Coloring Fast with Broadcasts

Coloring Fast with Broadcasts

Maxime Flin
Reykjavik University
Reykjavik, Iceland
maximef@ru.is

Mohsen Ghaffari
MIT
Cambridge, MA, USA
ghaffari@mit.edu

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

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

Alexandre Nolin
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
alexandre.nolin@cispa.de

ABSTRACT

We present an $O(\log^3 \log n)$-round distributed algorithm for the $(\Delta + 1)$-coloring problem, where each node *broadcasts* only one $O(\log n)$-bit message per round to its neighbors. Previously, the best such broadcast-based algorithm required $O(\log n)$ rounds. If $\Delta \in \Omega(\log^3 n)$, our algorithm runs in $O(\log^* n)$ rounds. Our algorithm's round complexity matches the state-of-the-art in the much more powerful CONGEST model [Halldórsson et al., STOC'21 & PODC'22], where each node sends one different message to each of its neighbors, thus sending up to $\Theta(n \log n)$ bits per round. This is the best complexity known, even if message sizes are unbounded.

Our algorithm is simple enough to be implemented in even weaker models: we can achieve the same $O(\log^3 \log n)$ round complexity if each node reads its received messages in a streaming fashion, using only $O(\log^3 n)$-bit memory. Therefore, we hope that our algorithm opens the road for adopting the recent exciting progress on sublogarithmic-time distributed $(\Delta + 1)$-coloring algorithms in a wider range of (theoretical or practical) settings.

CCS CONCEPTS

• **Theory of computation** → **Distributed algorithms**; • **Mathematics of computing** → *Graph coloring*.

KEYWORDS

CONGEST model, distributed graph coloring

ACM Reference Format:
Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. 2023. Coloring Fast with Broadcasts. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '23), June 17–19, 2023, Orlando, FL, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3558481.3591095

SPAA '23, June 17–19, 2023, Orlando, FL, USA

ACM ISBN 978-1-4503-9545-8/23/06…$15.00
https://doi.org/10.1145/3558481.3591095

1 INTRODUCTION

The coloring problem and its distributed motivations. Our focus is on $\Delta + 1$-coloring: the problem of assigning one color from $\{1, \ldots, \Delta{+}1\}$ to each node, such that no two neighboring nodes have the same color. Here Δ denotes the maximum degree of the graph. Coloring plays a pivotal role in distributed systems, as a clean way to divide access to non-shareable resources, resolve contention, and break symmetries. For instance, it is particularly important in wireless networking, for frequency allocation or channel assignment. A characteristic of wireless communication is that nodes broadcast their messages (reception is constrained by interference from other broadcasts).

Distributed models. The coloring problem has been studied extensively in distributed computing [9, 13, 22–25, 27–30, 36, 41]. Indeed, this problem was the subject of the celebrated paper by Linial [33], which introduced the LOCAL model of distributed computing. In this model, n processors form a graph $G = (V, E)$ where an edge exists only between processors that can communicate. The resulting graph is called the communication graph G and is the one to be colored. Per round, each node can send one unbounded-size message to each of its neighbors. The variant where the message sent to each neighbor is bounded to $O(\log n)$ bits is known as the CONGEST model [38].

Distributed coloring. Classic distributed algorithms for coloring [30, 34] achieved complexity $O(\log n)$ in the CONGEST model. There has been exciting recent progress on sublogarithmic time algorithms [9, 13, 23–25, 27–29], and the state of the art round complexity is $O(\log^3 \log n)$ rounds. This is also the best known in the more relaxed LOCAL model, which allows unbounded message sizes. However, unlike the earlier algorithm of [30], these faster algorithms make some nodes send one different message to each of their neighbors. Thus, each node may send up to $\Theta(n \log n)$ bits in one round. The research question at the core of this paper is to understand the extent to which one can compute a coloring fast if we constrain the set of outgoing messages. Specifically,

> *Can we compute a $(\Delta + 1)$-coloring as fast as in the* CONGEST *model if, in each round, each node must transmit the same $O(\log n)$-bit message to all its neighbors?*

To the best of our knowledge, with this restriction, the best round complexity known in general graphs remains the classic $O(\log n)$ bound [9, 30, 34].

1.1 Our Results

We give a fast $\Delta + 1$-coloring algorithm in the *broadcast congest* model (or BCONGEST) where, per round, each node *broadcasts one $O(\log n)$-bit message* to all of its neighbors.

Theorem 1. *Let $G = (V, E)$ be any n-node graph with maximum degree at most Δ. There is a distributed $O(\log^3(\log n))$-round algorithm that $\Delta + 1$-colors G with high probability, where each node broadcasts one $O(\log n)$-bit message in each round. If $\Delta \in \Omega(\log^3 n)$, the algorithm runs in $O(\log^* n)$ rounds.*

As a side remark, we note that the $O(\log n)$ complexity was the best bound known for general graphs even in the much more relaxed *broadcast congested clique* model, in which each node can send a $O(\log n)$ bit message to all other nodes. To emphasize, in this model, the communication graph is a complete graph and every two nodes are neighbors. The coloring is still with respect to the input graph G. This model is also sometimes known as the *shared blackboard model* with simultaneous messages and the *distributed sketching* model [5, 6, 16]. Our $O(\log^3 \log n)$-round complexity improves nearly exponentially over existing algorithms in this model.[1]

Even more basic models? The overarching goal in our work is not tied to any particular model. We would like to develop a distributed algorithm that assumes the least provided power from the theoretical model. The hope is that this makes the algorithm applicable in a wider range of (theoretical or practical) settings. To that end, we point out that our algorithm is basic enough to be implemented even with limited memory per node, with only small additional changes. Notice that a node may receive many messages from its neighbors, up to $\Omega(n \log n)$ bits overall in one round. In general, receiving so many bits would necessitate a significant memory for the node, and it also can complicate the task of simulating this algorithm in virtual graphs. [2] We show that our algorithm can be adapted to work with the same round complexity when each node processes its incoming messages in a streaming fashion, using only $\mathrm{poly}(\log n)$ memory. We refer to this model as BCStream. See Section 5 for a formal definition of the model.

Theorem 2. *There is a distributed $O(\log^3(\log n))$-round algorithm in* BCONGEST *for $\Delta + 1$-coloring graphs with high probability, even if each node reads its received messages through a stream and only has* $\mathrm{poly}(\log n)$ *memory.*

[1] If we increase the size of the message sent by each node in this BCC model from $O(\log n)$ to $O(\log^3 n)$ bits, then a celebrated work of Assadi, Khanna, and Chen [4] provides a one round algorithm.

[2] For instance, consider a frequent scenario in distributed graph algorithms: a virtual graph is formed by contracting low-depth clusters of the network, each forming one node of the virtual graph. Two clusters are neighbors if they contain adjacent network nodes. Usually, the communications of each cluster should be sent along a low-depth tree that spans the nodes of the cluster. If all the $\Omega(n \log n)$ bits should be delivered to the cluster center, this can require $\Omega(n)$ rounds, even for low-depth clusters.

1.2 Technical Contributions

1.2.1 Previous Algorithms & Challenges. We summarize the key concepts in previous fast coloring algorithms and emphasize the parts that do not work in the BCONGEST model.

A basic primitive in randomized coloring algorithms is a *random color trial*: each node selects a color from its *palette* (its set of available colors) uniformly at random and keeps the color if none of its neighbors picked the same. The (permanent) *slack* of a node is the excess number of colors in its palette compared to its degree. Sufficient slack speeds up coloring dramatically: each node can try *multiple colors* in each round, resulting in a $O(\log^* n)$-round coloring algorithm called MultiTrial [41]. As a color requires up to $O(\log n)$ bits to describe, trying more than a constant number of them is infeasible with $O(\log n)$ bandwidth. A solution by [28] was to use pseudorandomness: say each v tries a set of colors X_v, then v broadcasts a hash function h_v which each neighbor u of v uses to reply $h_v(X_u)$. A color that collides under h_v with none of its neighbors is safe to adopt. However, this approach requires individual responses $h_v(X_u)$ from each neighbor u. Therefore it does not work with single-message broadcasts.

> *Challenge 1: How can we perform* MultiTrial *with $O(\log n)$-bit broadcasts? The previous approaches [28, 41] require either large messages or individual responses.*

Slack can be generated for nodes with a sparse neighborhood, i.e., with $\Omega(\Delta^2)$ missing edges. The more difficult task in distributed $\Delta + 1$-coloring algorithms is to color the *dense nodes*. They can be partitioned into dense clusters called almost-cliques. The second key concept for fast coloring is to *synchronize* the colors tried within each almost-clique, in the following sense: the color suggested to each node should be random from the viewpoint of the nodes outside the almost-clique, but there should be no conflicts between nodes inside the almost clique. The earlier version of synchronized color trial (SCT for short) involved gathering all the information of the almost-clique for centralized processing [13, 29], requiring high bandwidth. A simpler form of SCT of [27] has a leader node permute its own palette and distribute the colors to the other nodes of the almost-clique. This still requires different messages to be sent along the different edges from the leader, making it incompatible with BCONGEST.

