Streaming Algorithms for Conflict-Free Coloring

Rogers Mathew
Department of Computer Science and Engineering, IIT Hyderabad, India

Fahad Panolan
School of Computer Science, University of Leeds, UK

Seshikanth
Department of Computer Science and Engineering, IIT Hyderabad, India

Abstract

Conflict-free coloring of a hypergraph $\mathcal{H} = (V, \mathcal{E})$ using k colors is a function $f : V \to \{1, 2, \ldots, k\}$ such that for all $E \in \mathcal{E}$, there exists a vertex $v \in E$ with a unique color. That is, $f(v) \neq f(u)$ for all $u \in E \setminus \{v\}$. The minimum k for which $\mathcal{H}$ has a conflict-free coloring using k colors is called the *conflict-free chromatic number* of $\mathcal{H}$. For a simple graph G, a conflict-free coloring of the hypergraph with vertex set $V(G)$ and edge set being the set of all closed neighborhoods of the vertices in G is called a *conflict-free closed neighborhood (CFCN) coloring* of G. *CFCN chromatic number*, denoted by $\chi_{CN}(G)$, is the minimum number of colors used in a conflict-free closed neighborhood coloring of G. Analogously, we define conflict-free open neighborhood (CFON) coloring and CFON chromatic number, $\chi_{ON}(G)$, of a graph G.

There are various works on proving upper and lower bounds of $\chi_{ON}(G)$ and $\chi_{CN}(G)$. In this work, we develop streaming algorithms for CFCN and CFON coloring of a graph where the number of colors used matches the best-known upper bounds of $\chi_{ON}(G)$ and $\chi_{CN}(G)$. Our algorithms use as input an edge stream of the graph G in the insertion-only model. Our results and the best-known bounds for $\chi_{ON}(G)$ and $\chi_{CN}(G)$ are given below.

1. Pach and Tardos [Combinatorics, Probability and Computing, 2009] showed that, for any n vertex graph G, $\chi_{CN}(G) = O(\ln^2 n)$. Glebov, Szabó and Tardos [Combinatorics, Probability and Computing, 2014] showed the existence of graphs G with $\chi_{CN}(G) = \Omega(\ln^2 n)$. We design a randomized single-pass semi-streaming algorithm (i.e., it uses $O(n \ln n)$ space[1] that, given an n-vertex graph G, outputs a CFCN coloring of G using $O(\ln^2 n)$ colors with probability at least $(1 - \frac{2}{n})$.
2. Bhyravarapu, Kalyanasundaram, Mathew [Journal of Graph Theory, 2021] showed that for a graph G with maximum degree Δ, $\chi_{CN}(G) = O(\ln^2 \Delta)$. The methods used by our algorithms give rise to a simpler, alternate proof for this bound.
3. It is known that $\chi_{ON}(G) \leq 1/2 + \sqrt{2n + 1/4}$ (See Pach and Tardos [Combinatorics, Probability and Computing, 2009] and Ph.D. thesis of Cheilaris). This bound is asymptotically tight.
   - We design a deterministic single-pass $O(n\sqrt{n})$ space streaming algorithm that, given a graph G on n vertices, finds a CFON coloring using $2\sqrt{n}$ colors.
   - We design a randomized, single-pass, semi-streaming algorithm to find a CFON coloring of a graph G using $O(\sqrt{n} \ln^2 n)$ colors with success probability at least $(1 - \frac{2}{n})$.
4. It is known that $\chi_{ON}(G) \leq \Delta + 1$, where Δ is the maximum degree of a vertex in G. Further, there are graphs G known with $\chi_{ON}(G) = \Delta + 1$. We design a randomized two-pass semi-streaming algorithm (uses $O(\frac{1}{\epsilon^2} n \ln^3 n)$ space) that outputs a CFON coloring of G using $(1+\epsilon)\Delta$ colors, for any $\epsilon > 0$, with a probability at least $(1 - \frac{1}{n})$.

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

Keywords and phrases Streaming algorithm, conflict-free coloring, vertex coloring, randomized algorithms

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.44

[1] The space complexity is in terms of number of words.

1 Introduction

Conflict-Free coloring (or CF coloring) of a hypergraph is a coloring of its vertices in which each hyperedge sees some color exactly once. The minimum number of colors required for the CF coloring of a hypergraph $\mathcal{H}$, denoted by $\chi_{CF}(\mathcal{H})$, is called the *conflict-free chromatic number* of $\mathcal{H}$. Motivated by its application in a frequency assignment problem in cellular networks, conflict-free coloring was introduced in 2002 by Even, Lokter, Ron, and Smorodinsky [17].

Cellular networks consist of base stations and mobile agents. Base stations, represented as vertices, are stationary and serve as the network's core, while mobile agents are clients served by base stations. Each base station operates on a fixed frequency, represented by a color. To establish a connection with a base station, an agent's device must tune itself to that base station's frequency. Agents can be within the range of multiple base stations. The range of communication for each agent is defined by a hyperedge, which represents the set of base stations it is capable of communicating with. To prevent interference, frequencies must be assigned to base stations in a conflict-free manner as described below. Every mobile agent should be able to find some base station in its vicinity that operates in a frequency that is distinct from that of all the other base stations in its vicinity. In combinatorial terms, this is about coloring the base stations in such a way that every agent sees a base station of a unique color in the hyperedge associated with it. While assigning n different frequencies to n base stations can solve the problem, it is expensive. Therefore, a scheme that allows the reusing of frequencies, whenever possible is preferred to minimize expenses.

Recently, conflict-free coloring and its variants were used in [19] to get improved results on a problem in coding theory, namely the Pliable Index Coding problem. Also, CF coloring has found applications in battery consumption analysis in sensor networks, in certain problems related to RFID protocols, in the vertex ranking problem (also known as ordered coloring, which finds applications in various fields, such as VLSI design and operations research), etc. See [14–19, 21, 23–25] for more details.

