Deterministic Graph Coloring in the Streaming Model*

Sepehr Assadi†
Rutgers University
USA
sepehr.assadi@rutgers.edu

Andrew Chen‡
Cornell University
USA
ac2337@cornell.edu

Glenn Sun§
UCLA
USA
glennsun@ucla.edu

ABSTRACT

Recent breakthroughs in graph streaming have led to design of semi-streaming algorithms for various vertex coloring problems such as $(\Delta+1)$-coloring, degeneracy-coloring, coloring triangle-free graphs, and others. These algorithms are all randomized in crucial ways and whether or not there is any deterministic analogue of them has remained an important open question in this line of work.

We settle this fundamental question by proving that there is no deterministic single-pass semi-streaming algorithm that given a graph G with maximum degree Δ, can output a proper coloring of G using any number of colors which is sub-exponential in Δ. Our proof is based on analyzing the multi-party communication complexity of a related communication game, using random graph theory type arguments that may be of independent interest.

We complement our lower bound by showing that one extra pass over the input allows one to recover an $O(\Delta^2)$ coloring via a deterministic semi-streaming algorithm. This result is extended to an $O(\Delta)$ coloring in $O(\log \Delta)$ passes even in dynamic streams.

CCS CONCEPTS

• **Theory of computation → Streaming, sublinear and near linear time algorithms**; **Graph algorithms analysis**.

KEYWORDS

Graph coloring, streaming, deterministic algorithms, lower bounds

ACM Reference Format:

Sepehr Assadi, Andrew Chen, and Glenn Sun. 2022. Deterministic Graph Coloring in the Streaming Model. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520016

*A full version of the paper is available on arXiv: https://arxiv.org/abs/2109.14891.

†Research supported in part by the NSF CAREER award CCF-2047061 and a gift from Google Research.

‡Research done primarily as part of 2020 REU program at DIMACS and Rutgers, supported by the NSF grant CCF-1852215.

§Research done primarily as part of 2021 REU program at DIMACS and Rutgers, supported by the NSF grant CCF-1836666.

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06…$15.00
https://doi.org/10.1145/3519935.3520016

1 INTRODUCTION

Coloring graphs with a small number of colors is a central problem in graph theory with a wide range of applications in computer science. A proper c-coloring of a graph $G = (V, E)$ assigns a color from the palette $\{1, \ldots, c\}$ to the vertices so that no edge is monochromatic. We study graph coloring in the semi-streaming model introduced by [23]: the edges of an n-vertex input graph are arriving one by one in a stream and the algorithm can make one (or a few) passes over the stream and use a limited memory of $O(n \cdot \text{polylog}(n))$ bits. At the end, it should output a proper coloring of the input graph. The semi-streaming model is particularly motivated by its applications to processing massive graphs and has received extensive attention in the last two decades.

Similar to the classical setting, it is known that approximating the minimum number of colors for proper coloring is quite intractable in the semi-streaming model [1, 17, 29]. As a result, the interest in this problem in graph streaming has primarily been on obtaining colorings with number of colors proportional to certain combinatorial parameters of input graphs, such as maximum degree or degeneracy. On this front, a breakthrough result of [3] gave the first semi-streaming algorithm for $(\Delta + 1)$ coloring of graphs with maximum degree Δ (see also the independent work of [11] that obtained $O(\Delta)$ coloring). Another remarkable result is that of [10] that gave a semi-streaming algorithm for $(\kappa + o(\kappa))$-coloring of graphs with degeneracy κ. See [2, 8, 12, 17] for other related results.

Perhaps, the single most common characteristic of all results in this line of work is that *they crucially rely on randomization*. For instance, one of the strongest tool for streaming graph coloring is the palette sparsification theorem of [3] which states the following: if we sample $O(\log n)$ colors from $\{1, \ldots, \Delta + 1\}$ for each vertex independently and uniformly at random, then with high probability, the entire graph can be colored using only the sampled colors of each vertex. This result immediately leads to a semi-streaming algorithm for $(\Delta + 1)$ coloring: after sampling $O(\log n)$ colors for each vertex, only $O(n \log^2(n))$ edges can potentially become monochromatic under *any* coloring of vertices from their sampled colors; thus, the algorithm can simply store these edges throughout the stream and find the desired coloring at the end (which is guaranteed to exist by the palette sparsification theorem). But the resulting algorithm is inherently randomized with this tool.

This state-of-affairs of graph coloring in admitting only randomized semi-streaming algorithms is rather unusual in the literature. Indeed, most problems of interest in the semi-streaming model such

as (minimum) spanning trees [23], edge/vertex connectivity [27], cut and spectral sparsifiers [31], spanners [23, 24] and weighted matchings [33] all admit deterministic algorithms with the same performance as best known randomized algorithms[1] (or altogether do not admit non-trivial algorithms even with randomization; see, e.g. [3, 6, 12, 17, 24] for various examples of such impossibility results). Consequently, there has been a general interest in derandomizing the semi-streaming algorithms for graph coloring, following the same recent trend in various related models such as distributed computing [14, 18, 25, 32] and Massively Parallel Computation (MPC) algorithms [19, 20]. This has led to the following important open question:

> *Can we design* **deterministic semi-streaming algorithms** *for graph coloring with similar guarantees as the randomized ones? In particular, are there deterministic semi-streaming algorithms for* $(\Delta+1)$*-coloring,* $O(\Delta)$ *coloring, or even* $\text{poly}(\Delta)$ *coloring?*

1.1 Our Contributions

Our main result is a strong negative answer to this fundamental open question: coloring graphs even with $\exp\left(\Delta^{o(1)}\right)$ colors is not possible with a deterministic semi-streaming algorithm!

> Result 1. *There does not exist any deterministic single-pass semi-streaming algorithm for coloring graphs of maximum degree* Δ *using at most* $\exp(\Delta^{o(1)})$ *colors (even when* Δ *is known to the algorithm at the beginning of the stream).*

Result 1 extends to the entire range of streaming algorithms with $o(n\Delta)$ space as well; see Corollary 4.7 for the formalization of this result and precise bounds. We emphasize that graph coloring, with more than Δ colors, is inherently a *search* problem not a *decision* one as all graphs can be colored with $\Delta+1$ colors after all. Thus, our lower bound in Result 1 says that even though we are certain that the input graph can be colored with $(\Delta+1)$ colors, we cannot find a coloring with even (almost) exponentially more number of colors.

Previously, no space lower bound was known for deterministic semi-streaming algorithms even for $(\Delta+1)$-coloring and even for dynamic streams that also allow for deleting edges from the stream[2] (but see Section 1.3 for a recent independent work). On the other hand, Result 1 effectively rules out any non-trivial algorithm for graph coloring: the best thing to do in $O(n\log^q(n))$ space is to either store the entire input graph when $\Delta \lesssim \log^q(n)$ and find a $(\Delta+1)$ coloring at the end, or color all vertices differently which results in $n \approx \exp\left(\Delta^{1/q}\right)$-coloring for $\Delta \gtrsim \log^q(n)$. Combined with the randomized algorithm of [3] for $(\Delta+1)$ coloring, Result 1 presents one of the strongest separations between deterministic and randomized algorithms in the semi-streaming model.

Given the strong impossibility result of Result 1, it is natural to consider standard relaxations of the problem. For this, we consider *multi-pass* algorithms that read the stream more than once. Multi-pass algorithms have also been studied extensively since the introduction of semi-streaming algorithms in [23]. We show that unlike in a single pass, deterministic semi-streaming multi-pass algorithms can indeed solve non-trivial graph coloring problems already in just two passes.

> Result 2. *There exist deterministic semi-streaming algorithms for coloring graphs of maximum degree* Δ *using* $O(\Delta^2)$ *colors in two passes or* $O(\Delta)$ *colors in* $O(\log\Delta)$ *passes. The algorithms can be implemented even in dynamic streams with edge deletions (still deterministically).*

Previously, no non-trivial deterministic semi-streaming algorithm was known for graph coloring. In light of Result 1, our algorithms in Result 2 also provide one of the strongest separation between two-pass and single-pass algorithms (see [5] for another example via min-cuts). Finally, our algorithms in Result 2 are among the first *deterministic* algorithms that work on *dynamic* streams.

All in all, our results collectively establish surprising aspects of graph coloring in the semi-streaming model, further cementing the role of this fundamental problem in capturing various different separations and properties in this model (say, search vs decision, determinism vs randomization, and single- vs two-pass algorithms).

1.2 Our Techniques

We now give a quick summary of our techniques here. More details can be found in the high-level overview of our approach in Section 2.

Lower bound of Result 1. Our Result 1 is proven by considering the *multi-party communication complexity* of the coloring problem: here, the edges of input graph are partitioned across the players and they can speak in turn, once each, to compute a proper coloring of the input using as small as possible number of colors. It is a standard fact that communication complexity lower bounds the space of streaming algorithms. The main technical contribution of our work is thus a communication lower bound for this problem.

We obtain our lower bound by designing an adversary that specifies the inputs of players via *random subgraphs* chosen *adaptively* based on the messages of prior players. The adaptivity in distribution of inputs allows us to prove a lower bound specifically for deterministic algorithms (as a non-adaptive distributional lower bound also works for randomized algorithms by Yao's minimax principle [34]). At the same time, working with these distributional inputs makes our arguments much simpler compared to using a typical counting argument over all possible graphs (we elaborate more on this in Section 2). One main ingredient of this proof is determining the power of communication protocols for "compressing

[1] There are some other exceptions to this rule also; moreover, in many cases, randomization can further help, e.g., by reducing the runtime of algorithms, but typically not that much with their space. We also emphasize that this "rough equivalence of power" of deterministic vs randomized algorithms only exist in the semi-streaming model: once we reduce the space to $o(n)$, deterministic algorithms are much weaker than randomized ones for most problems.

[2] Unlike insertion-only streams, *all* known algorithms in dynamic streams are randomized and for a crucial reason. It is easy to see that any non-trivial algorithm that should return a single edge from the graph cannot be deterministic in dynamic streams: one can simply use the memory of the algorithm to recover the entire input by passing each returned edge as a deletion to the algorithm, hence forcing it to return another edge of the graph, until we recover the entire graph. This means the memory of the algorithm has to be $\Omega(n^2)$ bits, enough to store the entire input. This approach however does not apply to $(\Delta+1)$-coloring at it does not require returning any edge as output.

non-edges" in a random subgraph, compared to standard approaches that bound the number of *edges* that can be recovered.

Algorithms of Result 2. Our algorithmic results are based on finding a way to *non-properly* color the graph using a small number of colors, so that the number of monochromatic edges is small. We can then store these edges explicitly and use them to further refine this non-proper coloring to a proper coloring of the entire graph (for $O(\Delta^2)$ coloring) or further extending a partial coloring and recurse (for $O(\Delta)$ coloring).