> *Challenge 2: How can we synchronize color trials with $O(\log n)$-bit broadcasts? The previous approaches [13, 27, 29] require either centralization or a node sending up to $\Omega(\Delta)$ messages.*

Finally, MultiTrial requires $\ell = \Omega(\log^{1+\Omega(1)} n)$ slack in order to fully color the graph with high probability. This is solved in [27] by putting aside mutually non-adjacent sets of ℓ nodes in very dense cliques, to be colored at the very end. [27] colors put-aside sets by gathering all their relevant information (list of uncolored neighbors and palette) and broadcasting the coloring from a leader node.

> *Challenge 3: How can we color the put-aside sets with $O(\log n)$-bit broadcasts? The previous approach [27] does not work as they require full information gathering and dissemination.*

Observe that Challenges 1 and 3 can easily be solved by increasing the bandwidth to a small $\mathrm{poly}(\log n)$. On the other hand, Challenge 2 seems to require greater effort to implement with the broadcast constraint, even with $\mathrm{poly}(\log n)$ bandwidth.

1.2.2 Our Algorithm. In this section, we give an overview of our solutions to each of the challenges described earlier.

Multi-Color Trial. A subset of a *known* universe can be sampled pseudorandomly in BCONGEST [26]. The problem is that when MultiTrial is applied after SCT, each node has a different palette, which is unknown to its neighbors. We solve this by *reserving* a subset of the color space for use by MultiTrial. Namely, each node v reserves the subset $[x(v)] = \{1, 2, \ldots, x(v)\}$, where $x(v)$ is a function of v's neighborhood density. Both slack generation and the synchronized color trial within v's almost-clique are restricted to using colors outside $[x(v)]$. The key is then to show that: a) using the colors $[\Delta + 1] \setminus [x(v)]$ suffices for these steps, and b) enough colors in $[x(v)]$ remain unused (by neighbors of v) for MultiTrial to succeed.

Synchronized Color Trial. Our solution for the synchronized color trial of an almost-clique K is to use the *clique palette* of K: the set of colors not used by nodes in K. We randomly permute this set, in a distributed manner, and assign each color to a single uncolored node of K. This introduces two types of errors: a) not all nodes receive a color to try, and b) nodes can receive non-usable colors (as a node's neighbors outside of K might already be using its assigned color). However, the errors are within acceptable bounds, and we are still able to show that after SCT, each node has an uncolored degree that is at most proportional to its slack, allowing for fast mop-up by MultiTrial.

To learn the clique palette $\Psi(K)$ in an almost-clique K, we randomly assign nodes of K into groups such that: a) every node is adjacent to at least one node of each group, and b) each group is connected and has a low diameter. Each group is tasked with learning a part of the clique palette, which it teaches to the rest of the almost-clique K.

We also randomly assign nodes into groups to randomly permute K. The random assignment roughly positions each node within the output permutation π. Each group, of much smaller size than K, then randomly permutes its members. The small size of each group, combined with relabeling its members with smaller IDs, makes the description of a permutation of its members fit within small bandwidth.

Coloring Put-Aside Sets. The put-aside set P_K of an almost-clique K has no edges to the put-aside sets in other almost-cliques. As such, coloring P_K can be done purely within K. Our algorithm first reduces the size of each P_K to sublogarithmic. Then, it gathers information about what remains of each P_K. One randomized color trial reduces $|P_K|$ by a constant factor with probability $1-e^{-\Theta(|P_K|)}$. We compress the equivalent of $O(\log\log n)$ iterations of this process into $O(1)$ rounds by sampling the colors of all iterations *in advance* and sending them all at once. To reach sublogarithmic size *with high probability*, we run $O(\log\log n)$ independent iterations in parallel. We avoid congestion issues by using few colors per iteration and by representing colors with few bits.

1.3 Related Work

Distributed $\Delta + 1$-Coloring. The best round complexity of randomized LOCAL $(\Delta+1)$-coloring, as a function of only the number of nodes n, progressed from $O(\log n)$ in the 80's [3, 30, 34], through $O(\sqrt{\log n})$ [29], to a recent $O(\log^3 \log n)$ [13]. The more recent work [13, 29] made heavy use of both the large bandwidth and the multiple-message transmission feature of the LOCAL model. A crucial concept in these algorithms is *shattering*. For coloring, shattering means coloring almost all the nodes such that each connected component of the set of nodes that remain uncolored has size at most $\mathrm{poly}(\log n)$. A similar concept was used originally by Beck [10]. The idea was introduced to the distributed setting in [9]. The dominating factor in the time complexity is the deterministic complexity of solving (a variant of) the problem on polylogarithmic-sized problems. As there are now polylogarithmic-time algorithms for deterministic coloring [40], with the fastest being $O(\log^3 n)$ [24], the randomized complexity is currently $O(\log^3 \log n)$ [13]. An $O(\log^5 \log n)$-round CONGEST algorithm was given in [25], improved to $O(\log^3 \log n)$ in [27]. These algorithms still require transmitting different messages to all $\Omega(\Delta)$ neighbors of a node.

Many distributed $(\Delta + 1)$-coloring algorithms work immediately in BCONGEST, including the folklore $O(\log n)$-round randomized algorithms [30] and the randomized part of [9]. The best deterministic algorithms known for small values of Δ, with complexity $\tilde{O}(\sqrt{\Delta}) + O(\log^* n)$ [8, 22, 35] use the full power of the LOCAL model, however. The $O(\log^3 n)$-round deterministic algorithm of [24] also works in BCONGEST, but it is sensitive to the palette size. When $\Delta \le \mathrm{poly}(\log n)$, [24] with the shattering of [9] colors in $O(\log^3 \log n)$ rounds of BCONGEST. Otherwise, if $\Delta \gg \mathrm{poly}(\log n)$, dependency on the palette size can be resolved by relabeling the palette, using network decomposition [23], as shown for coloring in [25]. Hence, there is a $O(\log \Delta + \mathrm{poly}(\log\log n))$-round BCONGEST algorithm for $(\Delta + 1)$-coloring.

Most known algorithms which work in BCONGEST were published as CONGEST algorithms, without making explicit that they also work with broadcast communication. Explicit mentions of the model are becoming more and more frequent in recent years, with examples in works on subgraph detection [32], flow and shortest paths problems [12, 14, 21], and proof labeling schemes [37].

Distributed Sketching and Broadcast Congested Clique. The palette sparsification theorem of [4] shows that even if each node uniformly samples $O(\log n)$ colors, the graph can still be $\Delta + 1$-colored while restricting each node to use only a sampled color. This has led to a (one-pass) streaming algorithm for $\Delta+1$-coloring using $O(n\,\mathrm{poly}(\log n))$ space. It was recently shown that the actual coloring can also be computed distributively, in $O(\log^2 \Delta + \log^3 \log n)$ rounds of CONGEST [20]. We utilize several technical lemmas from the work of [20], while the actual results are almost completely unrelated.

Palette sparsification is a one round/pass form of *distributed sketching* (or shared blackboard), a technique of considerable current interest [1, 6, 7]. The nomenclature that is closer to our setting is the *broadcast congested clique* [11, 16, 31]. Whereas there are no non-trivial lower bounds in the Congested Clique model for problems related to coloring, there is a recent $\Omega(\log\log n)$-round lower bound for the Maximal Independent Set problem in the broadcast congested clique [6].

1.4 Organization of the Paper

After preliminary definitions and results in Section 2, we formally describe our algorithm in Section 3 and give a proof of Theorem 1. Section 4 details the BCONGEST implementation of the synchronized color trial. We explain how to modify our algorithm for the BCStream model in Section 5. We defer some technical and lengthy proofs of new results to the full version [19].

2 PRELIMINARIES

Notation. For any integer $k \geq 1$, we denote the set $\{1, 2, \ldots, k\}$ by $[k]$. The communication network is $G = (V, E)$, we denote by $n = |V|$ its number of vertices, for each $v \in V$ we call $d(v)$ its degree and Δ the maximum degree of G. For a vertex $v \in V$, we denote by $N_G(v) = \{u \in V, uv \in E\}$ its neighbors in G. We assume nodes have $O(\log n)$-bit unique identifiers named $\mathrm{ID}(v)$. In the BCONGEST model, nodes of G communicate by broadcasting $O(\log n)$-bit messages in synchronous rounds.

A partial coloring is a function $C : V \to [\Delta + 1] \cup \{\perp\}$ such that for any edge $uv \in E$, its endpoints receive different colors $C(u) \neq C(v)$ unless $C(v)$ or $C(u)$ is $\perp$ – which stands for "not colored". With respect to any partial coloring C, we shall write $\widehat{d}(v)$ for the *uncolored* degree of v, i.e., its number of uncolored neighbors with respect to C. More generally, for any $S \subseteq V$, we write $\widehat{S}$ to denote the set of uncolored nodes in S (with respect to a partial coloring). Our algorithm colors monotonically: once we fix $C(v)$, it never changes.

When we say an event happens *with high probability*, or w.h.p. for short, we mean with probability $1 - n^{-c}$ for any suitably large constant $c > 0$. We implicitly choose the constant c large enough to union bound over polynomially many events.

2.1 Sparse-Dense Decomposition

The sparsity of a node counts the number of missing edges in its neighborhood, with the important detail that if a node has degree less than Δ, each "missing" neighbor counts as Δ missing edges.

Definition 1 (Sparsity). *The* sparsity ζ_v *of* $v \in V$ *is*

$$\zeta_v := \frac{1}{\Delta}\left(\binom{\Delta}{2} - m(N(v))\right),$$

where $m(N(v))$ *is the number of edges induced by* $N(v)$. *Node* v *is* ζ-sparse *if* $\zeta_v \geq \zeta$ *and* ζ-dense *if* $\zeta_v \leq \zeta$.