We define below two popular variants of the conflict-free coloring of graphs that have been extensively studied. Given a graph G, the *open neighborhood* of a vertex v in G, denoted by $N_G(v)$, is the set of vertices adjacent to v in G. The *closed neighborhood* of a vertex v in G, denoted by $N_G[v]$, is defined as $\{v\} \cup N_G(v)$. A coloring of the vertices of G using k colors is a *Conflict-Free Closed Neighborhood coloring* (*CFCN coloring*) if every vertex v sees some color exactly once in its closed neighborhood $N_G[v]$. The minimum k for which such a coloring exists is called the *Conflict-Free Closed Neighborhood chromatic number* (or *CFCN chromatic number*) of G. It is denoted by $\chi_{CN}(G)$. Analogously, one can define Conflict-Free Open Neighborhood coloring (CFON coloring) and Conflict-Free Open Neighborhood chromatic number (CFON chromatic number) of a graph by replacing "closed neighborhood" with "open neighborhood" in the above definitions. It is denoted by $\chi_{ON}(G)$. It is easy to see that for any graph G, $\chi_{CN}(G)$ is at most its chromatic number as in any proper coloring of G, every vertex v sees its color exactly once in its closed neighborhood $N_G[v]$. However, there are graphs G for which $\chi_{ON}(G)$ is arbitrarily larger than its chromatic number. For any positive integer r, Example 4 gives bipartite graphs (and hence its CFCN chromatic number is at most 2) with CFON chromatic number at least r. It is known that for any graph G with maximum degree Δ, (i) $\chi_{CN}(G) = O(\ln^2 \Delta)$ (see [12]), and (ii) $\chi_{ON}(G) \leq \Delta + 1$ (see [23]). Both these bounds are asymptotically tight (see Section 2 for more details).

In this paper, we focus on semi-streaming algorithms (Refer [22] for more information on graph streams) for conflict-free coloring of graphs. We consider only graph streams that are insert-only. Below, we define an *insert-only graph stream*. We consider streams consisting of a sequence of undirected edges $e = (u, v)$, where $u, v \in [n] = \{1, 2, \ldots, n\}$. Such a stream, $S = \langle e_1, e_2, \ldots, e_m \rangle$ naturally defines an undirected graph $G = (V, E)$, where $V = [n]$ and $E = \{e_1, \ldots, e_m\}$. We assume the stream elements are distinct, and therefore, the resulting graph is a simple graph. An algorithm for graphs is a *semi-streaming graph algorithm* if it takes as input a graph stream and does all the computations using $O(n \cdot \text{polylog}(n))$ bits of space, where n is the number of vertices of the graph. The efficiency of a graph algorithm in a streaming model is measured by the space it uses, the time it requires to process each edge, and the number of passes it makes over the graph stream. Extensive research has been conducted on semi-streaming algorithms for proper coloring of graphs, as demonstrated in several notable works [1, 3–5, 8, 9]. Similarly, edge coloring has been thoroughly explored in the context of streaming algorithms, with notable contributions found in [2, 7]. However, to the best of our knowledge, there is no streaming algorithm known for conflict-free coloring of graphs. We describe our results and methods in the section below.

2 Our contributions

First, we state a few results from the literature. Then we explain our results and briefly describe the key ideas involved. Please note that the space used by the algorithms mentioned below is in terms of the number of words.

Auxiliary results from the literature. The following results on conflict-free coloring are due to Pach and Tardos [23].

▶ **Proposition 1** (Theorem 1.1 in [23]). *Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph, where every vertex is present in at most Δ hyperedges. Then $\chi_{CF}(\mathcal{H}) \leq \Delta + 1$. Moreover, there is a deterministic $O(n\Delta)$ time (and hence $O(n\Delta)$ space) algorithm that obtains such a coloring.*

▶ **Proposition 2** (Theorem 1.2 in [23]). *For any positive integers t and Γ, the conflict-free chromatic number of any hypergraph in which each edge is of size at least $2t - 1$ and each edge intersects at most Γ others is $O(t\Gamma^{\frac{1}{t}} \ln \Gamma)$.*

As a corollary of Proposition 2, it is proved that $\chi_{ON}(G) \leq \frac{1}{2} + \sqrt{2n + \frac{1}{4}}$. Cheilaris gave an alternate proof to to this result [14].

▶ **Proposition 3** (Proposition 4.40 in [14]). *For every graph G, $\chi_{ON}(G) \leq 2\sqrt{n}$.*

From Example 4, we infer that the bound in Proposition 3 is asymptotically tight.

▶ **Example 4.** Let K_r^* denote the graph obtained from a complete graph on r vertices by subdividing every edge exactly once. This graph with $\binom{r}{2} + r$ vertices and with a maximum degree of $r - 1$ is known to have a CFON chromatic number of r. Here, subdividing an edge uv means introducing a new vertex w and replacing the edge uv with edges uw and wv.

Our methods and results. Pach and Tardos [23] showed that, for any graph G, $\chi_{CN}(G) = O(\ln^2 n)$ where n is the number of vertices in G. Glebov, Szabó and Tardos [18] showed the existence of graphs G with $\chi_{CN}(G) = \Omega(\ln^2 n)$.

Result 1: There is a randomized, single-pass, $O(n \ln n)$ space streaming algorithm that, given an n-vertex graph G, outputs a CFCN coloring of G using $O(\ln^2 n)$ colors with probability at least $(1 - \frac{2}{n})$.

Our algorithm partitions the vertex set into two parts: Part A, which is made of vertices of degree at least $2c \ln n$ for some constant c, and Part B, which is the set of all the remaining vertices (which are of degree less than $2c \ln n$). Part A is further partitioned into two parts, namely A' and A''. The set A'' is made of all vertices in A that have no neighbor in B. We choose a random coloring for the vertices in A from a geometric distribution. Then, we will show that, with high probability, every vertex in A'' sees some color exactly once in its open and closed neighborhood.

Next, we construct a hypergraph $\mathcal{H} = (B, \mathcal{E})$, where $\mathcal{E} = \{N_G[v] \cap B \ : \ v \in A' \cup B\} \cup \{N_G(v) \cap B \ : \ v \in A' \cup B, N_G(v) \cap B \neq \emptyset)\}$. Observe that the maximum degree of $\mathcal{H}$ is at most $2c \ln n$ and hence can be CF colored by the greedy procedure given in Proposition 1 using $2c \ln n + 1$ colors. Finally, we observe that the union of the two colorings explained above (let this coloring be c) is a CFCN coloring of G and obtain Result 1. Moreover, we observe that for many vertices, there is a unique color in its open neighborhood as well. More specifically, let $B_1 = \{b \in B : N_G(b) \cap B = \emptyset\}$. We prove that for any vertex $v \in V(G) \setminus B_1$, there is a unique colored vertex in $N_G(v)$. Also, notice that for all $b \in B_1$, $|N_G(b)| = O(\ln n)$ and $N_G(b) \subseteq A'$. So in the semi-streaming algorithm, we store the neighborhood information of the vertices in B_1 and obtain a CFON coloring c' of the graph induced on $A' \cup B_1$ using at most $O(\sqrt{n})$ colors. Then, the coloring c_o defined as $c_o(v) = (c(v), c'(v))$ for all v, is a CFON coloring of G using $O(\sqrt{n} \ln^2 n)$ colors. Here, for convenience, assume that $c'(v) = 1$ if $v \notin A' \cup B_1$. This gives us the following result on CFON coloring.