To be able to implement this strategy, we design families of coloring functions of small size so that for any given graph, at least one of these coloring functions lead to the desired non-proper coloring with a small number of monochromatic edges. These families are obtained via standard tools in de-randomization, namely, *near-universal hash functions*.

1.3 Recent Related Work

Independently and concurrently, [15] studied graph coloring in the semi-streaming model for *adversarially robust* algorithms (see [9, 15] for more context). They prove that no semi-streaming algorithm can be adversarially robust for $(\Delta^{2-\varepsilon})$-coloring for constant $\varepsilon > 0$. As all deterministic algorithms are adversarially robust, their result implies that no deterministic semi-streaming algorithm can achieve a $(\Delta^{2-\varepsilon})$-coloring. The authors of [15] state that: "A major remaining open question is whether this [lower bound] can be matched, perhaps by a deterministic semi-streaming $O(\Delta^2)$ coloring algorithm. In fact, it is not known how to get even a $\text{poly}(\Delta)$-coloring deterministically". Result 1 fully settles their open question for deterministic algorithms in negative. Incidentally, [15] provides a randomized but adversarially robust semi-streaming algorithm for $O(\Delta^3)$ coloring. Thus one cannot hope for our $\exp(\Delta^{o(1)})$ coloring lower bound in their model. Technique-wise, the two work are mostly disjoint: their lower bound is based on two-party communication complexity in a way that handles randomization for adversarially robust algorithms, while ours is based on multi-party communication complexity for deterministic algorithms (as their model admits an $O(\Delta^3)$ coloring, there is no "extension" of their two-party lower bound to a multi-party one to get the bounds in our paper – our approach even restricted to two-parties look quite different). Algorithms of the two work are entirely disjoint.

1.4 Further Related Work

Recently, there has been a surge of interest in graph coloring and related problems in graph streams [2–4, 7, 8, 10, 12, 16, 17, 28, 30]. Beside what already mentioned, another work related to ours is [2] that studied graph theoretic aspects of palette sparsification theorem of [3] and obtained semi-streaming algorithms for coloring triangle-free graphs and (deg +1)-coloring. Very recently, [7] proved an analogue of celebrated Brook's theorem [13] in the semi-streaming model: there is a randomized semi-streaming algorithm that can Δ-color any given graph besides cliques and odd cycles (which are not Δ-colorable). Moreover, [12] showed that some of the "easiest" problems in coloring are intractable in the semi-streaming model (even with randomization). See also [31] for an excellent overview of work on other problems in the semi-streaming model.

2 HIGH-LEVEL OVERVIEW

We give a streamlined overview of our approach in this section. We emphasize that this section oversimplifies many details and the discussions will be informal for the sake of intuition.

2.1 Lower Bound of Result 1

As stated earlier, the proof of Result 1 is by considering the multi-party communication complexity of the coloring problem. To start, let us consider the simple case of *two* players Alice and Bob, receiving edges of a graph G with maximum degree Δ. Alice sends a message M to Bob and Bob outputs a proper coloring of G using as small as possible number of colors. What is the best strategy of players for solving the problem with limited communication and small number of colors? As stated earlier, coloring with more than Δ colors is inherently a search problem, thus this question is basically asking how much Bob should learn about Alice's input to agree on a proper coloring of the *entire* graph (without knowing all edges of Alice). This view will be important throughout this discussion and our formal lower bound arguments.

Two-player communication complexity of coloring. There is a simple solution to our two-player communication game using $\approx n$ size messages and $O(\Delta^2)$ colors. Alice simply sends a $(\Delta+1)$ coloring of her input graph to Bob and Bob further finds a $(\Delta+1)$ coloring of each of Alice's color classes *individually* to obtain a proper $(\Delta+1)^2$ coloring of the entire input graph. Let us show that this is essentially the best one can do using $O(n)$ size messages and for a specific choice of $\Delta = \Theta(\sqrt{n})$ (neither of these assumptions are needed in our main lower bound).

Suppose Alice receives an arbitrary graph with maximum degree $\sqrt{n}$ and maps it to a message of size $O(n)$. As the graphs with maximum degree $\sqrt{n}$ are a *constant fraction* of graphs with $(n^{3/2}/2)$ edges[3], we have that there is a message, to which, Alice is mapping at least

$$\Omega(1) \cdot \binom{\binom{n}{2}}{\frac{n^{3/2}}{2}} \cdot 2^{-O(n)} \gtrsim \exp\left(\frac{n^{3/2}}{4} \cdot \ln(n)\right) \cdot 2^{-O(n)},$$

many different graphs. At the same time, given this message, Bob should avoid coloring any pairs of vertices the same if they appear in *some* graph mapped to this message. But having so many graphs mapped to the same message only allows for $O(n^{3/2})$ pairs of vertices to not have any edge at all in *any* of these graphs; this is because the total number of graphs with maximum degree $\sqrt{n}$ whose edges avoid a fixed set of $O(n^{3/2})$ pairs of vertices have size at most

$$\binom{\binom{n}{2} - O(n^{3/2})}{\frac{n^{3/2}}{2}} \lesssim \binom{\binom{n}{2}}{\frac{n^{3/2}}{2}} \cdot \left(1 - \frac{1}{O(\sqrt{n})}\right)^{\left(\frac{n^{3/2}}{2}\right)}$$

[3]Technically speaking, this sentence is *not* correct – one needs the number of edges to be $(1+o(1)) \cdot (n^{3/2}/2)$ instead, but this distinction is not relevant for our discussion here. Thus, to avoid the clutter, we omit this extra term in these informal calculations.

$$\lesssim \exp\left(\frac{n^{3/2}}{4}\cdot \ln(n)\right)\cdot 2^{-O(n)}.$$

At this point, this means that *from the perspective of Bob*, only $O(n^{3/2})$ pairs of vertices can be colored the same, even ignoring his own input graph (see Figure 1 for an illustration). Moreover, a Markov bound implies that half the vertices only have $O(\sqrt{n})$ non-edges from the perspective of Bob. Thus, Bob will "see" a set S of $\Theta(n)$ vertices where each one has at most $O(\sqrt{n})$ non-edges inside S. But recall that we are considering the case where maximum degree can be as large as $\Theta(\sqrt{n})$. So Bob's own input can simply contain all non-edges inside S while keeping the maximum degree of the graph still $O(\sqrt{n})$. At this point, the induced subgraph on vertices S, from the perspective of Bob, is simply a clique, and thus requires $|S| = \Omega(n)$ colors. Since $\Delta = \Theta(\sqrt{n})$, this gives us an $\Omega(\Delta^2)$ lower bound on the number of colors.

Multi-party communication complexity of coloring. Given the protocol mentioned earlier for two players, to prove Result 1, we need to consider a larger number of players. In general, the same strategy outlined above also implies a protocol for k players with $O(n)$ communication per player and an $O(\Delta^k)$ coloring. Our goal is to match this in our lower bound.

Suppose now we have k players $P_1, \ldots, P_k$ and the input edges are partitioned between them. Let us again present a graph of maximum degree $\approx \Delta/k$ to the first player. We can again use a similar counting argument to bound the number of non-edges in inputs mapped to a message of player P_1 (assuming that it has size, say, $O(n)$). We would like to continue this procedure, by choosing the input graph of player P_2 in a way that "destroys" many of these pairs, while having maximum degree of still $\approx \Delta/k$; then recourse on the third player and so on. However, continuing the above counting argument directly seems intractable at this point.

It turns out however that there is an easier way to implement this strategy by providing the input of players as *random subgraphs*. Specifically, the process goes as follows (see Figure 2):

- We present the first player P_1 with a random Erdős-Rényi graph with probability $\approx (\Delta/kn)$ for each edge (so max-degree $\approx \Delta/k$ with high probability). We prove that (see our **Compression Lemma** below) that there is some message M_1 of P_1 that creates $\lesssim k\cdot n^2/\Delta$ non-edges from the perspective of remaining players. We further *remove* all vertices with non-edge-degree $\gtrsim k^2 n/\Delta$ which by Markov bound are only $\lesssim n/k$.
- To player P_2, we give a *random subgraph* of (remaining) non-edges left by M_1 where each edge appears with probability $\approx (\Delta^2/k^3 n)$ now. By the bound of $\lesssim k^2 n/\Delta$ on the non-edge-degree of remaining vertices, it is easy to see that the input given to P_2 *still* has max-degree $\approx \Delta/k$ with high probability. We again use the **Compression Lemma** to find a message M_2 of P_2 that creates $\lesssim k^3 n^2/\Delta^2$ non-edges from the perspective of subsequent players, and continue. This way, each step to the next player will remove $\lesssim n/k$ vertices while reducing non-edge-degree of remaining vertices by a $\gtrsim \Delta/k^2$ factor.
- Eventually, we will be able to give a random subgraph of non-edges left by $M_1, \ldots, M_{k-1}$ to the player P_k with edge probability $\approx (\Delta^k/k^{2k} n)$, and bound the *total* maximum-degree of the graph by $k\cdot\Delta/k = \Delta$ as desired. But if we assume that $(\Delta/k^2)^k \approx n$ (again, this assumption is only for simplicity of exposition here), it means that we turned the *remaining* vertices of the graph, *from the perspective of* P_k, into a *clique* entirely[4]. Moreover, since we only removed $\lesssim n/k$ vertices for each player, we still have $\approx n/k$ vertices left in this clique. Thus, the number of colors needed by P_k to color this clique is $\approx n/k \gtrsim (\Delta/k^3)^k$ (which is larger than $\text{poly}(\Delta)$ for sufficiently large k).

Finally, we also state our compression lemma that is used to find the messages $M_1, \ldots, M_{k-1}$ that create "small" number of non-edges in the above discussion.

- **Compression Lemma:** Let H be any arbitrary graph and consider a distribution over subgraphs of H obtained by sampling each edge with probability p. Any compression scheme that maps the graphs sampled from this distribution into s-bit summaries will create a summary so that at most $O(s/p)$ edges are *missing* from *all* graphs mapped to this summary.

This bound should be contrasted with more standard compression arguments that in the same setting, prove that $O(s\cdot\log^{-1}(1/p))$ *edges* exist in *all* graphs mapped to the summary. The proof is a simple exercise in random graph theory plus showing that an s-bit compression cannot "capture" events that happen with probability $< 2^{-s}$ in the input distribution. This concludes the description of our lower bound approach for establishing Result 1.

2.2 Algorithms of Result 2

We now turn to our algorithmic results for multi-pass semi-streaming algorithms for graph coloring.