We decompose the graph into the locally sparse nodes and dense clusters called *almost-cliques*. Almost-cliques can be thought of as graphs that are ε-close to Δ-cliques, in a property-testing sense. Such decompositions are ubiquitous in randomized coloring [2, 4, 13, 25, 29, 39].

Definition 2. *For* $\varepsilon \in (0, 1/3)$, *an* ε*-almost-clique decomposition is a partition of* $V(G)$ *into sets* $V_{\mathrm{sparse}}, K_1, \ldots, K_k$ *such that*

(1) nodes in V_{sparse} *are* $\Omega(\varepsilon^2\Delta)$*-sparse,*
(2) for all $i \in [k]$, *almost-clique* K_i *satisfies:*
  (a) $|K_i| \leq (1 + \varepsilon)\Delta$,
  (b) $|N(v) \cap K_i| \geq (1 - \varepsilon)\Delta$ *for all* $v \in K_i$, *and*
  (c) $|N(v) \cap K_i| \leq (1 - \varepsilon/2)\Delta$ *for all* $v \notin K_i$.

Definition 3 (External and Anti-Degrees). *For a node* v *in an almost-clique* K, *we call* $e_v = |N(v) \setminus K|$ *its* external degree *and* $a_v = |K \setminus N(v)|$ *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.*

Property 2c is not typically included in prior work (e.g., [4, 25]). It was used recently in [7, 18]. We use it solely to prove Lemma 1. An *anti-edge* is a missing edge between two nodes, i.e., an edge in the complement graph. The following lemma is an immediate consequence of Properties 2b and 2c.

Lemma 1. *Let* K *be an almost-clique. Every* $v \in K$ *is* $(\varepsilon/2 \cdot e_v)$*-sparse.*

The first CONGEST algorithm to compute almost-clique decompositions in $O(1)$ rounds (when $\Delta \in \Omega(\log^2 n)$) was given by [25]. It was then improved by [28] to arbitrary Δ in the CONGEST model. [20] gives a simpler implementation of [28] that works in BCONGEST and BCStream.

Lemma 2 ([20]). *For any* $\varepsilon \in (0, 1/20)$, *there exists an algorithm computing an* ε*-almost-clique decomposition in* $O(\varepsilon^{-4})$ *rounds of* BCONGEST *with high probability.*

Colorful Matching. In a $\Delta + 1$-clique, the colors used in the clique are exactly the colors used in the neighborhood of each node. An almost-clique can have size larger than $\Delta + 1$. Thus, an almost-clique with uncolored nodes might actually have an empty clique palette. To solve this issue, [4] introduced the idea of colorful matching.

Definition 4 (Colorful Matching). *A* colorful *matching in a clique* K *(with respect to a partial coloring* C*) is a matching of anti-edges in* K *(edges in the complement graph) such that 1) endpoints of each anti-edge receive the same color, and 2) each anti-edge has a different color.*

Intuitively, if one contracts anti-edges of the colorful matching, one reduces the size of the almost-clique while maintaining a proper coloring. If the matching is large enough, the number of unused colors in K becomes greater than the number of uncolored nodes.

Definition 5 (Clique Palette). *For each* K, *let the* clique palette $\Psi(K) = [\Delta + 1] \setminus C(K)$ *be the set of colors not used in* K.

Claim 1. *Let* K *be an almost-clique and* M *a colorful matching in* K. *Then, for all* $v \in K$,

$$|\Psi(K)| \geq |\widehat{K}| + 1 + e_v - a_v + |M| .$$

Proof. The clique palette loses at most one color per colored node but saves one for each anti-edge in the colorful matching; hence, $|\Psi(K)| \geq \Delta + 1 - (|K| - |\widehat{K}|) + |M|$. On the other hand, observe that $\Delta \geq |N(v) \cap K| + e_v$ and $|K| = |N(v) \cap K| + a_v$. The claim follows. ∎

By computing a matching of size $\Theta(\overline{a}_K)$, the clique palette always contains colors for each node in $\widehat{K}$. Computing a colorful matching of size $\Theta(\overline{a}_K)$ can be done in $O(1)$ rounds as the clique contains $\Theta(\overline{a}_K\Delta)$ anti-edges and by trying colors, we expect $\Theta(\overline{a}_K)$ edges to join the matching.

Lemma 3 ([20]). *Let $\beta < 1/(18\varepsilon)$ be a constant. There exists a $O(\beta)$-round algorithm called* Matching *that computes a colorful matching of size $\beta \cdot \overline{a}_K$ with probability $1 - n^{-\Theta(C)}$ in every clique K with $\overline{a}_K \geq C \log n$. Furthermore, at most $2\beta \cdot \overline{a}_K$ nodes are colored in each almost-clique during this step.*

2.2 Distributed Coloring with Slack

Definition 6 (Palette). *The palette $\Psi(v)$ of node v, with respect to a partial coloring, is the set of colors not used by its neighbors.*

Definition 7 (Slack). *The* slack *$s_H(v)$ of a node v in a subgraph H is the difference between the size of its palette and its uncolored degree in this graph: $s_H(v) = |\Psi(v)| - \widehat{d}_H(v)$. When H is clear from context, we simply write $s(v)$.*

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 previously inactive neighbors become active, the node will lose the slack that those inactive neighbors were providing before. Elkin, Pettie, and Su [17] observed that by trying random colors, nodes receive slack proportional to their permanent sparsity.

Lemma 4 (Slack Generation, [17, Lemma 3.1]). *Let v be a ζ-sparse node for some ζ. Suppose each node of G independently decides w.p. $p_s = 1/200$ to try a uniform color in $[\Delta+1]$. Then, w.p. $1-e^{-\Theta(\zeta)}$, v has slack $s(v) \geq \gamma \cdot \zeta$ where $\gamma > 0$ is a (small) universal constant.*

Trying Colors From Lists. When we say a node *tries a random color*, we mean that it broadcasts a color uniformly sampled from some set (usually from its palette) and *adopts* the color if none of its neighbors with smaller ID tried the same color. We refer to this one-round procedure as TryColor. It is known that nodes with $\Omega(\log n)$ uncolored neighbors see a constant fraction of them get colored when they try random colors, w.h.p. [9].

Lemma 5. *Let H be a vertex-induced subgraph and $L(v) \subseteq \Psi(v)$ for each v. Suppose there exists a globally known constant $\alpha > 0$ such that every uncolored v satisfies $|L(v)| \geq \alpha \cdot \widehat{d}(v) \geq C \log n$. If nodes independently call* TryColor *w.p. $p_t = \alpha/3$* and samples a uniform color in *$L(v)$, then, w.p. $1 - n^{-\Theta(C)}$, the uncolored degree of every node has decreased by a factor $2/3$.*

Trying multiple colors to take advantage of extra colors (i.e., slack) was proposed originally by [41]. It is a key component of all recent fast randomized coloring algorithms [13, 27, 28]. A small tweak suffices to bring the technique to BCONGEST.

Lemma 6 (Multi-Color Trial, [26, 27]). *Let H be a vertex-induced subgraph of G. Suppose that for each $v \in H$, there is a $L(v)$ list of colors satisfying*
(1) $L(v)$ is known by each $u \in N_H(v)$,
(2) $|L(v) \cap \Psi(v)| \geq 2\widehat{d}_H(v)$, and
(3) $|L(v) \cap \Psi(v)| \geq \widehat{d}_H(v) + C \log^{1.1} n$ for some constant $C > 0$.
There exists an algorithm coloring every node of H in $O(\log^* n)$ rounds of BCONGEST *with probability $1 - n^{-\Theta(C)}$.*

Lemma 6 is a mere reformulation of [27, Lemma 1] with the notable exception that it works in BCONGEST because of the additional Property 1. This allows the use of *representative sets* [26]. At a high level, the technique is to save on the bandwidth necessary to send $\Theta(\log n)$ random colors by instead sending a pseudorandom sample. In BCStream, it can be implemented with $O(\log^3 n)$ memory but requires more work. We refer interested readers to [26, Section 7]. The main idea is that a set of $\Theta(\log n)$ random colors can be represented by a random walk on an implicit expander graph.

2.3 Concentration Inequalities

We use the following variants of Chernoff bounds for dependent random variables. The first one is obtained, e.g., as a corollary of Lemma 1.8.7 and Theorems 1.10.1 and 1.10.5 in [15].

Lemma 7 (Martingales). *Let $\{X_i\}_{i=1}^r$ be binary random variables, and $X = \sum_i X_i$. Suppose that for all $i \in [r]$ and $(x_1, \ldots, x_{i-1}) \in \{0,1\}^{i-1}$ with $\Pr(X_1 = x_1, \ldots, X_r = x_{i-1}) > 0$, $\Pr(X_i = 1 \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \leq q_i \leq 1$, then for any $\delta > 0$,*

$$\Pr\left(X \geq (1+\delta)\sum_{i=1}^r q_i\right) \leq \exp\left(-\frac{\min(\delta,\delta^2)}{3}\sum_{i=1}^r q_i\right). \tag{1}$$

Suppose instead that $\Pr(X_i = 1 \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \geq q_i$, $q_i \in (0,1)$ holds for $i, x_1, \ldots, x_{i-1}$ over the same ranges, then for any $\delta \in [0,1]$,

$$\Pr\left(X \leq (1-\delta)\sum_{i=1}^r q_i\right) \leq \exp\left(-\frac{\delta^2}{2}\sum_{i=1}^r q_i\right). \tag{2}$$

3 ALGORITHM AND ANALYSIS

In this section, we describe our algorithm and give the main technical ideas behind Theorem 1. Algorithm 1 gives a high-level description of our algorithm.

