Beyond Brooks: $(\Delta - 1)$-Coloring in Semi-Streaming

Maxime Flin
Aalto University, Espoo, Finland

Magnús M. Halldórsson
Reykjavik University, Iceland

Abstract

Reed [J. Comb. Theory B, 1999] showed that graphs of maximum degree $\Delta \geqslant 10^{14}$ without Δ-cliques are $(\Delta - 1)$-colorable. We design a one-pass semi-streaming algorithm for computing such a coloring. Additionally, we prove that any one-pass $(\Delta - k)$-coloring algorithm for $0 \leqslant k < (\Delta + 1)/2$ requires $\Omega(n(k + 1))$ space.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Graph coloring, streaming algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.92

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2605.07774 [30]

Funding *Maxime Flin*: Supported in part by the Research Council of Finland, Grants 359104 and 363558. Part of this work was done while M. Flin was working at Reykjavik University, funded by the Icelandic Research Fund, Grant 2310015-053.
Magnús M. Halldórsson: Supported by the Icelandic Research Fund, Grant 2511609.

1 Introduction

Graph coloring is a central problem in combinatorics and theoretical computer science. Given a graph $G = (V, E)$ and an integer $q \geqslant 1$, a q-coloring assigns a color from $\{1, \ldots, q\}$ to each vertex so that adjacent vertices receive different colors. Coloring has been extensively studied in diverse computational models, including the classical RAM [45, 41], distributed message-passing [43, 10, 36, 19], sublinear models [3, 18, 22, 7], dynamic algorithms [13, 37, 14, 11], communication complexity [31, 20], and streaming models [3, 1, 12, 2, 17, 4, 7]. In semi-streaming, the algorithm must produce a coloring after processing the edges of an n-vertex graph one at a time in an adversarial order while using $O(n \cdot \text{poly} \log n)$ space.

In this work, we study the streaming complexity of coloring with fewer than Δ colors, where Δ is the maximum degree. We give the first one-pass semi-streaming algorithm that outputs a $(\Delta - 1)$-coloring for graphs with large enough degree and no Δ-clique.

Coloring With $\Delta + 1$ or Δ Colors. The classic $(\Delta + 1)$-coloring problem is easy sequentially because one can always *extend a partial solution* to a full solution. Notwithstanding, the greedy approach does not work in semi-streaming; thus to $(\Delta + 1)$-color in semi-streaming, Assadi, Chen, and Khanna [3] had to invent the influential *palette sparsification* technique: if one samples $O(\log n)$ colors from $\{1, 2, \ldots, \Delta + 1\}$ independently for each vertex, then with high probability, all vertices can be colored with a color from their list. It has been further leveraged in sublinear algorithms [1, 4, 7], distributed algorithms [34, 27], dynamic algorithms [11], and discrete mathematics [39, 40, 23].

By a theorem of Brooks [16], Δ colors always suffice except for $(\Delta + 1)$-cliques or odd cycles. Assadi, Kumar, and Mittal [4] gave a semi-streaming algorithm for Δ-coloring, in spite of the fact that palette sparsification provably fails [3, Proposition C.2] and such an algorithm only exists when each edge appears at most once in the stream [4]. To that end, they introduced the novel sketching technique of sparse recovery.

In the scale of things, Δ-coloring is conceptually easy. It has spawned a great many solutions, starting with [46, 44], including newer linear-time algorithms [8, 52]; see comprehensive treatments in [21] and the book of [53]. *Modulo a single vertex*, it can be greedily colored, as there is an order under which it suffices to assign each node an arbitrary available color. The $(\Delta - 1)$-coloring problem does not have such a greedy or near-greedy property.

Beyond Brooks: $(\Delta - 1)$-Coloring. In [51], Reed showed that Brooks barely scratched the surface: in fact, as the size of the largest clique decreases, so does the chromatic number. Borodin and Kostochka [15] conjectured that every graph of maximum degree $\Delta \geqslant 9$ without a Δ-clique is $(\Delta - 1)$-colorable. Reed [50] proved the conjecture for $\Delta \geqslant 10^{14}$. The conjecture is otherwise open, but holds for claw-free graphs [21]. Despite significant advances in $(\Delta + 1)$- and even Δ-coloring, going further in the semi-streaming model has remained elusive.

Unsurprisingly, Reed's result is an order of magnitude more involved than the many simple proofs of Brooks' theorem. Reed's approach – based on technical structural observations and the probabilistic method – is highly non-trivial and remains (effectively) the *only one known*. As such, Reed's theorem is not merely a quantitative improvement but an exhibit for the existence of a conceptual leap between $(\Delta - 1)$-coloring and Δ-coloring. Indeed, unlike proofs of Brooks' Theorem (including that of [4]), Reed's argument is far from being greedy: it begins by transforming the graph – specifically, by adding edges. Given the already strenuous effort necessary to obtain a $(\Delta + 1)$-coloring and Δ-coloring in semi-streaming and the technical depth of Reed's theorem, it is natural to ask if Δ-coloring is the best we can hope for in semi-streaming:

> *Does there exist a (one-pass) semi-streaming algorithm to color with fewer than Δ colors any given graph of maximum degree Δ (large enough) and no Δ-cliques?*

Even if such an algorithm exists, the concern is that removing one more color may require an explosion of cases, considering, e.g., the much larger number of cases in the Δ-coloring algorithms [4, 26] than for $(\Delta + 1)$-coloring [3, 35].

1.1 Our Contributions

The main technical contribution of this paper is a semi-streaming algorithm for $(\Delta - 1)$-coloring graphs of sufficiently high degree. Concretely, we prove the following.

▶ **Theorem 1.** *There exists a universal constant Δ_0 for which the following holds. There exists a randomized one-pass semi-streaming algorithm that given any graph $G = (V, E)$ with maximum degree $\Delta \geqslant \Delta_0$ and containing no Δ-clique, outputs a $(\Delta - 1)$-coloring of G with high probability.*

During the streaming pass, akin to [4], our algorithm samples $\text{poly}(\log n)$-sized random lists and stores only the edges of the sparsified graph, meanwhile using their sparse recovery technique to learn about the densest regions of the graph. The coloring algorithm differs substantially: we begin by transforming the graph – both removing extremely dense subgraphs and carefully adding edges – and then leverage various kinds of probabilistic coloring arguments to construct a coloring.

Clearly, the impossibility results highlighted in [4] also apply to us. We therefore assume that each edge appears at most once in the stream. It is also worth noting that randomness is essential, as Assadi, Chen, and Sun [2] demonstrated that any deterministic semi-streaming algorithm requires $\exp(\Delta^{o(1)})$ colors.

As a final remark on Theorem 1, we observe that Δ_0 is in fact the smallest universal constant for which graphs of maximum degree $\Delta \geqslant \Delta_0$ and no Δ-clique admit a $(\Delta-1)$-coloring. Reed proved in [50] that $\Delta_0 \leqslant 10^{14}$. For graphs of maximum degree $\Delta \leqslant 10^{14}$, all edges $O(n\Delta) = O(n)$ in the graph can be stored, and an optimal coloring can be computed (although, not in polynomial time). We have not tried to reduce this constant; in [50, Section 5], Reed claims that $\Delta_0 = 10^6$ also works with more care, while $\Delta_0 < 100$ would need different techniques.

Using Fewer Colors. A natural follow-up question from Theorem 1 is to what extent this can be pushed further. Molloy and Reed [48] gave a characterization of $(\Delta-k)$-colorable graphs in terms of forbidden local subgraphs, for all $k \lesssim \sqrt{\Delta}$. This generalizes Reed's result and suggests an alternative approach. However, their structural decomposition is delicate and obtained through phases of contracting vertices and computing optimal partitions. We prove that, in fact, computing a $(\Delta-k)$-coloring requires $\Omega(n(k+1))$ space, prohibiting semi-streaming algorithms for $(\Delta-k)$-coloring unless k is polylogarithmic.

▶ **Theorem 2.** *Let $\Delta, c > 0$ be non-negative integers such that $(\Delta+1)/2 < c \leqslant \Delta$. Any (possibly randomized) one-pass streaming algorithm that outputs a c-coloring of given c-colorable n-vertex graph of maximum degree Δ requires $\Omega(n(\Delta-c+1))$ space.*

Very recently, Assadi, Sundaresan, and Yazdanyar [5] studied the problem of distinguishing graphs of low chromatic number (e.g., constant) from those with a large chromatic number (e.g., $\sqrt{n}$). They proved, amongst other things, that it requires $n^{2-o(1)}$ space in adversarial streams. Theorem 2 is orthogonal to (and much simpler than) their results since it considers graphs whose chromatic number is one less than their clique number.

1.2 Organization of the Paper

In Section 2, we give an overview of the challenges and technical ideas necessary to obtain Theorem 1. At the end of Section 2, we also provide a more detailed comparison with [4, 29]. In Section 4, we detail the information stored during the streaming pass and describe the coloring algorithm in Section 5. Theorem 2 is presented in Section 6.

2 Technical Introduction

We give an overview of the ideas behind Theorem 1. First, we present the three components for constructing $(\Delta-1)$-colorings, followed by a description of the high-level algorithm that combines them. We then explain how to implement these components in semi-streaming. For a complete step-by-step description, see Section 5. In this introduction, various concepts are simplified to aid intuition.

Pre-/Post-Processing. We start by identifying subgraphs in the graph that are easily colorable regardless of the coloring in the rest of the graph. During pre-processing, we remove those subgraphs and only color them at the very end during post-processing.

For instance, consider the $(\Delta+1)$-clique missing two disjoint edges, as illustrated in Figure 1a. In any $(\Delta-1)$-coloring of such a subgraph, both endpoints of each anti-edge must be assigned the same color. Once these pairs of vertices are colored the same, the remaining

vertices can be colored greedily. In general, the subgraphs shown in Figure 1 are such that any coloring of the outer vertices can be extended to a coloring of the inner vertices as well (they are, in fact, (deg -1)-choosable). This property allows us to defer their coloring to the very end of the algorithm.

To detect these subgraphs, during pre-processing, we decompose the graph into *(locally) sparse vertices* – which have many missing edges in their neighborhood – and *almost-cliques* – sets of vertices that can be turned into $(\Delta + 1)$-cliques by modifying an ε-fraction of their edges and vertices. We look for those choosable subgraphs only in the densest almost-cliques, as the other vertices can be colored by other means. We then distinguish two main types of choosable components: the *solitary* ones are inscribed inside an almost-clique (as on Figures 1a and 1b), and the *popular* ones for which the choosable component contains some vertices outside the almost-clique (as on Figure 1c).