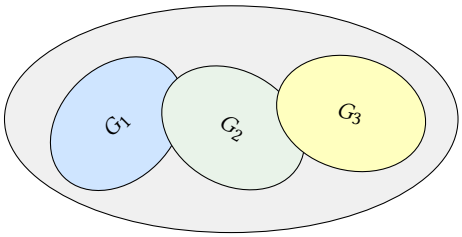
$O(\Delta^2)$ coloring in two passes. The key ingredient of this algorithm is the following family of coloring functions for any integers $n, \Delta \geq 1$:

- $\mathcal{C}(n,\Delta)$: there are $O(n)$ functions $C : V \to [\Delta]$ in the family so that given any n-vertex graph $G = (V, E)$ with max-degree Δ, there is *some* function C in the family such that assigning color $C(v)$ to each vertex v only creates $O(n)$ monochromatic edges. Moreover, each of these functions can be implicitly stored in $O(\log n)$ bits.

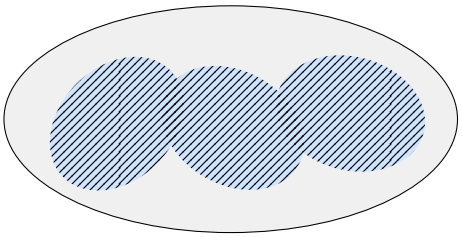
The proof of existence of this family is via probabilistic method by choosing these functions to be near-universal hash functions and a simple probabilistic analysis.

Now, consider the following simple two-pass algorithm. In the first pass, maintain $O(n)$ counters on the number of monochromatic edges of G for each of the functions $C \in \mathcal{C}(n,\Delta)$: the counter for function C simply needs to add one for each edge (u, v) appearing in the stream with $C(u) = C(v)$. This only requires $O(n)$ space. Given that we already know at least one of these counters only count up to $O(n)$ by the guarantee of $\mathcal{C}(n,\Delta)$, we will use the function C of that counter and *store* all monochromatic edges of G under C. Given that G had maximum-degree Δ, these monochromatic edges under C can themselves be properly colored using $(\Delta+1)$

[4] We emphasize that this clique is not part of a single input graph, but rather is a *union* of various inputs, which are all consistent with the view of player P_k based on the input and messages received.

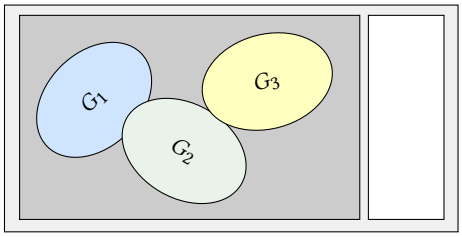


(a) Alice has to map several graphs to the same message. These graphs are *individually* "sparse": they have *max*-degree $\lesssim \sqrt{n}$.

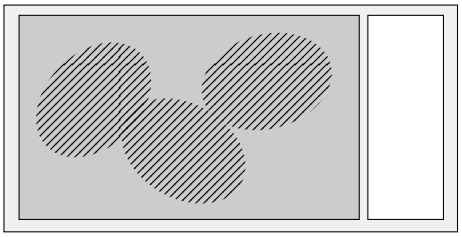


(b) Bob however "sees" all these edges as part of the input. So, from Bob's perspective, this subgraph is "dense": it has *min*-degree $\gtrsim n - \sqrt{n}$. Thus, even a "sparse" input to Bob with max-degree $\lesssim \sqrt{n}$, turns this subgraph into a clique.

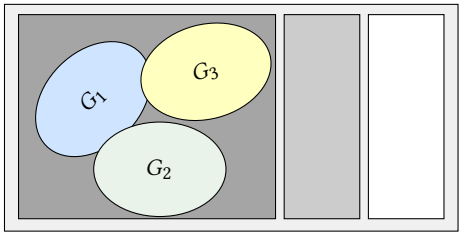
Figure 1: An illustration of the two-player communication lower bound.



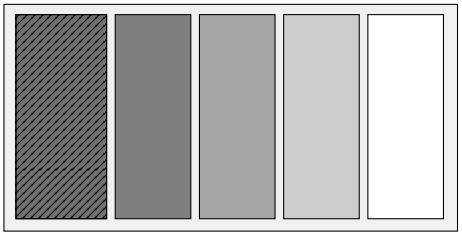
(a) Player P_i's different inputs that are mapped to the same message. The right (white) part are the vertices already removed from consideration and the left (dark) part are the "dense" subgraph of the input from the perspective of P_i.



(b) For player P_{i+1}, the left (dark) part "looks" even more "dense" than it was for player P_i, as multiple different graphs of P_i's input are mapped to the same message.



(c) We further remove "less dense" part of the input (middle layer) and provide the inputs of P_{i+1} inside the remaining subgraph.



(d) We continue like this until the last player; at that point, the remaining "super dense" part of the input (left most part) from the perspective of P_k is simply a clique.

Figure 2: An illustration of the multi-player communication lower bound.

colors. Taking the *product* of these two colorings then will give us an $O(\Delta^2)$ coloring as desired.

$O(\Delta)$ coloring in $O(\log \Delta)$ passes. The idea behind this algorithm is to *gradually* grow a coloring of G over multiple passes, using an extension of the ideas in the previous algorithm. For this, we need another family of coloring functions for integers n, Δ:

- $\mathcal{C}^\star(n, \Delta)$: there are $O(n)$ functions $C : V \to [O(\Delta)]$ in the family so that given any n-vertex graph $G = (V, E)$ with max-degree Δ and any partial (valid) coloring C_0 of some *subset* of vertices, there is *some* function C in the family such that assigning color $C(v)$ to every vertex v *uncolored* by C_0 only creates $o(n_0)$ monochromatic edges, where n_0 is the number of uncolored vertices by C_0. Moreover, each function can be implicitly stored in $O(\log n)$ bits.

The proof of existence of this family is again via probabilistic arguments although it needs a more detailed analysis.

The algorithm is then as follows. We start with a coloring C_0 that leaves all vertices uncolored. Then, iteratively, we first make one pass and use $O(n)$ counters to find a desired coloring function $C \in \mathcal{C}^\star(n, \Delta)$ as specified by the above result; in the second pass we pick all $o(n_0)$ monochromatic edges of this coloring with respect to

C_0. This allows us to color $(1-o(1))$ fraction of uncolored vertices of C_0 by C without creating *any* monochromatic edges. We continue this for $O(\log \Delta)$ iterations so that C_0 only leaves $O(n/\Delta)$ vertices uncolored. We make one final pass over the input and store all $O(n)$ edges incident on these remaining vertices and then at the end, simply color them greedily using $(\Delta+1)$ colors (as *any* partial coloring can be extended to a $(\Delta+1)$ coloring greedily). This gives our $O(\Delta)$ coloring algorithm.

We conclude this part by noting that even though both our algorithms turn out quite simple, their design, based on families $\mathcal{C}(n,\Delta)$ and $\mathcal{C}^\star(n,\Delta)$, requires a careful consideration to ensure one can also *verify* the guarantees of families in *limited space*[5].

3 PRELIMINARIES

Notation. For an integer $t \geq 1$, we define $[t] := \{1,2,\ldots,t\}$. For a tuple $(X_1,\ldots,X_t)$ and any $i \in [t]$, we define $X_{<i} := (X_1,\ldots,X_{i-1})$. For a distribution μ, $\text{supp}(\mu)$ denotes the support of μ.

For a graph $G=(V,E)$, we use $\Delta(G)$ to denote the maximum degree of G. For a vertex $v \in V$, we use $N(v)$ to denote the neighbors of v in G. For any integer $c \geq 1$, a **c-coloring function** is *any* function $C : V \to [c]$ and does *not* necessarily need to be a proper coloring of G. A monochromatic edge under C is any edge (u,v) of G with $C(u) = C(v)$. We further define a **partial c-coloring function** as any function $C : V \to [c] \cup \{\perp\}$; we refer to vertices v with $C(v) = \perp$ as *uncolored vertices* and do *not* consider edges (u,v) with $C(u) = C(v) = \perp$ as monochromatic edges.

We use the following standard form of Chernoff bound.

Proposition 3.1 (Chernoff bound; c.f. [22]). *Suppose $X_1,\ldots,X_m$ are m independent random variables with range $[0,1]$ each. Let $X := \sum_{i=1}^m X_i$ and $\mu_L \leq \mathbb{E}[X] \leq \mu_H$. Then, for any $\varepsilon > 0$,*

$$\Pr\left(X > (1+\varepsilon)\cdot\mu_H\right) \leq \exp\left(-\frac{\varepsilon^2\cdot\mu_H}{3+\varepsilon}\right)$$

$$\Pr\left(X < (1-\varepsilon)\cdot\mu_L\right) \leq \exp\left(-\frac{\varepsilon^2\cdot\mu_L}{2+\varepsilon}\right).$$

Finally, we use the following property of any proper coloring.

Proposition 3.2. *In any proper c-coloring of a graph $G=(V,E)$ for $c \leq \frac{n}{2}$, there are $\geq \frac{n^2}{4c}$ pairs of vertices that are colored the same.*

Proof. For any $i \in [c]$, let n_i vertices denote the number of vertices colored i in the given c-coloring.

$$\text{number of pairs colored the same} = \sum_{i=1}^{c}\binom{n_i}{2} = \frac{1}{2}\cdot\left(\sum_{i=1}^{c} n_i^2 - n_i\right)$$

$$= \frac{1}{2}\cdot\left(\sum_{i=1}^{c} n_i^2\right) - \frac{1}{2}\cdot n$$

(as $\sum_{i=1}^{c} n_i = n$ since all vertices are colored)

$$\geq \frac{1}{2}\cdot\left(c\cdot(\frac{n}{c})^2 - n\right) \geq \frac{n^2}{4c},$$

(as sum of quadratic-terms is minimized when they are all equal)

where the last inequality is by the assumption $c \leq n/2$. This concludes the proof. ∎ Proposition 3.2

4 THE LOWER BOUND

We present our lower bound in this section and formalize Result 1. We start by introducing a key tool used in our lower bound regarding a family of random graphs and its key compression aspect for our purpose. We then define the communication game we use in proving Result 1 formally, and next present the proof the communication lower bound.

4.1 A Random Graph Distribution

We introduce a basic random graph distribution in this subsection that forms an important component of the analysis of our lower bound. The key difference of our distribution from standard random graph models is that it generates random subgraphs of *arbitrary (base) graphs*, as opposed to subgraphs of cliques (which means some edges may never appear in the support of this distribution if they are not part of the base graph). Another (minor) change is that we ensure a *deterministic* bound on the maximum degree of the graphs sampled from this distribution.

