Fast Coloring Despite Congested Relays

Maxime Flin
Reykjavik University, Iceland

Magnús M. Halldórsson
Reykjavik University, Iceland

Alexandre Nolin
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract

We provide a $O(\log^6 \log n)$-round randomized algorithm for distance-2 coloring in CONGEST with $\Delta^2 + 1$ colors. For $\Delta \gg \text{poly} \log n$, this improves exponentially on the $O(\log \Delta + \text{poly} \log \log n)$ algorithm of [Halldórsson, Kuhn, Maus, Nolin, DISC'20].

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

Keywords and phrases CONGEST model, distributed graph coloring, power graphs

Digital Object Identifier 10.4230/LIPIcs.DISC.2023.19

Related Version *Full Version*: https://arxiv.org/abs/2308.01359 [17]

Funding *Maxime Flin*: Funded by the Icelandic Research Fund (grant 2310015).
Magnús M. Halldórsson: Partially supported by the Icelandic Research Fund (grant 217965).

1 Introduction

In the LOCAL model of distributed computing, we are given a communication network in the form of an n-node graph $G = (V, E)$, where each node has a unique $O(\log n)$-bit identifier. Time is divided into discrete intervals called rounds, during which nodes send/receive one message to/from each of their neighbors in G. In the CONGEST model, each message additionally is restricted to $O(\log n)$ bits.

Coloring problems are amongst the most intensively studied problems in the distributed graph literature for they capture the main challenges of symmetry breaking and resolving conflicts (see, e.g., [4]). We consider the *distance-2* $\Delta^2 + 1$-coloring problem in CONGEST, where Δ is the maximum degree of G. The task is to assign each node a color from $\{1, 2, \ldots, \Delta^2 + 1\}$ that is different from nodes within distance 2 in G. Namely, we want to color the square graph G^2 while communicating on G with $O(\log n)$-bit messages.

The distance-2 coloring problem is particularly interesting as a petri dish for examining the impact of bandwidth constraints. When we seek a coloring of G, each node can directly communicate with all nodes that it conflicts with, which are precisely its neighbors. In G^2, a node can conflict with Δ^2 other nodes, but only communicate directly with Δ of them. Thus, the bandwidth used by a node is at most $\Delta \log n$ bits, both for incoming and outgoing messages, which can be much smaller than the number of neighbors in G^2. In fact, it is altogether non-trivial to obtain even a $\text{poly}(\log n)$-round algorithm for distance-2 $\Delta^2 + 1$-coloring, which was only achieved in 2020 [27].

Distance-2 coloring is also interesting in its own right. In particular, it arises naturally when assigning frequencies to antennas in wireless networks. More generally, symmetry breaking on power graphs appears naturally in numerous settings [35, 6, 20, 21, 11, 13]. See, e.g., [38, Section 1.2] for a recent treatment.

Finally, the distance-2 coloring problem is the only explicit problem studied when the *conflict graph* H to be colored is different from the *communication graph* G. In the LOCAL model, this distinction is of little concern, as the square graph can be simulated within G with an overhead of factor 2 in the round complexity. This is what motivates the usual assumption that $H = G$. In CONGEST however, bandwidth constraints preclude such local reductions. This is a major challenge toward understanding the complexity landscape in CONGEST. In fact, such local transformations are ubiquitous in the distributed graph literature. Notable examples include reductions to Maximal Independent Set [4, Section 3.9], coloring algorithms based on the Lovász Local Lemma [41, 10, 8], or subroutines working with cluster graphs [24, 40, 13].

Our Contributions. We provide a $\operatorname{poly}\log\log n$-round randomized algorithm to find a distance-2 coloring of G. Our algorithm uses $\Delta^2 + 1$ colors, which is a natural analog to $\Delta + 1$ at distance-1.

▶ **Theorem 1.** *There is a randomized algorithm for distance-2 coloring any n-node graph G with maximum degree Δ, using $\Delta^2 + 1$ colors, and running in $O(\log^6 \log n)$ rounds of* CONGEST.

This is an exponential improvement over the previous best known bound $O(\log n)$ [28], as a function of n alone. Interestingly, for more general power graphs G^k with $k \geq 3$, it is provably hard to verify an arbitrary coloring [18]. Thus, any $\operatorname{poly}\log\log n$ algorithm coloring G^k when $k \geq 3$ and $\Delta \gg \operatorname{poly}\log\log n$ would need a different approach.

Theorem 1 requires non-constructive pseudorandom compression techniques, so can be viewed as either existential or requiring exponential local computation. However, we give an explicit and efficient algorithm that achieves such a coloring with $O(\log^2 n)$ bandwidth. We emphasize that even with $O(\log^2 n)$ bandwidth, it is not clear that fast coloring algorithms can be implemented at distance-2. Our $O(\log^2 n)$-bandwidth algorithm preserves the intuition behind our techniques. In fact, reducing the bandwidth to $O(\log n)$ is a technical issue that is almost entirely solved by previous work [32].

1.1 Related Work

Coloring has been extensively studied in the distributed literature [44, 6, 4, 33, 9, 31], and it was the topic of the paper of Linial [36] that defined the LOCAL model. The best round complexity of randomized $(\Delta + 1)$-coloring in LOCAL (as a function of n alone) progressed from $O(\log n)$ in the 80's [37, 2, 34], through $O(\log^3 \log n)$ [6, 33, 9], to the very recent $\widetilde{O}(\log^2 \log n)$ [22]. These algorithms made heavy use of both the large bandwidth and the multiple-message transmission feature of the LOCAL model.

In CONGEST, Halldórsson, Kuhn, Maus, and Tonoyan [29] gave a $O(\log^5 \log n)$-round CONGEST algorithm, later improved to $O(\log^3 \log n)$ in [32, 25]. Very recently, Flin, Ghaffari, Halldórsson, Kuhn, and Nolin [15] provided a $O(\log^3 \log n)$-round algorithm in *broadcast* CONGEST, in which nodes are restricted to broadcast one $O(\log n)$-bit message per round. While these algorithms drastically reduced the bandwidth requirements compared to their earlier LOCAL and CONGEST counterparts, they still use more bandwidth than what distance-2 coloring allows. Indeed, at distance-2 a node cannot receive a distinct message from each neighbor.

Recent years have seen several results for problems on power graphs in CONGEST [26, 27, 28, 38, 7]. Ghaffari and Portmann [26] gave the first sublogarithmic network decomposition algorithm with a mild dependency on the size of identifiers. Keeping a mild

dependency on the size of identifiers is crucial in CONGEST as a common technique, called *shattering*, is to reduce the problem to small $\operatorname{poly}\log n$-size instances on which we run a deterministic algorithm, typically a network decomposition algorithm. While the instance size decreases exponentially, identifiers remain of size $O(\log n)$ bit. Hence, deterministic algorithms with linear dependency on the size of identifiers, such as [43], yield no sub-logarithmic algorithms. The later $O(\log^5 n)$ CONGEST algorithm by [24] with mild dependency on the ID space was extended by [40] to work on power graphs with exponentially large IDs in time $O(\log^7 n)$. Very recently, [38] gave a $O(k^2 \log\Delta \log\log n + k^4 \log^5 \log n)$ randomized CONGEST algorithm to compute a maximal independent set in G^k. Along the way, [38] extended the faster $\widetilde{O}(\log^3 n)$ network decomposition of [23] to power graphs in CONGEST with a mild dependency on the ID space.

When $(1+\varepsilon)\Delta^2$ colors are available, the distance-2 coloring problem is much easier and is known to be solvable in $O(\log^4 \log n)$ rounds [30]. The first $\operatorname{poly}(\log n)$-round CONGEST algorithm for distance-2 (Δ^2+1)-coloring was given in [27], while a $O(\log\Delta)+\operatorname{poly}(\log\log n)$-round algorithm was given in [28]. The original publication of this last result had a higher dependence in n, later reduced by improved network decomposition results of [23, 38] and a faster deterministic algorithm of [25]. We state this for later use, and give more details in the full version [17, Appendix F]:

▶ **Proposition 2** ([28, Lemma 3.12+3.15] + [38, Appendix A] + [25]). *Let H be a subgraph of G^2 where G is the communication network, and suppose $\Delta(H) \leq \operatorname{poly}\log n$. Suppose each node v, of degree $d_H(v)$ in H, knows a list $L(v)$ of $d_H(v)+1$ colors from some color space $|\mathcal{U}| \leq \operatorname{poly}(n)$. There is a randomized algorithm coloring H in $O(\log^5 \log n)$ rounds of* CONGEST *such that each node receives a color from its list.*

The best bound known for a deterministic CONGEST algorithm using Δ^2+1 colors is $O(\Delta^2 + \log^* n)$ rounds [27]. Very recently, [7] gave a handful of deterministic coloring algorithms on power graphs, including a $O(\Delta^4)$-coloring algorithm in $O(\log\Delta \cdot \log^* n)$ rounds (which is an adaptation of Linial's algorithm) and a $O(\Delta^2)$-coloring algorithm in $O(\Delta\log\Delta + \log\Delta\cdot\log^* n)$ rounds (which is an adaptation of the $O(\sqrt{\Delta}\operatorname{poly}\log\Delta + \log^* n)$ algorithm of [19, 5, 39]) for the distance-2 setting.

Open Problems. Along the way we introduce various tools whose range of application extends to more general coloring problems. In particular, almost all steps of our algorithm work if each $v \in V$ uses colors $\{1, 2, \ldots, \widetilde{d}(v)+1\}$, for $\widetilde{d}(v)$ a locally computable upper bound on degrees. It would be interesting to know if $\widetilde{d}(v)+1$-coloring could be solved. The more difficult list variants of this problem, where nodes must adopt colors from lists of size Δ^2+1 or $\widetilde{d}(v)+1$, are also open. Key aspects of our approach fail for list coloring and, in fact, it is not even known if $\Delta+1$-list-coloring of G is achievable in $\operatorname{poly}\log\log n$ rounds of broadcast CONGEST. It would also be interesting to push the complexity of Δ^2+1-coloring of G^2 down to $O(\log^* n)$ when $\Delta \geq \operatorname{poly}\log n$, to fully match the state of the art at distance one. While we conjecture that minor modifications to our algorithm[1] might be able to reduce its complexity by one or more $\log\log n$ factors, achieving $O(\log^* n)$ would require a new approach, as many steps of our algorithm use $\Omega(\log\log\Delta)$ rounds.

[1] Such as reducing the nodes' uncolored degrees to $O(\log n/\log\log n)$ instead of $O(\log n)$, or slightly reducing the number of layers produced by one of the subroutines (SliceColor).

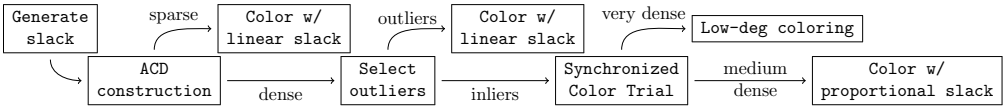

Figure 1 The structure of ultrafast coloring algorithms.

1.2 Our Techniques in a Nutshell

In this section, we highlight the main challenges and sketch the main ideas from our work. Precise definitions are in Section 2 and a more detailed but still high-level overview of our algorithm can be found in Section 3.

Fast Distributed Coloring. All sublogarithmic distributed coloring algorithms [33, 9, 29, 31, 32, 15] follow the overall structure displayed in Figure 1. The key concept is the one of *slack*: the slack of a node is the difference between the number of colors available to that node (i.e., not used by a colored neighbor) and its number of uncolored neighbors (see Definition 3). Nodes with slack proportional to their uncolored degree can be colored fast. The algorithm uses a combination of creating excess colors (by coloring two neighbors with the same color) and reducing the uncolored degree in order for all nodes to get a slack linear in their uncolored degree.