The main technical contribution is a $O(\log^* n)$-round algorithm for coloring graphs with $\Delta \in \Omega(\log^3 n)$. For low-degree graphs, a $O(\log^3 \log n)$-round algorithm that works in BCONGEST is known [9, 24]. We conjecture that our algorithm actually shatters the graph in $O(\log^* n)$ rounds when $\Delta = O(\log^3 n)$. If this were true, [9] would no longer be required for small Δ. This would make any improvement to the deterministic complexity of $(\deg+1)$-list-coloring, including beyond $o(\log n)$, carry over to our algorithm.

Algorithm 1. High Level Description of our Algorithm.
Parameters: Let $C = O(1)$ be a large enough constant,

$$\ell = C\log^{1.1} n\,, \quad \varepsilon = 10^{-5} \quad \text{and} \quad \beta = 401\,. \tag{3}$$

(1) **Setting up.** Compute an ε-almost-clique decomposition $V_{\text{sparse}}, K_1, \ldots, K_k$. Compute outliers O_K and inliers $I_K = K \setminus O_K$ in each clique K (see Definition 8), as well as put-aside sets P_K (see Lemma 8). We define a value $x(K) = \Theta(\overline{a}_K + \overline{e}_K + \log n)$ for each clique (see Eq. (5)). By extension, let $x(v) = x(K)$ for each $v \in K$.
Cliques are categorized as full, open, or closed (Definition 9). The following three steps aim at generating slack for each type:

(i) Slack Generation: each node tries a color in $[\Delta+1]\setminus[x(v)]$ w.p. $p_s = 1/200$.
(ii) Colorful Matching: by trying colors in $[\Delta+1]\setminus[x(K)]$ for $O(\beta)$ rounds, we color $\beta\overline{a}_K$ pairs of anti-edges in each K.
(iii) Put-Aside Sets: we find in each full clique K a set $P_K \subseteq I_K$ of size $\Theta(\ell)$ such that P_K has no edge to $P_{K'}$ for all $K \neq K'$.

Each sparse node has $\Omega(\Delta)$ permanent slack from the slack generation step; hence, we color them in $O(\log^* n)$ rounds with MultiTrial. We color outliers O_K with colors from $[\Delta+1]\setminus[x(K)]$ with MultiTrial using the $\Omega(\Delta)$ temporary slack provided by inactive inliers.

(2) **Synchronized Color Trial.** In each clique, we compute the clique palette $\Psi(K)$ and sample a permutation π of $\widehat{K}\setminus P_K$. Each node $v \in \widehat{K}\setminus P_K$ tries the $\pi(v)$-th color of $\Psi(K)$. In open cliques (see Definition 9), we run an extra $O(1)$ rounds of TryColor using only colors from $[\Delta+1]\setminus[x(K)]$.
(3) **Completing the Coloring.** Uncolored nodes now satisfy

$$|[x(v)] \cap \Psi(v)| \geq 2\widehat{d}(v) ,$$

and put-aside sets ensure that every node has slack $\Omega(\ell)$. Hence, inliers are colored in $O(\log^* n)$ rounds by MultiTrial with colors $[x(K)]$.
(4) **Coloring Put-Aside Sets.** We color put-aside sets in two steps: first, we reduce their size to $O(\log n/\log\log n)$ by running *non-adaptive* randomized color trial. Then, each node sends $|P_K|+1$ colors from a poly$(\log n)$-sized set of colors. This takes $O(1)$ rounds: $O(\log n/\log\log n)\times O(\log\log n)$ bits to send.

The key technical idea is to *reserve* colors $\{1,2,\dots,x(K)\}$ in each clique, where $x(K)$ is an integer that depends on the density of K (see Eq. (5)). It is straightforward to see that reserve colors $[x(K)]$ are not used during Steps 1 and 2. The value of $x(K)$ is chosen to be greater than nodes' degrees at the end of Step 2. This allows using lists $L(v) := [x(v)]$ for the MultiTrial in Step 3.

3.1 Step 1: Setting up

Assume we have an ε-almost-clique decomposition $V_{\text{sparse}}, K_1, \dots, K_k$ (see Definition 2). Sparse nodes can be colored in $O(\log^* n)$ rounds [26], so we focus our attention on almost-cliques. We call outliers the (possibly empty) set of nodes in each clique whose external degree or anti-degree derives more than a constant factor from the average.

Definition 8 (Inliers/Outliers). *For each K, we define its set of* outliers *as*

$$O_K = \{v \in K : e_v \geq 30\overline{e}_K \text{ or } a_v \geq 30\overline{a}_K\} . \quad (4)$$

We call the remaining uncolored *nodes $I_K = \widehat{K}\setminus O_K$* inliers.

In each clique, outliers represent only a small fraction of the vertices; hence, they can be colored beforehand with the temporary slack provided by their $\Omega(\Delta)$ uncolored neighbors in I_K. Claim 2 follows from Markov inequality and Chernoff bound (few nodes are colored by slack generation).

Claim 2. *For each K, after generating slack and computing a colorful matching, w.h.p. $|I_K| \geq 0.9\Delta$.*

We classify cliques in three categories, depending on the degree that nodes have after Step 2. Each type of clique receives slack from different sources: full cliques from put-aside sets, open cliques from the slack generation step, and closed cliques from the colorful matching.

Definition 9 (Full/Open/Closed Cliques). *For each $i \in [k]$, we say that $K = K_i$ is:*

- full *if $\overline{a}_K + \overline{e}_K < \ell$, where ℓ is defined in Eq. (3),*
- open *if K is not full and $2\overline{a}_K < \overline{e}_K$, and*
- closed *if K is neither full nor open.*

We denote by $\mathcal{K}_{\text{full}}$ (respectively $\mathcal{K}_{\text{open}}$ and $\mathcal{K}_{\text{closed}}$) the set of full cliques (respectively open and closed cliques).

In each clique, we reserve $x(K)$ colors depending on the clique's density. We will ensure that $[x(K)] \subseteq \Psi(K)$ until we color inliers with MultiTrial (Step 3). For a clique K, define

$$x(K) = \begin{cases} 200\ell & \text{if } K \in \mathcal{K}_{\text{full}} \\ 400\overline{a}_K & \text{if } K \in \mathcal{K}_{\text{closed}} \\ \gamma\varepsilon/8 \cdot \overline{e}_K & \text{if } K \in \mathcal{K}_{\text{open}} \end{cases} , \quad (5)$$

where γ is the constant from Lemma 4. By extension, we write $x(v) = x(K)$ for each $v \in K$.

Put-Aside Sets. Recall that to color in $O(\log^* n)$ rounds with MultiTrial, nodes need slack at least $\ell = \Theta(\log^{1.1} n)$ (Lemma 6, Property 3). Nodes from very dense cliques do not receive enough permanent slack from the slack generation phase. Following [27, Section 5.4], we overcome this issue by putting aside sets of $\Theta(\ell)$ nodes in each highly-dense clique to provide temporary slack. These sets remain uncolored until the very end of the algorithm. These are necessary only in highly-dense cliques, whose nodes have $O(\ell)$ external neighbors. It allows us to find put-aside sets such that no edge connects sets from different cliques. The lack of connections allows us to color each set independently at the very end. See [27, Lemma 5] for a proof of Lemma 8.

Lemma 8 (Put-Aside Sets). *There exists a $O(1)$-round BCONGEST algorithm finding subsets $P_K \subseteq I_K$ of size 201ℓ in each almost-clique $K \in \mathcal{K}_{\text{full}}$, such that P_K has no edges to other $P_{K'}$ for $K' \neq K$.*

3.2 Step 2: Synchronized Color Trial

The idea of the following Lemma 9 (which is a reformulation of [27]) is to distribute a set of colors to nodes in the clique. Each color has a unique recipient. This avoids in-clique conflicts, and a node can only fail to adopt the color it received due to choices of its external neighbors. Therefore, the expected number of nodes to fail is $\sum_{v\in K} O(e_v/\Delta) = O(\overline{e}_K)$.

Lemma 9 ([27, Section 5.5]). *Let x be an integer, K be a clique, and $S = \widehat{K}\setminus P_K$ be such that $0.75\Delta \leq |S| \leq |\Psi(K)| - x$. Suppose π is a uniform permutation of $[|S|]$. If for each $i \in [|S|]$ the i-th node in S tries the $\pi(i)$-th color in the set $\Psi(K)\setminus[x]$, then w.h.p. the number of nodes to remain uncolored is $8\max\{6\overline{e}_K, C\log n\}$. This holds even if the random bits outside of K are chosen adversarially.*

Lemma 10 shows that each clique has enough colors, even if when we reserve $x(K)$ colors.