Result 2: There is a randomized, single-pass, semi-streaming algorithm that, given a graph G on n vertices, finds a CFON coloring using $O(\sqrt{n} \ln^2 n)$ colors and in polynomial time with probability at least $(1 - \frac{2}{n})$.

From Proposition 3, we know that $\chi_{ON}(G) \leq 2\sqrt{n}$. This bound is asymptotically tight due to Example 4. Our next result, whose proof has been moved to the appendix due to the paucity of space, is the following.

Result 3: There is a deterministic, single-pass, $O(n\sqrt{n})$ space streaming algorithm that, given a graph G on n vertices, finds a CFON coloring using $2\sqrt{n}$ colors.

For a graph G, with maximum degree Δ, Pach and Tardos [23] in the year 2009 showed that $\chi_{CN}(G) = O(\ln^{2+\epsilon} \Delta)$ for any $\epsilon > 0$. As mentioned above, in 2014, Glebov, Szabó, and Tardos [18] proved that there exist graphs G on n vertices with $\chi_{CN}(G) = \Omega(\ln^2 n)$ (and thereby $\Omega(\ln^2 \Delta)$). In the year 2021, Bhyravarapu, Kalyanasundaram, and Mathew [12] tightened the upper bound to show that $\chi_{CN}(G) = O(\ln^2 \Delta)$. The methods we use in proving Result 1 give us an alternate, shorter proof (which is given below).

Proof of $\chi_{CN}(G) = O(\ln^2 \Delta)$. We partition $V(G) = A \uplus B$, where $A = \{v \in V(G) : deg_G(v) \geq 2t - 1\}$, $B = V(G) \setminus A$ and $t = \ln \Delta$. Further, let $A = A_1 \uplus A_2$ where $A_1 = \{a \in A : (N_G(a) \cap B) \neq \emptyset\}$ and $A_2 = A \setminus A_1$. We define a hypergraph $\mathcal{H}_2 = (A, \mathcal{E}_2)$ where $\mathcal{E}_2 = \{N_G[a] : a \in A_2\}$. From the definition of $A_2, N_G[a] \subseteq A, \forall a \in A_2$. Applying Proposition 2

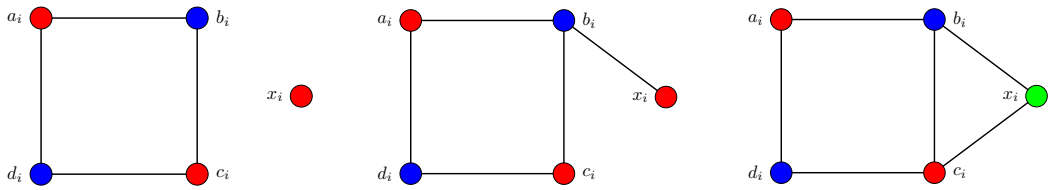
with $t = \ln\Delta, \Gamma = \Delta^2$, we get $\chi_{CN}(\mathcal{H}_2) = O(\log^2\Delta)$. This coloring of the vertices of A ensures that every vertex in A_2 sees some color exactly once in its closed neighborhood in G. We define another hypergraph $\mathcal{H}_1 = (B, \mathcal{E}_1), \mathcal{E}_1 = \{N_G[v] \cap B : v \in A_1 \cup B\}$. Since the degree of a vertex in B in graph G is at most $2\ln\Delta - 2$, we can conclude that every vertex in $\mathcal{H}_1$ is present in at most $2\ln\Delta - 1$ hyperedges. Applying Proposition 1 on $\mathcal{H}_1$, we get $\chi_{CN}(\mathcal{H}_1) \leq 2\ln\Delta$. We make sure that the set of colors used to color $\mathcal{H}_2$ is disjoint from the set of colors used to color $\mathcal{H}_1$. This coloring of the vertices of B ensures that every vertex in $A_1 \uplus B$ sees some color exactly once in its closed neighborhood in G. ◀

It is known that $\chi_{ON}(G) \leq \Delta + 1$, where Δ is the maximum degree of a vertex in G (follows from Proposition 1). From Example 4, we know that this bound is tight. Our next result is the following.

Result 4: There is a two-pass semi-streaming algorithm (uses $O(\frac{1}{\epsilon^2} n \ln^3 n)$ space) that outputs a CFON coloring of G using $(1+\epsilon)\Delta$ colors, for any $\epsilon > 0$, with probability at least $(1 - \frac{1}{n})$.

In our algorithm of Result 4, we use the pallete sparsification lemma of Assadi, Chen, and Khanna [4]. For each vertex $v \in V(G)$, we sample a set $L(v)$ of $O(\frac{1}{\epsilon}\ln n)$ colors from $[(1+\epsilon)\Delta]$. In the first pass of the stream, for each vertex v, we identify a special vertex $s(v)$ that will get a unique color among $N_G(v)$ in our coloring. In the second pass, using the list of colors and special vertices, we find a sketch H of G of size $O(\frac{1}{\epsilon^2} n \ln^2 n)$. Finally, we do a greedy procedure on H to get a CFON coloring.

We observe that any $O(1)$-pass streaming algorithm that determines if the CFCN (CFON) chromatic number of an input graph is at most 2 or not, requires $\Omega(n)$ space. The proofs of both the results are by reductions from the two player communication problem Disjointness (refer [20]). In this problem, Alice and Bob are given two strings of length n (say the strings given to them are $p_1 \dots p_n$ and $q_1 \dots q_n$, respectively), and they are disjoint if there is no $i \in [n]$ such that $p_i = 1$ and $q_i = 1$. It is known that any protocol (even randomized) solving Disjointness requires at least $\Omega(n)$ space.



$p_i = 0$, $q_i = 0$, $\chi_{\text{CN}} = 2$.