We first generate *slack* by a single-round randomized color trial. We next partition the nodes into the sparse nodes and dense clusters (called *almost-cliques*). Among the dense clusters, we separate a fraction of the nodes as *outliers*. Both the sparse nodes and the outliers can be colored fast using the *linear* slack available to them. The remaining *inliers* then go through a *synchronized color trial* (SCT), where the nodes are assigned a near-uniform random color which avoids color clashes between nodes in the same cluster. The remaining nodes now should have slack proportional to their number of uncolored neighbors. The ultra-dense clusters need a special treatment, but they induce a low-degree graph. The above structure is necessary for high-degree graphs, while for low-degree graphs one can afford less structured methods.

At the outset, several parts of this schema already exist for distance-2 coloring. In particular, generating slack is trivial, a $\text{poly}(\log\log n)$-round algorithm for coloring $\text{poly}(\log n)$-degree graphs is known from [28], and coloring with slack $\Omega(\Delta^2)$ follows from [30]. Almost-clique decompositions (ACD) have been well studied and need only a minor tweak here. We use a particularly simple form of SCT, introduced for the streaming and broadcast CONGEST settings [16, 15]: permute the colors of the *clique palette* – the set of colors not used within the almost-clique – and distribute them uniformly at random among the nodes. We produce the clique palette by giving the nodes data structure access for looking up their assigned color.

Challenges. The biggest challenge in deriving efficient algorithms for the distance-2 setting is that there are no efficient methods known (and possibly not existing) to compute (or approximate) basic quantities like the distance-2 degree of a node. One can easily count the number of 2-paths from a given node, but not how many distinct endpoints they have. This seriously complicates the porting of all previous methods from the distance-1 setting, as we shall see.

A related issue is that a node cannot keep track of all the colors used by its distance-2 neighbors, since it has Δ^2 of them but only bandwidth $O(\Delta \log n)$ bits. Hence, it cannot maintain its true palette (the set of available colors), which means that the standard method of coloring with proportional slack [44, 9] (that can be achieved in $O(\log^* n)$ rounds in distance-1) is not available.

Using Multiple Sources of Slack. We use slack from four sources in our analysis. The (usual) initial slack generation step gives the dense nodes slack proportional to their external degree – their degree to outside of their almost-clique. The method of *colorful matching* [3] provides slack proportional to the average anti-degree of the cluster, where anti-degree counts non-neighbors in one's almost-clique. And finally we get two types of slack for free: the discrepancy between the node's pseudodegree and its true degree on one hand, and the difference between Δ^2 and the pseudodegree on the other hand, where pseudodegrees are easy-to-compute estimates of distance-2 degrees. Only by combining all four sources can we ensure that the final step of coloring with proportional slack can be achieved fast.

Selecting Outliers. The outliers are nodes with exceptionally high degree parameters, either high *external degree* (to the outside of the almost-clique) or *anti-degree* (non-neighbors within the cluster). As we cannot estimate their true values, we work with *pseudodegrees*: the number of 2-paths to external neighbors, or how many additional 2-paths are necessary to reach all anti-neighbors. The selection of outliers is crucial for the success of the last step of the algorithm, where we need to ensure that nodes have true slack proportional to the number of uncolored neighbors. To select outliers, we use a sophisticated filtering technique, giving us bounds in terms of certain related parameters, that then can be linked to the slack that the nodes obtain.

Coloring Fast with Slack. With the right choice of inliers and suitable analysis of SCT, we argue that remaining uncolored nodes have slack proportional to their uncolored degree. We provide a new procedure to color these nodes, extending a method from the first fast CONGEST algorithm [29]. It needs to be adapted to biased sampling and to handle nodes with different ranges of slack. It outputs a series of low-degree graphs, which are then colored by the method of [28].

1.3 Organization of the Paper

After introducing some definitions and results from previous work in Section 2, we give a detailed overview of the full algorithm in Section 3. In Section 4, we go over the technical details involving the coloring of dense nodes, assuming a $O(\log^2 n)$ bandwidth. Various technical parts, as well as details on reducing bandwidth to $O(\log n)$, are in the full version [17].

2 Preliminaries & Definitions

Distributed Graphs. For any integer $k \geq 1$, let $[k]$ represent the set $\{1, 2, \ldots, k\}$. We denote by $G = (V, E)$ the communication network, $n = |V|$ its number of nodes, and Δ its maximum degree. The square graph G^2 has vertices V and edges between pairs $u, v \in V$ if $\text{dist}_G(u, v) \leq 2$. For a node $v \in V$, we denote its unique identifier by $\mathsf{ID}(v)$. For a graph $H = (V_H, E_H)$, the neighborhood in H of v is $N_H(v) = \{u \in V_H : uv \in E_H\}$. A subgraph $K = (V_K, E_K)$ of $H = (V_H, E_H)$ with $V_K \subseteq V_H$ is an *induced* subgraph if

$E_K = \{uv \in E_H : u, v \in V_K\}$, i.e., it contains all edges of E_H between nodes of V_K. We call *anti-edge* in H a pair $u, v \in V_H$ such that $uv \notin E_H$, i.e., an edge missing from H (or, equivalently, in the complement of H).

The degree of v in H is $d_H(v) = |N_H(v)|$, and we shall denote by $N^2(v) = N_{G^2}(v)$ the distance-2 neighbors of v, and the *distance-2 degree* of v by $d(v) = |N^2(v)|$. We also drop the subscript for distance-1 neighbors and write $N(v)$ for $N_G(v)$.

Distributed Coloring. A *partial* c-coloring $\mathcal{C}$ is a function mapping vertices V to colors $[c] \cup \{\perp\}$ such that if $uv \in E$, either $\mathcal{C}(u) \neq \mathcal{C}(v)$ or $\perp \in \{\mathcal{C}(u), \mathcal{C}(v)\}$. The coloring is complete if $\mathcal{C}(v) \neq \perp$ for all $v \in V$ (i.e., all nodes have a color). A $\deg+1$-list-coloring instance is an input graph $H = (V_H, E_H)$ where each node has a list $L(v)$ of $d_H(v)+1$ colors from some color space $\mathcal{U}$. A valid $\deg+1$-list-coloring is a proper coloring $\mathcal{C} : V_H \to \mathcal{U}$ such that $\mathcal{C}(v) \in L(v)$ for each $v \in V_H$.

Our algorithm computes a monotone sequence of partial colorings until all nodes are colored. In particular, once a node *adopts* a color, it never changes it. The palette of v with respect to the current partial coloring $\mathcal{C}$ is $\Psi(v) \stackrel{\text{def}}{=} [\Delta^2 + 1] \setminus \mathcal{C}(N^2(v))$, i.e., the set of colors that are not used by distance-2 neighbors. For a set $S \subseteq V$, we shall denote the uncolored vertices of S by $S^\circ \stackrel{\text{def}}{=} \{v \in S : \mathcal{C}(v) = \perp\}$ and, reciprocally, the colored vertices of S by $S^\bullet \stackrel{\text{def}}{=} S \setminus S^\circ$. We shall denote the uncolored (distance-2) degree with respect to $\mathcal{C}$ by $d^\circ(v) \stackrel{\text{def}}{=} |N^\circ_{G^2}(v)|$.

2.1 Slack Generation

A key notion to all fast randomized coloring algorithm is the one of slack. It captures the number of excess colors: a node with slack s will always have s available colors, regardless of the colors tried concurrently by neighbors. For our problem, the slack is more simply captured by the following definition.

▶ **Definition 3** (Slack). *Let H be an induced subgraph of G^2. The slack of v in H (with respect to the current coloring of G^2) is*

$$s_H(v) \stackrel{\text{def}}{=} |\Psi(v)| - d^\circ_H(v) \ .$$

There are three ways a node can receive slack: if it has a small degree originally, if two neighbors adopt the same color, or if an uncolored neighbor is inactive (does not belong to H). We consider the first two types of slack *permanent* because a node never increases its degree, and nodes never change their adopted color. On the other hand, the last type of slack is *temporary*: if some inactive neighbors become active, the node loses the slack which was provided by those neighbors.

The sparsity of a node counts the number of missing edges in its neighborhood. We stress that, contrary to previous work in $\Delta+1$-coloring [9, 29, 15], we use the *local sparsity* – defined in terms of the node's degree $d(v)$ – as opposed to the global sparsity, instead defined in term of Δ. This is to separate the contribution to slack of same-colored neighbors from the *degree slack*, $\Delta^2 - d(v)$. While global sparsity measures both, local sparsity focuses on the former.

▶ **Definition 4** (Local Sparsity, [1, 31]). *The sparsity of v (in the square graph G^2) is*

$$\zeta_v \stackrel{\text{def}}{=} \frac{1}{d(v)}\left(\binom{d(v)}{2} - |E(N^2(v))|\right) \ .$$

A node v is ζ-sparse if $\zeta_v \geq \zeta$; if $\zeta_v \leq \zeta$ it is ζ-dense.

For a node v, observe that each time that both endpoints of a missing edge in $N^2(v)$ are colored the same, the node v gains slack as its uncolored degree decreases by 2 while its palette loses only 1 color. Therefore, when a node has many missing edges in its neighborhood, it has the potential to gain a lot of slack [42, 12]. This potential for slack is turned into permanent slack by the following simple algorithm (GenerateSlack): each node flips a random coin (possibly with constant bias); each node whose coin flip turned heads picks a color at random and *tries* it, i.e., colors itself with it if none of its neighbors is also trying it. As we state the result with local sparsity (which is in terms of $d(v)$) while nodes try colors in $[\Delta^2+1]$, the next statement has a $d(v)/\Delta^2$ factor compared to previously published versions.

▶ **Proposition 5** (Slack Generation, [42, 12, 29]). *There exists a (small) universal constant $\gamma_{\mathsf{slack}} > 0$ such that after* GenerateSlack*, w.p. $exp(-\Omega(\zeta_v \cdot d(v)/\Delta^2))$, node v receives slack $\gamma_{\mathsf{slack}} \cdot \zeta_v \cdot \frac{d(v)}{\Delta^2}$.*

2.2 Sparse-Dense Decomposition

All recent fast randomized distributed coloring algorithms [33, 9, 29, 31, 14, 15] decompose the graph into a set of sparse nodes and several dense clusters. Such a decomposition was first introduced by [42].

▶ **Definition 6.** *For $\varepsilon \in (0, 1/3)$, a distance-2 ε-almost-clique decomposition (ACD) is a partition of $V(G)$ in sets $V_{\mathsf{sparse}}, K_1, \dots, K_k$ such that*

1. *nodes in V_{sparse} either are $\Omega(\varepsilon^2\Delta^2)$-sparse in G^2 or have degree $d(v) \leq \Delta^2 - \Omega(\varepsilon^2\Delta^2)$,*
2. *for all $i \in [k]$, sets K_i are called* almost-cliques*, and verify*
   a. *$|K_i| \leq (1+\varepsilon)\Delta^2$,*
   b. *for each $v \in K_i$, $|N^2(v) \cap K_i| \geq (1-\varepsilon)\Delta^2$.*

There are several ways to compute this decomposition in CONGEST [28, 29, 32, 16]. We refer the reader to the version of [32, Section 4.2]. The existing distance-2 algorithm of [28] uses $O(\log \Delta)$ rounds and the CONGEST algorithms by [29] require too much bandwidth at distance-2. We mention that [16] implements [32] without representative hash functions and that it can be done here as well. We refer the reader to the full version, [17, Section D], for more details.

▶ **Lemma 7** (Adaptation of [16, Section B.1]). *There exists a* CONGEST *randomized algorithm partitioning the graph into $V_{\mathsf{sparse}}, K_1, \dots, K_k$ for some integer $k \geq 0$ such as described in Definition 6. It runs in $O(\varepsilon^{-4})$ rounds.*

▶ **Definition 8** (External and Anti-Degrees). *For a node $v \in K$ and some almost-clique K, we call $e_v = |N^2(v) \setminus K|$ its* external degree *and $a_v = |K \setminus N^2(v)|$ its anti-degree. We shall denote by $\overline{e}_K = \sum_{v \in C} e_v/|K|$ the average external degree and $\overline{a}_K = \sum_{v \in K} a_v/|K|$ the average anti-degree.*

It was first observed by [29] that sparsity bounds external and anti-degrees.