Definition 4.1. For a **base graph** $G_{\text{Base}} = (V, E_{\text{Base}})$ and parameters $p \in (0,1), d \geq 1$, we define the **random graph distribution** $\mathbb{G} := \mathbb{G}(G_{\text{Base}}, p, d)$ as follows:

(i) Sample a graph G on vertices V and edges E by picking each edge of E_{Base} independently and with probability p in E;

(ii) Return G if $\Delta(G) < 2p\cdot d$, and otherwise repeat the process.

We will eventually set d to be approximately $\Delta(G_{\text{Base}})$. Since the average degree of a vertex is at most $p\cdot\Delta(G_{\text{Base}}) \approx p\cdot d$, the second condition of Definition 4.1 rarely kicks in by Chernoff bound, and thus this distribution is basically sampling random subgraphs of G_{Base}. We will make these statements more precise in the proof of Claim 4.4. We now consider algorithms that aim to "compress" graphs sampled from $\mathbb{G}$.

Definition 4.2. Consider $\mathbb{G}(G_{\text{Base}}, p, d)$ for a base graph $G_{\text{Base}} = (V, E_{\text{Base}})$ and parameters $p \in (0,1), d \geq 1$, and an integer $s \geq 1$. A **compression algorithm** with size s is any function $\Phi : \text{supp}(\mathbb{G}) \to \{0,1\}^s$ that maps graphs sampled from $\mathbb{G}$ into s-bit strings. For any graph $G \in \text{supp}(\mathbb{G})$, we refer to $\Phi(G)$ as the **summary** of G. For any summary $\phi \in \{0,1\}^s$, we define:

- $\mathbb{G}_\phi$ as the distribution of graphs mapped to ϕ by Φ, i.e., $\mathbb{G}_\phi := (G \sim \mathbb{G} \mid \Phi(G) = \phi)$.
- $G_{\texttt{Miss}}(\phi) = (V, E_{\texttt{Miss}}(\phi))$, called the **missing graph** of ϕ, as a graph on vertices V and edges missed by *all* graphs in $\mathbb{G}_\phi$, i.e., $E_{\texttt{Miss}}(\phi)$ is the set of all $(u,v) \in E_{\text{Base}}$ such that no graph in $\text{supp}(\mathbb{G}_\phi)$ contains the edge (u,v).

[5] For instance, a "more standard" guarantee instead of $C(n,\Delta)$ that bounds the *maximum-degree* of monochromatic edges can also be obtained via pair-wise independent hash functions (see, e.g. [10, 18]); but then that would require $\Theta(n)$ space *per* each function to verify whether or not the function satisfies the desired property.

We use the graphs from distribution $\mathbb{G}$ (for different base graphs and probability parameters) in the design of our lower bounds. The compression algorithms in Definition 4.2 then correspond to streaming algorithms that compress these graphs into their s-bit memory.

The notion of a missing graph is particularly useful for us, as from the perspective the streaming algorithm, only pairs of vertices with an edge in the missing graph are known to *not* have an edge in the original input. This implies that these are the only pairs of vertices that can be monochromatic in the final coloring without violating the correctness of the algorithm on *some* input.

The following lemma summarizes the main property of compression algorithms for our random graph distribution required in our main proof. Roughly speaking, it states that the *missing* graph of a "small-size" compression algorithm cannot have "many" edges

Lemma 4.3. *Let $G_{\text{Base}} = (V, E_{\text{Base}})$ be an n-vertex graph, $s \geq 1$ be an integer, and $p \in (0,1)$ and $d \geq 1$ be parameters such that $d \geq \max\{\Delta(G_{\text{Base}}), 4\ln(2n)/p\}$. Consider the distribution $\mathbb{G} := \mathbb{G}(G_{\text{Base}}, p, d)$ and suppose $\Phi : \text{supp}(\mathbb{G}) \to \{0,1\}^s$ is a compression algorithm of size s for $\mathbb{G}$. Then, there exist a summary $\phi^* \in \{0,1\}^s$ such that in the missing graph of ϕ^*, we have*

$$\left|E_{\texttt{Miss}}(\phi^*)\right| \leq \frac{\ln 2 \cdot (s+1)}{p}.$$

Proof. Define the distribution $\widetilde{\mathbb{G}}$ as the distribution of graphs in Line (i) of Definition 4.1 (i.e., without the check on max-degree and re-sampling step). This way, we have

$$\mathbb{G} = \left(G \sim \widetilde{\mathbb{G}} \mid \Delta(G) < 2p \cdot d\right). \tag{1}$$

We shall use this view in the following for bounding the probabilities of certain events. We have the following simple claim that bounds the probability of any event in $\mathbb{G}$ by twice the probability of the same event in $\widetilde{\mathbb{G}}$ – this is true simply because the graphs sampled from $\widetilde{\mathbb{G}}$ already satisfy the conditioned event above with high enough probability. The proof is standard and is postponed to the full version of the paper.

Claim 4.4. *For any event $\mathcal{E}$, $\Pr_{\mathbb{G}}(\mathcal{E}) \leq 2 \cdot \Pr_{\widetilde{\mathbb{G}}}(\mathcal{E})$.*

For any summary $\phi \in \{0,1\}^s$, its distribution $\mathbb{G}_\phi$, and its missing graph $G_{\texttt{Miss}}(\phi)$,

$$\begin{aligned}&\Pr_{\mathbb{G}}\left(G \text{ is sampled from } \mathbb{G}_\phi\right)\\ &\quad\leq \Pr_{\mathbb{G}}\left(\text{no edge of } G_{\texttt{Miss}}(\phi) \text{ is sampled in } G\right),\end{aligned} \tag{2}$$

because edges in $G_{\texttt{Miss}}(\phi)$ cannot belong to the graphs in the support of $\mathbb{G}_\phi$ by Definition 4.2. We can bound the RHS of Equation (2) using the distribution $\widetilde{\mathbb{G}}$ and apply Claim 4.4 to get the result for $\mathbb{G}$ also. By the independence in the choice of edges in $\widetilde{\mathbb{G}}$, we have,

$$\begin{aligned}&\Pr_{\widetilde{\mathbb{G}}}\left(\text{no edge of } G_{\texttt{Miss}}(\phi) \text{ is sampled in } G\right)\\ &\quad= \prod_{e \in E_{\texttt{Miss}}(\phi)} (1-p)\\ &\quad= (1-p)^{|E_{\texttt{Miss}}(\phi)|} \leq \exp\Big(-p \cdot |E_{\texttt{Miss}}(\phi)|\Big).\end{aligned}$$

Thus, combined with Claim 4.4 and Equation (2), for any summary $\phi \in \{0,1\}^s$, we have,

$$\Pr_{\mathbb{G}}\left(G \text{ is sampled from } \mathbb{G}_\phi\right) \leq 2 \cdot \exp\Big(-p \cdot |E_{\texttt{Miss}}(\phi)|\Big). \tag{3}$$

We now switch to lower bounding the LHS of Equation (3) instead. Since Φ maps each graph sampled from $\mathbb{G}$ to one of 2^s messages $\phi \in \{0,1\}^s$, we have,

$$\sum_{\phi \in \{0,1\}^s} \Pr_{\mathbb{G}}\left(G \text{ is sampled from } \mathbb{G}_\phi\right) = 1,$$

which means that there exist some $\phi^* \in \{0,1\}^s$ such that

$$\Pr_{\mathbb{G}}\left(G \text{ is sampled from } \mathbb{G}_{\phi^*}\right) \geq 2^{-s}.$$

Combining this with Equation (3), we have that

$$\exp(-s \cdot \ln 2) \leq \exp\Big(\ln 2 - p \cdot \left|E_{\texttt{Miss}}(\phi^*)\right|\Big),$$

which, by re-arranging the terms, implies the bound in the lemma statement. ∎ Lemma 4.3

4.2 The Coloring Communication Game

We prove our lower bound in Result 1 via communication complexity arguments in the following communication game.

Definition 4.5. For integers $n, \Delta, k \geq 1$, the Coloring(n, Δ, k) game is defined as:

i). There are k players $P_1, \ldots, P_k$. Each player P_i knows the vertex set V and receives a set E_i of edges. Letting $G = (V, E)$ where $E = E_1 \sqcup \cdots \sqcup E_k$, players are guaranteed that on every input $\Delta(G) \leq \Delta$ and their goal is to output a proper coloring of G.

ii). The communication is done using a shared blackboard. First player P_1 writes a message M_1 based on E_1 on the shared blackboard which will be visible to *all* subsequent players. Then, player P_2 writes the next message M_2 based on E_2 and M_1. The players continue like this until P_k writes the last message M_k which is a function of E_k and $M_{<k}$.

iii). The goal of the players is to output a valid coloring of the input graph G by P_k writing it last on the shared blackboard as the message M_k.

The **communication cost** of a protocol used by the players is defined as the *worst-case* number of bits written by *any single* player on the blackboard on any input.

Proposition 4.6. *Suppose there is a function $f : \mathbb{N}^+ \to \mathbb{N}^+$ and a deterministic streaming algorithm that on any n-vertex graph G with known maximum degree Δ, outputs an $f(\Delta)$-coloring of G using $s = s(n, \Delta)$ bits of space. Then, there also exists a deterministic protocol for* Coloring(n, Δ, k) *for any $k > 1$ with communication cost $O(s)$ bits that outputs an $f(\Delta)$-coloring of any input graph.*

Proof. The players simply run the streaming algorithm on their input by writing the content of the memory of the algorithm from one player to the next on the blackboard, so that the next player

can continue running the algorithm on their input. At the end, the last player computes the output of the streaming algorithm.

The maximum message size written on the blackboard is proportional to the size of memory of the streaming algorithm and is thus $O(s)$ as desired. ∎ Proposition 4.6

The following is the main technical result of our paper.

Theorem 1. *There are absolute constants $n_0, \eta_0 > 0$ such that the following is true. Consider any choice of the following parameters*

$$n \geq n_0, \quad \Delta \geq 64 \ln^2(2n), \quad 1 \leq k \leq \log_\Delta(n), \quad s \geq n \log \Delta.$$

Then no deterministic protocol for Coloring(n, Δ, k) *with communication cost s can color every input graph with fewer than*

$$\left(\frac{1}{\eta_0 \cdot k}\right)^{2k} \cdot \left(\frac{n \cdot \Delta}{s}\right)^{k} \text{ colors.}$$

As a corollary of this and Proposition 4.6, we can formalize Result 1.