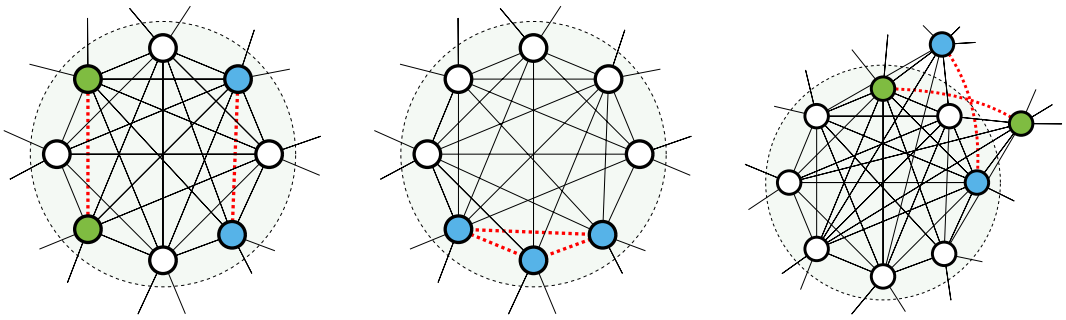

Figure 1 The three types of subgraphs $G[X]$ such that any coloring of $G[V - X]$ can be extended to a coloring of G. The vertices inside the gray circle are in the almost-clique. Figure 1a represent an almost-clique with an induced 2-anti-matching. Figure 1b represents an almost-clique with a 3-independent set. Figure 1c represents a $(\Delta - 1)$-clique with two vertices sharing a neighbor in the clique, each with a different anti-neighbor in the clique. To color each such graph, we must ensure that the highlighted pairs of vertices are same-colored, which is unlikely to occur through randomized coloring arguments.

For the remainder of this technical introduction, we focus on the case of Δ-regular nodes in disjoint $(\Delta - 1)$-cliques, as this already captures the main challenges and intuition for $(\Delta - 1)$-coloring.

Generating Slack Probabilistically. Rather than directly attempting to color all vertices, many coloring algorithms begin by coloring a few vertices randomly. The hope is that uncolored vertices have neighbors that pick the same color, which increases the flexibility when completing the coloring.

Consider the following random coloring process. Each vertex flips a coin and, if it is heads, randomly picks a color. A vertex keeps its color only if no neighbor selects the same one. If v is a vertex in a $(\Delta - 1)$-clique such as the one in Figure 2a, basic probabilistic analysis shows that v has a constant probability of saving two colors – meaning that v remains inactive, and both of its external neighbors retain colors used elsewhere in the clique. We say that v was successful.

For a $(\Delta - 1)$-clique with all (or most) of its external neighbors distinct, one can show that with high probability it contains two successful vertices. Then, we can defer the coloring of successful vertices and color the rest of the clique greedily. This technique generalizes to any almost-clique without an external node connected to most of its internal vertices.

Reed Transform. The most difficult case occurs when a $(\Delta-1)$-clique C has an external vertex – called a friend – with numerous neighbors in C, yet lacks any structure that yields probabilistic slack or choosability. To illustrate the challenge, consider the situation where the vertices are adjacent to either s_1 or s_2, but not both. If the probabilistic coloring procedure colors s_1 and s_2 the same (an event with probability $\approx 1/\Delta$), then the coloring cannot be extended to the rest of C (see Figure 2b). It is crucial to note that recoloring vertices might convert successful vertices into *un*successful ones, rendering the approach infeasible.

To handle such cliques, we apply the Reed Transform, based on a construction from [50]. Identify a specific substructure in C (vertices s, x, y, u, v as in Figure 2c), delete C from the graph, and add a single edge xy (in dashed gray in Figure 2c). Once the rest of the graph is colored – forcing x and y to have distinct colors – we reinsert and color C by coloring v as x and s as some anti-neighbor $z \in C - N(s)$. Naturally, x and y must be chosen carefully to avoid creating a Δ-clique when we connect them.

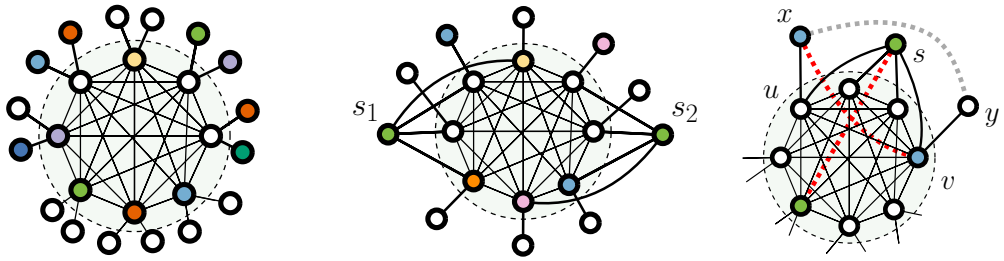

Figure 2 A $(\Delta-1)$-clique in different configurations. On Figure 2a it has many different external neighbors. On Figure 2b, all vertices are either adjacent to s_1 on the left or s_2 on the right. If slack generation colors s_1 and s_2 the same, there is no way to extend the coloring. On Figure 2c, a representation of the Reed Transform: the clique is removed, and the dashed edge added to the graph; the coloring can be later extended by same coloring the s vertex with some vertex of the clique and the v-vertex like the x-vertex. The edge added between the x- and y-vertex ensures this is possible.

The Coloring Algorithm. With this, we now describe the high-level structure of our $(\Delta-1)$-coloring algorithm. Despite our intricate analysis, each step is conceptually simple:

1. Identify and remove choosable components (solitary and popular as in Figure 1);
2. Apply the Reed Transform to troublesome $(\Delta-1)$-cliques;
3. Use probabilistic slack to color the remaining graph greedily;
4. Invert the Reed transform to color the removed cliques;
5. Complete the coloring of the choosable components.

Note how the steps pair with each other and correspond to one of the aforementioned three ideas. For the remainder of this technical introduction, we outline how each step can be implemented in semi-streaming, discuss the challenges, and present our solutions.

Streaming Implementation. We focus here on the implementation of our coloring algorithm after the streaming pass is done, for the streaming algorithm itself largely follows from [3, 4]: running palette sparsification and computing linear sketches. See Section 4.4 for a summary of the information recovered by the algorithm after the streaming pass.

First, we observe that we may focus on the densest almost-cliques, for the other ones can be colored by palette sparsification. Using the sparse-recovery technique of [4], we recover almost all the edges incident to the densest almost-cliques. In particular, it allows us to

identify the choosable subgraphs and the $(\Delta - 1)$-cliques that need to be removed by the Reed Transform. Implementing the Reed Transform requires that we avoid the creation of a Δ-clique and that we preserve invertibility for later steps. We must also ensure that we do not densify sparse neighborhoods, as this could inadvertently create the dense subgraphs we sought to isolate in Step (1). Remarkably, simple downsampling suffices: it limits the number of edges added per vertex while preserving enough flexibility to avoid creating a Δ-clique.

We avoid monochromatic edges by keeping a *serene coloring*: a coloring for which each node uses a color from its random list unless all its incident edges have been recovered, in which case it may choose its color freely. We frequently combine palette sparsification with sparse-recovery. For instance, when coloring popular almost-cliques, some nodes are colored using their random lists, while others are colored greedily, as in the classical setting. Overall, our algorithm makes significantly more use of the edges recovered by sparse recovery than [4]. Though subtle, this difference proves consequential: it frees us to focus more deeply on the graph's structural properties.

The second major challenge is a technical one: proving concentration on the probabilistic guarantees of the randomized coloring procedure described earlier. Indeed, to invert the Reed Transform, we require that each friend has at least $\Delta/\text{poly}(\log n)$ edges into the clique. Consequently, we can no longer guarantee that most external neighbors are distinct. At a high level, this undermines the apparent randomness of colors sampled outside the almost-clique – they may originate from too few vertices to provide sufficient diversity. Unfortunately, standard concentration inequalities fall short in capturing this behavior. Instead, we rely on a case analysis that combines multiple forms of slack-based concentration, sometimes even blending probabilistic and temporal slack to recover the guarantees we need.

Comparison with [4]. In [4], the authors design a semi-streaming algorithm for Δ-coloring and, in particular, introduces the sparse-recovery technique used by our algorithm. In contrast, while [4] would use it to recover edges incident on at most two vertices for each of the densest cliques (see Definitions 4.6 and 4.7 of [4]), we heavily use that we can recover all edges incident on the densest cliques, most notably for implementing and inverting the Reed Transform.

Coloring with fewer than $(\Delta - 1)$-colors. Molloy and Reed [48] generalized Reed's result to $(\Delta - k)$-colorable graphs, for all values of k up to roughly $\sqrt{\Delta}$. As shown in [9, 29], this can be obtained in the LOCAL model [43, 38] in a near-optimal round complexity of $\text{poly}(\log\log n)$. The lower bound of Theorem 2 proves that $(\Delta - k)$-coloring is not attainable in semi-streaming unless $k = \text{poly}(\log n)$.

Our work leaves open the question of $(\Delta - k)$-coloring in the streaming model for $k \geqslant 2$, specifically whether it can be achieved in $n \cdot \text{poly}(k, \log n)$ space. Conceptually, as k increases, the number of cases involving dense cliques grows rapidly, as it no longer suffices to track two anti-edges per dense clique.

To avoid this combinatorial explosion, the main alternative is to follow the approach of [48], which provides a systematic treatment for arbitrary k via a generalization of the Reed transform. This framework starts from a coarse structural decomposition (akin to the almost-clique decomposition) and progressively refines it through increasingly intricate local transformations. These transformations appear to require the full knowledge of the local topology as they rely, among other things, on the optimal coloring of dense induced subgraphs. While it is conceivable that, using the sparse recovery technique with $\text{poly}(k)$-sparse vectors (thus, $n \cdot \text{poly}(k, \log n)$ space), one could recover most of the edges necessary to implement this intricate transformation, the feasibility of this approach remains open.

3 Preliminaries

Notation. For an integer $k \geqslant 1$, we use $[k]$ to denote $\{1, 2, \dots, k\}$. For sets A and B, we write $A - B = \{a \in A : a \notin B\}$. For succinctness, we abuse notation slightly and write $A - x$ and $A + x$ for $A - \{x\}$ and $A \cup \{x\}$ respectively. For a function $f : X \to Y$, let $f(A) = \{f(a) : a \in A\}$ and $f^{-1}(B) = \{x : f(x) \in B\}$ where $A \subseteq X$ and $B \subseteq Y$ respectively. We abuse notation slightly and occasionally write $f^{-1}(y)$ instead of $f^{-1}(\{y\})$ when $y \in Y$.