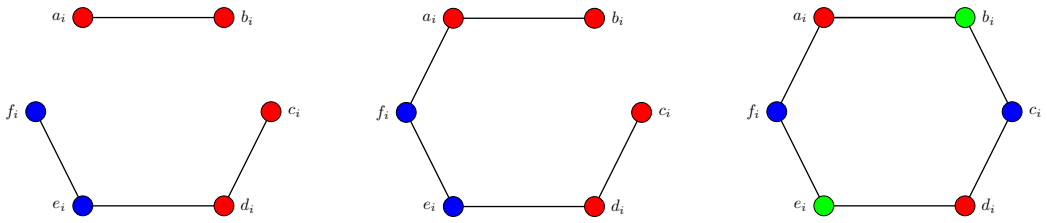
$p_i = 1$, $q_i = 0$, $\chi_{\text{CN}} = 2$. We get similar graph for $p_i = 0$, $q_i = 1$.

$p_i = 1$, $q_i = 1$, $\chi_{\text{CN}} = 3$.

Figure 1 An illustration of the cases in the construction of the graph $G(I)$. The CFCN chromatic number of the rightmost graph is 3, and the CFCN chromatic number of the other two graphs is 2.

Proof of lower bound for CFCN. Consider an instance I of the Disjointness Problem where Alice is given an n-bit string $p_1 \dots p_n$ and Bob is given an n-bit string $q_1 \dots q_n$. Together they construct a graph $G(I)$ on $5n$ vertices whose vertices are a_i, b_i, c_i, d_i, x_i, where $1 \leq i \leq n$. For every i, Alice adds the edges a_ib_i, b_ic_i, c_id_i, d_ia_i to form n 4-cycles. Further, for every i, (i) if $p_i = 1$, then Alice adds the edge b_ix_i, and (ii) if $q_i = 1$, then Bob adds the

edge $c_i x_i$. For each i, they create one or two components of $G(I)$ which is isomorphic to one of the graphs in Figure 1. The graph when both the corresponding bits intersect (i.e., $p_i = q_i = 1$) require at least 3 colors for any CFCN coloring. It can be verified that the graph $G(I)$ thus constructed has its CFCN chromatic number at most 2 if and only if I is a YES instance of the DISJOINTNESS problem. ◀



$p_i = 0$, $q_i = 0$, $\chi_{\text{ON}} = 2$.

$p_i = 1$, $q_i = 0$, $\chi_{\text{ON}} = 2$. We get similar graph for $p_i = 0$, $q_i = 1$.

$p_i = 1$, $q_i = 1$, $\chi_{\text{ON}} = 3$.

Figure 2 An illustration of the cases in the construction of the graph $H(I)$. The CFON chromatic number of the rightmost graph is 3, and the CFON chromatic number of the other two graphs is 2.

Proof of lower bound for CFON. For this purpose, given an instance I, Alice and Bob together construct a graph $H(I)$ on $6n$ vertices, namely $a_i, b_i, c_i, d_i, e_i, f_i$, $1 \leq i \leq n$, as described below: For every i, Alice adds the edges $a_i b_i, c_i d_i, d_i e_i, e_i f_i$. Further, for every i, (i) if $p_i = 1$, then Alice adds the edge $a_i f_i$, and (ii) if $q_i = 1$, the Bob adds the edge $b_i c_i$. If there exists i such that $p_i = q_i = 1$, then $H(I)$ has a connected component which is a cycle of length 6 and we need at least 3 colors in any CFON coloring of it. Otherwise each connected component of $H(I)$ is a path and its CFON chromatic number is at most 2. See Figure 2 for an illustration. ◀

3 Semi-streaming algorithms for CFCN and CFON colorings

In this section, we prove the following theorems.

▶ **Theorem 5.** *There exists a randomized, single-pass, semi-streaming algorithm that, given a graph G with n vertices, with probability at least $\left(1 - \frac{2}{n}\right)$, finds a CFCN coloring using $O(\ln^2 n)$ colors, using $O(n \ln n)$ space and in polynomial time.*

▶ **Theorem 6.** *There exists a randomized, single-pass, semi-streaming algorithm that, given a graph G on n vertices, with probability at least $(1 - \frac{2}{n})$, finds a CFON coloring using $O(\sqrt{n} \ln^2 n)$ colors, using $O(n \ln n)$ space and in polynomial time.*

Towards that, we first prove the following lemma about conflict-free coloring. The bound obtained on the number of colors used to do a conflict-free coloring of the hypergraph defined in this lemma can be obtained by directly substituting the values $t = 2 \ln r$ and $\Gamma = 2n$ in the statement of Proposition 2 that was proved in [23]. However, Proposition 2 would only ensure that such a coloring exists with positive probability. In the lemma below, we modify the proof of this proposition to obtain such a coloring with high probability. For the sake of completeness, we provide the proof of this lemma in the appendix.

▶ **Lemma 7** (⋆). [2] *Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph with $|V| = n$, $|\mathcal{E}| \leq 2n$, and $|E| \geq 4 \ln r - 1$ for all $E \in \mathcal{E}$, where $r \geq max\{7, n\}$ is a positive integer. Then, Algorithm 1, which takes as input V runs in $O(n \ln^2 r)$ time, uses $O(n)$ space and with probability greater than $\left(1 - \frac{2}{r}\right)$ returns a conflict-free coloring of $\mathcal{H}$ that uses at most $\lceil 16e \ln^2 r \rceil$ colors.*

Algorithm 1 Randomized Algorithm for CF coloring of $\mathcal{H}$.

Input: The vertex set of hypergraph $\mathcal{H} = (V, \mathcal{E})$, where each hyperedge E satisfies $|E| \geq 4 \ln r - 1, \forall E \in \mathcal{E}$, and $r \geq n = |V|$.
Output: A CF coloring function $c : V \to [\lceil 16e \ln^2 r \rceil]$

```
Let T = ⌈16e ln² r⌉ and let c(v) = 0, ∀v ∈ V
Set i := 0
while i < T do
    i := i + 1;
    for each vertex v ∈ V with c(v) = 0 do
        Independently toss a coin that gives Head with probability 1/(8e ln r)
        if the outcome is Head then
            c(v) := i
return c
```

Next, we prove the following lemma that gives a coloring such that a subset of the vertices with "large" degree gets a unique color in its open and closed neighborhoods.