▶ **Lemma 9** ([29, Lemmas 6.2]). *There exists two constants $C_{\mathsf{ext}} = C_{\mathsf{ext}}(\varepsilon) > 0$ and $C_{\mathsf{anti}} = C_{\mathsf{anti}}(\varepsilon) > 0$ such that for all $v \in K$, the bounds $e_v \leq C_{\mathsf{ext}}\zeta_v$ and $a_v \leq C_{\mathsf{anti}}\zeta_v$ holds.*

2.3 Pseudo-degrees

Bandwidth constraints, such as that of the CONGEST model, can severely restrict nodes in their ability to learn information about their neighborhood in a power graph of G. This includes a node's palette (which colors are not yet used by its neighbors in the power graph) but also its degree and related quantities. This motivates the use of similar, but readily computable quantities.

▶ **Definition 10** (Distance-2 Pseudo-Degrees). *In the distance-2 setting, for any node $v \in V$, let its* pseudo-degree $\widetilde{d}(v)$ *and its* uncolored pseudo-degree $\widetilde{d}^{\circ}(v)$ *be*

$$\widetilde{d}(v) \stackrel{\text{def}}{=} \sum_{u \in N_G(v)} |N_G(u)| \quad \text{and} \quad \widetilde{d}^{\circ}(v) \stackrel{\text{def}}{=} |N_G^{\circ}(v)| + \sum_{u \in N_G(v)} |N^{\circ}_{G \setminus \{v\}}(u)| \ . \tag{1}$$

For a dense node $v \in K$, its pseudo-external degree $\widetilde{e}_v$ *and its* pseudo-anti degree $\widetilde{a}_v$ *are*

$$\widetilde{e}_v \stackrel{\text{def}}{=} \sum_{u \in N_G(v)} |N_G(u) \setminus K| \quad \text{and} \quad \widetilde{a}_v \stackrel{\text{def}}{=} |K| - \sum_{u \in N_G(v)} |N_G(u) \cap K| \ . \tag{2}$$

Note that pseudo-degree and pseudo-external degree are *overestimates* of a node's actual degree and external degree, while pseudo-anti degree is an *underestimate* of a dense node's actual anti-degree. The estimates are accurate for nodes with a tree-like 2-hop neighborhood.

For dense nodes, we also introduce notation for the deviations between the pseudo-degrees and actual G^2-degrees. Such deviations result in slack, which we exploit later in the paper.

$$\theta_v^{\mathsf{ext}} \stackrel{\text{def}}{=} \widetilde{e}_v - e_v \ , \quad \theta_v^{\mathsf{anti}} \stackrel{\text{def}}{=} a_v - \widetilde{a}_v \ , \quad \text{and} \quad \theta_v \stackrel{\text{def}}{=} \theta_v^{\mathsf{ext}} + \theta_v^{\mathsf{anti}} = \widetilde{d}(v) - d(v) \ .$$

We also write $\theta_K^{\mathsf{ext}} = \sum_{v \in K} \theta_v^{\mathsf{ext}} / |K|$ for the average value within a clique.

Pseudo-degrees partially allow nodes to estimate their *degree slack*, the number of colors that v is guaranteed to always have available due to the palette being larger than its degree. Intuitively, the deviations θ_v^{ext} and θ_v^{anti} capture the part of its degree slack that a dense node v does not know about.

$$\underbrace{\Delta^2 + 1 - d(v)}_{\text{degree slack}} = \underbrace{\Delta^2 + 1 - \widetilde{d}(v)}_{\text{known to } v} + \underbrace{\theta_v^{\mathsf{ext}} + \theta_v^{\mathsf{anti}}}_{\text{unknown to } v} \tag{3}$$

3 Detailed Overview of the Full Algorithm

We now give a streamlined overview of our algorithm and describe with some details the technical ideas behind it. See Algorithm 1 for a high-level description of its steps. Since there exists a $O(\log^5 \log n)$-round algorithm when $\Delta \leq \operatorname{poly} \log n$ (Proposition 2), we assume $\Delta \geq \Omega(\log^{3.5} n)$. Henceforth, we assume we are given the almost-clique decomposition $V_{\mathsf{sparse}}, K_1, \ldots, K_k$ (Lemma 7).

Coloring Sparse Nodes (Steps 2 & 3). The coloring of sparse nodes was already handled in [30]. After GenerateSlack, all sparse nodes have slack proportional to Δ^2 (Proposition 5). In particular, their palettes always represent a constant fraction of the color space $[\Delta^2 + 1]$. This allows them to sample colors in their palette efficiently without learning most of their distance-2 neighbors' colors. The algorithm is summarized by the following proposition:

▶ **Proposition 11** (Coloring Nodes with Slack Linear in Δ^2, [30]). *Suppose $\Delta \geq \Omega(\log^{3.5} n)$. Let H be an induced subgraph of G^2 for which all nodes have slack $\gamma \cdot \Delta^2$ for some universal constant $\gamma > 0$ known to all nodes. There exists an algorithm coloring all nodes of H in $O(\log^* n)$ rounds.*

Algorithm 1 High-Level Algorithm.

Input : Graph G with $\Delta \geq \Omega(\log^{3.5} n)$
Output : A distance-2 coloring $\mathcal{C}$ of G

1 $V_{\mathsf{sparse}}, K_1, \ldots, K_k = \mathsf{ComputeACD}(\varepsilon)$ (Section 2.2)
2 GenerateSlack (Proposition 5)
3 ColoringSparseNodes (Proposition 11)
4 Matching ([17, Appendix C])
5 ComputeOutliers ([17, Section 6])
6 ColorOutliers (Proposition 11)
7 SynchColorTrial (Section 4.2)
8 $L_1, \ldots, L_\ell \leftarrow$ SliceColor, for some $\ell = O(\log \log n)$ (Section 4.3 and [17, Section 5])
9 **foreach** $i \in [\ell]$ **do**
10 LearnPalette (Section 4.4)
11 ColorSmallDegree(L_i) (Proposition 2)

Reducing Degrees with Slack. Since coloring sparse nodes is already known, from now on, we focus our attention on dense nodes. Reducing coloring problems to low-degree instances that one then solves with an algorithm that benefits from the low degree is a common scheme in randomized algorithms for distributed coloring [6, 9]. In particular, when nodes have slack linear in their degree, it was observed by [44, 12, 9] that if nodes try *multiple colors from their palette*, degrees decrease exponentially fast, resulting in a $O(\log^* n)$-round algorithm in LOCAL. This observation motivates the structure of all ultrafast coloring algorithms: 1) generate $\Omega(e_v)$ slack with GenerateSlack, 2) reduce degrees to $O(e_v)$ with SynchColorTrial, and 3) complete the coloring with slack. Unfortunately, this approach is not feasible for us because it requires too much bandwidth. As a result, we do something intermediate that takes advantage of slack but only tries a single color at a time to accommodate our bandwidth limitations. In $O(\log \log n)$ rounds, our method creates $O(\log \log n)$ instances of the maximum degree $O(\log n)$.

Another key technical detail of these methods is that nodes try colors *from their palettes*. At distance-2, perfect sampling in one's palette is not feasible for nodes do not have sufficient bandwidth. We show that they can nevertheless sample colors from a good enough approximation of their palette, in the sense that it preserves the slack. Our involved sampling process requires our degree reduction algorithm to work with weaker guarantees than previous work [29, 31].

▶ **Lemma 12** (Slice Color). *Let $C, \alpha, \kappa > 0$ be some universal constants. Suppose each node knows an upper bound $b(v) \geq d^\circ(v)$ on its uncolored degree. Suppose that for all nodes with $b(v) \geq C \log n$, and a value $s(v) \geq \alpha \cdot b(v)$, there exists an algorithm that samples a color $\mathsf{C}_v \in \Psi(v) \cup \{\perp\}$ (where $\perp$ represents failure) with the following properties:*

$$\Pr(\mathsf{C}_v = \perp) \leq 1/\operatorname{poly}(n) \ , \tag{4}$$

$$\Pr(\mathsf{C}_v = c \mid \mathsf{C}_v \neq \perp) \leq \frac{\kappa}{d^\circ(v) + s(v)} \ . \tag{5}$$

Then, there is a $O(\log \log \Delta + \kappa \cdot \log(\kappa/\alpha))$-rounds algorithm extending the current partial coloring so that uncolored vertices are partitioned into $\ell = O(\log \log \Delta)$ layers $L_1, \ldots, L_\ell$ such that each uncolored node knows to which layer it belongs and each $G[L_i]$ has uncolored degree $O(\log n)$.

Coloring Dense Nodes. We assume the sparse nodes are colored (Step 3) and focus on the dense nodes (Steps 4 to 11). Dense nodes receive slack proportional to their external degree (Step 2, Proposition 5 and Lemma 9) in all but the densest almost-cliques.

Steps 4, 5 & 6: Setting up (Section 4.1). We begin by two pre-processing steps to ensure uncolored nodes have useful properties further in the algorithm. Computing a colorful matching (Step 4, Proposition 17) creates $\Theta(\overline{a}_K)$ slack in the clique palette $\Psi(K)$. This is a crucial step to ensure we can approximate nodes palettes (see Step 8). We then compute a (small) fraction $O_K \subseteq K$ of atypical nodes called *outliers* (Step 5, Lemma 14). Outliers have $\Omega(|K|)$ slack from their inactive inlier neighbors, and can thus be colored in $O(\log^* n)$ rounds (Step 6, Proposition 11). *Inliers* $I_K \stackrel{\text{def}}{=} K \setminus O_K$ verify $\widetilde{e}_v \leq O(\overline{e}_K + \theta_K^{\mathsf{ext}})$ and $\widetilde{a}_v \leq O(\overline{a}_K)$ (Equation (6)).

While the colorful matching algorithm is rather straightforward to implement even at distance-2, computing outliers is a surprisingly challenging task. The reason is that, contrary to distance-1, nodes do not know good estimates for $\overline{a}_K$. Fortunately, a node only overestimates its anti-degree (i.e., $\widetilde{a}_v \geq a_v$) and we know that $0.9|K|$ nodes have $a_v \leq 100\overline{a}_K$. By learning approximately the distributions of anti-degrees, we can set a threshold τ such that all nodes with $\widetilde{a}_v \leq \tau$ verify $\widetilde{a}_v \leq 200\overline{a}_K$.

Step 7: Synchronized Color Trial (Section 4.2). This now standard step exploits the small external degree of dense nodes to color most of them. We distributively sample a permutation π of $[|I_K|]$ such that the i-th node in I_K (with respect to any arbitrary order) knows $\pi(i)$ (Lemma 28). Each node then learns the $\pi(i)$-th color in $\Psi(K)$ and tries that color (Lemma 29). This leaves $O(\overline{a}_K + \overline{e}_K + \log n)$ uncolored node in each almost-clique (Lemma 18). To implement these steps, we split nodes into small random groups to spread the workload. The main technical novelty here is an algorithm to aggregate the partial information of each group (Lemma 20).

Step 8: Slice Color (Section 4.3). In the densest almost-cliques, the synchronized color trial already leaves $O(\log n)$ nodes uncolored with uncolored degree $O(\log n)$. In other almost-cliques, nodes have slack proportional to their uncolored degree $O(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$: $\Theta(\overline{a}_K)$ slack from the colorful matching (Step 4), $\Omega(e_v) + \theta_v^{\mathsf{ext}} = \Omega(\widetilde{e}_v)$ from slack generation and pseudo-external-degree. If these are not large enough, it must be that $\Delta^2 - \widetilde{d}(v) \geq \Omega(\overline{e}_K)$, i.e., the node has enough slack from its small degree.

While at distance-1 we could use slack to color fast, doing the same at distance-2 requires more work because nodes do not know their palettes. The first key observation, is that *the clique-palette preserves the slack*. More precisely, for all inliers $v \in I_K$, we have $|\Psi(K) \cap \Psi(v)| \geq d^\circ(v) + \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$ (Lemma 23). The proof of this statement is very technical and requires careful balancing of all four sources of slack: the colorful matching, the sparsity slack, the pseudo-degree slack and the degree slack. We also emphasize this is why inliers need to verify $\widetilde{a}_v \leq O(\overline{a}_K)$: when we use colors from the clique-palette, we lose up to a_v colors used by anti-neighbors, which we compensate using the colorful matching and pseudo-anti-degree slack $\Theta(\overline{a}_K) + \theta_K^{\mathsf{anti}}$.