Lemma 10. *For all K, $|\Psi(K)| - x(K) \geq |\widehat{K} \setminus P_K|$.*

Proof. We consider each type of clique separately. In a full clique K, recall that we computed a set P_K of put-aside nodes of size $201\ell = \Theta(\log^{1.1} n)$ that remain uncolored (Lemma 8). The set S of nodes participate in the synchronized color trial is $|S| = |\widehat{K} \setminus P_K| \geq 0.75\Delta$ (by of Claim 2 and $\Delta \gg \ell$). The number of colors used in K is bounded by the number of colored nodes; hence, $|\Psi(K)| \geq \Delta - (|K| - |\widehat{K}|)$. Since each full clique has size at most $\Delta + \ell$, we infer $|\Psi(K)| \geq |\widehat{K}| - \ell$. Put-aside sets have size $|P_K| = 201\ell$, so

$$|\widehat{K} \setminus P_K| = |\widehat{K}| - 201\ell \leq |\Psi(K)| - 200\ell = |\Psi(K)| - x(v)\ . \quad \text{(by Eq. (5))}$$

Suppose that K is open, i.e. $\overline{a}_K \leq \overline{e}_K/2$ (Definition 9). By summing on each $v \in K$ over the bounds $\Delta \geq |K \cap N(v)| + e_v$ and $|K| = |K \cap N(v)| + a_v$, we get $\Delta - |K| \geq \overline{e}_K - \overline{a}_K \geq \overline{e}_K/2$. By our choice of $x(K)$,

$$|\Psi(K)| - x(K) \geq |\widehat{K}| + \overline{e}_K/2 - x(K) \geq |\widehat{K}|\ .$$

Suppose now that K is closed. Denote by t the number of nodes colored during the slack generation step or as outliers. In closed clique, we compute a colorful matching of size $\beta\overline{a}_K$. Hence $|\Psi(K)| \geq \Delta - t - \beta\overline{a}_K$. On the other hand, each edge in the matching colors two nodes. Therefore, the number of uncolored nodes is

$$\begin{aligned}
|\widehat{K}| &\leq |K| - t - 2\beta\overline{a}_K \\
&\leq (\Delta - t - \beta\overline{a}_K) - (\beta - 1)\overline{a}_K && \text{(because } |K| \leq \Delta + \overline{a}_K) \\
&\leq |\Psi(K)| - x(K)\ . \quad \blacksquare && \text{(by definition of } \beta \text{, Eq. (3))}
\end{aligned}$$

We now claim that each node has enough slack after SCT. Details of its implementation and related proofs are postponed to a later section (Section 4, Lemmas 16 and 19).

Lemma 11. *At the end of Step 2, w.h.p. each $v \in \widehat{K}$ satisfies $|[x(v)] \cap \Psi(v)| \geq 2\widehat{d}(v)$.*

Proof. By Lemma 10, cliques carry more colors than nodes they try to color during SCT, and by Lemma 9, at most $O(\overline{e}_K + \log n)$ nodes remain uncolored per clique. Simple counting shows the following claim.

Claim 3. *After the synchronized color trial, every uncolored $v \in K$ satisfies*

- *$2\widehat{d}(v) + e_v \leq x(v)$ if $v \in \mathcal{K}_{\text{full}} \cup \mathcal{K}_{\text{closed}}$, and*
- *$\widehat{d}(v) \leq 80\overline{e}_K$ if $K \in \mathcal{K}_{\text{open}}$.*

Observe that, since $x(v)$ has the same value for each $v \in K$, and colors from $[x(K)]$ are not used to color nodes of K, the only reason some $c \in [x(v)]$ might not belong to $\Psi(v)$ is if it is used by an external neighbor of v. For all $v \in K$ with $K \in \mathcal{K}_{\text{full}} \cup \mathcal{K}_{\text{closed}}$, Eq. (6) follows from Claim 3:

$$|[x(v)] \cap \Psi(v)| \geq x(v) - e_v \geq 2\widehat{d}(v)\ . \tag{6}$$

For $v \in K$ with $K \in \mathcal{K}_{\text{open}}$, we need $O(1)$ additional rounds of TryColor to ensure Eq. (6). However, we need to preserve $[x(K)] \subseteq \Psi(K)$. Thus, nodes of K try random colors in $\Psi(v) \setminus [x(v)]$. We now show it is enough to reduce the uncolored degree.

Let $v \in K$ for any $K \in \mathcal{K}_{\text{open}}$. By Claim 3, $\widehat{d}(v) \leq 80\overline{e}_K$; we show that $|\Psi(v)| - x(v) \geq \Omega(\overline{e}_K)$. By Lemma 5, even when using only colors from $\Psi(v) \setminus [x(v)]$, after one call to TryColor the uncolored degree of each node decreases by a constant factor. After $O(1)$ rounds, with high probability, the uncolored degree of each v verifies the desired equation.

Claim 4. *For each $v \in K$, $\Delta - d(v) + e_v \geq \overline{e}_K/2$.*

If $e_v \leq C\log n$, by Claim 4, $s(v) \geq \Delta - d(v) \geq \overline{e}_K/2 - C\log n \geq \overline{e}_K/3$ because $\overline{e}_K \geq \ell/2 \gg C\log n$. If $e_v \geq C\log n$, vertex v receives $\gamma\varepsilon/2 \cdot e_v$ permanent slack from the slack generation step w.p. $1 - n^{-\Theta(C)}$ (by Lemma 4). Overall, nodes use lists of size

$$\begin{aligned}
|\Psi(v)| - x(v) &\geq \Delta - d(v) + \gamma\varepsilon/2 \cdot e_v - x(v) \\
&\geq \gamma\varepsilon/2 \cdot (\Delta - d(v) + e_v) - x(v) && (\gamma\varepsilon/2 < 1) \\
&\geq \gamma\varepsilon/4 \cdot \overline{e}_K - x(v) && \text{(by Claim 4)} \\
&\geq \gamma\varepsilon/8 \cdot \overline{e}_K\ . && \text{(by Eq. (5))}
\end{aligned}$$

By Lemma 5 with $\alpha = \gamma\varepsilon/640$, after TryColor the uncolored degree of each node reduces by a constant factor with high probability. $\blacksquare$ Lemma 11

3.3 Step 4: Coloring Put-Aside Sets

Our goal, in this section, is to reduce the size of put-aside sets to $O(\log n/\log\log n)$. Once this is achieved, coloring their remaining nodes only takes $O(1)$ rounds, as the next lemma shows.

Lemma 12. *Suppose all nodes are colored except put-aside sets P_K in each $K \in \mathcal{K}_{\text{full}}$ of size $O(\log n/\log\log n)$. Then, w.h.p. we can complete the coloring in $O(1)$ rounds of BCONGEST.*

Proof. Recall that no edges exist between put-aside sets. Hence, we color each put-aside set independently. We can assume without loss of generality that $|\Psi(K)| = O(\log^3 n)$. Indeed, since nodes have $O(\log^{1.1} n)$ external and anti-degree, any $D \subseteq \Psi(K)$ of size $\Theta(\log^3 n)$ works as replacement for the clique palette when $\Psi(K)$ is larger. Nodes use Algorithm 2 to learn $\Psi(K)$ in $O(1)$ rounds (Lemma 16).

Therefore, describing a color $c \in \Psi(K)$ takes $O(\log\log n)$ bits. If $\overline{a}_K \geq C\log n$, the clique palette has enough colors for every node, i.e., $|\Psi(K) \cap \Psi(v)| \geq |P_K| + 1$. If $\overline{a}_K < C\log n$, lists $L(v) = \Psi(K) \cup C(K \setminus N(v))$ have $|P_K| + 1$ colors (Claim 1 with an empty matching and a_v extra colors). Since lists have size $|P_K| + 1 = O(\log n/\log\log n)$ and each color takes $O(\log\log n)$ bits, nodes can broadcast their list in $O(1)$ rounds. Nodes complete the coloring without additional communication, simulating a greedy sequential algorithm with the lists. $\blacksquare$

The following technical claim (which is a direct application of Chernoff) allows us to assume we have global communication within almost-clique if the number of messages to send is small enough. In particular, nodes can learn all the identifiers from P_K, therefore relabel nodes with $O(\log\log n)$-bit.

Claim 5 (Many-to-All Broadcast). *Let K be an almost-clique with $O(\Delta/\log n)$ nodes with an $O(\log n)$-bit message to send to everyone in K. Suppose each node with a message broadcasts it, before* *each node in K broadcasts $O(1)$ messages it received, picked randomly. Then, w.h.p., all messages are received by every node in K.*

The key difficulty in coloring put-aside sets lies in reducing their sizes to $O(\log n/\log\log n)$. We use a procedure CompressTry, which simulates a sequential algorithm where nodes of the put-aside set, in the order of their IDs, each perform $O(\log n/\log\log n)$ times a *non-adaptive* TryColor with slack z. The following technical lemma analyzes the performance of CompressTry. We defer its proof and the exact description of CompressTry to the full version [19].

Lemma 13. *Let $K \in \mathcal{K}_{\text{full}}$ and fix a set $S \subseteq \widehat{K}$ of size $O(\log^{1.1} n)$. Furthermore, suppose each $v \in S$ has a list $L(v)$ of at most $C\log^{1.1} n$ colors known to every $u \in S$, and such that $|L(v) \cap \Psi(v)| \geq |S| + z$ for a fixed $z \geq C\log n/\log\log n$. Then, w.p. $1 - e^{-z} - 1/\text{poly}(n)$,* CompressTry *colors all but z nodes in S. Furthermore,* CompressTry *uses $O(\log n/\log\log n)$ bandwidth.*

Lemma 14 shows how we use CompressTry to reduce the size of the put-aside sets. In cliques with colorful matching, nodes have $\overline{a}_K \in \Omega(\log n)$ slack; CompressTry directly reduces P_K to $O(\log n/\log\log n)$ nodes by using the clique palette. In cliques where $\overline{a}_K < C\log n$, we first put-aside $O(\log n)$ nodes to reduce P_K to $O(\log n)$ using the clique palette. Then, nodes add colors used by their anti-neighbors to their list, and CompressTry finishes to reduce P_K to $O(\log n/\log\log n)$.

