian Portmann. Improved network decompositions using small messages with applications on MIS, neighborhood covers, and beyond. In *Proc. 33rd Int. Symp. on Distributed Computing (DISC)*, pages 18:1–18:16, 2019.

23 Magnús M. Halldórsson, Fabian Kuhn, and Yannic Maus. Distance-2 coloring in the CONGEST model. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 233–242, 2020.

24 Magnus M. Halldorsson, Fabian Kuhn, Yannic Maus, and Alexandre Nolin. Coloring fast without learning your neighbors' colors, 2020. `arXiv:2008.04303`.

25 David G Harris, Johannes Schneider, and Hsin-Hao Su. Distributed $(\Delta + 1)$-coloring in sublogarithmic rounds. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 465–478, 2016.

26 Ö. Johansson. Simple distributed $\Delta + 1$-coloring of graphs. *Inf. Process. Lett.*, 70(5):229–232, 1999.

27 Fabian Kuhn. Faster deterministic distributed coloring through recursive list coloring. In *Proc. 31st ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1244–1259, 2020.

28 Fabian Kuhn and Roger Wattenhofer. On the complexity of distributed graph coloring. In *Proc. 25th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 7–15, 2006.

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

30 Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. on Computing*, 15:1036–1053, 1986.

31 Steffen Mecke. MAC layer and coloring. In D. Wagner and R. Wattenhofer, editors, *Algorithms for Sensor and Ad Hoc Networks*, pages 63–80, 2007.

32 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

33 Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 350–363, 2020.