It remains to sample uniform colors in $\Psi(K) \cap \Psi(v)$. Based on Lemma 23, it can be observed that $|\Psi(K) \cap \Psi(v)| \geq \Omega(|\Psi(K)|)$. Hence, each node v finds a random color $\mathsf{C}_v \in \Psi(K) \cap \Psi(v)$ to try w.h.p. by sampling $\Theta(\log n)$ uniform colors in $\Psi(K)$. With $\Theta(\log^2 n)$ bandwidth, this step can easily be implemented (Lemma 24) by sampling indices in $|\Psi(K)|$ and using the same tools as for the synchronized color trial (Step 7). With $\Theta(\log n)$

bandwidth, we use *representative hash functions* [30]. Intuitively, we use a poly(n)-sized family of hash functions mapping $[\Delta^2+1]$ to some $[\Theta(|\Psi(K)|)]$. To "sample" colors, we take a hash function at random and pick as sampled colors those hashing below $\Theta(\log n)$. Since a hash function h can be described in $O(\log n)$ bits and the hashes $h(\Psi(K)) \cap [\sigma]$ and $h(\mathcal{C}(N(v) \setminus K)) \cap [\sigma]$ can be described using a $O(\log n)$-bitmap, the algorithm works in CONGEST.

The above allows us to apply SliceColor after SynchColorTrial. In $O(\log\log n)$ rounds, we compute $\ell = O(\log\log n)$ layers $L_1, L_2, \ldots, L_\ell$ such that the maximum uncolored degree in each induced graph $G[L_i]$ is $O(\log n)$ (Lemma 12).

Steps 10 & 11: Coloring Small Degree Instances (Section 4.4). We go through each layer $L_1, L_2, \ldots, L_\ell$ sequentially, each time coloring all nodes in L_i. Actually constructing small degree instances for solving with a deterministic algorithm requires the nodes to learn colors from their palette – a tough ordeal in the distance-2 setting. Our argument is two-fold: in not-too-dense almost-cliques, a simple sampling argument works (Lemma 26). In very dense almost-cliques where $\overline{a}_K, \overline{e}_K, \theta_K^{\mathsf{ext}} \leq O(\log n)$, we use a different argument exploiting the very high density of the cluster to disseminate colors fast (Lemma 27). We point out that at this step, it is crucial that uncolored nodes have typical degrees $a_v, e_v \leq O(\log n)$, which is ensured by our inlier selection (Step 5). Once nodes know a list of $d^\circ(v)+1$ colors from their palettes, we can use a small-degree algorithm from [28, 23, 38, 25] to complete the coloring of L_i in $O(\log^5\log n)$ rounds (Proposition 2). Overall, coloring small degree instances needs $O(\log^6\log n)$ rounds, which dominates the complexity of our algorithm.

4 Coloring Dense Nodes

Henceforth, we assume that we are given an ε-almost-clique decomposition $V_{\mathsf{sparse}}, K_1, \ldots, K_k$ for $\varepsilon = 10^{-5}$ [2], where V_{sparse} is already colored. We further assume we ran GenerateSlack and that each node v with $\zeta_v \geq \Omega(\log n)$ has slack $\Omega(\zeta_v)$ (Proposition 5). In this section, we describe an algorithm that colors dense nodes. More formally, we prove the following result:

▶ **Proposition 13** (Coloring Dense Nodes). *After* GenerateSlack *and coloring sparse nodes, there is a $O(\log^6\log n)$-round randomized algorithm for completing a Δ^2+1-coloring of the dense nodes, w.h.p.*

We assume access to $O(\log^2 n)$ bandwidth throughout the rest of the paper, and defer of how to achieve $O(\log n)$ bandwidth to the full version [17, Section 7]. The use of extra bandwidth is very limited and explicitly stated. The reduction in bandwidth only introduces minor changes to the algorithm, and is mostly achieved through techniques from [32].

4.1 Leader, Outliers & Colorful Matching

A useful property of almost-cliques, used by [29, 31, 15], is their relative uniform sparsity. The first step of these algorithms is to dissociate the typical nodes, called *inliers*, from the atypical ones, called *outliers*. At distance-2, however, detecting outliers is difficult. For instance, the algorithm of [15] requires to keep only nodes with anti-degree $a_v \leq O(\overline{a}_K)$. Such a trivial task at distance one requires work at distance-2 because nodes are unable to approximate their degree accurately (up to a constant factor). To circumvent this limitation of the distance-2 setting, we instead compute outliers using *pseudo-degrees* (Definition 10).

[2] Note that we made no attempt to optimize the constants.

▶ **Lemma 14** (Compute Outliers). *We compute in $O(\log\log\Delta)$ rounds a set O_K in each almost-clique K such that $I_K \stackrel{\text{def}}{=} K \setminus O_K$ has size $0.95|K|$ and each $v \in I_K$ verifies that*

$$\widetilde{e}_v \leq 200(\overline{e}_K + \theta_K^{\mathsf{ext}}) \ , \quad \textit{and} \quad \widetilde{a}_v \leq 200\overline{a}_K \ . \tag{6}$$

The general idea behind Lemma 14 is that a large fraction of the almost-clique has a typical sparsity, external degree and anti-degree. By learning approximately the distribution of pseudo-external degrees and pseudo-anti-degrees, the leader can select a large enough fraction of K verifying Equation (6). As the proof of Lemma 14 is quite technical, we defer it to the full version of the paper, [17, Section 6].

Outliers can be colored in $O(\log^* n)$ rounds, thanks to the $\Omega(\Delta^2)$ slack provided by their inactive inlier neighbors. Starting from Section 4.2, we will assume outliers are all colored, thus focus on coloring inliers.

Colorful Matching. A major issue when coloring dense nodes in G^2 is that they do not know their palette. We overcome this by using the *clique palette* as an approximation.

▶ **Definition 15** (Clique Palette). *For an almost-clique K, define its* clique palette *as $\Psi(K) = [\Delta^2+1] \setminus \mathcal{C}(K)$, i.e., the set of colors in $\{1, 2, \ldots, \Delta^2+1\}$ that are not already used by a node of K.*

This idea was first (implicitly) used by [3] to prove their palette sparsification theorem on almost-cliques. This was since used formally in [16, 15]. Note that in large almost-cliques (such that $|K| = (1+\varepsilon)\Delta^2$), the clique-palette can be empty after coloring the outliers. To remedy this issue, [3] compute first a colorful-matching:

▶ **Definition 16** (Colorful Matching). *In a clique K, a colorful matching M is a set of anti-edges in K (edges in the complement) such that both endpoints are colored the same.*

Flin, Ghaffari, Halldórsson, Kuhn and Nolin gave a CONGEST algorithm to compute a colorful matching of size $\Theta(\overline{a}_K/\varepsilon)$ in $O(1/\varepsilon)$ rounds in cliques with a high average anti-degree [15]. We review this algorithm and argue it can be implemented on G^2 with constant overhead in the full version of the paper (see [17, Appendix C]).

▶ **Proposition 17** (Distance-2 Colorful Matching). *Let $\beta \leq O(1/\varepsilon)$. There exists a $O(\beta)$-round randomized algorithm* Matching *that computes a colorful matching of size $\beta\overline{a}_K$ in all almost-cliques of G^2 with $\overline{a}_K \geq \Omega(\log n)$.*

4.2 Synchronized Color Trial

Synchronizing color trials in dense components is a fundamental part of all known sub-logarithmic algorithm [33, 9, 29, 31]. We implement a variant of [31] where a uniform permutation determines which node tries which color. Contrary to [31], we use colors from the clique palette $\Psi(K)$ (Definition 15), which is easier to implement in our setting. This approach was also used by [15] to implement the synchronized color trial in Broadcast-CONGEST. A major difference with [15] is that at distance-2, nodes cannot learn the whole clique-palette $\Psi(K)$.

▶ **Lemma 18** (Synchronized Color Trial, [31]). *Let K be an almost-clique with $|I_K| \geq \Omega(|K|)$ inliers. Fix the randomness outside K arbitrarily. Let π be a uniform random permutation of $[|I_K|]$. If the i-th node in I_K (for any arbitrary order) tries the $\pi(i)$-th color in $\Psi(K)$ (if it exists), then, with high probability, at most $O(\overline{e}_K + \overline{a}_K + \log n)$ are uncolored in K.*

To implement the synchronized color trial, a node v needs only to know $\pi(v)$ and the $\pi(v)$-th color of $\Psi(K)$. We use an approach similar to [15]: randomly partition nodes into groups $T_1, \dots, T_k$ to spread the workload. Concretely, we use the following fact, which is a straightforward consequence of Chernoff and Definition 6.2b.

▶ **Fact 19.** *Let K be an almost-clique and $k \leq |K|/(C \log n)$ for some large enough $C > 0$. Suppose each $v \in K$ samples $t(v) \in [k]$ uniformly at random. Then, w.h.p., each $T_i = \{v \in K : t(v) = i\}$ satisfies that any $u, w \in K$ have $|N^2(u) \cap N^2(w) \cap T_i| \geq (C/4) \log n$. We say set T_i* 2*-hop connects K.*

Note that the two hops mentioned in Fact 19 are in G^2, i.e., for two nodes $u, w \in K$ where T_i 2-hop connects K, u and w can be at distance 4 in G.

Contrary to [15], at distance-2, nodes do not have the bandwidth to learn the whole clique-palette nor the full random permutation. Fortunately, they only need to know their position in the permutation and the one corresponding color. The main technical novelty in our distance-2 implementation lies in an algorithm to compute prefix-sums $\sum_{j<i} x_j$ where each random group T_i holds a value x_i (Lemma 20). We first explain how to aggregate such prefix sums and then show it is enough for implementing the synchronized color trial.

▶ **Lemma 20** (Prefix Sums). *Let $T_1, \dots, T_k \subseteq K$ be disjoint sets that* 2*-hop connect K. If each T_i holds a* $\operatorname{poly}\log n$*-bit integer x_i, then there is a $O(1)$-round algorithm such that for all $i \in [k]$, each $v \in T_i$ learns $\sum_{j<i} x_j$.*

Proof. Compute a BFS tree rooted at some arbitrary $w_K \in K$ and spanning $N^2(w_K) \cap K$. We order distance-2 neighbors of w_K with the *lexicographical order induced by the BFS tree*: distance-2 neighbors $u \in N_{G^2}(w_K)$ are ordered first by $\mathsf{ID}(v)$, where v is the parent of u in the BFS tree, and then by $\mathsf{ID}(u)$. Call $u_1, u_2, \dots, u_{|N_{G^2}(w_K) \cap K|}$ distance-2 neighbors of w_K with respect to that ordering.For each $i \in [k]$, *node u_i learns x_i*. Since T_i 2-hop connects K, there must exist a node $r \in N(u_i) \cap N(T_i)$ which can relay x_i from its neighbor in T_i to u_i. For each distance-1 neighbor $v_j \in N(w_K) \cap K$ of w_K (i.e., depth-1 nodes in the BFS tree), let $u_{i_j}, u_{i_j+1}, \dots, u_{i_{j+1}-1}$ be its children in the BFS tree. Each v_j can learn all values $x_{i_j}, \dots, x_{i_{j+1}-1}$ with a broadcast. Node v_j then sends the sum $S_j \stackrel{\text{def}}{=} \sum_{k=i_j}^{i_{j+1}-1} x_k$ to w_K, which responds with $\sum_{k<j} S_j = \sum_{k<i_j} x_k$. For each child u_{i_j+t} with $0 \leq t \leq i_{j+1} - i_j$, the node v_j communicates

$$\sum_{k<j} S_k + \sum_{k=i_j}^{i_j+t-1} x_k = \sum_{k<i_j+t} x_k$$

to u_{i_j+t}, which is exactly the prefix sum it had to learn. Each u_i can then transmit its prefix sum to T_i using the same path it used to learn x_i. ◀

We briefly sketch the algorithm for implementing the synchronized color trial using Lemma 20 and random groups in the following lemma. See Appendix A for more details.