Corollary 4.7. *For any $q \geq 1$, $\alpha \in (0,1)$, and sufficiently large $n > 1$, no deterministic single-pass streaming algorithm can obtain a proper coloring of every graph with maximum degree at most Δ for the following parameters:*

i). $O(n \cdot \log^q n)$ space and fewer than $\exp\left(\Delta^{1/4q}\right)$ colors for $\Delta = 200 \log^{q+1}(n)$;

ii). $O(n^{1+\alpha})$ space and fewer than $\Delta^{1/3\alpha}$ colors for $\Delta = n^{2\alpha}$.

Proof. We prove both parts by Proposition 4.6 and using different parameters in Theorem 1.

i). Set $k = \sqrt{\log_\Delta n} = \Theta(\sqrt{\frac{\log n}{\log\log n}})$ and $s = O(n \cdot \log^q n)$. These parameters, plus n and Δ, satisfy the hypotheses of Theorem 1. As such, we get that the minimum number of colors needed to color the input graph in this case is at least

$$\left(\frac{1}{\eta_0 \cdot k}\right)^{2k} \cdot \left(\frac{n \cdot \Delta}{s}\right)^{k} = \left(\frac{\log\log n \cdot \log n}{\Theta(1) \cdot \log n}\right)^{\Theta\left(\sqrt{\frac{\log n}{\log\log n}}\right)} > \exp\left(\Theta(1) \cdot \sqrt{\frac{\log n}{\log\log n}}\right) \gg \exp\left(\Delta^{1/4q}\right),$$

by a simple calculation of the parameters in these bounds.

ii). Set $k = \log_\Delta n = 1/2\alpha$ and $s = O(n^{1+\alpha})$. These parameters, plus n and Δ, satisfy the hypotheses of Theorem 1. As such, we get that the minimum number of colors needed to color the input graph in this case is at least

$$\left(\frac{1}{\eta_0 \cdot k}\right)^{2k} \cdot \left(\frac{n \cdot \Delta}{s}\right)^{k} = \left(\frac{n^\alpha}{\Theta(1)}\right)^{1/2\alpha} = \Theta(\sqrt{n}) \gg \Delta^{1/3\alpha},$$

again by a simple calculation. This concludes the proof. ∎

4.3 A Communication Lower Bound for Coloring

Before getting to the lower bound construction, we specify a recursive set of parameters.

Parameters. Our construction is governed by the following two parameters:

- p_i: the probability parameter used in defining the graph of each player P_i from the random graph distribution $\mathbb{G}$ for base graphs chosen by the adversary;
- d_i: a threshold on maximum degree of base graph (used in $\mathbb{G}$) chosen by the adversary for each player P_i.

These parameters are defined recursively as follows (these expression would become clear shortly from the description and analysis of the lower bound):

$$d_1 = n, \quad p_1 := \frac{\Delta}{2k \cdot n},$$
$$\text{and for } i > 1: \quad d_i = \frac{2 \ln 2 \cdot (s+1) \cdot 2k}{p_{i-1} \cdot n}, \quad p_i = \frac{\Delta}{2k \cdot d_i}. \tag{4}$$

It is easier for us to work with the recursive definitions of these parameters in most of the analysis (as their closed form is tedious to work with). But, we also compute them explicitly (the proof is postponed to the full version).

Claim 4.8. *For any $i > 1$, we have,*

$$d_i = n \cdot \left(\frac{2 \ln 2 \cdot (s+1) \cdot (2k)^2}{n \cdot \Delta}\right)^{i-1}$$
$$p_i = \frac{\Delta}{2k \cdot n} \cdot \left(\frac{n \cdot \Delta}{2 \ln 2 \cdot (s+1) \cdot (2k)^2}\right)^{i-1}.$$

The lower bound construction is as follows (see Figure 3).

An **adversary** that generates the "hard" input of players (using parameters in Equation (4)).

(i) Let $G_{\mathsf{Base}}(1)$ be a clique on n vertices $V_1 = V$.

(ii) For $i = 1$ to k:

(a) Let $\mathbb{G}_i := \mathbb{G}(G_{\mathsf{Base}}(i), p_i, d_i)$ and let
$$\Phi_i = \Phi_i(\mathbb{G}_i, M^*_{<i}) : \text{supp}(\mathbb{G}_i) \to \{0,1\}^s$$
be the function generating the message of player P_i after seeing messages $M^*_{<i}$ of the first $i-1$ players; we ensure that the input graph of player P_i given previous messages $M^*_{<i}$ is chosen from $\mathbb{G}_i$ and thus this is well defined.

(b) Notice that Φ_i is a compression algorithm. Apply Lemma 4.3 and let M^*_i be the special summary of this compression algorithm which is message for the player P_i. We shall verify the hypotheses of the lemma in Lemma 4.9.

(c) Let V_{i+1} be the set of vertices in $G_{\mathsf{Miss}}(M^*_i)$ with degree at most d_{i+1} and $G_{\mathsf{Base}}(i+1)$ be the subgraph of $G_{\mathsf{Miss}}(M^*_i)$ induced on V_{i+1}.

(iii) Let $G_i = (V_i, E_i) \in \text{supp}(\mathbb{G}_i)$ be such that $\Phi_i(G_i) = M^*_i$ for all $i \in [k]$. Give player P_i the edge set E_i as the adversarial input. We shall verify that this is a valid input in Lemma 4.10.

Notice that in this construction, we allow the players to know that their inputs come from a smaller distribution $\mathbb{G}_i$, not the entire

space of edges. This is convenient for our analysis, and since this only makes the players' jobs easier (as they can simply ignore this information), this can only strengthen our lower bound.

It is useful to note the containment relationships between various edge sets in the lower bound construction. For all $i \in [k]$, the edge sets E_i and $E_{\texttt{Miss}}(M_i^*)$ are *disjoint* because G_i was mapped to M_i^*, and both are subsets of $E_{\texttt{Base}}(i)$ by definition. Also, $E_{\texttt{Base}}(i)$ itself is a subset of $E_{\texttt{Miss}}(M_{i-1}^*)$ by Line (*ii*)c of the construction, obtained by removing "high degree" vertices in $E_{\texttt{Miss}}(M_{i-1}^*)$. A visual is provided in Figure 3 for reference.

We start with two lemmas verifying that the above construction produces a valid input and satisfies the hypotheses of Lemma 4.3.

Lemma 4.9. *For all $i \in [k]$, the parameters p_i and d_i in the distribution $\mathbb{G}_i = \mathbb{G}(G_{\texttt{Base}}(i), p_i, d_i)$ satisfy the hypotheses of Lemma 4.3. That is, $d_i \geq \max\{\Delta(G_{\texttt{Base}}(i)), 4\ln(2n)/p_i\}$ and $p_i \in (0,1)$.*

Proof. The fact that $d_i \geq \Delta(G_{\texttt{Base}}(i))$ follows from $d_1 = n$ in the case $i = 1$, and directly from the construction of $G_{\texttt{Base}}(i)$ in Line (*ii*)c for all other i.

To show that $d_i \geq 4\ln(2n)/p_i$, we first note that $p_i \cdot d_i = \frac{\Delta}{2k}$ by definition of p_i in Equation (4). Hence it suffices to show that $\Delta \geq 8k\ln(2n)$. Referencing the constraints in the statement of Theorem 1, we have $\Delta \geq 64\ln^2(2n)$ which implies $\sqrt{\Delta} \geq 8\ln(2n)$, and we have $\sqrt{\Delta} \geq k$, which combined with the latter inequality, implies $\Delta \geq 8k\ln(2n)$. This proves the bound for d_i.

We now prove the bound for p_i. For this, it is easier to work with the closed-form of p_i in Claim 4.8. We have,

$$p_i = \frac{\Delta}{2k \cdot n} \cdot \left(\frac{n \cdot \Delta}{2\ln 2 \cdot (s+1) \cdot (2k)^2}\right)^{i-1} < \frac{\Delta^i}{n} \leq \frac{\Delta^k}{n} \leq 1,$$

as $s \geq n$ and $k > 1$ for the first inequality and by the upper bound of $k \leq \log_\Delta(n)$ for the last one. It is also clear that $p_i > 0$, thus concluding the proof. ∎ Lemma 4.9

Lemma 4.10. *Any graph G constructed by the adversary has $\Delta(G) \leq \Delta$ and no parallel edges.*

Proof. Consider each graph G_i as input to player P_i. We have,

$$\Delta(G_i) < 2p_i d_i = \frac{\Delta}{k},$$

where the first inequality is by Definition 4.1 for $\mathbb{G}(G_{\texttt{Base}}(i), p_i, d_i)$ and the second equality is by the definition of p_i in Equation (4). This implies that the graph G_i presented to each player has maximum degree at most Δ/k. Given that there are k players in the game, this means the final graph has maximum degree at most Δ.

To show that there are no parallel edges, note that $E_1, \ldots, E_k$ are pairwise disjoint by the edge set containments. ∎ Lemma 4.10

We start proving the communication lower bound. First, we show that the set V_{k+1} obtained at the end, i.e., after presenting last player's input, still is "quite large".

Lemma 4.11. *For any $i \in [k+1]$, we have $|V_i| \geq n - (i-1) \cdot \frac{n}{2k}$.*

Proof. The proof is by induction on i. For $i = 1$, we simply have $V_1 = V$ and thus $|V_1| = n$; hence, the base case holds. For the inductive step, it suffices to show that at most $\frac{n}{2k}$ vertices are removed after every player. By Lemma 4.3, for which we verified the hypotheses in Lemma 4.9, we have that M_i^* satisfies

$$\left|E_{\texttt{Miss}}(M_i^*)\right| \leq \frac{\ln 2 \cdot (s+1)}{p_i}.$$

Recall that V_{i+1} is the set of vertices with degree at most d_{i+1} in $G_{\texttt{Miss}}(M_i^*)$. Since any vertex in $V_i \setminus V_{i+1}$ contributes at least d_{i+1} edges to $E_{\texttt{Miss}}(M_i^*)$ (and each edge can be contributed at most twice), we have,

$$\left|E_{\texttt{Miss}}(M_i^*)\right| \geq \frac{1}{2} \cdot |V_i \setminus V_{i+1}| \cdot d_{i+1},$$

implying that

$$|V_i \setminus V_{i+1}| \leq \frac{2\ln 2 \cdot (s+1)}{p_i \cdot d_{i+1}} = \frac{n}{2k},$$

by the choice of p_i and d_i in Equation (4). ∎ Lemma 4.11

We now formalize the idea we alluded to after defining the missing graph in Definition 4.2, where we described how only the edges appearing in the missing graph can have the same color assigned to both endpoints. Some extra care is needed here to account for the fact that the players have their own compression algorithm which is defined based on the messages of previous players.