▶ **Lemma 8.** *Let G_1 be a graph on n_1 vertices, where $V(G_1) = X \uplus Y$. Suppose for all $x \in X, deg_{G_1}(x) \geq 4 \ln r_1 - 1$, where r_1 is a positive integer with $r_1 \geq max\{n_1, 7\}$. Let c_1 be the coloring $c_1 : V(G_1) \to [\lceil 16e \ln^2 r_1 \rceil]$ obtained by invoking Algorithm 1 with $V(G_1)$ as input. Then, with probability $1 - \frac{2}{r_1}$, every vertex in X sees some color exactly once in its open and closed neighborhoods.*

Proof. Proof follows from the application of Lemma 7 with $\mathcal{H}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ defined as $\mathcal{V}_1 = V(G_1)$ and $\mathcal{E}_1 = \{N_{G_1}[v] : v \in X\} \cup \{N_{G_1}(v) : v \in X\}$. ◀

▶ **Lemma 9.** *Let $r_2 \in \mathbb{Z}^+$ and let G_2 be a graph on n_2 vertices, where $V(G_2) = X \uplus Y$ with the following properties: (i) X is an independent set, (ii) for all $x \in X, N_{G_2}(x) \cap Y \neq \emptyset$, and (iii) for all $y \in Y, deg_{G_2}(y) < 4\lceil \ln r_2 \rceil - 1$. Then, Algorithm 2 is a greedy, deterministic algorithm that runs in $O(n_2 \ln r_2)$ time and $O(n_2 \ln r_2)$ space, and outputs a coloring of vertices in Y using at most $(8\lceil \ln r_2 \rceil)$ colors with the following properties.*

(a) *Every vertex in $V(G_2)$ sees some color exactly once in its closed neighborhood.*
(b) *Every vertex $v \in V(G_2)$ with $N_{G_2}(v) \cap Y \neq \emptyset$ sees some color exactly once in its open neighborhood.*

Proof. The hypergraph $\mathcal{H}_2$ constructed in step 1 of Algorithm 2 takes $O(n_2 \ln r_2)$ space because the degree of this hypergraph is at most $8\lceil \ln r_2 \rceil$. It is easy to see that a conflict-free coloring of $\mathcal{H}_2$ will ensure properties (a) and (b). Proposition 1 gives a conflict-free coloring of $\mathcal{H}_2$ in $O(n_2 \ln r_2)$ time and $O(n_2 \ln r_2)$ space using at most $8\lceil \ln r_2 \rceil$ colors. ◀

[2] Proof of results marked with ⋆ are deferred to the appendix.

Algorithm 2 Deterministic Algorithm for conflict-free coloring of G_2.

Input: The vertex set and edge set of the graph G_2 with
1. Partition $V(G_2) = X \uplus Y$, where X is an independent set,
2. $\forall x \in X, N_{G_2}(x) \cap Y \neq \emptyset$ and
3. $\forall y \in Y, \deg_{G_2}(y) < (4\lceil \ln r_2 \rceil - 1)$

Output: A coloring function $c_2 : Y \to [8\lceil \ln r_2 \rceil]$ that satisfies properties (a) and (b) mentioned in Lemma 9.

1 Construct a hypergraph $\mathcal{H}_2 = (Y, \mathcal{E})$ with

$$\mathcal{E} = \{N_{G_2}[v] \cap Y : v \in V(G_2)\} \cup \{N_{G_2}(v) \cap Y : v \in V(G_2), N_{G_2}(v) \cap Y \neq \emptyset\}.$$

Note that the degree of this hypergraph is at most $8\lceil \ln r_2 \rceil$ because for all $y \in Y$, $\deg_{G_2}(y) < 4\lceil \ln r_2 \rceil$. Use the algorithm given in Proposition 1 to get a coloring $c_2 : Y \to 8[\lceil \ln r_2 \rceil]$ of the vertices of $\mathcal{H}_2$ such that every hyperedge sees some color exactly once.

2 Return the coloring c_2.

▶ **Lemma 10.** *Let G be a graph on n vertices, where*
- *$V(G) = A \uplus B$, $A = A' \uplus A''$, $A' = \{a \in A : N_G(a) \cap B \neq \emptyset\}, A'' = A \setminus A'$,*
- *for all $a \in A$, $deg_G(a) \geq (4\lceil \ln n \rceil - 1)$, and*
- *for all $b \in B$, $deg_G(b) < (4\lceil \ln n \rceil - 1)$.*

Algorithm 3 takes as input (i) the sets A', A'', B, (ii) $N_G(a') \cap B$ for all $a' \in A'$, and (iii) $N_G(b)$ for all $b \in B$, runs in $O(n \cdot \ln^2 n)$ time, uses $O(n \ln n)$ space and gives a coloring c of $V(G)$ using $O(\ln^2 n)$ colors such that with probability greater than $\left(1 - \frac{2}{n}\right)$ the following holds.

(i) *c is a CFCN coloring of G.*

(ii) *For every $v \in V(G) \setminus \{b \in B : N_G(b) \cap B = \emptyset\}$, v sees some color exactly once in its open neighbourhood.*

Proof. We apply Lemma 8 with the following inputs: $G_1 = (A, E_1)$, where $E_1 = \{(u, v) : u, v \in A\}$, $X = A''$, $Y = A'$, and $r_1 = n$. The algorithm mentioned in Lemma 7, runs in $O(n \ln^2 n)$ time, as $n_1 \leq r_1 = n$. It uses $O(n)$ space and outputs the coloring function $c_1 : A \to [16\lceil e \ln^2 n \rceil]$. With probability greater than $(1 - \frac{2}{n})$, every vertex in $A''(= X)$ will see some color exactly once in its open and closed neighborhoods under the coloring c_1 (as well as in c).

Next, we invoke Algorithm 2 with the following inputs: the graph $G_2 = (A' \uplus B, E_2)$, where $X = A'$, $Y = B$, $r_2 = n$, and $E_2 = E(G) \setminus E_1$. It is important to note that (i) G_2 is well-defined as every edge in E_2 has both its endpoints in $A' \cup B$, and (ii) A' is an independent set in G_2. According to Lemma 9, Algorithm 2 outputs the coloring $c_2 : B \to \{16\lceil e \ln^2 n \rceil + 1, \ldots, 16\lceil e \ln^2 n \rceil + 8\lceil \ln n \rceil\}$ in $O(n \ln n)$ time, using $O(n \ln n)$ space, since $n_2 \leq r_2 = n$. By property (a) of Lemma 9, every vertex in $A' \cup B$ sees some color exactly once in its closed neighborhood under the coloring c_2 (as well as in c). Therefore, we have proved that c is a CFCN coloring of G.