Let $G = (V, E)$ be a simple graph. We denote by $n = |V|$ and $m = |E|$ its respective number of vertices and edges. For a set $S \subseteq V$, the induced subgraph $G[S]$ is the graph on vertex set S and all edges of E between pairs of vertices in S. The neighborhood of $v \in V$ is $N_G(v) = \{u \in V : \{u, v\} \in E\}$ and $\deg(v, G) = |N_G(v)|$. The closed neighborhood of v is $N_G[v] = N_G(v) + v$. When G is clear from context, we write $N(v)$ and $\deg(v)$. The maximum degree of G is denoted by Δ. We refer to a pair $\{u, v\}$ as a non-edge or an anti-edge when no edge connects u and v. A k-anti-matching is a set of k anti-edges such that each vertex belongs to at most one anti-edge. A k-independent set is a set of k vertices that are not connected by any edge.

For an integer $q \geqslant 1$, a partial q-coloring is a mapping $\varphi : V \to [q] \cup \{\bot\}$ such that $\varphi(u) \neq \varphi(v)$ or $\bot \in \{\varphi(u), \varphi(v)\}$ for all $\{u, v\} \in E$. The set of colored vertices is the domain $\operatorname{dom}\varphi = \{v \in V : \varphi(v) \neq \bot\}$. Given a partial coloring φ, the uncolored degree of v is $\deg_\varphi(v, G) = |N_G(v) - \operatorname{dom}\varphi|$. Throughout the paper, we denote by $L_\varphi(v) = [\Delta - 1] - \varphi(N(v))$ the set of colors unused by neighbors of v with respect to a (possibly partial) coloring φ. We say that a coloring ψ is an extension of φ if $\psi(v) = \varphi(v)$ for all $v \in \operatorname{dom}\varphi$.

We say that an event occurs *with high probability* (in n) if it occurs with probability at least $1 - n^{-c}$, for a desirably large constant $c > 0$. An event occurs *with exponentially high probability* in x if it occurs with probability at least $1 - \exp(-\Omega(x))$. We abridge "with probability", "with high probability" and "with exponentially high probability" as "w.p.", "w.h.p." and "w.e.h.p." respectively.

3.1 Sparse-Dense Decomposition

We use the variant of the sparse-dense decomposition from [36, 3, 6, 4] (inspired by [51, 47]). It partitions vertices between (locally) sparse vertices – with many anti-edges in their neighborhood – and dense vertices clustered in almost-cliques – subgraphs that resemble a $(\Delta + 1)$-clique.

▶ **Definition 3.** *A vertex v is* ζ-sparse *in G if $G[N(v)]$ contains at most $\binom{\Delta}{2} - \zeta\Delta$ edges.*

▶ **Definition 4.** *A set $C \subseteq V$ is an* (ε, δ)-almost-clique *if*

1. $(1 - \varepsilon/2)\Delta \leqslant |C| \leqslant (1 + \varepsilon/2)\Delta$,
2. $|C - N[v]|, |N(v) - C| \leqslant \varepsilon\Delta$ *for all $v \in C$, and*
3. *every $v \notin C$ has $|C - N(v)| \geqslant \delta\Delta$.*

We stress that (3) is not always included as part of the definition of almost-clique (e.g., in [36, 3]). It is however essential for us that nodes outside an almost-clique have $\delta\Delta$ non-neighbor inside the almost-clique (we will always have $\delta = \Omega(\varepsilon)$).

Let v be a vertex in some almost-clique C. We call external neighbors $E(v) = N(v) - C$ its $e(v) = |E(v)|$ neighbors outside C. The vertices of C that are not adjacent to v are its anti-neighbors $A(v) = C - N[v]$; let $a(v) = |A(v)|$ be the anti-degree of v. Part (2) in Definition 4 means that $e(v), a(v) \leqslant \varepsilon\Delta$.

▶ **Definition 5.** *For $\varepsilon, \eta \in (0,1)$, an $(\eta, \varepsilon, \delta)$-almost-clique decomposition of G is a vertex-partition $V_{sparse}, C_1, C_2, \ldots, C_k$ such that*

1. *every $v \in V_{sparse}$ is $\eta\varepsilon^2\Delta$-sparse;*
2. *for every $i \in [k]$, the set C_i is an (ε, δ)-almost-clique.*

Vertices of V_{sparse} are called sparse, and those of $V - V_{sparse}$ are called dense. It follows almost directly from the definitions that dense vertices have sparsity proportional to their anti-degree and external-degree. For a proof, we refer readers to [33, Lemma 6.2] for the bound in terms of anti-degrees and to [28, Lemma 3.6] for the bound in terms of external degrees[1].

▶ **Lemma 6.** *If $V_{sparse}, C_1, \ldots, C_k$ is an $(\eta, \varepsilon, \delta)$-almost-clique decomposition, then every $v \in C_i$ with $e(v) \geqslant 2/(\eta\varepsilon^2)$ is $\max\{\eta\varepsilon^2/4 \cdot e(v), (1-3\varepsilon)/2 \cdot a(v)\}$-sparse.*

A semi-streaming algorithm for computing an almost-clique decomposition exists, as given by [4, Proposition 3.5].

▶ **Proposition 7.** *There are universal constants $\eta, \varepsilon_0 \in (0,1)$ such that the following holds. For any $0 < \varepsilon < \varepsilon_0$, there is a streaming algorithm using $O(\varepsilon^{-2} n \log n)$ space that computes an $(\eta, \varepsilon, \varepsilon)$-almost-clique decomposition.*

3.2 Palette Sparsification

Our streaming algorithm uses the palette sparsification technique from [3] to color some almost-cliques. Assadi, Chen and Khanna view the coloring of an almost-clique as a perfect matching in a bipartite graph, where one side of the bipartition represents vertices and the other side the colors. An edge exists between a vertex and a color in this graph if the color is available to the vertex with respect to the current coloring. When we apply palette sparsification, each edge is retained in the graph independently with some probability, and they prove that, under the right assumptions, it is possible to match all vertices with high probability.

We use the following formulation of the statement, from [4, Lemma 3.10], in a black-box manner. We refer readers to [3, Section 3.5] for a detailed proof. We say that a matching is an $\mathscr{L}$-perfect matching iff it matches all vertices of $\mathscr{L}$.

▶ **Lemma 8.** *Let $\mathscr{G} = (\mathscr{L} \cup \mathscr{R}, E)$ be a bipartite graph with the following properties:*

1. *$k = |\mathscr{L}|$ and $k \leqslant |\mathscr{R}| \leqslant 2k$;*
2. *every vertex $v \in \mathscr{L}$ has $\deg(v, \mathscr{R}) \geqslant 2k/3$; and*
3. *for every set $S \subseteq \mathscr{L}$ with $|S| \geqslant k/2$ we have that $\sum_{v \in S} \deg(v, \mathscr{R}) \geqslant |S| \cdot k - k/4$.*

For any $\delta \in (0,1)$, the subgraph obtained by sampling each edge in $\mathscr{G}$ independently with probability at least $\frac{20}{k}(\log k + \log(1/\delta))$ contains an $\mathscr{L}$-perfect matching with probability at least $1 - \delta$.

As observed by [4], every almost-clique C such that $G[C]$ contains at least $10^7\varepsilon\Delta$ anti-edges (they call them holey) can be colored using palette sparsification. Note that the extension exists with high probability regardless of the coloring outside C.

[1] In contrast to our definition, which includes sparse vertices as external neighbors, in [33], the authors count as external neighbors only vertices from other almost-cliques. The original proof of Lemma 6 is in a technical report [35].

▶ **Lemma 9.** *Let C be an almost-clique inducing at least $10^7\varepsilon\Delta$ anti-edges and let $\mathbf{L}(v)$ be a list of $O_\varepsilon(\log n)$ random colors in $[\Delta-1]$ for each $v \in C$. For any partial $(\Delta-1)$-coloring φ outside C (i.e., $C \cap \operatorname{dom}\varphi = \emptyset$), there exists, w.h.p., an extension ψ of the coloring to C such that $\psi(v) \in \mathbf{L}(v)$ for every $v \in C$.*

We stress that Lemma 9 assumes that C is uncolored. Since the proof is almost verbatim that of [4, Lemma 5.10] or [3], we skip details and refer interested readers to [4, Appendix B.3].

3.3 Concentration

We use the classic Chernoff Bound on sums of independent random variables:

▶ **Proposition 10** (Chernoff bounds, [24, Theorem 1.10.1 and 1.10.5])**.** *Let $X_1, \dots, X_n$ be a family of independent random variables with values in $[0,1]$, and let $X = \sum_{i\in[n]} X_i$. Suppose $\mu_\mathsf{L} \leqslant \mathbf{Exp}[X] \leqslant \mu_\mathsf{H}$, then*

$$\text{for all } \delta \geqslant 0, \text{ we have that} \qquad \mathbf{Pr}[X > (1+\delta)\mu_\mathsf{H}] \leqslant \exp\left(-\frac{\delta^2}{2+\delta}\mu_\mathsf{H}\right). \tag{1}$$

$$\text{for all } \delta \in (0,1), \text{ we have that} \qquad \mathbf{Pr}[X < (1-\delta)\mu_\mathsf{L}] \leqslant \exp\left(-\frac{\delta^2}{2}\mu_\mathsf{L}\right). \tag{2}$$

Let $Y_1, \dots, Y_n$ be boolean random variables (with values in $\{0,1\}$). We say they form a *read-k family* if they can be expressed as a function of independent random variables $X_1, \dots, X_m$ such that each X_j influences at most k variables Y_i. More formally, there are sets $P_i \subseteq [m]$ for each $i \in [n]$ such that: (1) for each $i \in [n]$, the variable Y_i is a function of $\{X_j : j \in P_i\}$ and (2) $|\{i : j \in P_i\}| \leqslant k$ for each $j \in [m]$.

▶ **Proposition 11** (read-k bound, [32])**.** *Let $Y_1, \dots, Y_n$ be a read-k family of boolean variables, and let Y be their sum. Then, for any $\delta > 0$,*

$$\mathbf{Pr}\left[\left|Y - \mathbf{Exp}[Y]\right| > \delta n\right] \leqslant 2\exp\left(-\frac{2\delta^2 n}{k}\right).$$

4 The Streaming Algorithm

In this section, we describe the information we collect during the streaming pass. This amounts to the almost-clique decomposition (Proposition 7), the sparsified graph (Section 4.1) and some linear sketches to recover edges in the densest regions of the graph (Section 4.3). Overall, we use $\widetilde{O}(n)$ space with high probability. In Section 4.2, we introduce structural definitions essential to our coloring algorithm.