Lemma 14. *There is a $O(1)$-round* BCONGEST *algorithm reducing the number of uncolored nodes in P_K to $O(\log n/\log\log n)$ with high probability.*

Proof. For cliques such that $\overline{a}_K \geq C\log n$, Lemma 13 allows us to directly reduce P_K to a set of size $z := C\log n/\log\log n$. This is because, in such cliques, we compute a colorful matching of size $\beta\overline{a}_K \geq \overline{a}_K + a_v$, for each $v \in P_K$ (which are inliers). Therefore, using lists $L(v) := \Psi(K)$, by Claim 1, $|L(v) \cap \Psi(v)| \geq |P_K| + \overline{a}_K \geq |P_K| + z$. Note that the clique palette can be publicly learned in $O(1)$ rounds by Lemma 16. CompressTry succeeds only w.p. $1 - e^{-z}$, but by repeating independently $\log\log n$ times, the probability that at least one instance succeeds is $1 - e^{-z\log\log n} = 1 - n^{-C}$. Overall, we need $O(\log n/\log\log n \cdot \log\log n) = O(\log n)$ bandwidth.

Henceforth, we assume that $\overline{a}_K < C\log n$. The main difference is that we do not have a colorful matching, so the clique palette does not approximate $\Psi(v)$ well. We settle this in two steps.

From $O(\log^{1.1} n)$ to $O(\log n)$. Let $S \subseteq P_K$ be an arbitrary subset of P_K of $31C\log n$ nodes. By Claim 1, $|\Psi(K) \cap \Psi(v)| \geq |P_K| - a_v \geq |P_K \setminus S| + C\log n$. Therefore, CompressTry with lists $L(v) = \Psi(K)$ and $z = C\log n$ reduces P_K w.h.p. to size $32C\log n$ (the $C\log n$ nodes left uncolored in $P_K \setminus S$ by CompressTry and the $31C\log n$ uncolored nodes of S).

From $O(\log n)$ to $O(\log n/\log\log n)$. Now, instead of using only the clique palette, we augment lists with colors of anti-neighbors. Let $L(v) := \Psi(K) \cup C(K \setminus N(v))$. Since we are adding a_v colors to each list, Claim 1, even with an empty matching, gives us, $|L(v) \cap \Psi(v)| = |\Psi(K) \cap \Psi(v)| + a_v \geq |P_K|$. If we now put-aside a set $S \subseteq P_K$ of $z := C\log n/\log\log n$ nodes, lists $L(v)$ verify $|L(v) \cap \Psi(v)| \geq |P_K \setminus S| + z$. To conclude, it remains to explain how nodes learn lists $L(v)$.

Since $\overline{a}_K < C\log n$, each node has at most $30C\log n$ anti-neighbors in the clique. If we relabel nodes of P_K using identifiers in $[|P_K|]$ (with Claim 5), every $u \in K$ can describe the set $P_K \setminus N(v)$ with a bit-map in one $O(\log n)$-bit message. Note that only $O(\log^2 n)$ nodes will need to send a bit-map, i.e. at most $O(\log n)$ per node in P_K. By Claim 5, all messages can be disseminated in $O(1)$ rounds to all nodes in K. Thus, all lists are known and we make $\log\log n$ independent calls to CompressTry. ∎

4 SYNCHRONIZED COLOR TRIAL IN BCONGEST

At its core, synchronized color trial is simply about creating a random bijection between (most of) a set of colors and (most of) the uncolored nodes of a clique. Our implementation uses the clique palette as a set of colors and randomly permutes the nodes. The order of each node in the permutation tells it which color to take in the clique palette. This entails two difficulties. Firstly, to make use of its order in the sampled permutation, each node needs to know the matching color in the clique palette. We show that $O(1)$ rounds of BCONGEST suffice for all nodes to learn their clique palette. The second issue is sampling the permutation, and entails a more involved process. For simplicity, we describe an $O(\log\log n)$ permutation sampling procedure in the main text, which suffices for Theorems 1 and 2, and defer a more involved $O(1)$ procedure to the full version [19].

We will need the following technical lemma (which is an immediate consequence of Property 2b and Chernoff).

Lemma 15. *Let K be an almost-clique and an integer $k \leq \Delta/(C\log n)$ for some large enough $C > 0$. Suppose each $v \in K$ samples $t(v) \in [k]$ uniformly at random. Then, with high probability, for each $i \in [k]$, the set $T_i = \{v \in K : t(v) = i\}$ satisfies that for any $u, w \in K$, $|T_i \cap N(u) \cap N(w)| \geq (C/4)\log n$. We say that T_i* 2-hop connects *K in that each pair of nodes in K has a common neighbor in T_i.*

Note that since $T_i \subseteq K$, each T_i also 2-hop connects itself, thus has diameter 2.

Learning the clique palette. We learn the clique palette by dividing the color space into $O(\Delta/\log n)$ contiguous subpalettes. Given a 2-hop connecting set of nodes to handle each subpalette – with a trivial construction due to Lemma 15 – each node learns $\Psi(K)$ in $O(1)$ rounds. Recall that $C(S)$ denotes the set of colors currently assigned to a set S of nodes.

Algorithm 2. Procedure LearnPalette, in almost-clique K.
Parameters: Let $C = O(1)$ be a large enough constant, $k = \lfloor \Delta/(C\log n) \rfloor$.
Assume K to be split into k 2-hop connecting sets $T_1, \ldots, T_k$. Let $R_i := \{1 + \lfloor (i-1)\cdot(\Delta+1)/k \rfloor, \ldots, \lfloor i\cdot(\Delta+1)/k \rfloor\}$, i.e., $R_1, \ldots, R_k$ partition the color space $[\Delta + 1]$.

(1) Each v encodes $R_{t(v)} \cap C(N(v) \cap K)$ into a $C\log n$-sized bit-map and broadcasts it.

(2) For each $i \in [k]$, each $v \in K$ combines the bit-maps received from its neighbors in T_i, i.e., computes

$$\bigcup_{u \in N(v) \cap T_i} \left(R_i \cap C(N(u) \cap K) \right)$$

and takes it for $R_i \cap C(K)$.

Lemma 16. *Let K be an almost-clique of palette $\Psi(K)$.* LearnPalette *has each $v \in K$ learn $\Psi(K)$ in $O(1)$ rounds of* BCONGEST.

Proof. In $\Delta + 1$-coloring, learning $\Psi(K)$ is equivalent to learning the *used* colors $C(K)$. LearnPalette requires $O(1)$ rounds of BCONGEST, as each node in K only sends one $C \log n$-bit message. Let us consider a color $c \in C(K)$, a node $v \in K$, and argue that v learn c. Let R_i be such that $c \in R_i$, and $u \in K$ a node with color c. Since T_i 2-hop connects K, there exists a node in $T_i \cap N(u) \cap N(v)$. Such a node contains c in the bitmap it computes in Step 1 of LearnPalette, and v receives this bitmap in Step 2. As this works for every $c \in C(K)$ and $v \in K$, all $v \in K$ learn $C(K)$. ∎

Sampling the permutation. At a high level, the $O(\log \log n)$ algorithm for permuting the nodes presented in this section has the nodes undergo two shuffling steps. Nodes first undergo a "rough shuffling", which puts them into buckets, roughly positioning them in the permutation. Each group then does a "fine shuffling" to give each node its exact position.

An important step in both our $O(\log \log n)$ and our $O(1)$ implementation is giving nodes $O(\log \log n)$-bit labels unique within their buckets. Using the smaller labels instead of the original node IDs allows each bucket to save a multiplicative $\Theta(\log n / \log \log n)$ factor when describing a permutation of its elements.

Algorithm 3. Procedure Relabel, in 2-hop connected set of nodes $T \subseteq V$, for subset $S \subseteq T$.
Parameters: Let $C = O(1)$ be a large enough constant, $x := \lceil C \log n / \log \log n \rceil$.

(1) Each $v \in S$ samples and broadcasts x labels in $[|S|^2 \log n]$, picked u.a.r. and independently.
(2) Each $v \in T$ broadcasts an x-sized bit-map indicating, for each $j \in [x]$, whether multiple nodes in $S \cap N(v)$ have the same jth label.
(3) If for a minimum $j \in [x]$, all nodes in S have distinct jth labels, S uses them as new labels.

Lemma 17. *Suppose S has size* $\text{poly}(\log n)$. Relabel *succeeds at relabeling S in $O(1)$* BCONGEST *rounds, w.h.p.*

Proof. First, note that $O(1)$ BCONGEST rounds suffice to compute $|S|$ for Step 1, as T is 2-hop connected. Since $|S|^2 \log n \in \text{poly}(\log n)$, each label sent by a node $v \in S$ during Step 1 is representable with $O(\log \log n)$ bits. Thus, $x \in O(\log n / \log \log n)$ labels can be transmitted in $O(1)$ rounds.