Furthermore, based on the reasoning above, our algorithm takes $O(n \ln^2 n)$ time and $O(n \ln n)$ space. We have already proved that with probability greater than $(1 - \frac{2}{n})$, every vertex in A'' will see some color exactly once in its open neighborhoods under the coloring c. By the property (b) of Lemma 9, we know that every vertex v in $\{u \in V(G_2) : N_{G_2}(u) \cap Y \neq \emptyset\} = A' \cup \{b \in B : N_G(b) \cap B \neq \emptyset\}$, sees some vertex exactly once in its open neighborhood under the coloring c_2 (as well as in c). This proves property (ii) mentioned in the lemma. ◀

Algorithm 3 Algorithm for conflict free coloring.

Input: (i) vertex set $V(G)$ of the graph G on n vertices and the partition $V(G) = A \uplus B$, where $A = A' \uplus A''$, (ii) $N_G(a') \cap B$ for all $a' \in A'$, and (iii) $N_G(b)$ for all $b \in B$. The input satisfies the conditions of Lemma 10.

Output: A coloring function $c : V(G) \to [16\lceil e \ln^2 n \rceil + 8\lceil \ln n \rceil]$.

1 Consider the graph $G_1 = (A, E_1)$, where $E_1 = \{(u, v) : u, v \in A\}$. We apply the algorithm mentioned in Lemma 8 with G_1 as the input. Let $X = A''$, $Y = A'$, and $r_1 = n$. Notice that for all $x \in X = A''$, $\deg_{G_1}(x) \geq (4\lceil \ln n \rceil - 1)$. Let $c_1 : A \to [16\lceil e \ln^2 n \rceil]$ be the coloring returned by the algorithm mentioned in Lemma 8.

2 Consider the graph $G_2 = (A' \uplus B, E_2)$, where $E_2 = E(G) \setminus E_1$. Note that A' is an independent set in G_2. Invoke Algorithm 2 with G_2 as the input. Set $X = A'$, $Y = B$, and $r_2 = n$. Let the coloring function returned by Algorithm 2 be $c_2 : B \to \{16\lceil e \ln^2 n \rceil + 1, \ldots, 16\lceil e \ln^2 n \rceil + 8\lceil \ln n \rceil\}$.

3 Return $c : V(G) \to [16\lceil e \ln^2 n \rceil + 4\lceil \ln n \rceil - 1]$, defined as follows:

$$c(v) = \begin{cases} c_1(v), & \text{if } v \in A \\ c_2(v), & \text{if } v \in B \end{cases}$$

Proof of Theorem 5. The proof follows from the application of Lemma 10 and the following partitioning of the vertex set $V := V(G)$. We start by partitioning the vertex set V into two sets, namely A and B, where $A = \{v \in V : \deg_G(v) \geq (4\lceil \ln n \rceil - 1)\}$ and $B = V \setminus A$ using the following procedure . Here, A represents the set of high-degree vertices, while B represents the set of low-degree vertices.

To facilitate the partitioning process, we maintain a list $L(v) = \{u : u \in N_G(v)\}$. Initially, we set $A := \emptyset$ and $L(v) = \emptyset$. As each edge (u, v) passes through the stream, we update $L(u) = L(u) \cup \{v\}$ if $u \in V \setminus A$ and $L(v) = L(v) \cup \{u\}$ if $v \in V \setminus A$. Additionally, if a vertex w satisfies $|L(w)| \geq (4\lceil \ln n \rceil - 1)$, then we update $A := A \cup \{w\}$. At the end of the stream, we define $B := V \setminus A$. It is important to note that for any vertex $b \in B$, $|L(b)| < (4\lceil \ln n \rceil - 1)$. Up to this point, the storage requirement for $\bigcup_{v \in V} L(v)$ is $O(n \ln n)$ space.

Next, for all $a \in A$, update $L(a)$ as $L(a) = \{b \in B \ : \ a \in L(b)\}$. That is $L(a)$ is the set $N_G(a) \cap B$. Observe that updating $L(a)$'s can only double the total space used so far. After updating $L(a)$, for all $a \in A$, we define $A' = \{a \in A \ : \ L(a) \neq \emptyset\}$ and $A'' = A \setminus A'$. We have thus obtained (i) the sets A', A'', B, (ii) $N_G(a') \cap B$, for all $a' \in A'$, and (iii) $N_G(b)$ for all $b \in B$. We can now invoke Algorithm 3 with the required inputs to obtain a CFCN coloring function $c : V(G) \to [16\lceil e \ln^2 n \rceil + 8\lceil \ln n \rceil]$. From Lemma 10, we know that Algorithm 3 requires only $O(n \ln n)$ space and runs in polynomial time. ◀

Proof of Theorem 6. Given a graph $G = (V, E)$, we follow the same partitioning procedure mentioned in Theorem 5. That is, $A = \{v \in V : \deg_G(v) \geq (4\lceil \ln n \rceil - 1)\}$, and $B = V(G) \setminus A$. We again partition $B \subseteq V$ into two sets: $B = B_1 \uplus B_2$, where $B_1 = \{b \in B : N_G(b) \cap B = \emptyset\}$ and $B_2 = B \setminus B_1$ That is, using similar steps as in the case of Theorem 5, we get the partition of $V(G)$ into $A' \uplus A'' \uplus B_1 \uplus B_2$. Moreover, we also get $L(b) = N_G(b)$ for all $b \in B$ in $O(n \ln n)$ space. Now we invoke Algorithm 3 with the required inputs to obtain a coloring $c : V(G) \to [16\lceil e \ln^2 n \rceil + 8\lceil \ln n \rceil]$. From Lemma 10, we know that Algorithm 3 requires only

$O(n \ln n)$ space and runs in polynomial time. Moreover, with probability greater than $1 - \frac{2}{n}$, for every $v \in V(G) \setminus \{b \in B : N_G(b) \cap B = \emptyset\}$, v sees some color exactly once in its open neighbourhood.