Parameters. Let us define some parameters:

$\alpha = 150$	a conveniently large constant,
$\varepsilon = 10^{-8}$	the parameter of the almost-clique decomposition in Proposition 7
$\rho = \frac{c\log n}{\varepsilon^2}$	the sparsity threshold above which palette sparsification works,

where the constant c is large enough to have $\exp(-\Omega(\varepsilon^2\rho)) \leqslant 1/\mathrm{poly}(n)$. We use the dependency on ε of ρ in Section 5.4 to ensure that all dense vertices have two same-colored neighbors. At a high-level, the parameter ρ is chosen so that we can color nodes with greater sparsity using palette sparsification, and those with less sparsity using information recovered from vectors with $O(\rho)$ non-zero values. By choosing $\rho = \Theta(\varepsilon^{-2}\log n)$, the algorithm uses $\widetilde{O}(n\rho) = \widetilde{O}(n)$ space while the probabilistic guarantees from slack generation and palette sparsification hold with high probability.

Assumption on Δ. We assume Δ is known in advance and that $\Delta \geqslant 2\alpha\rho^3$ so that $\exp(-\Omega(\Delta/\rho^2))$ and $\exp(-\Omega(\Delta/\rho))$ are at most $1/\text{poly}(n)$. When Δ is not known in advance, we run the algorithm $\lceil \log n \rceil$ times in parallel with Δ set to each power of two. After the streaming pass, we know Δ and we thus use the information from the streaming pass corresponding to the largest power of two smaller than Δ. When $\Delta \leqslant \rho^3$, we can afford to store all edges of the graph and use that, by [50], a $(\Delta - 1)$-coloring exists for $\Delta \geqslant \Delta_0$, for some large enough universal constant Δ_0.

4.1 Palette Sampling

To avoid storing all edges, our algorithm uses palette sparsification to color some of the vertices. Similarly to [3, 4], we partition the list of colors $\mathbf{L}(v)$ sampled by v into multiple independent random lists. This is done to simplify the analysis by removing dependencies between the successive coloring steps. The exact sampling process for each of the $\mathbf{L}_i(v)$ is therefore chosen to be convenient in the analysis of Section 5, where the index i corresponds to the step at which the coloring algorithm uses $\mathbf{L}_i(v)$. For each vertex $v \in V$, let

- $\mathbf{L}_2(v)$: sample one color in $[\Delta - 1]$ uniformly at random,
- $\mathbf{L}_3(v)$: each color of $[\Delta - 1]$ is sampled w.p. ρ/Δ,
- $\mathbf{L}_4(v)$: each color of $[\Delta - 1]$ is sampled independently w.p. ρ/Δ,
- $\mathbf{L}_5(v)$: each color of $[\Delta - 1]$ is sampled independently w.p. ρ^2/Δ,
- $\mathbf{L}_6(v)$: each color of $[\Delta - 1]$ is sampled independently w.p. ρ^3/Δ.

and $\mathbf{L}(v)$ is the union of those lists. The random lists of colors are sampled before the beginning of the streaming pass.

During the streaming pass, we store all the edges of the sparsified graph.

▶ **Definition 12.** *For a graph $G = (V, E)$ and random lists $\mathbf{L}$, the sparsified graph is the subgraph of G with vertex set V and all edges $\{u, v\} \in E$ such that $\mathbf{L}(u) \cap \mathbf{L}(v)$ is not empty.*

The following lemma is direct by applications of the Chernoff Bound. We refer readers to [3, Lemma 4.1] for details. In particular, it shows that w.h.p. the sparsified graph requires only $\widetilde{O}(n)$ space to store.

▶ **Lemma 13.** *W.h.p., each $\mathbf{L}(v)$ contains at most $O(\rho^3)$ colors and the sparsified graph contains at most $O(n\rho^6)$ edges.*

4.2 Structural Decomposition

First, partition almost-cliques based on their size.

▶ **Definition 14.** *We say the almost-clique C is* large *if $|C| \geqslant \Delta + 1$,* small *if $|C| \leqslant \Delta + 1 - \rho$ and* critical *otherwise.*

Our algorithm focuses almost entirely on coloring critical almost-cliques. The large will be easy to color because they always contain some anti-edges. This motivates the following definition.

▶ **Definition 15.** *The* core of C *is a $K \subseteq C$ such that $G[K]$ is a clique of maximal size. We say C is* solitary *iff the core of C has size at most $|C| - 2$.*

If the largest clique of $G[C]$ is not unique, we pick an arbitrary one as the core of C. When C is not solitary, the core is uniquely defined unless $G[C]$ contains exactly one anti-edge. We call core-critical the vertices in the core of a critical almost-clique. When we have an

almost-clique decomposition $V_{sparse}, C_1, C_2, \ldots, C_k$, we write K_i for the core of C_i for all $i \in [k]$. Since G does not contain a Δ-clique, a large almost-clique is solitary. What makes the solitary almost-cliques easy to color is the following fact.

▶ **Fact 16.** *C is solitary iff $G[C]$ contains a 2-anti-matching or a 3-independent set.*

Solitary almost-cliques are such that any coloring of $V - C$ can be extended to C (see Figures 1a and 1b). The second criteria that determines how we color almost-cliques relates to how they are connected to the vertices outside (see Figure 1c for an example of a popular clique and Figures 2b and 2c for examples of friendly cliques, where s_1, s_2 and s are friends).

▶ **Definition 17.** *Let C be a non-solitary almost-clique with core K, $v \in N(C) - K$ and $k \in \mathbb{R}_{>0}$. We say v is a k-friend of C if it has at least Δ/k neighbors in K and at least one non-neighbor in K.*

If C has two k-friend u and v with a shared neighbor in K and there exists $u' \neq v' \in K$ such that $\{u, u'\}$ and $\{v, v'\}$ are not edges, we say that C is k-popular.

If C has at least one k-friend but is not k-popular, we say that C is k-friendly.

4.3 Sparse Recovery

To color the densest regions of the graph, it is necessary to recover some of their structure. For a solitary C, the information contained in the solitary helper structure allows us to extend any coloring of $V - C$ to C.

▶ **Definition 18.** *Let C be a solitary almost-clique. A solitary helper structure for C is one of the two following types of tuple:*

- *$(u_1, v_1, u_2, v_2, N_G(u_1), N_G(u_2))$ such that $\{u_1v_1, u_2v_2\}$ is a 2-anti-matching of $G[C]$; or*
- *$(u_1, u_2, v, N_G(u_1), N_G(u_2))$ such that $\{u_1, u_2, v\}$ is a 3-independent set in $G[C]$.*

We do not define other types of helper structure, for we can implicitly recover all the edges incident to the remaining dense vertices. Formally, we recover the following:

▶ **Lemma 19.** *W.h.p., `sparse-recovery` uses $\widetilde{O}(n)$ memory and after the streaming pass, we recover*

- *the sets $A(v)$ and $E(v)$ for every vertex $v \notin V_{sparse}$ with $a(v) + e(v) \leqslant 4\rho$, and*
- *a solitary helper structure for every non-small solitary almost-clique.*

We say that "we know $N_G(v)$" if we can iterate over the neighbors $u \in N_G(v)$ in time $O(\deg(v))$ and $\text{poly}(\log n)$ space after the streaming pass. So, we know $N_G(u_1)$ and $N_G(u_2)$ where u_1 and u_2 are vertices from the solitary helper structure described in Definition 18. Lemma 19 implies that, w.h.p., we also know $N_G(v)$ for every dense vertex such that $a(v) + e(v) \leqslant 4\rho$. Indeed, to iterate over the neighbors of $v \in C$, it suffices to iterate over $E(v)$ and then $C - A(v)$, which is possible since C, $E(v)$ and $A(v)$ are stored in memory. We remark that we only store $N_G(v)$ implicitly, and hence the space usage remains $\widetilde{O}(n)$.

Since the sparse recovery is almost verbatim that of [4, Section 4.4], we skip details here and refer readers to [30, Appendix A]. Observe how using Lemma 19 allows us to recover all relevant information for non-small and popular almost-cliques.

▶ **Corollary 20.** *Let C with core K be critical and not solitary. From `sparse-recovery`, we recover K and $N(v) \cap K$ for all vertices $v \in V$.*

4.4 Summary of the Information Collected by the Algorithm

Before moving on with the analysis of our coloring algorithm, we recapitulate all the information gathered during the streaming pass. With high probability, at the end of the stream, we have the following information available:

- the random lists of colors $\mathbf{L}(v)$ as described in Section 4.1,
- all the edges of the sparsified graph defined in Definition 12,
- a vertex partition $V_{sparse}, C_1, \ldots, C_k$ as in Definition 5 (by Proposition 7),
- the sets $A(v)$ and $E(v)$ for all $v \in C$ with $a(v) + e(v) \leqslant 4\rho$ (by Lemma 19),
- a solitary-helper structure for every non-small solitary almost-clique (by Lemma 19),
- the core of every critical and not solitary almost-clique (by Corollary 20),
- the sets of critical k-popular and k-friendly almost-cliques for all $k > 0$.

It is an easy observation that the set of k-popular critical almost-cliques can be inferred, such cliques are not solitary and all the edges incident to their core are known.

It should also be clear that the algorithm uses $\widetilde{O}(n)$ space as the sparsified graph contains $O(n \log^6 n)$ edges (by Lemma 13), and sparse recovery uses $\widetilde{O}(n)$ space (by Lemma 19).

5 The Coloring Algorithm

In this section, we explain how to color the graph after the streaming pass. We have two means to ensure that we do not create monochromatic edges when we color vertices. Either they use a color from their random list $\mathbf{L}(v)$, and we can check for conflicts by looking at the edges of the sparsified graph (Definition 12). Or we color a vertex with a color that is not from its random list $\mathbf{L}(v)$, in which case we must be able to iterate over all its neighbors. To emphasize that aspect, we define the notion of serene coloring and maintain that our coloring is one.

▶ **Definition 21.** *We say that a (possibly partial) coloring φ is* serene *if for all vertices v with $\varphi(v) \neq \perp$, either $\varphi(v) \in \mathbf{L}(v)$ or we know $N_G(v)$ after the streaming pass.*

Detailed Overview. Let us explain the steps of our coloring algorithm.