As T 2-hop connects itself (a fortiori S), two nodes of S with a common jth label are necessarily detected by a common neighbor during Step 2. Taking the AND of all x-sized bitmaps sent in this step, the nodes in T all learn for which $j \in [x]$ it holds that all nodes of S picked distinct jth labels.

We now analyze the probability that the relabeling succeeds, i.e., that a $j \in [x]$ as used in Step 3 exists. For each $j \in [x]$, each jth sampled label in S has probability less than $1/(|S| \log n)$ of conflicting with one of the other $|S| - 1$ jth labels. Hence, by union bound, the jth labels have a collision with probability at most $1/(\log n)$. Having x independent samples implies success with probability at least $1 - (\log n)^{-x} = 1 - 2^{-x \log \log n} = 1 - 2^{-C \log n} = 1 - n^{-C}$, i.e., w.h.p. ∎

Algorithm 4. Procedure Permute, in almost-clique K, on subset $S \subseteq K$ of the nodes.
Parameters: Let $C = O(1)$ be a large enough constant, $k := \lfloor \Delta / (C \log n) \rfloor$, $x := \lceil C \log n / \log \log n \rceil$.

(1) **Rough bucketing.** Each $v \in K$ independently picks a random $t(v) \in [k]$ u.a.r.
For each $i \in [k]$, let $T_i := \{v \in K : t(v) = i\}$ and $S_i := T_i \cap S$.
(2) **Counting buckets.** For each $i \in [k]$, the nodes in T_i compute and broadcast $|S_i|$.
(3) **Relabeling.** Within each T_i, $i \in [k]$, use Relabel on S_i.
(4) **Permuting within buckets.** Within each T_i, the maximum ID node gathers the new labels of S_i, picks a random permutation ρ_i of S_i, and sends it to T_i, all along a BFS tree.
(5) **Output.** Each $v \in S_i$ takes $\pi(v) := \rho_i(v) + \sum_{j<i} |S_j|$ as its index in the output π.

Lemma 18. *With high probability,* Permute *outputs a permutation of S in $O(\log \log n)$ rounds. For each permutation π of S, the probability of sampling π is bounded by $\frac{1}{(1-1/\text{poly}(n)) \cdot |S|!}$*

Proof. By Lemma 15, the sets T_i computed in Step 1 2-hop connect K, w.h.p., and in particular have diameter 2. Assuming this holds, Step 2 only takes $O(1)$ rounds using a aggregation and dissemination on the depth-2 BFS tree within each T_i. This allows each $v \in S_i$ to compute $\sum_{j<i} |S_j|$ for the last step of the algorithm.

In addition, it also holds w.h.p. that each $S_i \subseteq T_i$ has size $O(\log n)$. Assuming this holds, running Relabel in Step 3 only requires $O(1)$ rounds per Lemma 17, and it succeeds w.h.p. Finally, the process takes $O(\log \log n)$ rounds due to Step 4, during which a leader node within each T_i broadcasts $O(\log n)$ labels of $O(\log \log n)$ bits each.

We now argue the approximate uniformity of the sampling. Consider the random process in which each node in S picks a random ordered bucket independently and u.a.r, and then each bucket is permuted uniformly at random. Let μ be the distribution of the permutation generated by this process. Clearly, μ is the uniform distribution. Permute is the same as this process, except it does not output anything if some high probability event $\mathcal{E}$ does not hold. More precisely, the high probability event $\mathcal{E}$ corresponds to all buckets being 2-connected, all buckets being of $O(\log n)$ size, and Relabel succeeding. Let μ_1 be the distribution μ conditioned on $\mathcal{E}$ holding, and μ_2 be μ conditioned on $\mathcal{E}$ not holding. Distribution μ_1 is the output distribution of Permute, and we have $\mu = (1 - 1/\text{poly}(n))\mu_1 + (1/\text{poly}(n))\mu_2$. Thus, for each permutation π, $\mu_1(\pi) \le \mu(\pi)/(1 - 1/\text{poly}(n)) = 1/((1 - 1/\text{poly}(n))|S|!)$. ∎

Reducing the complexity to a constant. Our $O(1)$ implementation improves on the running time by splitting buckets from the first "rough shuffling" into sub-buckets, and arguing that most such buckets satisfy properties allowing them to use a leader to permute themselves as in Algorithm 4, while buckets that fail this second sub-bucketing are few enough that they can be efficiently permuted with the help of the whole almost-clique. We sketch the main ideas behind Lemma 19 here and defer all necessary details to the full version [19].

Lemma 19. *There is an algorithm simulating the permutation sampling step of the synchronized color trial in $O(1)$ rounds of* BCONGEST.

Proof sketch. For simplicity, let us assume we want to permute the whole almost-clique K. The overall structure of our $O(1)$ algorithm for permuting K is as follows:

(1) each $v \in K$ picks random $t(v) \in [k]$ and $t'(v) \in [k']$,
(2) Let $S_i := \{v \in K : t(v) = i\}$ and $S_{i,i'} := \{v \in S_i : t'(v) = i'\}$. Each $S_{i,i'}$ generates a random permutation $\rho_{i,i'}$ of its elements,
(3) a node $v \in S_{i,i'}$ takes $\pi(v) := \sum_{(j,j')<(i,i')}|S_{i,i'}| + \rho_{i,i'}(v)$ as index in the output permutation π.

The numbers k and k' used in our random assignments are chosen of order $k \in \Theta(\Delta/\log n)$ and $k' \in \Theta(\log\log n)$. For every node $v \in K$, in expectation, its neighborhood contains $\Theta(\log n)$ members of each S_i and $\Theta(\log n/\log\log n)$ members of each $S_{i,i'}$. We can claim that each node in K has $\Theta(\log n)$ neighbors in each S_i, w.h.p. For the sets $S_{i,i'}$, we argue that most of them have size $\Theta(\log n/\log\log n)$ and diameter 2.

We permute the sets $S_{i,i'}$ of size $\Theta(\log n/\log\log n)$ and diameter 2 the same way that sets S_i are permuted in the $O(\log\log n)$ algorithm: a node in them chooses a random permutation. The sets being smaller makes the process $\Theta(\log\log n)$ times faster.

Other sets $S_{i,i'}$ use the assistance of the whole almost-clique to permute themselves. Nodes in such sets $S_{i,i'}$ each pick a big random number, send it to all other nodes, and order themselves within their $S_{i,i'}$ according to the big random numbers. We argue that there are few enough sets $S_{i,i'}$ to be permuted using this method that Many-to-All Broadcast (Claim 5) can be used.

Like the $O(\log\log n)$ algorithm, the output distribution of the $O(1)$ algorithm is close-to-uniform, with the same argument. ∎

5 COLORING IN STREAMING-CONGEST

Definition 10. *We define* BCStream *to be the* BCONGEST *model in which, per round, each node receives the messages from its neighbors in a streaming fashion, using $O(\log^c n)$ memory for some fixed $c > 0$.*

Note that results in BCStream constrain the size of the messages more than equivalent results in CONGEST or BCONGEST. In the latter models, the size of the messages can be freely changed between $c\log n$ and $c'\log n$ for two positive constants c and c' without changing $\omega(1)$ asymptotic complexities. This is because, without a memory constraint, for $c > c' > 0$, nodes can simulate an algorithm using $c\log n$-bit messages by buffering the $c'\log n$-bit messages received from each neighbor over $\lceil c/c'\rceil$ rounds. Such buffering uses $\Theta(\Delta\log n)$ memory and is impossible in BCStream. In BCStream, having a T-round algorithm for a given problem means that there exist constants $c, c' > 0$ s.t. given that nodes can send messages of size $c\log n + c'$, they can solve the problem in T rounds.

Running a randomized color trial remains feasible under BCStream constraints. As this consists of the core of our algorithm, most steps carry over to this model. The technical difficulties to overcome are: (1) (high-degree) nodes cannot store all colors used in their neighborhood, in order to know their palette; and (2) dense nodes cannot learn the full clique palette nor the full permutation π during the synchronized color trial.

Dealing with the first issue is fairly straightforward since in order to overcome the broadcast constraint, nodes sample colors in publicly known sets of colors (e.g., $[\Delta+1]$ or $[x(v)]$). After sampling colors in such a set, a node can learn which sampled colors belong to its palette in one communication round (where each colored node broadcasts its color).

The synchronized color trial (Step 2 of Algorithm 1) requires more care. Note that a node v merely needs to know its index in the permutation $\pi(v)$ and the $\pi(v)$-th color in the clique-palette. Lemmas 16 and 18 are both based on the idea of "random bucketing". Let us focus on the permutation and consider Algorithm 4. As each bucket contains $O(\log n)$ nodes, Relabel requires only poly $\log n$ memory (Algorithm 3). What remains, then, is to compute the prefix sum $\sum_{j<i}|S_j|$ counting the number of elements in buckets of lower indices (Step 5 of Algorithm 4). Compared to BCONGEST, the challenge is to avoid double counting. Indeed, in Step 2 of Permute, nodes receive $\Theta(\log n)$ times each term $|S_j|$ of the sum.