We now invoke Proposition 3 (or an algorithmic version of it (see Theorem 18)) on the graph $G' = (A' \cup B_1, E')$ where $E' = \{(a, b) \in E(G) : a \in A', b \in B_1\}$ and get a CFON coloring $c' : V(G') \to [\lceil 2\sqrt{n} \rceil]$ of G'. Notice that $|E'| = O(n \ln n)$ and hence c' can be obtained in $O(n \ln n)$ space and polynomial time (see Theorem 18). Now, consider the coloring c_o on $V(G)$ defined as follows.

$$c_o(v) = \begin{cases} (c(v), c'(v)), & \text{if } v \in A' \cup B_1 \\ (c(v), 1), & \text{otherwise} \end{cases}$$

Now it is easy to see that c_o is a CFON coloring because for each $v \in V(G) \setminus B_1$, there is a unique color in the open neighborhood of v due to the coloring c and for each $v \in B_1$, there is a unique color in the open neighborhood of v due to the coloring c'. Notice that the number of colors used in c_o is $O(\sqrt{n} \ln^2 n)$. ◀

4 Semi-streaming algorithm for CFON coloring using $(1 + \epsilon)\Delta$ colors

In this section we prove a two-pass randomized semi-streaming algorithm for finding a CFON coloring of the input graph G using $(1 + \epsilon)\Delta$ colors, where Δ is the maximum degree of G. The theorem is formally stated below.

▶ **Theorem 11.** *There is a two-pass randomized semi-streaming algorithm (uses $O(\frac{1}{\epsilon^2} n \ln^3 n)$ space) that given a graph G and $\epsilon > 0$, outputs a CFON coloring of G using $(1 + \epsilon)\Delta$ colors or reports fail. It outputs a CFON coloring with probability at least $(1 - \frac{1}{n^3})$. Here, Δ is the maximum degree of G.*

We would like to mention that prior knowledge of Δ is not required. During the first pass of the edge stream, we find the maximum degree Δ of G, and for each vertex $u \in V(G)$, a special vertex $s(u)$ as explained below. Let $V(G) = \{v_1, \ldots, v_n\}$ and let Π be a fixed order $v_1, v_2, \ldots, v_n$. For each $u \in V(G)$, $s(u) = v_i$, where i is the least index such that $v_i \in N_G(u)$. We would like to mention that if our algorithm outputs a coloring, then for each vertex $u \in V(G)$, $s(u)$ will get a unique color among $N_G(u)$. It is easy to note that one can find the maximum degree Δ from the first pass using $O(n)$ space by keeping n counters, one per vertex. One can find out the special vertices from the first pass of the edge stream using the following procedure. We maintain an array (call it "a") of size n in memory. The array keeps track of the special vertices, i.e., the minimum indexed vertex in the open neighborhood of every vertex in the graph seen until now. Initially, all the elements in the array are set to NULL. When an edge (v_i, v_j) comes in the stream, we update $a[i]$ as follows.

- If $a[i]$ =NULL, then set $a[i] = v_j$.
- If $a[i] = w$ for some vertex $w \in V(G)$, then we set $a[i] = v_j$ iff $v_j <_\Pi w$.
- Similarly, we update the entry $a[j]$.

Notice that the space used in the first pass is $O(n \ln n)$. In the second pass of the stream, we construct a sketch H of G. Before the beginning of the second pass, for each vertex $v \in V(G)$, we sample a set of colors $L(v)$ of size $t = \frac{4}{\epsilon} \ln n$ from $[(1 + \epsilon)\Delta]$ uniformly and independently at random with replacement. Note that $L(v)$ is a multiset as the colors are sampled at random with replacement.

Algorithm 4 Algorithm for CFON coloring of graph G.

Data: Stream of edges of a graph G and $\epsilon > 0$
Result: coloring $c : V(G) \to [(1+\epsilon)\Delta]$ where Δ is the maximum degree of G

1 Fix an ordering $\Pi = v_1, \ldots, v_n$ of $V(G)$.
2 For all $v \in V$, sample a set $L(v)$ of size $t = \frac{4}{\epsilon} \ln n$ colors from $[(1+\epsilon)\Delta]$ uniformly and independently at random with replacement.
3 In the first pass of the edge stream, compute the maximum degree Δ of G and special vertices $s(u)$ for all $u \in V(G)$.
4 In the second pass of the edge stream, construct the sketch H of G with respect L (see Definition 12).
5 Do coloring $c : V(G) \to [(1+\epsilon)\Delta]$ using a greedy procedure on H as explained below.
6 In the i^{th} step, we have colored all the vertices $v_1, \ldots, v_{i-1}$. Now we color v_i as follows.
- Let $S_i = \{s(u) \ : \ u \in N_H(v_i)\} \cap \{v_1, \ldots, v_{i-1}\}$ and $Z_i = L(v_i) \setminus \{c(w) \mid w \in S_i\}\}$
- If $Z_i = \emptyset$, then report FAIL.
- Otherwise $c(v_i) = \min Z$.

▶ **Definition 12** (Sketch of G with respect to L)**.** *A sketch of G with respect to L is the subgraph H of G defined as follows: $V(H) = V(G)$, and*

$$E(H) = \{(u,v) : (L(u) \cap L(s(v)) \neq \emptyset) \vee ((L(s(u)) \cap L(v)) \neq \emptyset))\}$$

It is easy to store the sketch H of G in the second pass. For each edge (u,v) encountered in the edge stream, we check if $(u,v) \in E(H)$ by checking the condition $(L(u) \cap L(s(v)) \neq \emptyset) \vee ((L(s(u)) \cap L(v)) \neq \emptyset))$. Thus, by the end of the second pass, we have the sketch H of G with respect to L.

Finally, we do a coloring $c : V(G) \to [(1+\epsilon)\Delta]$ using a greedy procedure on H explained as follows. We color the vertices in the order $v_1, \ldots, v_n$. In the ith step we color the vertex v_i as follows. Let $S_i = \{s(u) \ : \ u \in N_G(v_i)\} \cap \{v_1, \ldots, v_{i-1}\}$.

Note that for each $u \in N_G(v_i), s(u) \leq_\Pi v_i$ and S_i is the set of special vertices, of "all" the neighbors of v_i, that are on the left side of v_i in the ordering Π. Let $Z_i = L(v_i) \setminus \{c(w) \mid w \in S_i\}$. That is, Z_i is the set of colors from $L(v_i)$ that are not used to color the vertices in S_i. Now we color v_i as per the following.

- If $Z_i = \emptyset$, then report FAIL. That is no color is available in the list $L(v_i)$ that is not equal to the color of the special vertices in S_i.
- Otherwise $c(v_i) = \min Z_i$.

For convenience, the pseudocode of the algorithm is available in Algorithm 4. Next we prove the following lemma which will be used to prove Theorem 11.