- **Step 1: Reed Transform**. First, we remove all non-small solitary and non-small $\alpha\rho^2$-popular almost-cliques from G and obtain the graph G'. Then, we remove all 2ρ-friendly almost-cliques with a $(\Delta - 1)$-clique, possibly adding some extra edges to the graph. The resulting graph is called H and has the same almost-clique decomposition (apart from those that got removed), but contains no large, critical solitary, or critical ρ-popular almost-clique, and the almost-cliques with a core of size $\Delta - 1$ are not ρ-friendly.
- **Step 2: Slack Generation.** Every vertex in H gets activated with constant probability and samples a uniform color in $[\Delta - 1]$. If that color was sampled by no neighbor, we color the vertex with it.
- **Step 3: Coloring Critical Almost-Cliques of H.** Recall that all edges incident to the core of critical almost-cliques are known. Hence, we color those vertices serenely without using palette sparsification. We argue that after slack generation, all critical almost-cliques have two vertices whose coloring can be delayed, because (1) they have two pairs of same-colored neighbors, or (2) they have one pair of same-colored neighbor and an uncolored neighbor whose coloring can be deferred even further. Because the coloring of some vertices must be delayed, we actually color critical almost-cliques in two steps.

- **Step 4: Coloring Sparse Vertices & Small Almost-Cliques.** After Step 3, all critical almost-cliques have been colored. Observe that in Step 1, we removed all large almost-cliques from H, because they are solitary. So it remains to color the small almost-cliques and the sparse vertices. The sparse vertices are colored as in [3]. Each dense vertex from this set gets $\Omega(\log n)$ pairs of same-colored neighbors from slack generation. As such, they can be colored using palette sparsification.
- **Step 5: Inverse Reed Transform.** Steps 3 and 4 colored all vertices of H. From a proper coloring of H, we construct a coloring of G'. Step 5 focuses on inverting the Reed Transform, during which we removed 2ρ-friendly almost-cliques with a $(\Delta-1)$-core from the graph, and possibly added some edge. By same-coloring two pairs of vertices manually, we can extend the coloring to all such almost-cliques. Note that, from then on, we may recolor vertices. In particular, the pairs of same-colored neighbors produced by slack generation are no longer guaranteed to exist.
- **Step 6: Coloring Non-Small Popular & Non-Small Solitary.** We complete the coloring by handling the solitary and friendly almost-cliques removed during the phase 1 of the Reed Transform. Structurally, those almost-cliques have pairs of vertices that, if same-colored, allow us to extend the coloring greedily (recall Figure 1). Thanks to sparse recovery, we can identify those pairs and same-color them with colors from their random lists. Then, they can be colored using either a greedy algorithm or palette sparsification.

Table 1 For each type of almost-clique, the step of the algorithm at which it is colored. We stress that popular almost-cliques are not solitary and that the "Friendly with a $(\Delta-1)$ Core" are neither solitary nor popular.

	Solitary	**Popular**	**Friendly with a $(\Delta-1)$ Core**	**Others**
Small	Step 4			Step 4
Large	Step 6			
Critical	Step 6	Step 6	Step 5	Step 3

5.1 Step 1: Preprocessing and the Reed Transform

This first step removes from the graph all vertices that cannot be colored with slack generation. To ensure that the almost-cliques removed from the graph can be colored later, we may add some edges. The resulting graph is called H.

Concretely, we construct the graph H in two phases. First, a preprocessing phase computes the graph G' by removing from G all almost-cliques that are either large, critical solitary, or critical $\alpha\rho^2$-popular. This step is obvious from the information gathered during the streaming pass (Section 4.4) and needs not further discussion. Then, the Reed Transform computes the graph H by removing from G' all 2ρ-friendly almost-cliques that contain a $(\Delta-1)$-clique and adding selected edges. We choose the edges to add to H in Algorithm 1.

▶ Remark 22. Some important observations about Algorithm 1 before we analyze it:

- the random set A_{RT} is introduced in Algorithm 1 to break-symmetry and simplify the inversion of the Reed Transform; it ensures that an x- or y-vertex is not selected as a u- or v-vertex for some other almost-clique and vice versa;

Algorithm 1 Reed Transform.

Input: a graph G' with no large, critical solitary, or critical $\alpha\rho^2$-popular almost-cliques.
Output: a graph H such that: a) almost-cliques with $|K_i| = \Delta - 1$ are not ρ-friendly and b) solitary and $\alpha\rho^2$-popular almost-cliques are small.

For each 2ρ-friendly almost-cliques C_i with a $(\Delta-1)$-core (i.e., $|K_i| = \Delta - 1$), do:

- **i)** Let s_i be a (2ρ)-friend of C_i; if $C_i \neq K_i$, take s_i inside C_i.
- **ii)** Let $D_i = N_{G'}(s_i) \cap C_i$ be the set of neighbors of s_i in C_i.
- **iii)** For each $u \in D_i$, let $f(u)$ denote the external neighbor of u that is not s_i, if any.
- **iv)** Sample each vertex into the set A_{RT} independently w.p. $p_{RT} = 1/10$.
  Let $S_i = \{u \in D_i \cap A_{RT} : f(u) \notin A_{RT}\}$.
- **v)** If all nodes of D_i are of degree Δ *and* $f(S_i)$ is an independent set then
  - **a.** Sample each vertex of S_i into a set T_i w.p. $p_{ds} = 1/\rho$.
  - **b.** Let $u_i \neq v_i$ be any pair of nodes in T_i such that $x_i = f(u_i)$ and $y_i = f(v_i)$ are distinct, and not members of the same critical almost-clique.
  - **c.** Add the pair $\{x_i, y_i\}$ to the set E_{new} and add i into the set I.

Return the graph H where $V(H) = V(G') - \bigcup_{i\in I} C_i$ and $E(H) = G'[V(H)] \cup E_{new}$.

- the algorithm can be implemented after the streaming pass because the decomposition is known (Proposition 7) and the core as well as all edges incident to core vertices of non-small almost-cliques are known (Corollary 20).

▶ **Lemma 23.** *Assume that $\Delta \geqslant \alpha\rho^3$ and $\varepsilon \geqslant \max\{8/\rho, 4\sqrt{2}/\sqrt{\eta\rho}\}$. Let G' be a graph with an $(\eta, \varepsilon, \varepsilon)$-almost-clique decomposition $V_{sparse}, C_1, C_2, \ldots, C_k$ that has no large, critical solitary, or critical $\alpha\rho^2$-popular almost-cliques. W.h.p., the graph H produced by Algorithm 1 with the same vertex-partition restricted to $V(H)$ is such that:*

(RT.1) *it is an $(\eta/2, \varepsilon, \varepsilon/2)$-almost-clique decomposition;*
(RT.2) *all solitary or ρ-popular almost-cliques in H are small;*
(RT.3) *no almost-clique in H with a $(\Delta-1)$-core is ρ-friendly; and*
(RT.4) *H has maximum degree Δ and does not contain a Δ-clique.*

Proof. We first verify that each $u \in D_i$ of degree Δ has exactly one external neighbor $f(u)$ different from s_i. Indeed, u is in the $(\Delta-1)$-clique K_i, thus has $\Delta - 2$ core neighbors. Since u is of degree Δ, it has one additional neighbor besides s_i, which must be external because s_i is the only vertex of $C_i - K_i$ (when $C_i \neq K_i$).

▶ **Fact 24.** *Every $u \in D_i$ with degree Δ has exactly one external neighbor $f(u)$ other than s_i.*

We need that a constant fraction of $u \in D_i$ belong to S_i, i.e., are in A_{RT} while their external neighbor $f(u)$ is not. We show that this is a high probability event.

▷ Claim 25. W.h.p. over the randomness of A_{RT}, we have $|S_i| \geqslant |D_i|/24$.

Proof of claim. For $u \in D_i$, let X_u be the indicator random variable of the event that $u \in A_{RT}$ and $f(u) \notin A_{RT}$. We have $\mathbf{Exp}[X_u] = p_{RT}(1 - p_{RT})$ and by the linearity of expectation their sum has expectation $\mu = p_{RT}(1-p_{RT})|D_i| \geqslant |D_i|/12$. The activation of u affects only X_u and the activation of a vertex in $f(D_i)$ affects at most $\Delta/\alpha\rho^2$ other vertices in D_i, for C_i would otherwise be $\alpha\rho^2$-popular. So we may use the read-k bound (Proposition 11) with $k = \Delta/(\alpha\rho^2)$ and $\delta = 1/24$ to show concentration as

$$\mathbf{Pr}\left[\left|\sum_{u\in D_i} X_u - \mu\right| \geqslant |D_i|/24\right] \leqslant \exp\left(-\Omega\left(\frac{|D_i|}{k}\right)\right) \leqslant \exp(-\Omega(\rho)) \leqslant 1/\mathrm{poly}(n)\ ,$$

where we use that $|D_i| \geqslant \Delta/(2\rho)$. ◁

Henceforth, fix A_{RT} such that Claim 25 holds and let $S = \bigcup_{i \in I} S_i$.

In Step va, vertices are downsampled to ensure that the algorithm does not add an excessive number of edges to any single vertex.

▷ Claim 26. After Step va, over the randomness $T = \bigcup_i T_i$, we have that

- **i)** For each C_i, we have that $|T_i| \geqslant \Delta/(50\rho^2)$ w.p. $\exp(-\Omega(\Delta/\rho^2))$.
- **ii)** For each node $u \in V(G')$, we have that $|f^{-1}(u) \cap T| \leqslant 2\Delta/\rho$ w.p. $\exp(-\Omega(\Delta/\rho))$.
- **iii)** For each C_i, we have that $|f(T) \cap K_i| \leqslant 4\Delta/\rho$ w.p. $\exp(-\Omega(\Delta/\rho))$.

Proof of claim. Consider an almost-clique C_i. Observe that $|T_i|$ is the sum of independent binary variables over the nodes in S_i, each with probability p_{ds}. By linearity of expectation, $\mathbf{Exp}[|T_i|] = |S_i|p_{ds} \geqslant |D_i|/24 \cdot 1/\rho \geqslant \Delta/(48\rho^2)$, using Claim 25 and that $|D_i| \geqslant \Delta/(2\rho)$. By Chernoff, $|T_i| \geqslant \Delta/(50\rho^2)$, w.p. $\exp(-\Omega(\Delta/\rho^2))$, establishing i).

Consider an arbitrary vertex u of the graph. Each neighbor $v \in f^{-1}(u) \cap S$, independently joins T w.p. $p_{ds} = 1/\rho$. Hence, by linearity of expectation, we have $\mathbf{Exp}[|f^{-1}(u) \cap T|] \leqslant \Delta/\rho$. By Chernoff, we get that $|f^{-1}(u) \cap T|$ is at most $2\Delta/\rho$ w.p. $\exp(-\Omega(\Delta/\rho))$, establishing ii).

Each vertex of K_i has at most two external neighbors, so $|E(K_i)| \leqslant 2(\Delta+1)$. Each vertex of $E(K_i)$ joins T independently with probability at most $1/\rho$ so $\mathbf{Exp}[|E(K_i) \cap T|] \leqslant 2(\Delta+1)/\rho$. By Chernoff, at most $4\Delta/\rho$ vertices of $E(K_i)$ join T w.p. $\exp(-\Omega(\Delta/\rho))$. It implies iii) because each external neighbor in T maps to at most one vertex of K_i through f, i.e., $|f(T) \cap K_i| \leqslant |E(K_i) \cap T|$. ◁

We deduce that the algorithm is well-defined.