Lemma 4.12. *For any two vertices $u, v \in V_{k+1}$ that have the same color in the output of P_k, the edge (u,v) exists in $E_{\texttt{Miss}}(M_k^*)$.*

Proof. Suppose toward a contradiction that $(u,v) \notin E_{\texttt{Miss}}(M_k^*)$. We first show that there exists i such that $(u,v) \notin E_{\texttt{Miss}}(M_i^*)$ and $(u,v) \in E_{\texttt{Base}}(i)$. Recalling that $E_{\texttt{Miss}}(M_k^*) \subseteq \cdots \subseteq E_{\texttt{Miss}}(M_1^*)$, either $(u,v) \notin E_{\texttt{Miss}}(M_1^*)$, in which case taking $i = 1$ suffices, or $(u,v) \in E_{\texttt{Miss}}(M_{i-1}^*) \setminus E_{\texttt{Miss}}(M_i^*)$ for some $i > 1$. Referencing the containments illustrated in Figure 3, either $(u,v) \in E_{\texttt{Base}}(i)$ or $(u,v) \in E_{\texttt{Miss}}(M_{i-1}^*) \setminus E_{\texttt{Base}}(i)$. By Line (*ii*)c of the construction, the second case happens only when u or v is dropped when restricting to V_i, which is impossible because $u, v \in V_{k+1} \subseteq V_i$, so $(u,v) \in E_{\texttt{Base}}(i)$ as desired.

Because $(u,v) \notin E_{\texttt{Miss}}(M_i^*)$ and $(u,v) \in E_{\texttt{Base}}(i)$, there should exists some graph $G_i' \in \text{supp}(\mathbb{G}_i)$ that contains the edge (u,v) and is mapped to M_i^* by player P_i, i.e., $\Phi_i(G_i') = M_i^*$. Consider giving the graphs $G_1, \ldots, G_{i-1}, G_i', G_{i+1}, \ldots, G_k$ as input to the players $P_1, \ldots, P_k$, respectively. Because the same messages are generated as in the original construction, P_k also outputs the same coloring. But now (u,v) is in the input graph, so u and v should be colored differently. ∎ Lemma 4.12

In this final lemma, we bound the number of colors that can be used by player P_k to color G.

Lemma 4.13. *Player P_k requires*

$$c \geq \frac{n^2}{16\ln 2 \cdot (s+1)} \cdot p_k$$

colors to color the graph G constructed by the adversary above.

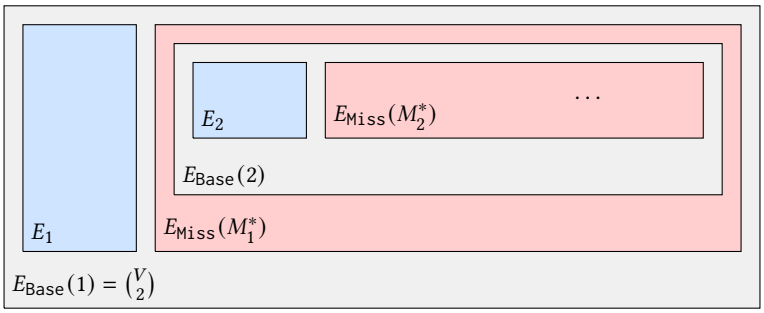


Figure 3: An illustration of edge set containments in the adversary construction.

Proof. Consider the number of pairs of vertices in V_{k+1} that are assigned the same color by the proper c-coloring created by player P_k. Because V_{k+1} has at least $\frac{n}{2}$ vertices by Lemma 4.11, the number of pairs is at least $\frac{n^2}{16c}$ by Proposition 3.2. (We have $c \leq \frac{n}{2}$ by the choice of $s > n$.) At the same time, the number of pairs of vertices that can be colored the same is at most $|E_{\texttt{Miss}}(M_k^*)|$ by Lemma 4.12, which by Lemma 4.3 is at most $\frac{\ln 2 \cdot (s+1)}{p_k}$. In conclusion,

$$\frac{n^2}{16c} \leq \frac{\ln 2 \cdot (s+1)}{p_k},$$

which rearranges to our desired bound. ∎ Lemma 4.13

Finally, by plugging in the explicit value of p_k in Claim 4.8 in the bounds of Lemma 4.13, we have that the minimum number of colors c used by the protocol is at least

$$\begin{aligned} c &\geq \frac{n^2}{16 \ln 2 \cdot (s+1)} \cdot \frac{\Delta}{2k \cdot n} \cdot \left(\frac{n \cdot \Delta}{2 \ln 2 \cdot (s+1) \cdot (2k)^2}\right)^{(k-1)} \\ &= \frac{k}{4} \cdot \left(\frac{n \cdot \Delta}{2 \ln 2 \cdot (s+1) \cdot (2k)^2}\right)^{k} \\ &\geq \left(\frac{1}{\eta_0 \cdot k}\right)^{2k} \cdot \left(\frac{n \cdot \Delta}{s}\right)^{k}, \end{aligned}$$

for some absolute constant $\eta_0 < 100$. This concludes the proof of Theorem 1.

5 THE ALGORITHMS

We present our algorithmic results in this section that complement our strong lower bound for single-pass algorithms. Our first algorithm achieves an $O(\Delta^2)$-coloring in only two passes.

Theorem 2. *There exists a deterministic algorithm that given any n-vertex graph G with maximum degree Δ presented in an insertion-only stream, can find an $O(\Delta^2)$-coloring of G in two passes and $O(n \log n)$ bits of space.*

Our second algorithm builds on the ideas developed for the first one and reduces the number of colors to $O(\Delta)$, at the cost of increasing the number of passes to $O(\log \Delta)$.

Theorem 3. *There exists a deterministic algorithm that given any n-vertex graph G with maximum degree Δ presented in an insertion-only stream, can find an $O(\Delta)$-coloring of G in $O(\log \Delta)$ passes and $O(n \log n)$ bits of space.*

Remark 5.1. We prove Theorem 3 with a leading constant of 6 in the $O(\Delta)$-coloring, i.e., obtain a 6Δ-coloring of the graph in $O(\log \Delta)$ passes. This constant is in no way sacrosanct and is simply chosen to eliminate the need to consider any corner cases (e.g., n not being a power of 2 or Δ not being a prime and alike – this will become clear from the algorithm). In fact, a somewhat more detailed analysis of the same approach gives a $(1+\varepsilon) \cdot \Delta$ coloring algorithm in $O(\log \Delta)$ passes for any constant $\varepsilon > 0$. However, as this result is not the main focus of our work, we opted to present the proof of the most direct approach in this theorem for brevity and showcasing the main idea.

In the following, we first present two families of coloring functions that create few monochromatic edges in different settings, needed for our algorithms, and then present each of our algorithms. Further extensions of our results such as to dynamic streams are presented at the end of this section. These results collectively formalize Result 2.

5.1 Families of Coloring Functions with Few Monochromatic Edges

We start with the following simple result that shows existence of a fixed family of Δ-coloring functions that allows for coloring any graph G with $O(n)$ monochromatic edges via at least one of the functions in the family. We use this result in our two-pass algorithm.

Lemma 5.2. *For any integers $n, \Delta \geq 1$, there exists a family $\mathcal{C} := \mathcal{C}(n, \Delta)$ of size at most $(2n)$ consisting of Δ-coloring functions such that for any n-vertex graph $G = (V, E)$ with maximum degree Δ, there is a coloring function $C \in \mathcal{C}$ such that G has at most $(4n)$ monochromatic edges under C. Moreover, each function in $\mathcal{C}$ can be generated via $O(\log n)$ bits.*

Proof. The proof is by a probabilistic method. Let p be the smallest prime number larger than n and note that we have $p < 2n$ by Bertrand's postulate. We simply pick $\mathcal{C}$ to be the following standard family of near-universal hash functions:

$$\{C_a(v) = ((a \cdot v \bmod p) \bmod \Delta) + 1 \; \forall v \in V \mid a \in \{0, 1, \ldots, p-1\}\}.$$

As such, since $\mathcal{C}$ is a near-universal hash family, for any two vertices $u, v \in V$, we have,

$$\Pr_{C \in \mathcal{C}}(C(u) = C(v)) \leq \frac{2}{\Delta}. \tag{5}$$

See the full version of the paper for the standard that proves Equation (5). Using Equation (5), for any graph G, we have,

$$\mathop{\mathbb{E}}_{C \in \mathcal{C}}\left[\#\text{ of monochromatic edges of } G \text{ under } C\right] = \sum_{(u,v)\in E} \Pr_{C\in\mathcal{C}}(C(u) = C(v)) \leq 2n\Delta \cdot \frac{2}{\Delta} = 4n.$$

Consequently, for any given graph G, there should exist a choice of $C \in \mathcal{C}$ with at most $(4n)$ monochromatic edges. Finally, any coloring function in $\mathcal{C}$ is specified uniquely by an integer in $\{0, 1, \ldots, p-1\}$ which requires $O(\log n)$ bits to store. ∎ Lemma 5.2

We next present our second family of functions which is used in our $O(\Delta)$ coloring algorithm.

Definition 5.3. Let $C_1 : V \to [c] \cup \{\perp\}$ be a partial c-coloring function of a graph $G = (V, E)$ that has no monochromatic edges. Let $C_2 : V \to [c]$ be a c-coloring function of V (not necessarily a proper one). We define the **extension of** C_1 **by** C_2 as the c-coloring function $C_3 : V \to [c]$ such that for any $v \in V$,

$$C_3(v) = \begin{cases} C_1(v) & \text{if } C_1(v) \neq \perp \\ C_2(v) & \text{otherwise} \end{cases},$$

i.e., C_3 uses C_1 to color vertices v with $C_1(v) \neq \perp$ and use C_2 to color the remaining vertices.

The following family of coloring functions has the following property: for any graph G and a partial coloring C_1 of G, there is a coloring function C in the family with a small number of monochromatic edges in the extension of C_1 by C. Formally,

Lemma 5.4. *For any $n, \Delta \geq 1$, there is a family $\mathcal{C}^\star := \mathcal{C}^\star(n, \Delta)$ of size at most $(2n)$ consisting of (6Δ)-coloring functions such that the following is true. For any n-vertex graph $G = (V, E)$ with maximum degree Δ and any partial coloring function C_1 of G with no monochromatic edges, there is a (6Δ)-coloring function $C \in \mathcal{C}^\star$ such that extension of C_1 by C has $\leq (\frac{n_0}{3})$ monochromatic edges where $n_0 := |\{v \in V \mid C_1(v) = \perp\}|$, is the number of uncolored vertices by C_1. Each function in $\mathcal{C}^\star$ can be generated via $O(\log n)$ bits.*

Proof. The proof is again by the probabilistic method similar to that of Lemma 5.2. Let p be the smallest prime number larger than n and note that we have $p < 2n$ by Bertrand's postulate. We pick $\mathcal{C}^\star$ to be the following family of near-universal hash functions:

$$\{C_a(v) = ((a \cdot v \bmod p) \bmod 6\Delta) + 1 \;\forall v \in V \mid a \in \{0, \ldots, p-1\}\}.$$

Since $\mathcal{C}^\star$ is a near-universal hash family, for any two vertices $u, v \in V$ and any fixed color $c \in [6\Delta]$,

$$\Pr_{C_2 \in \mathcal{C}^\star}(C_2(u) = C_2(v)) \leq \frac{2}{6\Delta} = \frac{1}{3\Delta}$$
$$\Pr_{C_2 \in \mathcal{C}^\star}(C_2(u) = c) \leq \frac{2}{6\Delta} = \frac{1}{3\Delta}. \tag{6}$$

The proof is similar to Equation (5) (see the full version).