▶ **Lemma 21.** *Let $v_1, \dots, v_{|I_K|} \in I_K$ be the inliers of some almost-clique K. In $O(\log \log n)$ rounds of* CONGEST*, with high probability,*

1. *we can sample a uniform permutation π of $[|I_K|]$ such that v_i knows $\pi(i)$; and*
2. *each v_i can learn i_v-th color of $\Psi(K)$, for any $i_v \in [\Delta + 1]$ (and fail if $|\Psi(K)| < i_v$).*

Proof Sketch. We begin by explaining how to sample the permutation. Each node $v \in I_K$ picks an integer $t(i) \in [\Theta(|K|/\log n)]$ at random. Let $T_i = \{v \in S : t(v) = i\}$. By Chernoff bound, w.h.p., $|T_i| = O(\log n)$ and 2-hop connects K (Fact 19). In particular, each T_i has

hop-diameter at most 4 and we can relabel nodes of T_i using $O(\log\log n)$-bit labels in $O(1)$ rounds. Using small labels, each T_i can sample a permutation ρ_i of itself in $O(\log\log n)$ rounds. The permutation of I_K is defined by $\pi(v) = \sum_{j<t(v)} |T_j| + \rho_{t(v)}(v)$; hence, using the prefix sum algorithm (Lemma 20), nodes learn $\sum_{j<t(v)} |T_j|$ and have the required information to compute their position in π.

For learning colors, we split nodes into $\Theta(\Delta^2/\log n)$ groups randomly. Random group T_i is tasked with learning which colors in range $R_i = \{i \cdot \Theta(\log n), \dots, (i+1) \cdot \Theta(\log n) - 1\}$ are used by a node in K. With high probability, each T_i 2-hop connects K; thus, using $O(\log n)$-bitmaps and simple aggregation, nodes in T_i learn $R_i \cap \mathcal{C}(K)$ in $O(1)$ rounds. Hence, each node in T_i learns $R_i \setminus \mathcal{C}(K) = R_i \cap \Psi(K)$. By computing prefix sums $\sum_{j<i} |R_j \cap \Psi(K)|$, nodes of T_i learn which range of queries $i \in [|\Psi(K)|]$ they must respond. Since each T_i 2-hop connects K, each node in K has a relay to the group which must answer its query. ◀

4.3 Slack Color (with extra bandwidth)

After the synchronized color trial, uncolored nodes have degree proportional to the slack they received from GenerateSlack (Proposition 5). Contrary to [9, 29, 31], nodes cannot trivially try colors from their palettes, for they lack direct knowledge of it. In this section, we give a solution that uses $O(\log^2 n)$ bandwidth and refer to [17] for the CONGEST implementation. The idea is to sample $\Theta(\log n)$ colors from the clique palette, which is accessible by Lemma 29. Note that this step is needed only in high-sparsity cliques: if $\overline{a}_K + \overline{e}_K \leq O(\log n)$, then its remaining uncolored nodes after the synchronized color trial have degree $O(\log n)$. This motivates the following definition:

▶ **Definition 22** ($\mathcal{K}_{\mathsf{mod}}$, $\mathcal{K}_{\mathsf{very}}$)**.** *Let $C > 0$ be a large enough constant. We say that almost-clique K is* very dense *if $\overline{a}_K < C\log n$ and $\overline{e}_K, \theta^{\mathsf{ext}}_K < 4C\log n$. Reciprocally, we say K is* moderately dense *if it is not very dense. We call $\mathcal{K}_{\mathsf{mod}}$ the set of moderately dense almost-cliques and $\mathcal{K}_{\mathsf{very}}$ the very dense ones.*

Lemma 23 shows that in moderately dense almost-cliques, the clique palette preserves the slack provided by early steps of the algorithm (slack generation, colorful matching and degree slack).

▶ **Lemma 23** (The Clique Palette Preserves Slack)**.** *After* GenerateSlack *and* Matching*, for all inlier $v \in I_K$ with $K \in \mathcal{K}_{\mathsf{mod}}$, we have $|\Psi(v) \cap \Psi(K)| \geq d^\circ(v) + \Omega(\widetilde{e}_v + \overline{e}_K + \overline{a}_K)$. In particular, for any such $v \in I_K$ with $K \in \mathcal{K}_{\mathsf{mod}}$, we have $|\Psi(v) \cap \Psi(K)| \geq \Omega(|\Psi(K)|)$.*

Proof. Clearly, $|K| = |N^2(v) \cap K| + a_v$. We carefully add all contributions to the degree slack of a node

$$\Delta^2 = (\Delta^2 - \widetilde{d}(v)) + \widetilde{d}(v) = |N^2(v) \cap K| + e_v + (\Delta^2 - \widetilde{d}(v)) + \theta^{\mathsf{ext}}_v + \theta^{\mathsf{anti}}_v \ .$$

The clique palette loses one color for each colored node but saves one for each edge in the colorful matching. Recall that K° denotes the uncolored part of K. The clique palette has size at least

$$|\Psi(K)| \geq \Delta^2 - (|K| - |K^\circ|) + |M| \geq |K^\circ| + e_v + |M| - a_v + (\Delta^2 - \widetilde{d}(v)) + \theta^{\mathsf{ext}}_v + \theta^{\mathsf{anti}}_v \ .$$

Let s be the slack received w.h.p. by v after GenerateSlack: if $e_v \geq C\log n$ then $s \stackrel{\text{def}}{=} \Omega(\zeta_v) \geq \Omega(e_v)$ (Proposition 5 and Lemma 9), otherwise $s = 0$. The palette of v is of size at least

$$|\Psi(v)| \geq d^\circ(v) + s + (\Delta^2 - \widetilde{d}(v)) + \theta^{\mathsf{ext}}_v + \theta^{\mathsf{anti}}_v \ .$$

Notice $|\Psi(v) \setminus \Psi(K)| \leq a_v$ and $|\Psi(K) \setminus \Psi(v)| \leq e_v^\bullet$ (recall $e_v^\bullet$ is the number of *colored* external neighbors). A double counting argument bounds the number of colors in both v's palette and the clique palette:

$$\begin{aligned} 2|\Psi(v) \cap \Psi(K)| &= |\Psi(v)| + |\Psi(K)| - |\Psi(v) \setminus \Psi(K)| - |\Psi(K) \setminus \Psi(v)| \\ &\geq d^\circ(v) + |K^\circ| + (e_v - e_v^\bullet) + s + |M| + 2(\theta_v^{\mathsf{anti}} - a_v + \theta_v^{\mathsf{ext}} + \Delta^2 - \widetilde{d}(v)) \\ &\geq 2d^\circ(v) + s + |M| - 2\widetilde{a}_v + 2\theta_v^{\mathsf{ext}} + 2(\Delta^2 - \widetilde{d}(v)) \ , \end{aligned} \tag{7}$$

where the second inequality uses $|K^\circ| + (e_v - e_v^\bullet) \geq d^\circ(v)$ and $\theta_v^{\mathsf{anti}} - a_v = (a_v - \widetilde{a}_v) - a_v = -\widetilde{a}_v$.

The remaining of this proof is a careful case analysis to show that Equation (7) implies $|\Psi(v) \cap \Psi(K)| \geq d^\circ(v) + \Omega(\widetilde{e}_v + \overline{e}_K + \overline{a}_K)$. Each case of our analysis corresponds to a regime for $\overline{a}_K$ and $\overline{e}_K$, since v receives slack from the coloring matching only when $\overline{a}_K > \Omega(\log n)$ (Proposition 17) and from slack generation when $e_v \geq \Omega(\log n)$ (Proposition 5). We defer the detail of this case analysis to Appendix B.

Constant density. Observe that if $|\Psi(K)| > 2e_v$ then $|\Psi(v) \cap \Psi(K)| \geq |\Psi(K)| - e_v \geq |\Psi(K)|/2$. Otherwise if $e_v \geq |\Psi(K)|/2$, we use $|\Psi(v) \cap \Psi(K)| \geq \Omega(e_v)$ (which we just proved) to deduce that $|\Psi(v) \cap \Psi(K)| \geq \Omega(e_v) \geq \Omega(|\Psi(K)|)$. ◀

Lemma 24 is the main implication of Lemma 23. It states that we can use random sampling in the clique palette, instead of nodes' palettes, to try colors in SliceColor (Lemma 12). In particular, after SynchColorTrial (Step Algorithm 1 in Algorithm 1), SliceColor with the sampling process described in Lemma 24 reduces degrees to $O(\log n)$ in $O(\log \log n)$ rounds.

▶ **Lemma 24.** *There is an $O(1)$-round algorithm (using $O(\log^2 n)$ bandwidth) that when run after* GenerateSlack *and* Matching*, achieves the following: It samples a random color $\mathsf{C}_v \in \Psi(v) \cup \{\perp\}$ for all uncolored dense nodes $v \in K \in \mathcal{K}_{\mathsf{mod}}$ such that $\Pr(\mathsf{C}_v = \perp) \leq 1/\operatorname{poly}(n)$ and $\Pr(\mathsf{C}_v = c) \leq \frac{1}{d^\circ(v) + \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e_v})}$ for all colors $c \in \Psi(v) \cap \Psi(K)$.*

Proof. Fix a node $v \in K$. Nodes of K can learn $|\Psi(K)|$ by Lemma 29. Then v samples $x = \Theta(\log n)$ indices in $[|\Psi(K)|]$. By Lemma 29, using $O(\log^2 n)$ bandwidth, each node can learn in $O(1)$ rounds the colors corresponding to the indices they sampled. They broadcast this list of colors (using $O(\log^2 n)$ bandwidth) and drop all colors used by neighbors (i.e., that are not in their palette). Finally, node v picks C_v uniformly at random among the remaining ones. Since $|\Psi(K) \cap \Psi(v)| \geq \Omega(|\Psi(K)|)$, by sampling $\Theta(\log n)$ colors, we sample at least one color $\Psi(K) \cap \Psi(v)$ with high probability (i.e., $\Pr(\mathsf{C}_v = \perp) \leq 1/\operatorname{poly}(n)$). To argue about the uniformity (Equation (5)), we observe that sampling $x = \Theta(\log n)$ indices in $[|\Psi(K)|]$ and then trying a random one of those is equivalent to sampling a uniform permutation π of $[|\Psi(K)|]$ (the x sampled indices are $\pi^{-1}(1), \dots, \pi^{-1}(x)$) and trying the color $c \in \Psi(K) \cap \Psi(v)$ with the smallest $\pi(c)$ (if $\min \pi(\Psi(K) \cap \Psi(v)) < x$). Hence, if we call $Z = \min \pi(\Psi(K) \cap \Psi(v))$, we have

$$\begin{aligned} \Pr(\mathsf{C}_v = c) &= \Pr(Z < x \ \wedge \ \pi(c) = Z) \\ &\leq \Pr(\pi(c) = Z) = \frac{1}{|\Psi(K) \cap \Psi(v)|} \\ &\leq \frac{1}{d^\circ(v) + \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)} \ . \qquad \text{(by Lemma 23)} \end{aligned}$$

◀

4.4 Learning Small Palettes (with extra bandwidth)

Assume we are given sets $L_1, \dots, L_\ell$ for some $\ell = O(\log\log n)$ such that the maximum uncolored degree in each $G[L_i]$ is at most $O(\log n)$. We explain how nodes learn a list $L(v)$ of $d^\circ(v)+1$ colors in their palette, with respect to the current coloring of G^2.

▶ **Lemma 25** (Learn Palette). *Let H be an induced subgraph of G^2 with maximum uncolored degree $O(\log n)$. There is a $O(\log\log n)$-round algorithm at the end of which each node in H knows a set $L(v) \subseteq \Psi(v)$ of $d^\circ_H(v)+1$ colors with high probability.*

The argument is two-fold, we deal with $v \in K \in \mathcal{K}_{\mathsf{mod}}$ nodes and very dense nodes $v \in K \in \mathcal{K}_{\mathsf{very}}$ separately. We again assume $O(\log^2 n)$ bandwidth. The $O(\log n)$ bandwidth argument can be found in [17].