▷ Claim 27. W.h.p., for each C_i, there is a pair u_i, v_i as required in Step vb of Algorithm 1.

Proof of claim. Let u_i be any node in T_i. If $x_i = f(u_i)$ is in a critical almost-clique C_j, then $f(S_i)$ includes at most one other vertex $z \in C_j \cap f(S_i)$ because $f(S_i)$ is an independent set and C_j is not small, hence not solitary. Both x_i and z (if it exists) have at most $\Delta/\alpha\rho^2$ neighbors in D_i, for otherwise C_i would be $\alpha\rho^2$-popular. From Claim 26-i), we have that $|T_i| \geqslant \Delta/(50\rho^2)$, w.h.p. Thus, if we exclude from consideration the nodes in T_i that map by f to either x_i or z, we are left with $|T_i| - 2\Delta/(\alpha\rho^2) \geqslant \Delta/(\alpha\rho^2) \geqslant \rho$ suitable candidates (using that $\alpha \geqslant 150$). Any of these can be chosen as v_i, thereby establishing the claim. ◁

It is now easy to verify that RT.1 to RT.3 of Lemma 23 hold. It is immediate from Claim 26-ii) that at most $2\Delta/\rho$ edges are added into H that are incident on any given vertex. So, a vertex $v \in V_{sparse}$ gains at most $2\Delta/\rho$ new neighbors and each of its previous neighbors gains at most $2\Delta/\rho$ incident edges (by Claim 26-ii). So the sparsity of v decreases by at most $4\Delta/\rho$, staying above $\eta\varepsilon^2\Delta - 4\Delta/\rho \geqslant (\eta/2)\varepsilon^2\Delta$. Similarly, we add few enough edges to ensure that Part 3 of Definition 5 continues to hold, up to a constant factor.

Using the properties of the almost-clique decomposition RT.1, we deduce that H contains no Δ-clique.

▷ Claim 28. The graph H has maximum degree Δ and contains no Δ-clique.

Proof of claim. The maximum degree remains at most Δ, for each edge added incident to v, another edge incident to v is removed. So we focus on proving that H contains no Δ-clique.

A Δ-clique in H must be within an almost-clique, since its nodes must be dense, and it can't involve nodes from different almost-cliques (by Definition 4 (2)). But the almost-clique decomposition did not change RT.1 and no edges were added inside critical almost-cliques (by Step vb), so the Δ-clique would have to have been contained in G', a contradiction. ◁

This concludes the proof of Lemma 23. ◀

5.2 Step 2: Slack Generation

Formally, when we color some graph, the slack of a vertex is the difference between the number of colors it has available and its number of (uncolored) neighbors. For $(\Delta - 1)$-coloring, vertices start with -1 slack and each time we same-color a pair of neighbors, we increase the slack by one. We use a well-known technique for generating pairs of same-colored neighbors: let vertices pick random colors. The algorithm is as follows; recall that we run it on the graph H obtained from Step 1.

Algorithm 2 `slack-generation`.

Input: a graph H
Output: a proper partial $(\Delta - 1)$-coloring

i) vertices join A_{SG} w.p. $p_{SG} = 1/10$.
ii) vertices[2]of A_{SG} sample $\chi(v)$ uniformly at random in $[\Delta - 1]$.
iii) if $\chi(v) \notin \chi(N(v))$, then set $\varphi(v)$ to $\chi(v)$ and otherwise let v uncolored, i.e., $\varphi(v) = \perp$.

To implement Algorithm 2 after the streaming pass, each vertex uses the one color from $\mathbf{L}_2$ as the random color $\chi(v)$. It therefore suffices to look at the edges of the sparsified graph to know if a vertex retains its color.

The activation probability ensures that not too many vertices get colored by slack generation.

▶ **Fact 29.** *After Algorithm 2, w.h.p., every C_i contains at most $\Delta/5$ colored vertices.*

We use three flavors of slack generations. First, we have the well-known result (see, e.g., [47, 25, 33]) that ζ-sparse vertices (cf. Definition 3) receive $\Omega(\zeta)$ slack w.e.h.p. in ζ. This explains why sparse vertices and small almost-cliques can be colored easily.

▶ **Proposition 30** ([33, Lemma 6.1])**.** *There exists a universal constant $\gamma \in (0, 1)$ for which the following holds. Suppose H is a graph of maximal degree $\Delta \gg 1$. If φ is the coloring produced by Algorithm 2, then a ζ-sparse vertex v with $\zeta \gg 1$ has that*

$$|L_\varphi(v)| \geqslant \deg_\varphi(v, H) + \gamma \cdot \zeta$$

with probability at least $1 - \exp(-\Omega(\zeta))$.

In very dense critical almost-cliques, the sparsity of individual vertices is too small for Proposition 30 to provide vertices with extra colors (by same-coloring neighbors) with high enough probability. However, when the almost-clique has many different external neighbors, a very similar argument shows that many vertices of the almost-clique get one external neighbor colored the same as a vertex inside the almost-clique. For simplicity, we state the result for a clique K, rather than an almost-clique.

▶ **Proposition 31** (Rephrasing Lemma 5.4 in [26])**.** *Let K be a clique and M a matching between K and $N(K) - K$. After Algorithm 2, w.e.h.p. in $|M|$, at least $\Omega(|M|)$ vertices in $V(M) \cap K$ have two non-adjacent neighbors colored the same, i.e., one unit of slack.*

[2] For consistency, set $\chi(v) = \perp$ for every $v \notin A_{SG}$.

We conclude this anthology of results about slack generation with one from Reed's original paper on $(\Delta-1)$-coloring. It serves the same purpose as Proposition 31 except that it counts vertices that receive two units of slack. Consequently, instead of asking for a matching between C and $N(C)-C$, we require a one-to-two matching. Formally, we define a triple for C as a tuple (u,v,w) such that $v \in C$, $u \neq w \in E(v)$. We say that a set of triples is disjoint if no vertex belongs to more than one.

▶ **Proposition 32** ([50, Section 4] or [47, Chapter 11.3]). *For every $\varepsilon, \delta \in (0, 1/10)$, there is a universal constant $\gamma = \gamma(\varepsilon, \delta, p_{SG})$ for which the following holds. Let C be an (ε,δ)-almost-clique and $\{(u_i, v_i, w_i)\}_{i=1}^{t}$ be a set of disjoint triples for C. Let Z be the random variable counting number of triples for which v_i is uncolored, and u_i and w_i are colored the same as some vertex in C by slack generation. Then,*

$$\mathbf{Pr}[Z \leqslant \gamma \cdot t] \leqslant \exp\left(-\Theta\left(\frac{t^2}{\Delta}\right)\right)$$

Note that the concentration from Proposition 32 is effective only when t is asymptotically larger than $\sqrt{\Delta}$. This is a consequence of the fact that, contrary to Propositions 30 and 31, counting colors does not suffice to count the number of vertices with two-units of slack.

5.3 Step 3: Coloring Critical Cliques in H

In this step, we prove that all critical almost-cliques of H can be colored after slack generation. Since the formal argument is involved, let us provide first a bird's-eye view of the proof. It is in three parts:

- Firstly, we argue that if a subgraph of H induced by a set of critical almost-cliques is such that each almost-clique has two adjacent vertices with respectively one and two units of slack, we can serenely color that subgraph greedily.
- Then we analyze how the critical vertices of H get slack from slack generation (Algorithm 2). The analysis is three-fold: when many low-degree vertices exist, or when many vertices have external neighbors with few edges to the almost-clique, or finally when most of the vertices have degree Δ and external neighbors with many edges to the almost-clique. In the first two cases, we show that w.h.p. at least two vertices get slack. For the last case, we argue that w.h.p. the almost-clique has many vertices with one pair of same-colored neighbors and one uncolored external neighbor with 2ρ edges to the almost-clique.
- Finally, we color the critical vertices of H in two stages. That is to ensure that, in the last type of critical almost-cliques, the uncolored external neighbors with 2ρ neighbors in the almost-clique C remain inactive while we color C, thereby providing one more unit of slack. The trick is that such external neighbors, if dense, must have 2ρ external neighbors. However, critical almost-cliques of H, since they are not solitary, contain at most one such vertex.

In the end, we get the following result. For the proof, we refer the readers to the full version [30, Section 5.3].

▶ **Lemma 33.** *Let H be the graph obtained from Step 1 and φ be the coloring produced by Algorithm 2 in Step 2. With high probability over the randomness of $\mathbf{L}_2(V)$ and $\mathbf{L}_3(V)$, there exists a serene extension of φ to all critical almost-cliques of H.*

5.4 Step 4: Coloring Sparse Vertices & Small Almost-Cliques

We color the non-critical vertices of H in the following order: first, the sparse vertices; second, the small almost-cliques with fewer than $10^7\varepsilon\Delta$ anti-edges; last, the small almost-cliques with at least $10^7\varepsilon\Delta$ anti-edges. The treatment of sparse vertices and almost-cliques with many missing edges is the same as in [3]. For the small almost-clique with few missing edges, we observe that their nodes must be $\Omega(\rho)$-sparse, and hence they can be colored by palette sparsification. We refer readers to [30, Section 5.4] for more details.

5.5 Step 5: Inverting the Reed Transform

In this step, we take the coloring of H computed thus far and inverse the Reed Transform from Algorithm 1 to deduce a coloring of G', i.e., of the all vertices except those in non-small solitary or non-small $\alpha\rho^2$-popular almost-cliques. We emphasize that we may change colors given by slack generation (Algorithm 2), thus guarantees of Propositions 30–32 do not apply henceforth.

The algorithm same-color two pairs of vertices in each almost-clique removed by the Reed Transform. This guarantees that the remaining vertices can be serenely colored (recall that we know the neighborhood of core-critical vertices). We argue that the pairs can be same-colored greedily, in fact they always have $\Omega(\Delta/\rho)$ colors available. When certain pairs of vertices to same-color overlap, then we same-color entire independent sets at once.

▶ **Lemma 34.** *Let H be the graph from Algorithm 1. Suppose φ is a total proper serene coloring of H. With high probability, we compute a serene coloring of G', i.e., of the graph G where all non-small solitary and non-small $\alpha\rho^2$-popular almost-cliques are removed.*

Proof. The goal is to same-color all pairs of vertices x_iv_i and s_iz_i, where z_i is an anti-neighbor of s_i in K_i. Then the coloring can be serenely extended to all 2ρ-friendly critical almost-cliques with a $(\Delta-1)$-clique, by [30, Claim 5.14]. If a vertex in D_i does not have degree Δ, we do not have any vertices x_i and y_i; same-coloring the s_iz_i pair suffices. If $f(S_i)$ is not an independent set, we pick any pair $u_i \neq v_i \in S_i$ such that $x_i = f(u_i)$ and $y_i = f(v_i)$ are adjacent in G'.