For any edge $(u, v) \in E$ to be monochromatic in the *extension* C_3 of C_1 by C_2, we should have that at least one of $C_1(u)$ or $C_1(v)$ is $\perp$; otherwise, both retain u and v their colors in C_1 which contains no monochromatic edges. By symmetry suppose $C_1(u) = \perp$ and so u will be colored by C_2 in the extension C_3. If $C_1(v) = \perp$ also, then to get a monochromatic edge, we need $C_2(u) = C_2(v)$ which happens with probability at most $1/3\Delta$ by the first part of Equation (6). Conversely, if $C_1(v) \neq \perp$, then to get a monochromatic edge, we need $C_2(u) = C_1(v)$ which again happens with probability at most $1/3\Delta$ by the second part of Equation (6). All in all, only edges incident on $\{v \in V \mid C_1(v) = \perp\}$ can be monochromatic and each one will become so with probability at most $1/3\Delta$. Hence,

$$\mathop{\mathbb{E}}_{C_2 \in \mathcal{C}^\star}\left[\#\text{ of monochromatic edges of } G \text{ in extension of } C_1 \text{ by } C_2\right] \leq \sum_{v: C_1(v) = \perp} \sum_{u \in N(v)} \frac{1}{3\Delta} = n_0 \cdot \Delta \cdot \frac{1}{3\Delta} = \frac{n_0}{3}.$$

Hence, for any G and C_1, there should exist a choice of $C_2 \in \mathcal{C}$ with at most $(n_0/3)$ monochromatic edges in the extension of C_1 by C_2. Also, any coloring function in $\mathcal{C}$ is specified uniquely by an integer in $\{0, 1, \ldots, p-1\}$ which requires $O(\log n)$ bits to store, concluding the proof. ∎ Lemma 5.4

5.2 A Two-Pass $O(\Delta^2)$-Coloring Algorithm

We now present our two-pass semi-streaming algorithm for $O(\Delta^2)$ coloring and prove Theorem 2. The key tool we use in this result is the coloring functions of Lemma 5.2.

Algorithm 1. A two-pass deterministic semi-streaming algorithm for $O(\Delta^2)$ coloring.

(*i*) Let $\mathcal{C} = \mathcal{C}(n, \Delta) = \{C_1, \ldots, C_k\}$ be the family of Δ-coloring functions guaranteed by Lemma 5.2 for some $k \leq 2n$.

(*ii*) In the first pass, for any $i \in [k]$, maintain a *counter* ϕ_i that counts the number of monochromatic edges of G under the coloring C_i, i.e.,

$$\phi_i = |\{(u, v) \in G \mid C_i(u) = C_i(v)\}|.$$

Let $C_{i^\star} \in \mathcal{C}$ be the coloring function with the smallest value of $\phi_{i^\star}$, i.e., $i^\star \in \arg\min_{i \in [k]} \phi_i$.

(*iii*) In the second pass, store all monochromatic edges of G under $C_{i^\star}$. Compute a $(\Delta + 1)$ coloring C of the stored edges and return the following coloring function $C^\star$ as the answer:

$$\text{for all } v \in V\text{: } C^\star(v) = (C_{i^\star}(v) - 1) \cdot (\Delta + 1) + C(v).$$

Lemma 5.5. *Space complexity of Algorithm 1 is $O(n \log n)$ bits.*

Proof. The first pass of this algorithm requires storing $O(n)$ counters of size $O(\log n)$ bits each, and can be implemented in $O(n \log n)$ bits of space. The second pass requires storing only $O(n)$ edges by the guarantee of Lemma 5.2 which again can be done in $O(n \log n)$ bits of space. ∎ Lemma 5.5

We now argue that the final coloring $C^\star$ returned by the algorithm is a proper coloring of G, i.e., it does not contain any monochromatic edges.

Lemma 5.6. *Algorithm 1 always outputs a proper $O(\Delta^2)$ coloring of any given input graph with maximum degree Δ.*

Proof. Firstly, since maximum degree of G is Δ, we clearly have that maximum degree of stored edges is also at most Δ, and consequently, the algorithm can always find a $(\Delta+1)$ coloring of the stored edges. For any edge $(u,v) \in G$, if $C_{i^\star}(u) \neq C_{i^\star}(v)$,

$$\begin{aligned}\left|C^\star(u) - C^\star(v)\right| &\geq |C_{i^\star}(u) - C_{i^\star}(v)| \cdot (\Delta+1) - |C(u) - C(v)| \\ &\geq (\Delta+1) - \Delta = 1,\end{aligned}$$

thus $C^\star(u) \neq C^\star(v)$ and so (u,v) will not be monochromatic. For any edge $(u,v) \in G$ with $C_{i^\star}(u) = C_{i^\star}(v)$, the algorithm stores (u,v) in the second pass and thus by the coloring it finds, we have $C(u) \neq C(v)$, making $C^\star(u) \neq C^\star(v)$ also.

Finally, since the total number of colors used by $C^\star$ is $\Delta \cdot (\Delta+1)$, we obtain an $O(\Delta^2)$ coloring as desired. ∎ Lemma 5.6

This concludes the proof of Theorem 2.

5.3 An $O(\log \Delta)$-Pass $O(\Delta)$-Coloring Algorithm

This section includes our $O(\log \Delta)$-pass semi-streaming algorithm for $O(\Delta)$ coloring, i.e., the proof of Theorem 3. The key tool we use in this result is the coloring functions of Lemma 5.4.

Algorithm 2. An $O(\log \Delta)$-pass deterministic semi-streaming algorithm for (6Δ) coloring.

(*i*) Let $\mathcal{C}^\star = \mathcal{C}^\star(n,\Delta) = \{C_1, \ldots, C_k\}$ be the family of (6Δ)-coloring functions guaranteed by Lemma 5.4 for some $k \leq 2n$.

(*ii*) Let C be a partial coloring function, initially set to map all vertices to $\perp$.

(*iii*) While C has more than n/Δ uncolored vertices:

(a) In one pass, for any $i \in [k]$, maintain a *counter* ϕ_i that counts the number of monochromatic edges of G under the extension C'_i of C by C_i, i.e.,

$$\phi_i = \left|\left\{(u,v) \in G \mid C'_i(u) = C'_i(v)\right\}\right|.$$

Let $C_{i^\star} \in \mathcal{C}$ be the coloring function with the smallest $\phi_{i^\star}$, i.e., $i^\star \in \arg\min_{i \in [k]} \phi_i$ and $C'_{i^\star}$ be the extension of C by $C_{i^\star}$.

(b) In another pass, store all monochromatic edges of G under $C'_{i^\star}$. For any vertex $v \in V$, if no monochromatic edges incident on v are stored, then set $C(v) = C'_{i^\star}(v)$.

(*iv*) In the last pass, store all edges incident on the uncolored vertices of C. Greedily color all the remaining uncolored vertices with a color not assigned to their neighbors.

We first note a direct invariant of the algorithm that will be used in our analysis.

Lemma 5.7. *At any point of time in Algorithm 2, there are no monochromatic edges between vertices colored by C.*

Proof. This is simply because we always work with the extensions of C and thus if a vertex is colored by C, we never change its color, and since we only color a vertex by C if it does not have any monochromatic edges. ∎ Lemma 5.7

Note that *if* the while-loop finishes, then the coloring C computed greedily by the algorithm is a proper (6Δ) coloring of G as C contained no monochromatic edges throughout (by Lemma 5.7), and the last step of using greedy coloring, only requires $(\Delta+1)$ colors since we have stored all edges incident on uncolored vertices. We thus want to show that the while-loop indeed finishes. This is the main part of the analysis.

Lemma 5.8. *There are $O(\log \Delta)$ iterations of the while-loop in Algorithm 2 before it terminates.*

Proof. Fix an iteration of the while-loop and let

$$n_0 := |\{v \in V \mid C(v) = \perp\}|$$

denote the number of uncolored vertices by C at the beginning of this iteration. By the guarantee of Lemma 5.4 (and since Lemma 5.7 verifies the hypothesis of this lemma), we know that the coloring $C'_{i^\star}$ computed by the algorithm in this iteration has at most $n_0/3$ monochromatic edges. This means that at least $n_0 - 2n_0/3 = n_0/3$ vertices not colored by C have zero monochromatic edges under $C'_{i^\star}$. All these vertices will be colored by C at the end of this iteration.

By the above discussion, the number of uncolored vertices reduces by a factor of at most $2/3$ in each iteration. As a result, after $O(\log \Delta)$ iterations, the number of uncolored vertices by C drops below n/Δ and thus the while-loop terminates. ∎ Lemma 5.8

Finally, we analyze the space complexity of the algorithm.

Lemma 5.9. *Space complexity of Algorithm 2 is $O(n \log n)$ bits.*

Proof. The first pass of each iteration of while-loop of Algorithm 2 requires maintaining $O(n)$ counters of size $O(\log n)$ bits each, and can be implemented in $O(n \log n)$ bits of space. The second pass requires storing only $O(n)$ edges by the guarantee of Lemma 5.4 (and since Lemma 5.7 verifies the hypothesis of this lemma) which again can be done in $O(n \log n)$ bits of space. Finally, at the end we are storing at most Δ edges for each of the remaining n/Δ uncolored vertices and thus we can store them in $O(n \log n)$ bits as well. ∎ Lemma 5.9

This concludes the proof of Theorem 3.

5.4 Further Extensions: Dynamic Streams and Unknown Δ

Dynamic streams. In order to implement our algorithms in dynamic streams, we simply need a way of recovering the $O(n)$ monochromatic edges in each step of each one. (Maintaining the counters is straightforward by simply adding and subtracting their values based on insertion and deletion of monochromatic edges – recall that we already know the coloring we need to work with and thus upon update of an edge, we know whether or not it is a monochromatic edge).