Computing prefix sums $\sum_{j<i}|S_j|$ can be done in $O(\log\log n)$ rounds of BCStream. To achieve this, we progressively merge together the S_i's into larger groups, keeping track of the groups' sizes as they merge. Say groups have size z, the main idea is to merge $z^{1/2}$ groups together. Computing the size of the result of this merge involves summing $z^{1/2}$ group sizes. In each group, nodes choose a term to learn in the sum at random (among the $z^{1/2}$ terms). In expectation, $z^{1/2}$ nodes are assigned to each term. Because of the highly connected structure of almost-cliques, we can elect a *unique* node for each term, allowing us to aggregate all values without double counting. Since the sizes of the groups grow polynomially, after $O(\log\log n)$ rounds, all sums have been computed. As the proof is quite technical, we defer it to the full version [19].

Lemma 20. *Let T_i be a family of sets such as described in Lemma 15. Suppose nodes of each group T_i knows some value $y_i \le \text{poly}(n)$. There is a $O(\log\log n)$-round* BCStream *algorithm such that w.h.p. all nodes in T_i learn $\sum_{j<i} y_j$.*

The same idea allows nodes to find the i-th color in the clique palette. When the only remaining nodes are from the put-aside sets, the algorithm only requires poly $\log n$ memory. Indeed, we can assume the clique palette has size $O(\log^3 n)$ and we sample $O(\log^3 n)$ colors at each step of the process. Observe that the communication procedure described in Claim 5 works in BCStream if nodes know in advance which messages they need to store (e.g., the i-th color in the clique palette) or if the total number of messages is poly $\log n$ (e.g., when coloring the put-aside sets).

ACKNOWLEDGMENTS

This work was supported by the Icelandic Research Fund grants 217965 and 2310015-051.

REFERENCES

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Graph sketches: sparsification, spanners, and subgraphs. In *PODS*. ACM, 5–14. https://doi.org/10.1145/2213556.2213560

[2] Noga Alon and Sepehr Assadi. 2020. Palette Sparsification Beyond $(\Delta + 1)$ Vertex Coloring. In *APPROX/RANDOM (LIPIcs, Vol. 176)*. LZI, 6:1–6:22. https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2020.6

[3] Noga Alon, László Babai, and Alon Itai. 1986. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *J. of Algorithms* 7, 4 (1986), 567–583. https://doi.org/10.1016/0196-6774(86)90019-2

[4] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In *SODA*. SIAM, 767–786. https://doi.org/10.1137/1.9781611975482.48

[5] Sepehr Assadi, Gillat Kol, and Rotem Oshman. 2020. Lower bounds for distributed sketching of maximal matchings and maximal independent sets. In *PODC*. ACM, 79–88. https://doi.org/10.1145/3382734.3405732

[6] Sepehr Assadi, Gillat Kol, and Zhijun Zhang. 2022. Rounds vs communication tradeoffs for maximal independent sets. In *FOCS*. IEEE, 1193–1204. https://doi.org/10.1109/FOCS54457.2022.00115

[7] Sepehr Assadi, Pankaj Kumar, and Parth Mittal. 2022. Brooks' theorem in graph streams: a single-pass semi-streaming algorithm for Δ-coloring. In *STOC*. ACM, 234–247. https://doi.org/10.1145/3519935.3520005

[8] Leonid Barenboim. 2016. Deterministic $(\Delta + 1)$-Coloring in Sublinear (in Δ) Time in Static, Dynamic, and Faulty Networks. *J. ACM* 63, 5 (2016), 47:1–47:22. https://doi.org/10.1145/2979675

[9] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2016. The Locality of Distributed Symmetry Breaking. *J. ACM* 63, 3 (2016), 20:1–20:45. https://doi.org/10.1145/2903137

[10] József Beck. 1991. An algorithmic approach to the Lovász local lemma. I. *Random Structures & Algorithms* 2, 4 (1991), 343–365. https://doi.org/10.1002/rsa.3240020402

[11] Florent Becker, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. 2020. The Impact of Locality in the Broadcast Congested Clique Model. *SIAM Journal on Discrete Mathematics* 34, 1 (2020), 682–700. https://doi.org/10.1137/18M1233534

[12] Ruben Becker, Sebastian Forster, Andreas Karrenbauer, and Christoph Lenzen. 2021. Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models. *SIAM Journal on Computing* 50, 3 (2021), 815–856. https://doi.org/10.1137/19M1286955

[13] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. 2020. Distributed $(\Delta + 1)$-Coloring via Ultrafast Graph Shattering. *SIAM Journal on Computing* 49, 3 (2020), 497–539. https://doi.org/10.1137/19M1249527

[14] Shiri Chechik and Doron Mukhtar. 2019. Reachability and Shortest Paths in the Broadcast CONGEST Model. In *DISC (LIPIcs, Vol. 146)*. LZI, 11:1–11:13. https://doi.org/10.4230/LIPIcs.DISC.2019.11

[15] Benjamin Doerr. 2020. *Probabilistic Tools for the Analysis of Randomized Optimization Heuristics*. Springer International Publishing, 1–87. https://doi.org/10.1007/978-3-030-29414-4_1

[16] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. 2014. On the power of the congested clique model. In *PODC*. ACM, 367–376. https://doi.org/10.1145/2611462.2611493

[17] Michael Elkin, Seth Pettie, and Hsin-Hao Su. 2015. $(2\Delta - 1)$-Edge-Coloring is Much Easier than Maximal Matching in the Distributed Setting. In *SODA*. SIAM, 355–370. https://doi.org/10.1137/1.9781611973730.26

[18] Manuela Fischer, Magnús M. Halldórsson, and Yannic Maus. 2023. Fast Distributed Brooks' Theorem. In *SODA*. SIAM, 2567–2588. https://doi.org/10.1137/1.9781611977554.ch98

[19] Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. 2023. *Coloring Fast with Broadcasts*. Technical Report 2304.09844. arXiv. https://doi.org/10.48550/arXiv.2304.09844 Full version of this paper.

[20] Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. 2023. *A Distributed Palette Sparsification Theorem*. Technical Report 2301.06457. arXiv. https://doi.org/10.48550/arxiv.2301.06457

[21] Sebastian Forster and Tijn de Vos. 2022. The Laplacian Paradigm in the Broadcast Congested Clique. In *PODC*. ACM, 335–344. https://doi.org/10.1145/3519270.3538436

[22] Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. 2016. Local Conflict Coloring. In *FOCS*. IEEE Computer Society, 625–634. https://doi.org/10.1109/FOCS.2016.73

[23] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. 2021. Improved Deterministic Network Decomposition. In *SODA*. SIAM, 2904–2923. https://doi.org/10.1137/1.9781611976465.173

[24] Mohsen Ghaffari and Fabian Kuhn. 2021. Deterministic Distributed Vertex Coloring: Simpler, Faster, and without Network Decomposition. In *FOCS*. IEEE Computer Society, 1009–1020. https://doi.org/10.1109/FOCS52979.2021.00101

[25] Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. 2021. Efficient Randomized Distributed Coloring in CONGEST. In *STOC*. ACM, 1180–1193. https://doi.org/10.1145/3406325.3451089

[26] Magnús M. Halldórsson and Alexandre Nolin. 2023. Superfast coloring in CONGEST via efficient color sampling. *Theor. Comput. Sci.* 948 (2023), 113711. https://doi.org/10.1016/j.tcs.2023.113711

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

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

[29] David G. Harris, Johannes Schneider, and Hsin-Hao Su. 2018. Distributed $(\Delta + 1)$-coloring in sublogarithmic rounds. *J. ACM* 65 (2018), 19:1–19:21. https://doi.org/10.1145/3178120

[30] Öjvind Johansson. 1999. Simple Distributed $\Delta + 1$-Coloring of Graphs. *Inf. Process. Lett.* 70, 5 (1999), 229–232. https://doi.org/10.1016/S0020-0190(99)00064-2

[31] Tomasz Jurdziński and Krzysztof Nowicki. 2018. Connectivity and minimum cut approximation in the broadcast congested clique. In *SIROCCO (LNCS, Vol. 11085)*. Springer, 331–344. https://doi.org/10.1007/978-3-030-01325-7_28

[32] Janne H. Korhonen and Joel Rybicki. 2017. Deterministic Subgraph Detection in Broadcast CONGEST. In *OPODIS (LIPIcs, Vol. 95)*. LZI, 4:1–4:16. https://doi.org/10.4230/LIPIcs.OPODIS.2017.4

[33] Nathan Linial. 1992. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing* 21, 1 (1992), 193–201. https://doi.org/10.1137/0221015

[34] M. Luby. 1986. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing* 15 (1986), 1036–1053. https://doi.org/10.1137/0215074

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

[36] Alessandro Panconesi and Aravind Srinivasan. 1997. Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding Bounds. *SIAM Journal on Computing* 26, 2 (1997), 350–368. https://doi.org/10.1137/S0097539793250767

[37] Boaz Patt-Shamir and Mor Perry. 2022. Proof-labeling schemes: Broadcast, unicast and in between. *Theor. Comput. Sci.* 923 (2022), 179–195. https://doi.org/10.1016/j.tcs.2022.05.006

[38] David Peleg. 2000. *Distributed Computing: A Locality-Sensitive Approach*. SIAM.

[39] Bruce A. Reed. 1998. ω, Δ, and χ. *J. Graph Theory* 27, 4 (1998), 177–212.

[40] Václav Rozhoň and Mohsen Ghaffari. 2020. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *STOC*. ACM, 350–363. https://doi.org/10.1145/3357713.3384298

[41] Johannes Schneider and Roger Wattenhofer. 2010. A new technique for distributed symmetry breaking. In *PODC*. ACM, 257–266. https://doi.org/10.1145/1835698.1835760