Denote by $\mathcal{S}$, $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{U}$ and $\mathcal{V}$ the set vertices respectively containing all s_i, x_i, y_i, u_i and v_i. Recall that Algorithm 1 chooses $\mathcal{X}$ and $\mathcal{V}$ disjoint (because $\mathcal{V} \subseteq A_{RT}$ while $\mathcal{X} \cap A_{RT} = \emptyset$). For w, call $S(w)$ the set of $i \in I$ for which $w = s_i$. First, let us explain how we select the z_i vertices. The proof can be found in the full version of this paper, see [30, Section 5.5].

▷ Claim 35. If $|S(w)| \geqslant 1$, then there exist vertices $z_i \in K_i - N(w)$ for all $i \in S(w)$ and such that

$$\mathcal{I}(w) := \{w\} \cup \{z_i : w = s_i\} \cup \{v_i : w = x_i \text{ or } z_j = x_i \text{ for some } j \in S(w)\}$$

is an independent set.

Call z_i the vertex given by Claim 35 in C_i and call $\mathcal{Z}$ the set of all z_i vertices. Note that $\mathcal{X}$ can overlap with $\mathcal{S}$ and $\mathcal{Z}$. The nodes in $\mathcal{Z}, \mathcal{V}$ are core-critical but not those in $\mathcal{S}$. We same-color the pairs in the following order:

1. uncolor all vertices of $\mathcal{S}$,
2. color all v_i with $\varphi(x_i)$ where x_i is colored,
3. color all pairs s_iz_i and x_iv_i where $x_i \in \mathcal{S} \cup \mathcal{Z}$, and finally
4. color all remaining pairs x_iv_i, i.e., where $x_i \notin \mathcal{S} \cup \mathcal{Z}$ and $x_i \in C_j$ with $j \in I$.

Step (2) cannot create a color conflict because v_i has at most one colored external neighbor y_i (after uncoloring S) and H (from the Reed Transform) contained an edge between x_i and y_i.

Let us explain how we perform Step (3). We form a virtual graph Q_1 by contracting sets of vertices in G. More precisely, for a set of *disjoint* sets of vertices $\mathscr{I} \subseteq 2^{V(G)}$, denote by $G/\mathscr{I}$ the graph with vertex set $\mathscr{I}$ and an edge between two vertices if the corresponding sets contain adjacent vertices in G. Let $\mathscr{I}_1$ be the sets of $\mathcal{I}(w)$ for all $w \in S$ and $Q_1 = G/\mathscr{I}_1$. For a vertex $\mathcal{I}$ of Q_1 (i.e., $\mathcal{I} \in \mathscr{I}_1$), let $L(\mathcal{I})$ be the intersection of the lists of available colors in G overall the vertices in $\mathcal{I}$, i.e., $L(\mathcal{I}) = \bigcap_{w \in \mathcal{I}} L_\varphi(w)$.

Since each $\mathcal{I}(w)$ is an independent set, a coloring of Q_1 using lists $L(I)$ induces a proper extension of the coloring to G where vertices from the same $\mathcal{I}(w) \in \mathscr{I}_1$ are colored the same. Since each s_i has at least Δ/ρ uncolored neighbors, while each element $\mathcal{I}(s_i)$ other than s_i gives rises to $O(1)$ edges, we do obtain a $(\deg + \Omega(\Delta/\rho))$-list-coloring instance. The proof can be found in the full version of this paper, see [30, Section 5.5].

▷ Claim 36. For all $\mathcal{I} \in \mathscr{I}_1$, we have that $|L(\mathcal{I})| \geqslant \deg(\mathcal{I}, Q_1) + \Delta/4\rho$.

To implement this in streaming, recall that we know all edges incident to core-critical vertices, which include $\mathcal{Z}$ and $\mathcal{V}$. In particular, we can compute the sets $\mathcal{I}(w)$ described in Claim 35, which thereby implies that we know Q_1. We color vertices of Q_1 greedily. When it comes to $\mathcal{I} = \mathcal{I}(w)$ with $w \in S$, we claim that we can color $\mathcal{I}$ with some color from $\mathbf{L}_5(w)$ with high probability. Indeed, by Claim 36, for any partial coloring of Q_1, the vertex $\mathcal{I}$ still has $\Omega(\Delta/\rho)$ available colors, thus

$$\mathbf{Pr}[\mathbf{L}_5(w) \cap L(\mathcal{I}) = \emptyset] \leqslant \left(1 - \frac{\rho^2}{\Delta}\right)^{\frac{\Delta}{4\rho}} \leqslant \exp(-\Omega(\rho)) \leqslant 1/\mathrm{poly}(n) \ .$$

In particular, when we color vertices of $\mathcal{I}$ with that color, the coloring remains serene because (1) w is the only non-core-critical vertex of this set, and it gets a color from its random list, and (2) we know the neighborhood of all other vertices from sparse recovery.

The pairs that remain to be same-colored are of the form $x_i v_i$ such that $x_i \notin \mathcal{S} \cup \mathcal{Z}$ and $x_i \in C_j$ for some $j \in I$. As such, it has at least $\Delta - O(\rho)$ available colors and at most two external neighbors. It can be that $x_i = x_j = x$, in which case we must same color the three vertices x, v_i and v_j. Note that v_i and v_j cannot be adjacent as $\mathcal{V}$ is an independent set. Since the uncolored vertices in $\mathcal{X}$ and $\mathcal{V}$ belong to $(\Delta - 1)$-cliques of which at most two vertices are colored, the vertices to same-color share at least $\Delta - O(1)$ available colors.

Claim 26-*iii*) implies that $|\mathcal{X} \cap C_i| \leqslant 4\Delta/\rho$ for all $i \in I$ because $\mathcal{X} \subseteq f(T)$. Hence, the degree of virtual vertices is at most $12\Delta/\rho$, and we can use $(\deg + 1)$-list-coloring. Recall that all the x_i that remains are core-critical, hence we know their neighborhood and can serenely color them with any color. ◀

5.6 Step 6: Post-processing

Finally, we assume that φ is a serene coloring of G', i.e., of all the vertices except for the non-small solitary and non-small $\alpha\rho^2$-popular almost-cliques. We iterate over such almost-cliques and apply either [30, Lemma 5.22] or [30, Lemma 5.23]. The full statement and proofs can be found in the full version of this paper, see [30, Section 5.6]. They start by same coloring some pairs of vertices (recall Figure 1) using colors from their random lists, w.h.p., and serenely extend the coloring to the rest of the almost-clique.

6 Lower bound for $(\Delta-k)$-Coloring in Semi-Streaming

Recall that by a result of Molloy and Reed [48], graphs of maximum degree Δ (larger than some sufficiently large constant) that do not have certain forbidden local subgraphs can be $(\Delta-k)$-colored for all k such that $(k+1)(k+2) \leqslant \Delta$. Very recently, [29] proved that such a $(\Delta-k)$-coloring can be computed in $\text{poly}(\log\log n)$ rounds of LOCAL for all such k but the largest one (for which the problem requires $\Omega(n/\Delta)$ rounds).

In this section, we show that such a result cannot be obtained in semi-streaming.

▶ **Theorem 2.** *Let $\Delta, c > 0$ be non-negative integers such that $(\Delta+1)/2 < c \leqslant \Delta$. Any (possibly randomized) one-pass streaming algorithm that outputs a c-coloring of given c-colorable n-vertex graph of maximum degree Δ requires $\Omega(n(\Delta-c+1))$ space.*

The proof is via a reduction to the one-way communication complexity of the INDEX_m problem. In this problem, Alice is given a bit-string $x = (x_1, x_2, \ldots, x_m)$ and Bob is given an index $i \in [m]$. Alice sends one message to Bob and so that Bob correctly outputs x_i with probability at least $2/3$. It is well-known that Alice needs to send $\Omega(m)$ bits to Bob (see [42, Theorem 3.7] or [49, Chapter 6] for a more recent treatment).

We construct a gadget with $\Theta(\Delta)$ vertices such that Bob can infer the value of x_i from a proper c-coloring of that graph. The graph induced by C is a $(c-3)$-clique, and the graph induced by $A \cup B$ will be the complement of that induced by $\overline{A} \cup \overline{B}$. We select m pairs $e_1, e_2, \ldots, e_m$ between A and B, but Alice includes the edge e_j iff $x_j = 1$ (and the corresponding edge between $\overline{A}$ and $\overline{B}$ is added iff $x_j = 0$). Then Bob connects the endpoints of e_i in $A \cup B$, and he connects all the vertices of C and the endpoints of e_i to the endpoints of e_i in $\overline{A} \cup \overline{B}$. See Figure 3. The key observation is that the graph induced by C plus the endpoints of e_i in $A \cup B$ and $\overline{A} \cup \overline{B}$ form a $(c+1)$-clique minus one of the edge in $A \cup B$ or $\overline{A} \cup \overline{B}$, depending on whether $x_i = 0$ or $x_i = 1$. Since any c-coloring of this graph must same-color the endpoints of this missing edge, Bob recovers the value of x_i from a proper c-coloring of this gadget. It follows that c-coloring this gadget requires $\Omega(\Delta(\Delta-c+1))$ bits. To obtain Theorem 2, we create $g = \frac{m}{\Delta(\Delta-c+1)}$ parallel gadgets, each encoding a different contiguous block of $\Delta(\Delta-c+1)$ bits from x. A formal proof of Theorem 2 can be found in [30, Section 6].

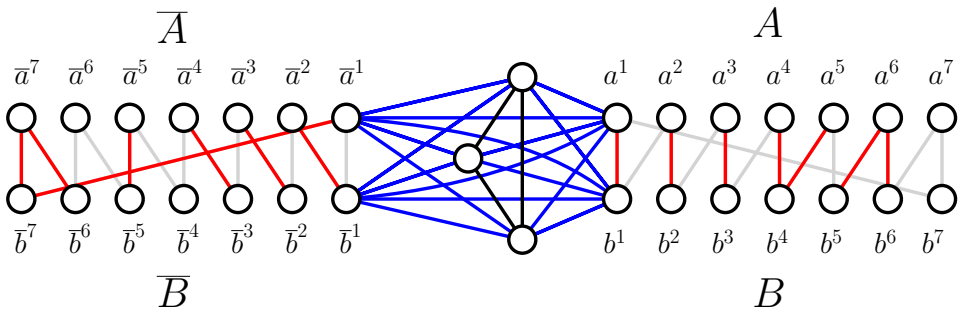

Figure 3 One gadget from the lower bound with $\Delta = 7$ and $c = 6$. In black are the edges of C, known to both players, and in gray the pairs used to encode the bits of x. The edges known/added by Alice are represented in red, and the edges known/added by Bob are represented in blue. Here $x = 10101011011000$ as read on the right with the edges between A and B, and $i = 1$ since the edges of Bob are incident to the first gray pair.