To recover these $O(n)$ monochromatic edges, we can simply use any standard **deterministic sparse recovery** algorithm [26] over

dynamic streams. The following result is folklore (we provide a short proof sketch for completeness).

Proposition 5.10 (Folklore; cf. [21, 26]). *There exists a deterministic algorithm that given an integer $k \geq 1$ and a dynamic stream of of edge insertions and deletions that define an n-vertex graph $G = (V, E)$, uses $O(k \cdot \log n)$ bits of space and at the end of the stream recovers all edges under the promise that G has at most k edges (the answer can be arbitrary if the promise is not satisfied).*

Proof Sketch. Consider the characteristic vector $x \in \{0,1\}^{\binom{n}{2}}$ of edges of G being updated in the stream. The promise of the proposition is that x is going to be k-sparse.

Let p be a prime larger than $m := \binom{n}{2}$ and consider the field $\mathbb{F}_p$. Let A be the (transpose) of Vandermonde matrix over $\mathbb{F}_p$ defined as follows:

$$A := \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2^1 & 3^1 & \cdots & m^1 \\ 1 & 2^2 & 3^2 & \cdots & m^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{2k-1} & 3^{2k-1} & \cdots & m^{2k-1} \end{bmatrix}$$

where or for all $i \in [2k]$ and $j \in m$, we set $A_{i,j} = j^{i-1} \bmod p$. Given that A can be described explicitly using only $O(\log m)$ bits, we can maintain $y = A \cdot x$ throughout the stream by linearity: for any update to the i-th entry of x, we simply add or subtract the i-th column of A from y to have the updated value of $A \cdot x$.

We now argue that at the end of the stream, one can recover x from $A \cdot x$. Consider any two k-sparse vectors $x_1 \neq x_2 \in \mathbb{F}_p^m$. We argue that $A \cdot x_1 \neq A \cdot x_2$. Suppose not, then we have $A \cdot (x_1 - x_2) = 0$. This in turn means that the at most $2k$ columns of A corresponding to the union of the support of x_1 and x_2 are *not* linearly independent. But this is a contradiction as in the Vandermonde matrix, every $2k$ columns are linearly independent. This already implies that unique recovery of x is possible from $A \cdot x$ for any k-sparse vector $x \in \mathbb{F}_p^{\binom{n}{2}}$. Finally, one can also use *syndrome decoding* from coding theory to implement this decoding step in polynomial time. We refer the interested reader to, e.g. [21], for more details. ∎

As in Algorithm 1 and Algorithm 2, we need to find monochromatic edges that are guaranteed to be at most $O(n)$ many, we can simply use Proposition 5.10 to recover these edges in $O(n \log n)$ bits even in dynamic streams (we only need to define the underlying graph as insertions and deletions *between* monochromatic pairs and set $k = O(n)$ and apply the proposition).

This immediately extends both our Theorems 2 and 3 to dynamic streams with the same asymptotic space complexity and the same exact number of passes.

Removing the knowledge of Δ. Our algorithms in the previous part are described assuming the knowledge of Δ. For our $O(\Delta)$ coloring algorithm this is simply without loss of generality as we can increase the number of passes by one and compute Δ in the first pass—given that we report the number of passes asymptotically anyway, this does not change anything. But the same approach for our $O(\Delta^2)$ coloring algorithm increases the number of passes to three instead.

Nevertheless, there is a simple way to fix $O(\Delta^2)$ coloring algorithm without changing the number of passes. In the first pass, pick $O(\log n)$ choices of for Δ in geometrically increasing values and maintain the counters for $C(n, \cdot)$ for these $O(\log n)$ choices; in parallel, also compute Δ in this pass. At the end of the first pass, we know Δ and can focus on the right choice of counters for $C(n, \Delta')$ where $\Delta' \geq \Delta \geq \frac{1}{2} \cdot \Delta'$. The rest of the algorithm and its proof are exactly as before.

ACKNOWLEDGEMENT

Sepehr Assadi would like to thank Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl for helpful discussions regarding [15] and its connection to our work. We thank the organizers of DIMACS REU in Summers 2020 and 2021, in particular Lazaros Gallos, for making this collaboration possible and all their help and encouragements along the way. We are also thankful to the anonymous reviewers of STOC 2022 for their helpful comments.

REFERENCES

[1] Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Ami Paz. 2019. Smaller Cuts, Higher Lower Bounds. *CoRR* abs/1901.01630 (2019).
[2] Noga Alon and Sepehr Assadi. 2020. Palette Sparsification Beyond (Δ + 1) Vertex Coloring. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference (LIPIcs, Vol. 176)*, Jaroslaw Byrka and Raghu Meka (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:22.
[3] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Sublinear Algorithms for (Δ + 1) Vertex Coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. 767–786.
[4] Sepehr Assadi and Aditi Dudeja. 2021. Ruling Sets in Random Order and Adversarial Streams. In *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference) (LIPIcs, Vol. 209)*, Seth Gilbert (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:18.
[5] Sepehr Assadi and Aditi Dudeja. 2021. A Simple Semi-Streaming Algorithm for Global Minimum Cuts. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, Hung Viet Le and Valerie King (Eds.). SIAM, 172–180.
[6] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2016. Tight bounds for single-pass streaming complexity of the set cover problem. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. 698–711.
[7] Sepehr Assadi, Pankaj Kumar, and Parth Mittal. 2022. Brooks' Theorem in Graph Streams: A Single-Pass Semi-Streaming Algorithm for Δ-Coloring. *CoRR* abs/2203.10984. In STOC 2022. (2022).
[8] Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Marina Knittel, and Hamed Saleh. 2019. Streaming and Massively Parallel Algorithms for Edge Coloring. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany (LIPIcs, Vol. 144)*, Michael A. Bender, Ola Svensson, and Grzegorz Herman (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 15:1–15:14.
[9] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. 2020. A Framework for Adversarially Robust Streaming Algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, Dan Suciu, Yufei Tao, and Zhewei Wei (Eds.). ACM, 63–80.
[10] Suman K. Bera, Amit Chakrabarti, and Prantar Ghosh. 2020. Graph Coloring via Degeneracy in Streaming and Other Space-Conscious Models. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*. 11:1–11:21.
[11] Suman Kalyan Bera and Prantar Ghosh. 2018. Coloring in Graph Streams. *CoRR* abs/1807.07640 (2018).
[12] Anup Bhattacharya, Arijit Bishnu, Gopinath Mishra, and Anannya Upasana. 2021. Even the Easiest(?) Graph Coloring Problem Is Not Easy in Streaming!. In *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8,*

2021, Virtual Conference (LIPIcs, Vol. 185), James R. Lee (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 15:1–15:19.

[13] Rowland Leonard Brooks. 1941. On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 37. Cambridge University Press, 194–197.

[14] Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. 2020. Derandomizing local distributed algorithms under bandwidth restrictions. *Distributed Comput.* 33, 3-4 (2020), 349–366.

[15] Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. 2022. Adversarially Robust Coloring for Graph Streams. 215 (2022), 37:1–37:23.

[16] Graham Cormode, Jacques Dark, and Christian Konrad. 2018. Approximating the Caro-Wei Bound for Independent Sets in Graph Streams. In *Combinatorial Optimization - 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11-13, 2018, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 10856)*, Jon Lee, Giovanni Rinaldi, and Ali Ridha Mahjoub (Eds.). Springer, 101–114.

[17] Graham Cormode, Jacques Dark, and Christian Konrad. 2019. Independent Sets in Vertex-Arrival Streams. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 45:1–45:14.

[18] Artur Czumaj, Peter Davies, and Merav Parter. 2020. Simple, Deterministic, Constant-Round Coloring in the Congested Clique. In *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, Yuval Emek and Christian Cachin (Eds.). ACM, 309–318.

[19] Artur Czumaj, Peter Davies, and Merav Parter. 2021. Graph Sparsification for Derandomizing Massively Parallel Computation with Low Space. *ACM Trans. Algorithms* 17, 2 (2021), 16:1–16:27.

[20] Artur Czumaj, Peter Davies, and Merav Parter. 2021. Improved Deterministic (Δ + 1) Coloring in Low-Space MPC. In *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen (Eds.). ACM, 469–479.

[21] Abhik Kumar Das and Sriram Vishwanath. 2013. On finite alphabet compressive sensing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 5890–5894.

[22] Devdatt P. Dubhashi and Alessandro Panconesi. 2009. **Concentration of Measure for the Analysis of Randomized Algorithms**. Cambridge University Press.

[23] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2005. On graph problems in a semi-streaming model. *Theor. Comput. Sci.* 348, 2-3 (2005), 207–216.

[24] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2008. Graph Distances in the Data-Stream Model. *SIAM J. Comput.* 38, 5 (2008), 1709–1727.

[25] Mohsen Ghaffari and Fabian Kuhn. 2020. Deterministic Distributed Vertex Coloring: Simpler, Faster, and without Network Decomposition. *CoRR* abs/2011.04511. To appear in FOCS 2021 (2020).

[26] Anna C. Gilbert and Piotr Indyk. 2010. Sparse Recovery Using Sparse Matrices. *Proc. IEEE* 98, 6 (2010), 937–947.

[27] Sudipto Guha, Andrew McGregor, and David Tench. 2015. Vertex and Hyperedge Connectivity in Dynamic Graph Streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. 241–247.

[28] Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. 2016. Streaming Algorithms for Independent Sets in Sparse Hypergraphs. *Algorithmica* 76, 2 (2016), 490–501.

[29] Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. 2012. Streaming and Communication Complexity of Clique Approximation. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 7391)*, Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer (Eds.). Springer, 449–460.

[30] Christian Konrad, Sriram V. Pemmaraju, Talal Riaz, and Peter Robinson. 2019. The Complexity of Symmetry Breaking in Massive Graphs. In *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary (LIPIcs, Vol. 146)*, Jukka Suomela (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 26:1–26:18.

[31] Andrew McGregor. 2014. Graph stream algorithms: a survey. *SIGMOD Rec.* 43, 1 (2014), 9–20.

[32] Merav Parter. 2018. (Δ + 1) Coloring in the Congested Clique Model. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic (LIPIcs, Vol. 107)*, Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 160:1–160:14.

[33] Ami Paz and Gregory Schwartzman. 2017. A (2+ε)-Approximation for Maximum Weight Matching in the Semi-Streaming Model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, Philip N. Klein (Ed.). SIAM, 2153–2161.

[34] Andrew Chi-Chih Yao. 1977. Probabilistic Computations: Toward a Unified Measure of Complexity (Extended Abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*. IEEE Computer Society, 222–227.