Moderately Dense Almost-Cliques. Using the sampling algorithm from Lemma 24, nodes can sample $C\log n$ many colors in their palette in $O(1)$ rounds, for any arbitrarily large constant $C > 0$. Since uncolored degrees in H are $O(\log n)$, this suffices for Lemma 25.

▶ **Lemma 26.** *Let H be an induced subgraph of G^2 with maximum uncolored degree $C'\log n$ for a large constant $C' > 0$. There is a $O(1)$-round algorithm (using $O(\log^2 n)$ bandwidth) for $v \in K \in \mathcal{K}_{\mathsf{mod}}$ to learn a list $L(v)$ of $d^\circ_H(v)+1$ colors from their palettes.*

Very Dense Almost-Cliques. In very dense almost-cliques, the clique palette $\Psi(K)$ does not approximate their palette well enough: Lemma 23 does not apply. We correct for that by adding colors used by anti-neighbors. They filter out colors used by external neighbors with $O(\log^2 n)$ bandwidth, because they have $O(\log n)$ such neighbors.

▶ **Lemma 27.** *Suppose each $K \in \mathcal{K}_{\mathsf{very}}$ has $O(\log n)$ uncolored nodes (hence $d^\circ(v) \le O(\log n)$ for all $v \in K \in \mathcal{K}_{\mathsf{very}}$). There is a $O(\log\log n)$-round randomized algorithm (using $O(\log^2 n)$ bandwidth) for all uncolored nodes $v \in K \in \mathcal{K}_{\mathsf{very}}$ to learn a list $L(v)$ of $d^\circ_H(v)+1$ colors from their palettes.*

Proof. By repeating messages randomly, we can broadcast any $O(\log^2 n)$-many messages to all nodes in K (see Lemma 31). In particular, when $|\Psi(K)| = O(\log^2 n)$, all nodes in K learn all colors in $\Psi(K)$ in $O(1)$ rounds (see Lemma 32). If $|\Psi(K)| \ge \Omega(\log^2 n)$, nodes learn in $O(1)$ rounds a set $D \subseteq \Psi(K)$ of $\Theta(\log^2 n)$ colors.

Assume first that nodes learned all colors of $\Psi(K)$. Recall nodes have $e_v = O(\log n)$ external neighbors (because $K \in \mathcal{K}_{\mathsf{very}}$ and $v \in I_K$); hence, they can learn all colors used by their external neighbors in $O(1)$ rounds by using $O(\log^2 n)$ bandwidth. Since each uncolored v knows $\Psi(K)$ and the colors of its external neighbors, it thereby knows $\Psi(K) \cap \Psi(v)$.

With a BFS, we can relabel uncolored node of K in the range $[O(\log n)]$. Since uncolored nodes are inliers, they have anti-degree $a_v \le O(\overline{a}_K) \le O(\log n)$ each. At most $O(\log^2 n)$ nodes in K are anti-neighbors of (at least one) uncolored node. We can run $O(\log n)$ BFS in parallel, one rooted at each uncolored node, such that each node knows to which uncolored node it is connected (at distance-2). This takes $O(\log\log n)$ rounds, even with bandwidth $O(\log n)$, because each BFS uses $O(\log\log n)$-bit messages (thanks to the relabeling) and we run $O(\log n)$ of them. Then, the $O(\log^2 n)$ nodes with an uncolored anti-neighbor can describe their list of uncolored anti-neighbors using a $O(\log n)$-bitmap. Using Lemma 31, they broadcast this information as well as their color to all nodes.

Nodes use lists $L(v) \stackrel{\text{def}}{=} (\Psi(K) \cup \mathcal{C}(K \setminus N^2(v))) \cap \Psi(v)$, i.e., the clique palette augmented with the colors of their anti-neighbors, minus colors used by external neighbors. Adding $\Delta^2+1 \geq |N^2(v) \cap K| + e_v$ and $|K| \leq |N^2(v) \cap K| + a_v$, we get $|\Psi(K)| \geq \Delta^2+1-(|K|-|K^\circ|) \geq |K^\circ| + 1 + e_v - a_v$. Since each colored external neighbor removes as most one color, lists have size (recall $e_v^\bullet$ and $a_v^\bullet$ are the *colored* external degree and anti-degrees respectively)

$$|L(v)| \geq |\Psi(K)| - e_v^\bullet + a_v^\bullet \geq |K^\circ| + (e_v - e_v^\bullet) - (a_v - a_v^\bullet) + 1 = d^\circ(v) + 1 \ .$$

Suppose we are in the second case of Lemma 26, i.e., nodes learn a set $D \subseteq \Psi(K)$ of $\Theta(\log^2 n)$ colors. Nodes set $L(v) = D \setminus \mathcal{C}(N^2(v) \setminus K) = D \cap \Psi(v)$, i.e., remove colors used by external neighbors. Since they have $e_v \leq O(\log n)$, this provides large enough lists. For the detailed analysis, see the end of Appendix C. ◀

References

1. Noga Alon and Sepehr Assadi. Palette sparsification beyond $(\Delta + 1)$ vertex coloring. In *APPROX/RANDOM*, volume 176 of *LIPIcs*, pages 6:1–6:22. LZI, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.6.
2. 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. doi:10.1016/0196-6774(86)90019-2.
3. Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta+1)$ vertex coloring. In *SODA*, pages 767–786. SIAM, 2019. doi:10.1137/1.9781611975482.48.
4. Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool Publishers, 2013. doi:10.2200/S00520ED1V01Y201307DCT011.
5. Leonid Barenboim, Michael Elkin, and Uri Goldenberg. Locally-iterative distributed $(\Delta + 1)$-coloring and applications. *J. ACM*, 69(1):5:1–5:26, 2022. doi:10.1145/3486625.
6. Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM*, 63(3):20:1–20:45, 2016. doi:10.1145/2903137.
7. Leonid Barenboim and Uri Goldenberg. Speedup of distributed algorithms for power graphs in the CONGEST model. Technical Report 2305.04358, arXiv, 2023. doi:10.48550/arXiv.2305.04358.
8. Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. The complexity of distributed edge coloring with small palettes. In *SODA*, pages 2633–2652. SIAM, 2018. doi:10.1137/1.9781611975031.168.
9. Yi-Jun Chang, Wenzheng Li, and Seth Pettie. Distributed $(\Delta+1)$-coloring via ultrafast graph shattering. *SIAM J. Computing*, 49(3):497–539, 2020. doi:10.1137/19M1249527.
10. Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. *Distributed Comput.*, 30(4):261–280, 2017. doi:10.1007/s00446-016-0287-6.
11. Michael Elkin and Shaked Matar. Near-additive spanners in low polynomial deterministic CONGEST time. In *PODC*, pages 531–540. ACM, 2019. doi:10.1145/3293611.3331635.
12. Michael Elkin, Seth Pettie, and Hsin-Hao Su. $(2\Delta - 1)$-edge-coloring is much easier than maximal matching in the distributed setting. In *SODA*, pages 355–370. SIAM, 2015. doi:10.1137/1.9781611973730.26.
13. Salwa Faour, Mohsen Ghaffari, Christoph Grunau, Fabian Kuhn, and Václav Rozhoň. Local distributed rounding: Generalized to MIS, matching, set cover, and beyond. In *SODA*, pages 4409–4447. SIAM, 2023. doi:10.1137/1.9781611977554.ch168.
14. Manuela Fischer, Magnús M. Halldórsson, and Yannic Maus. Fast distributed Brooks' theorem. In *SODA*, pages 2567–2588. SIAM, 2023. doi:10.1137/1.9781611977554.ch98.

15 Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. Coloring fast with broadcasts. In *SPAA*, pages 455–465. ACM, 2023. doi:10.1145/3558481.3591095.

16 Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. A distributed palette sparsification theorem. Technical Report 2301.06457, arXiv, 2023. doi:10.48550/arxiv.2301.06457.

17 Maxime Flin, Magnús M. Halldórsson, and Alexandre Nolin. Fast coloring despite congested relays. Technical Report 2308.01359, arXiv, 2023. Full version of this paper. doi:10.48550/arxiv.2308.01359.

18 Pierre Fraigniaud, Magnús M. Halldórsson, and Alexandre Nolin. Distributed testing of distance-k colorings. In *SIROCCO*, volume 12156 of *LNCS*, pages 275–290. Springer, 2020. doi:10.1007/978-3-030-54921-3_16.

19 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In *FOCS*, 2016. doi:10.1109/FOCS.2016.73.

20 Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *SODA*, pages 270–277. SIAM, 2016. doi:10.1137/1.9781611974331.ch20.

21 Mohsen Ghaffari. Distributed maximal independent set using small messages. In *SODA*, pages 805–820. SIAM, 2019. doi:10.1137/1.9781611975482.50.

22 Mohsen Ghaffari and Christoph Grunau. Faster deterministic distributed MIS and approximate matching. *STOC*, abs/2303.16043, 2023. doi:10.48550/arXiv.2303.16043.

23 Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhoň. Improved distributed network decomposition, hitting sets, and spanners, via derandomization. In *SODA*, pages 2532–2566. SIAM, 2023. doi:10.1137/1.9781611977554.ch97.

24 Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. Improved deterministic network decomposition. In *SODA*, 2021. arXiv:2007.08253.

25 Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *FOCS*, pages 1009–1020. IEEE Computer Society, 2021. doi:10.1109/FOCS52979.2021.00101.

26 Mohsen Ghaffari and Julian Portmann. Improved network decompositions using small messages with applications on MIS, neighborhood covers, and beyond. In *DISC*, volume 146 of *LIPIcs*, pages 18:1–18:16. LZI, 2019. doi:10.4230/LIPIcs.DISC.2019.18.

27 Magnús M. Halldórsson, Fabian Kuhn, and Yannic Maus. Distance-2 coloring in the CONGEST model. In *PODC*, pages 233–242. ACM, 2020. doi:10.1145/3382734.3405706.

28 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Alexandre Nolin. Coloring fast without learning your neighbors' colors. In *DISC*, pages 39:1–39:17. LZI, 2020. doi:10.4230/LIPIcs.DISC.2020.39.

29 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In *STOC*, pages 1180–1193. ACM, 2021. doi:10.1145/3406325.3451089.

30 Magnús M. Halldórsson and Alexandre Nolin. Superfast coloring in CONGEST via efficient color sampling. *Theor. Comput. Sci.*, 948:113711, 2023. doi:10.1016/j.tcs.2023.113711.

31 Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. Near-optimal distributed degree+1 coloring. In *STOC*, pages 450–463. ACM, 2022. doi:10.1145/3519935.3520023.

32 Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. Overcoming congestion in distributed coloring. In *PODC*, pages 26–36. ACM, 2022. doi:10.1145/3519270.3538438.

33 David G. Harris, Johannes Schneider, and Hsin-Hao Su. Distributed $(\Delta + 1)$-coloring in sublogarithmic rounds. *Journal of the ACM*, 65:19:1–19:21, 2018. doi:10.1145/3178120.

34 Öjvind Johansson. Simple distributed $\Delta + 1$-coloring of graphs. *Inf. Process. Lett.*, 70(5):229–232, 1999. doi:10.1016/S0020-0190(99)00064-2.

35 Sven Oliver Krumke, Madhav V. Marathe, and S. S. Ravi. Models and approximation algorithms for channel assignment in radio networks. *Wirel. Networks*, 7(6):575–584, 2001. doi:10.1023/A:1012311216333.

36 Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

37 M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Computing*, 15:1036–1053, 1986. doi:10.1137/0215074.

38 Yannic Maus, Saku Peltonen, and Jara Uitto. Distributed symmetry breaking on power graphs via sparsification. In *PODC*, pages 157–167. ACM, 2023. full version available at arxiv:2302.06878. doi:10.1145/3583668.3594579.

39 Yannic Maus and Tigran Tonoyan. Linial for lists. *Distributed Comput.*, 35(6):533–546, 2022. doi:10.1007/s00446-022-00424-y.