References

1 Noga Alon and Sepehr Assadi. Palette sparsification beyond (Δ+1) vertex coloring. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 6:1–6:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.6.

2 Sepehr Assadi, Andrew Chen, and Glenn Sun. Deterministic graph coloring in the streaming model. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 261–274. ACM, 2022. doi:10.1145/3519935.3520016.

3 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for (Δ + 1) vertex coloring. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 767–786. SIAM, 2019. doi:10.1137/1.9781611975482.48.

4 Sepehr Assadi, Pankaj Kumar, and Parth Mittal. Brooks' theorem in graph streams: A single-pass semi-streaming algorithm for Δ-coloring. *TheoretiCS*, 2, 2023. doi:10.46298/theoretics.23.9.

5 Sepehr Assadi, Janani Sundaresan, and Helia Yazdanyar. Coloring graphs with few colors in the streaming model. In Kasper Green Larsen and Barna Saha, editors, *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026*, pages 6065–6132. SIAM, 2026. doi:10.1137/1.9781611978971.217.

6 Sepehr Assadi and Chen Wang. Sublinear time and space algorithms for correlation clustering via sparse-dense decompositions. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.10.

7 Sepehr Assadi and Helia Yazdanyar. Simple sublinear algorithms for (Δ + 1) vertex coloring via asymmetric palette sparsification. In Ioana Oriana Bercea and Rasmus Pagh, editors, *2025 Symposium on Simplicity in Algorithms, SOSA 2025, New Orleans, LA, USA, January 13-15, 2025*, pages 1–8. SIAM, 2025. doi:10.1137/1.9781611978315.1.

8 Bradley Baetz and David R Wood. Brooks' vertex-colouring theorem in linear time, 2014. doi:10.48550/arXiv.1401.8023.

9 Étienne Bamas and Louis Esperet. Distributed coloring of graphs with an optimal number of colors. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 10:1–10:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.STACS.2019.10.

10 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *J. ACM*, 63(3):20:1–20:45, 2016. doi:10.1145/2903137.

11 Soheil Behnezhad, Rajmohan Rajaraman, and Omer Wasim. Fully dynamic (Δ + 1)-coloring against adaptive adversaries. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 4983–5026. SIAM, 2025. doi:10.1137/1.9781611978322.169.

12 Anup Bhattacharya, Arijit Bishnu, Gopinath Mishra, and Anannya Upasana. Even the easiest(?) graph coloring problem is not easy in streaming! In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 15:1–15:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.15.

13 Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. Dynamic algorithms for graph coloring. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1–20. SIAM, 2018. doi:10.1137/1.9781611975031.1.

14 Sayan Bhattacharya, Fabrizio Grandoni, Janardhan Kulkarni, Quanquan C. Liu, and Shay Solomon. Fully dynamic $(\Delta + 1)$-coloring in $O(1)$ update time. *ACM Trans. Algorithms*, 18(2):10:1–10:25, 2022. doi:10.1145/3494539.

15 Oleg V Borodin and Alexandr V Kostochka. On an upper bound of a graph's chromatic number, depending on the graph's degree and density. *Journal of Combinatorial Theory, Series B*, 23(2-3):247–250, 1977. doi:10.1016/0095-8956(77)90037-5.

16 Rowland Leonard Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2):194–197, 1941. doi:10.1017/S030500410002168X.

17 Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Adversarially robust coloring for graph streams. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 37:1–37:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.37.

18 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The complexity of $(\Delta+1)$ coloring in congested clique, massively parallel computation, and centralized local computation. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 471–480. ACM, 2019. doi:10.1145/3293611.3331607.

19 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. Distributed $(\Delta+1)$-coloring via ultrafast graph shattering. *SIAM J. Comput.*, 49(3):497–539, 2020. doi:10.1137/19M1249527.

20 Yi-Jun Chang, Gopinath Mishra, Hung Thuan Nguyen, and Farrel D. Salim. Round and communication efficient graph coloring. In Alkida Balliu and Fabian Kuhn, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025*, pages 360–371. ACM, 2025. doi:10.1145/3732772.3733508.

21 Daniel W. Cranston and Landon Rabern. Coloring claw-free graphs with delta-1 colors. *SIAM J. Discret. Math.*, 27(1):534–549, 2013. doi:10.1137/12088015X.

22 Artur Czumaj, Peter Davies, and Merav Parter. Simple, deterministic, constant-round coloring in congested clique and MPC. *SIAM J. Comput.*, 50(5):1603–1626, 2021. doi:10.1137/20M1366502.

23 Abhishek Dhawan. Palette sparsification for graphs with sparse neighborhoods. *arXiv preprint arXiv:2408.08256*, 2024. doi:10.48550/arXiv.2408.08256.

24 Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation - Recent Developments in Discrete Optimization*, Natural Computing Series, pages 1–87. Springer, 2020. doi:10.1007/978-3-030-29414-4_1.

25 Michael Elkin, Seth Pettie, and Hsin-Hao Su. $(2\Delta - 1)$-edge-coloring is much easier than maximal matching in the distributed setting. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 355–370. SIAM, 2015. doi:10.1137/1.9781611973730.26.

26 Manuela Fischer, Magnús M. Halldórsson, and Yannic Maus. Fast distributed brooks' theorem. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2567–2588. SIAM, 2023. doi:10.1137/1.9781611977554.ch98.

27 Maxime Flin, Mohsen Ghaffari, Magnús M. Halldórsson, Fabian Kuhn, and Alexandre Nolin. A distributed palette sparsification theorem. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*. SIAM, 2024. doi:10.1137/1.9781611977912.142.

28 Maxime Flin and Magnús M. Halldórsson. Faster dynamic $(\Delta+1)$-coloring against adaptive adversaries. In *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*, volume 334 of *LIPIcs*, pages 79:1–79:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.79.

29 Maxime Flin, Magnús M. Halldórsson, Manuel Jakob, and Yannic Maus. Sublogarithmic distributed vertex coloring with optimal number of colors, 2026. To appear at STOC 2026. doi:10.48550/arXiv.2603.28637.

30 Maxime Flin and Magnús M. Halldórsson. Beyond brooks: (Δ-1)-coloring in semi-streaming, 2026. arXiv:2605.07774.

31 Maxime Flin and Parth Mittal. (Δ + 1) vertex coloring in o(n) communication. *Distributed Comput.*, 38(1):19–29, 2025. doi:10.1007/s00446-024-00475-3.

32 Dmitry Gavinsky, Shachar Lovett, Michael E. Saks, and Srikanth Srinivasan. A tail bound for read-k families of functions. *Random Struct. Algorithms*, 47(1):99–108, 2015. doi:10.1002/rsa.20532.

33 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1180–1193. ACM, 2021. doi:10.1145/3406325.3451089.

34 Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. Near-optimal distributed degree+1 coloring. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 450–463. ACM, 2022. doi:10.1145/3519935.3520023.

35 Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. Ultrafast distributed coloring of high degree graphs, 2021. doi:10.48550/arXiv.2105.04700.

36 David G. Harris, Johannes Schneider, and Hsin-Hao Su. Distributed (Δ +1)-coloring in sublogarithmic rounds. *J. ACM*, 65(4):19:1–19:21, 2018. doi:10.1145/3178120.

37 Monika Henzinger and Pan Peng. Constant-time dynamic (Δ +1)-coloring. *ACM Trans. Algorithms*, 18(2):16:1–16:21, 2022. doi:10.1145/3501403.

38 Juho Hirvonen and Jukka Suomela. *Distributed Algorithms*. Aalto University, 2020. URL: https://jukkasuomela.fi/da2020/.

39 Jeff Kahn and Charles Kenney. Asymptotics for palette sparsification. *arXiv preprint arXiv:2306.00171*, 2023. doi:10.48550/arXiv.2306.00171.

40 Jeff Kahn and Charles Kenney. Asymptotics for palette sparsification from variable lists. *arXiv preprint arXiv:2407.07928*, 2024. doi:10.48550/arXiv.2407.07928.

41 Sanjeev Khanna, Nathan Linial, and Shmuel Safra. On the hardness of approximating the chromatic number. *Combinatorica*, 20(3):393–415, 2000. doi:10.1007/s004930070013.

42 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Comput. Complex.*, 8(1):21–49, 1999. doi:10.1007/s000370050018.

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

44 László Lovász. Three short proofs in graph theory. *Journal of Combinatorial Theory, Series B*, 19(3):269–271, 1975. doi:10.1016/0095-8956(75)90089-1.

45 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. Assoc. Comput. Mach.*, 41(5):960–981, 1994. doi:10.1145/185675.306789.

46 L. S. Melnikov and V. G. Vizing. New proof of brooks' theorem. *Journal of Combinatorial Theory*, 7(4):289–290, 1969. doi:10.1016/S0021-9800(69)80057-8.

47 Michael Molloy and Bruce Reed. *Graph colouring and the probabilistic method*, volume 23 of *Algorithms and Combinatorics*. Springer, 2002. doi:10.1007/978-3-642-04016-0.

48 Michael Molloy and Bruce A. Reed. Colouring graphs when the number of colours is almost the maximum degree. *J. Comb. Theory B*, 109:134–195, 2014. doi:10.1016/j.jctb.2014.06.004.

49 Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020. doi:10.1017/9781108671644.

50 Bruce Reed. A strengthening of brooks' theorem. *Journal of Combinatorial Theory, Series B*, 76(2):136–149, 1999. doi:10.1006/jctb.1998.1891.

51 Bruce A. Reed. ω, Δ, and χ. *J. Graph Theory*, 27(4):177–212, 1998. doi:10.1002/(SICI)1097-0118(199804)27:4\%3C177::AID-JGT1\%3E3.0.CO;2-K.

52 Gopalan Sajith and Sanjeev Saxena. On brooks' theorem, 2022. doi:10.48550/arXiv.2208.02186.

53 Michael Stiebitz and Bjarne Toft. Brooks' theorem. In Lowell W. Beineke and Robin J. Wilson, editors, *Topics in Chromatic Graph Theory*, pages 36–55. Cambridge University Press, 2015. doi:10.1017/CBO9781139519793.005.