40 Yannic Maus and Jara Uitto. Efficient CONGEST algorithms for the Lovász local lemma. In *DISC*, volume 209 of *LIPIcs*, pages 31:1–31:19. LZI, 2021. doi:10.4230/LIPIcs.DISC.2021.31.

41 Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Inf. Comput.*, 243:263–280, 2015. doi:10.1016/j.ic.2014.12.018.

42 Bruce A. Reed. ω, Δ, and χ. *J. Graph Theory*, 27(4):177–212, 1998. doi:10.1002/(SICI)1097-0118(199804)27:4<177::AID-JGT1>3.0.CO;2-K.

43 Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *STOC*, pages 350–363. ACM, 2020. doi:10.1145/3357713.3384298.

44 Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In *PODC*, pages 257–266. ACM, 2010. doi:10.1145/1835698.1835760.

A Missing Details in Synchronized Color Trial

In this section, we expand on the proof sketch of Lemma 21 in the main text and fill in some of the missing details of how we implement Synchronized Color Trial in the distance-2 setting. Similarly to how we used random groups to compute prefix sums in Lemma 20, permuting the nodes (Lemma 28) and learning the colors used in the almost-clique (Lemma 29) are performed by assigning nodes randomly to groups which each perform a chunk of the workload. Assembling together the work done in each group is done using our algorithm for computing prefix sums (Lemma 20), which was the most novel part of the implementation. Algorithmic ideas behind Lemmas 28 and 29 are very similar to the ones of [15], and we discuss them more briefly.

▶ **Lemma 28** (Permute). *There is an algorithm that samples a uniform permutation π of $[|I_K|]$ in $O(1)$ rounds with high probability. The i-th node in I_K (with respect to any ordering where v knows its index) learns $\pi(i)$.*

Proof. Each node $v \in I_K$ picks an integer $t(i) \in [\Theta(|K|/\log n)]$ at random. Let $T_i = \{v \in S : t(v) = i\}$. By Chernoff bound, w.h.p., $|T_i| = O(\log n)$ and 2-hop connects K (Fact 19). In particular, each T_i has hop-diameter at most 4. Let w_i be the node of minimum ID in T_i. Each T_i computes a spanning tree rooted at its w_i. This is performed in parallel for all groups, by having nodes forward the minimum ID they received from a group T_i to other members of T_i. Note that an edge only needs to send information concerning the two groups of its endpoints. Each T_i then relabels itself using small $O(\log\log n)$-bit identifiers in the range $[|T_i|]$. w_i samples a permutation ρ_i of $|T_i|$ and broadcasts it to T_i. Since the permutation of a group needs $O(\log n) \times O(\log\log n)$ bits, after $O(\log\log n)$ rounds each $v \in T_i$ knows $\rho_i(v)$. Then, using Lemma 20, each v learns $\sum_{j<i} |T_j|$. Finally, node v sets its position to $\pi(v) = \sum_{j<i} |T_j| + \rho_i(v)$. ◀

▶ **Lemma 29** (Free Color). *Suppose each node in $v \in K$ holds an integer $i_v \in [\Delta^2+1]$. There is $O(1)$-round algorithm at the end of which each v knows the i_v-th color of $\Psi(K)$ (with respect to any globally known total order of $\Psi(K)$). Furthermore, all nodes can learn $|\Psi(K)|$ in the process.*

Proof. Each node $v \in K$ picks an integer $t(v) \in [\Theta(\Delta^2/\log n)]$. Let $T_i = \{v \in K : t(v) = i\}$. Again, w.h.p., $|T_i| = O(\log n)$ and T_i 2-hop connects K. Each node broadcasts its color (if it adopted one) and its group number $t(v)$. Let $R_i = \{i \cdot \Theta(\log n), \dots, (i+1) \cdot \Theta(\log n) - 1\}$. Let $S_{u,i} = R_i \cap \mathcal{C}(N(u) \cap K)$ be the colors from range R_i used by neighbors of u. For each $i \in [k]$, node u can describe $S_{u,i}$ to each neighbor in T_i using a $O(\log n)$-bitmap. Since each T_i has diameter 4 and 2-hop connects K, after $O(1)$ rounds of aggregation on bitmaps using a bitwise OR, each node in T_i knows $R_i \cap \mathcal{C}(K)$, i.e., all colors from range R_i used in K. Note that this also allows them to compute $R_i \setminus \mathcal{C}(K) = R_i \cap \Psi(K)$, i.e., the colors of R_i that are *not* used by a node of K. By Lemma 20, nodes of T_i learn $\sum_{j<i} |R_j \cap \Psi(K)|$ in $O(1)$ rounds. Finally each v broadcasts i_v and each $u \in T_i$ broadcasts i, $\sum_{j<i} |R_j \cap \Psi(K)|$ and $R_i \setminus \mathcal{C}(K)$. Since each set T_i 2-hop connects K, if the i_v-th color of $\Psi(K)$ belongs to range R_i (i.e., $\sum_{j<i} |R_j \cap \Psi(K)| \leq i_v < \sum_{j \leq i} |R_j \cap \Psi(K)|$), then there exists a $u \in T_i$ and $w \in N(u) \cap N(v)$ which knows both i_v and the color it corresponds to. Then w can transmit that information to v.

To learn $|\Psi(K)|$, nodes aggregate the sum of all $|R_i \cap \Psi(K)|$. This can easily done with a BFS (and electing a leader in each group to avoid double counting). ◀

B Missing Details in the Proof of Lemma 23

▶ **Lemma 23** (The Clique Palette Preserves Slack). *After* GenerateSlack *and* Matching*, for all inlier $v \in I_K$ with $K \in \mathcal{K}_{\mathsf{mod}}$, we have $|\Psi(v) \cap \Psi(K)| \geq d^\circ(v) + \Omega(\widetilde{e}_v + \overline{e}_K + \overline{a}_K)$. In particular, for any such $v \in I_K$ with $K \in \mathcal{K}_{\mathsf{mod}}$, we have $|\Psi(v) \cap \Psi(K)| \geq \Omega(|\Psi(K)|)$.*

In this section, we give the details of a case analysis which we skipped in the proof of Lemma 23 in the main text. In Section 4.3, we show Equation (7):

$$2|\Psi(v) \cap \Psi(K)| \geq 2d^\circ(v) + s + |M| - 2\widetilde{a}_v + 2\theta_v^{\mathsf{ext}} + 2(\Delta^2 - \widetilde{d}(v)) \ .$$

To complete the proof, we show that Equation (7) implies that

$$|\Psi(v) \cap \Psi(K)| \geq d^\circ(v) + \Omega(\widetilde{e}_v + \overline{e}_K + \overline{a}_K) \ .$$

Henceforth, slack implicitly refers to the slack in the clique palette, i.e., node v has slack x if $|\Psi(v) \cap \Psi(K)| \geq d^\circ(v) + x$. When both quantities are too small, the following fact implies that nodes must have slack from a low degree.

▶ **Fact 30.** *If $\overline{a}_K, \widetilde{e}_v \leq \overline{e}_K/4$, then $\Delta^2 - \widetilde{d}(v) > \overline{e}_K/2$.*

Proof. For all $v \in K$, we have $|K| = |N^2(v) \cap K| + a_v$ and $\Delta^2 = (\Delta^2 - \widetilde{d}(v)) + |N^2(v) \cap K| + \theta_v + e_v$, Equation (8) holds:

$$\Delta^2 - |K| = (\Delta^2 - \widetilde{d}(v)) + \theta_v + e_v - a_v = (\Delta^2 - \widetilde{d}(v)) + \widetilde{e}_v - \widetilde{a}_v \ . \tag{8}$$

Since this holds for all nodes, it also holds on average:

$$\Delta^2 - |K| \geq \theta_K + \overline{e}_K - \overline{a}_K \ . \tag{9}$$

We conclude by replacing Equation (9) in Equation (8):

$$\begin{aligned}\Delta^2 - \widetilde{d}(v) &\geq (\Delta^2 - |K|) - \widetilde{e}_v && \text{(by Equation (8))}\\ &\geq \overline{e}_K - \overline{a}_K - \widetilde{e}_v && \text{(by Equation (9))}\\ &\geq \overline{e}_K/2\ . && \text{(because } \overline{a}_K, \widetilde{e}_v \leq \overline{e}_K/4)\end{aligned}$$

◀

When one or both of the quantities are larger, we do a case analysis with four cases. Large anti-degree implies that the colorful matching provides slack w.h.p. Large external degree implies that GenerateSlack created slack at the beginning of the algorithm, w.h.p. A careful analysis allows to claim that sufficient slack is guaranteed to exist w.h.p.

Case 1 (high anti-degree, low external degree). If $\overline{a}_K > C\log n$ and $\overline{e}_K < 4C\log n$. We compute a colorful matching of size $|M| \geq 402\overline{a}_K$. Thus, all nodes have slack $|M| - 2\widetilde{a}_v \geq 2\overline{a}_K$, because $\widetilde{a}_v \leq 200\overline{a}_K$ for all inliers (Lemma 14). If $e_v \geq \overline{a}_K > C\log n$, then v receives slack $\Omega(e_v)$ from slack generation; hence it has $\Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$ slack by Equation (7). Otherwise, if $\overline{a}_K > e_v$, it gets enough slack from the colorful matching.

Case 2 (high anti-degree and external degree). If $\overline{a}_K > C\log n$ and $\overline{e}_K \geq 4C\log n$. Similarly to case 1, nodes have slack $\overline{a}_K$. If $\overline{a}_K > \overline{e}_K/4$ or $\theta_v^{\mathsf{ext}} \geq \overline{e}_K/8$, then it has enough slack. Finally, if $e_v > \overline{e}_K/8 > \Omega(\log n)$, then v received $\Omega(e_v)$ slack from GenerateSlack; hence has slack $\Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$. The only remaining possibility is that $\overline{a}_K, \widetilde{e}_v \leq \overline{e}_K/4$. Then, Fact 30 shows that $\Delta^2 - \widetilde{d}(v) \geq \overline{e}_K/2 \geq \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$ and we are done.

Case 3 (low anti-degree, high external degree). If $\overline{a}_K < C\log n$ and $\overline{e}_K > 4C\log n$. If $e_v > \overline{e}_K/8 \geq \Omega(\log n)$, then v has slack $\theta_v^{\mathsf{ext}} + \Omega(e_v) \geq \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$ from slack generation, so we are done. If $\theta_K^{\mathsf{ext}} > \overline{e}_K/8$, then again we are done. Otherwise, $\overline{a}_K, \widetilde{e}_v \leq \overline{e}_K/4$ and by Fact 30 we conclude that all nodes have enough degree slack.

Case 4 (low anti-degree and external degree). If $\overline{a}_K < C\log n$ and $\overline{e}_K < 4C\log n$. Since $K \in \mathcal{K}_{\mathsf{mod}}$, it must be that $\theta_K^{\mathsf{ext}} > 4C\log n$. If $\widetilde{e}_v$ is greater than $\theta_K^{\mathsf{ext}}/8$, then v has slack $\Omega(\widetilde{e}_v) \geq \Omega(\widetilde{e}_v + \theta_K^{\mathsf{ext}}) \geq \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$ and we are done. So we can assume $\overline{a}_K, \widetilde{e}_v < \theta_K^{\mathsf{ext}}/4$. We argue that the degree slack must be large. Similarly to Fact 30, we have

$$\begin{aligned}\Delta^2 - \widetilde{d}(v) &\geq (\Delta^2 - |K|) - \widetilde{e}_v && \text{(by Equation (8))}\\ &\geq \theta_K^{\mathsf{ext}} - \overline{a}_K - \widetilde{e}_v && \text{(by Equation (9))}\\ &\geq \theta_K^{\mathsf{ext}}/2 \geq \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)\ . && \text{(by assumption)}\end{aligned}$$

C Random Broadcast in Almost-Cliques

In this section, we explain how nodes in an almost-clique can all learn $\Theta(\log^2 n)$ messages each originating from a different node in $O(1)$ rounds, as used in the proof of Lemma 27. We use the following broadcast primitive: Each node forwards along each outgoing edge a (independently) random message received.

▶ **Lemma 31** (Distance-2 Many-to-All Broadcast). *Let K be an almost-clique in G^2 and $S \subseteq K$ be a subset of k vertices such that each $x \in S$ has a message m_x. Suppose $\Delta \geq k\log n$ and $\Delta^2 \geq k^3 \log n$. After four rounds of the broadcast primitive, every node in K received all messages $\{m_x\}_{x\in S}$, w.h.p.*

Proof. Let $u \in S$ and $v \in K$. Recall that u and v both have at least $(1-\varepsilon)\Delta^2$ d2-neighbors in K. Since $|K| \leq (1+\varepsilon)\Delta^2$, there are at least $(1-3\varepsilon)\Delta^2$ common d2-neighbors of u and v in K. Let $W \stackrel{\text{def}}{=} N^2(u) \cap N^2(v)$ be this set.

We attribute a unique "relay" to each node $w \in W$, connecting it to v. For each $w \in W$, let r_w be the common d1-neighbor of w and v of lowest ID. For each d1-neighbor r of v, let $W_r \subseteq W$ be the nodes of W for which r is the chosen relay to v. Assume $\varepsilon < 1/12$. Using that $|W_r| \leq \Delta$, and by a simple Markov-type argument, there are at least $(1-6\varepsilon)\Delta \geq \Delta/2$ d1-neighbors r of v for which $|W_r| \geq \Delta/2$. Let R be the set of those "heavy" relays.

After the first round, each d1-neighbor of u receives m_u. Consider some heavy relay $r \in R$. Each node $w \in W_r$ receives the message m_u from a d1-neighbor it shares with u with probability at least $1/k$, independently from other nodes. Thus, with probability at least $1 - \exp(-\Delta/(24k)) = 1 - 1/\operatorname{poly}(n)$, $\Delta/(4k)$ or more nodes in W_r receive m_u.

Assume at least $\Delta/(4k)$ nodes in W_r received m_u. Then, in the third round of the broadcast primitive, r fails to receive m_u with probability at most:

$$\left(1 - \frac{1}{k}\right)^{\Delta/(4k)} \leq e^{-\Delta/(4k^2)} .$$

When $\Delta/(4k^2) \geq 1/2$, this probability is bounded by $e^{-1/2} \leq 2/3$. In that case, $\Delta/6$ or more nodes in R should receive m_u in expectation, and so at least $\Delta/12$ heavy relays receive m_u w.p. $1 - \exp(-\Delta/72)$. The probability that those relays all fail in sending m_u to v in the fourth round of the broadcast primitive is at most $(1-1/k)^{\Delta/72} \leq \exp(-\Delta/(72k)) = 1/\operatorname{poly}(n)$.

If $\Delta/(4k^2) \leq 1/2$, then $e^{-\Delta/(4k^2)} \leq 1 - \Delta/(8k^2)$. In expectation, at least $\Delta^2/(16k^2)$ heavy relays receive m_u, and $\Delta^2/(32k^2)$ of them receive m_u w.h.p. All those relays fail to send m_u to v with probability at most $(1-1/k)^{\Delta^2/(32k^2)} \leq \exp\bigl(-\Delta^2/(32k^3)\bigr) = 1/\operatorname{poly}(n)$. ◀

In particular, this allows us to learn all colors remaining in the clique palette, because at this step of the algorithm, only $\operatorname{poly}\log n$ colors should remain available in $\Psi(K)$. If not, we learn nonetheless a set of $\operatorname{poly}\log n$ colors from $\Psi(K)$ which will act as a replacement.

▶ **Lemma 32.** *Assume $\Delta \geq \Omega(\log^{3.5} n)$. There is an $O(1)$-round algorithm (with $O(\log n)$ bandwidth) such that, in each almost-clique K, either*

1. *all nodes $v \in K$ learn all colors in $\Psi(K)$, or*
2. *all nodes learn a set $D \subseteq \Psi(K)$ of $\Theta(\log^2 n)$ colors.*

Proof. If $|\Psi(K)| \leq O(\log^2 n)$, then let $D \stackrel{\text{def}}{=} \Psi(K)$. Otherwise, if $|\Psi(K)| \geq \Omega(\log^2 n)$, let D be the $\Theta(\log^2 n)$ first colors of $\Psi(K)$. Recall that all nodes can learn $|\Psi(K)|$ in $O(1)$ rounds (Lemma 20); hence, nodes know in which of the two case they are.

Assign indices of $[|\Psi(K)|]$ to arbitrary nodes $u_1, \dots, u_{|D|}$ of K (with a BFS for instance). This is feasible because $|K| \geq \Delta^2/2 > \Theta(\log^2 n) = |D|$. Then, each u_i learns the i-th color of D in $O(1)$ rounds (Lemma 29). Each u_i then crafts a message m_i containing that color and distributes it to all nodes of K by Many-to-All broadcast. By assumption, there are only $|D| = O(\log^2 n)$ messages, and since $|K| \geq \Delta^2/2 \geq \Theta(\log^7 n) = |D|^3 \times \Theta(\log n)$, we meet the requirements of Lemma 31. Thus, in $O(1)$ rounds, all the nodes in K know all colors of D. ◀

This immediately leads to a good approximation $L(v) = D \cap \Psi(v) = D \setminus \mathcal{C}(N^2(v) \setminus K)$ for $v \in K \in \mathcal{K}_{\mathsf{very}}$ after synchronized color trial. Suppose that we are in the second case of Lemma 32, i.e., nodes learn a set $D \subseteq \Psi(K)$ of $\Theta(\log^2 n)$ colors. Since $K \in \mathcal{K}_{\mathsf{very}}$, the

average node has few external connections, $\overline{e}_K + \theta_K^{\mathsf{ext}} = O(\log n)$ (Definition 22). Moreover, because uncolored nodes are all inliers, $e_v = O(\overline{e}_K + \theta_K^{\mathsf{ext}})$ (Equation (6)). Finally, node v loses at most one color in D per external neighbor, hence

$$|D \cap \Psi(v)| \geq |D| - e_v \geq \Theta(\log^2 n) - O(\overline{e}_K + \theta_K^{\mathsf{ext}}) \geq \Theta(\log^2 n) \geq d^\circ(v) + 1 \ .$$

The last inequality holds because, at this point of the algorithm, nodes have $d^\circ(v) = O(\log n)$.

D Proof of Theorem 1

In this section, we put together all results from other sections to prove our main theorem. Only missing are the technicalities of reducing the bandwidth from $O(\log^2 n)$ to $O(\log n)$, which we tackle in the full version [17, Section 7].

Proof. Let $C > 0$ be some large universal constant. By Lemma 7, computing the almost-clique decomposition $V_{\mathsf{sparse}}, K_1, \ldots, K_k$ of G^2 takes $O(1)$ rounds of CONGEST (Step 1). Generating slack and coloring V_{sparse} takes $O(\log^* n)$ rounds (Propositions 5 and 11). Putting together all results from Section 4, we prove the proposition stated earlier, which implies Theorem 1.

▶ **Proposition 13** (Coloring Dense Nodes). *After* GenerateSlack *and coloring sparse nodes, there is a $O(\log^6 \log n)$-round randomized algorithm for completing a $\Delta^2 + 1$-coloring of the dense nodes, w.h.p.*

Steps 4, 5 & 6. By Proposition 17, we compute a colorful matching of size $402\overline{a}_K$ in $O(1)$ rounds, in all almost-cliques with $\overline{a}_K > C \log n$. By Lemma 14, we can compute sets O_K and $I_K = K \setminus O_K$ in all almost cliques, such that all $v \in I_K$ verify Equation (6) and $|I_K| > (1-5/100)|K|$. Let H_1 be the subgraph of G^2 induced by $\bigcup_K O_K$. Note that each $v \in O_K$, for some almost-clique K, has at least $(1-5/100)|K| - \varepsilon\Delta^2 > (1-5/100)(1-\varepsilon)\Delta^2 - \varepsilon\Delta^2 \geq \Delta^2/2$ neighbors in I_K, for ε small enough. Hence, each outlier has $\Delta^2/2$ slack in H_1 and can thus be colored in $O(\log^* n)$ rounds by Proposition 11.

Step 7 & 8. Order nodes of I_K with a BFS. By Lemma 28, w.h.p., the i-th node of I_K can learn $\pi(i)$, where π is a uniformly random permutation of $[|I_K|]$. By Lemma 29, each node can learn, thus try, the i-th color of $\Psi(K)$ (if it exists). With high probability, by Lemma 18, each almost-clique K has $O(\overline{a}_K + \overline{e}_K + \log n)$ uncolored nodes. This implies, the uncolored degree of a dense node $v \in K$ is $O(e_v + \overline{a}_K + \overline{e}_K + \log n)$. In particular, if $v \in K \in \mathcal{K}_{\mathsf{very}}$ (Definition 22), it has uncolored degree $d^\circ(v) \leq O(\log n)$. Since Step 8 intend to reduce the uncolored degree to $O(\log n)$, we can focus on moderately dense almost-cliques. Let $H_2 = \bigcup_{K \in \mathcal{K}_{\mathsf{mod}}} K$. The following fact shows conditions of Lemma 12 are verified by the sampler of Lemma 24.

▶ **Fact 33.** *There exists a universal constant $\alpha > 0$ such that $s(v) \geq \alpha \cdot b(v)$ for all $v \in H_2$, where $b(v) = \widetilde{e}_v + |K^\circ|$ and $s(v) \geq \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$ from Lemma 24 such that $\mathsf{Pr}(\mathsf{C}_v = c) \leq \frac{1}{d^\circ(v)+s(v)}$.*

Proof of Fact 33. Let K be the almost-clique of v. The quantity $b(v)$ only requires $O(1)$ rounds to compute: To compute its pseudo-external degree $\widetilde{e}_v$, a node only needs to receive from each of its direct neighbors $u \in N_G(v)$ the value $|(N_G(u) \cup \{u\}) \setminus K|$; for $|K^\circ|$, the number of uncolored nodes in K, a simple BFS within K suffices to count $|K^\circ|$ and broadcast it to the whole almost-clique.

We now show $b(v)$ satisfies the hypotheses. After SCT, by Lemma 18, at most $O(\overline{e}_K + \overline{a}_K + \log n)$ nodes are left uncolored in K, so $b(v) \in O(\widetilde{e}_v + \overline{e}_K + \overline{a}_K + \log n)$. By Lemma 24, there exist $s(v) \in \Omega(\overline{a}_K + \overline{e}_K + \widetilde{e}_v)$ s.t. Equation (5) holds. If $b(v) \leq C \log n$, then $d^\circ(v) < C \log n$, hence the uncolored degree is already $O(\log n)$. Otherwise, when $b(v) \geq C \log n$, it must be that $b(v) \in \Theta(\widetilde{e}_v + \overline{e}_K + \overline{a}_K)$, and so, there exists a universal constant α s.t. $s(v) \geq \alpha \cdot b(v)$. ◀

Hence, we can use the sampling algorithm of Lemma 24 to run SliceColor (Lemma 12) in H_2. Therefore, in $O(\log \log n)$ rounds, we produce a coloring and a partition $L_1, \ldots, L_\ell$ of uncolored nodes in H_2 such that the maximum uncolored degree of each $G[L_i]$ for $i \in [\ell]$ is $O(\log n)$. We also define $L_0 = \bigcup_{K \in \mathcal{K}_{\mathsf{very}}} K$ which has maximum uncolored degree $O(\log n)$ after the synchronized color trial.

Steps 10 & 11. We go through layers $L_0, L_1, \ldots, L_\ell$ sequentially. In L_0, nodes learn lists of $\deg+1$ colors from their palette by Lemma 27. In each L_i for $i \in [\ell]$, nodes are moderately dense, hence learn their palette from sampling by Lemma 26. Solve each of these $\deg+1$-list-coloring instance of Proposition 13 with the small degree algorithm of Proposition 2. Since learning palettes takes $O(\log \log n)$ and each $\deg+1$-list-coloring instance is solved in $O(\log^5 \log n)$, the total round complexity of this step is $O(\log^6 \log n)$, which dominates the complexity of the algorithm. ◀