▶ **Lemma 13** (⋆)**.** *With probability at least* 0.99, $|E(H)| \leq \frac{1600}{\epsilon^2} n \ln^2 n$.

▶ **Lemma 14.** *If Algorithm 4 produces a coloring c then c is a CFON coloring of G. Moreover, Algorithm 4 produces a CFON coloring with probability at least* $(1 - \frac{1}{n^3})$.

Proof. Suppose Algorithm 4 outputs a coloring c. To prove that it is a CFON coloring of G, we need to prove that for each $v \in V(G)$, there is a vertex in $N_G(v)$ with unique color. We will prove that for each vertex $v \in V(G)$, $s(v)$ becomes the uniquely colored vertex in $N_G(v)$. Recall that $s(v)$ is the first vertex in Π among $N_G(v)$. So Algorithm 4 colors $s(v)$ before coloring the vertices in $N_G(v) \setminus \{s(v)\}$. Step 6 of Algorithm 4 implies that each vertex in $N_G(v) \setminus \{s(v)\}$ gets a color different from the color of $s(v)$. So, c is the CFON coloring of G.

We now find the probability that Algorithm 4 reports fail. Suppose Algorithm 4 reports FAIL in the ith iteration of step 6. It means that $Z_i = \emptyset$. That is, $L(v_i) \subseteq \{c(x) \mid x \in S_i\}$. We know that $|S_i| \leq |N_G(v_i)| \leq \Delta$. Here, we want to calculate $\Pr[L(v_i) \subseteq S_i]$. This is calculated as follows.

$$\Pr[L(v_i) \subseteq S_i] = \left(\frac{|S_i|}{((1+\epsilon)\Delta)}\right)^t \leq \left(\frac{\Delta}{(1+\epsilon)\Delta)}\right)^t = \frac{1}{(1+\epsilon)^t} = \frac{1}{(1+\epsilon)^{\frac{4}{\epsilon}\cdot \ln n}} \leq \frac{1}{n^4}$$

Here, the second inequality follows from the fact that $|S_i| \leq \Delta$. Notice that there are n iterations, and by the union bound, the probability that the algorithm reports FAIL in any one of these iterations is at most $\frac{1}{n^3}$. ◀

Proof of Theorem 11. We run Algorithm 4 $\ln n$ times parallelly. In each execution of the algorithm, if the space used by it exceeds $\frac{1600}{\epsilon^2} n \ln n$, then we abort it. Finally, if any of the execution outputs a coloring, we output one such coloring. Otherwise, we output FAIL. The probability that each execution is either aborted or output FAIL is upper bounded by $\frac{1}{100} + \frac{1}{n^3} \leq 0.02$. The probability that all the executions fail with probability is at most $0.02^{\ln n} \leq \frac{1}{n^3}$. This implies that the overall probability that Algorithm 4 succeeds is at least $(1 - \frac{1}{n^3})$. Notice that each execution of Algorithm 4 takes $O(\frac{1}{\epsilon^2} n \ln^2 n)$ space and hence the total space is $O(\frac{1}{\epsilon^2} n \ln^3 n)$. ◀

5 Concluding remarks

Most of the study on CFCN and CFON coloring so far has been towards proving tighter bounds in terms of various graph parameters like maximum degree, order of the graph, pathwidth, etc. Showing improved bounds for special graph classes like planar graphs, line graphs, unit-disc graphs, etc. has also been explored. CFCN and CFON coloring problems have also been explored from a parameterized complexity setting [10, 11, 13]. To the best of our knowledge, this is the first paper exploring semi-streaming algorithms for these coloring problems. The bounds (on the number of colors used) we obtain in Theorems 5, 11, and 18 are asymptotically tight. Yet these leave us with many more intriguing open questions, some of which are listed below. Given as input an edge stream of an n-vertex graph G with maximum degree Δ,

1. Is it possible to obtain a deterministic single pass semi-streaming algorithm that does a CFCN coloring using $O(\ln^2 n)$ colors? The algorithm given by Theorem 5 is randomized.
2. Is it possible to improve Theorem 6 to obtain a semi-streaming (deterministic or randomized) algorithm that gives a CFON coloring using $O(\sqrt{n})$ colors?
3. Is it possible to obtain a semi-streaming algorithm that outputs a CFON coloring of G using $\Delta + 1$ colors? Theorem 11 gives a two-pass randomized semi-streaming algorithm that outputs a CFON coloring using at most $(1+\epsilon)\Delta$ colors. And, Theorem 18 can be modified to obtain a deterministic single-pass streaming algorithm for CFON coloring of G using $\Delta + 1$ colors that uses $O(n\sqrt{n})$ space (the algorithm given by the theorem works, if $\Delta \geq 2\sqrt{n} - 1$. Otherwise, we store the entire graph and color it offline). As far as classical coloring is concerned, there are randomized single-pass semi-streaming algorithms that output a $\Delta + 1$-coloring with high probability (see [6]). At the same time, it is known (see [8]) that any single-pass streaming algorithm that outputs a $(d+1)$-coloring of G requires at least $\Omega(n^2)$ space, where d is the degeneracy of G (the graph G is k-degenerate if its vertices can be arranged in a line such that every vertex has at most k neighbors to its left. The minimum k for which G is k-degenerate is called its degeneracy. Clearly, this parameter is at most Δ). Coming back to the CFON coloring problem, any CFON

coloring of G would want some vertex, say $s(v)$, in the neighborhood of each vertex v to receive a color distinct from that of the other neighbors of v. If we construct an auxiliary graph of G by adding edges between $s(v)$ and all the other neighbors of v, for every v, such a graph would have a maximum degree of $\Theta(\Delta^2)$ and degeneracy of Δ. And a classical coloring of this auxiliary graph would yield a CFON coloring of G. This makes the question of obtaining a semi-streaming algorithm for $(\Delta+1)$-CFON coloring of G all the more interesting.

References

1. Noga Alon and Sepehr Assadi. Palette sparsification beyond $(\Delta + 1)$ vertex coloring. *International Workshop and International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques 2020*, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.6.
2. Mohammad Ansari, Mohammad Saneian, and Hamid Zarrabi-Zadeh. Simple streaming algorithms for edge coloring. In *30th Annual European Symposium on Algorithms (ESA 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.8.
3. Sepehr Assadi, Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Coloring in graph streams via deterministic and adversarially robust algorithms. *PODS '23: Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2022.
4. Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta+1)$ vertex coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 767–786. SIAM, 2019. doi:10.1137/1.9781611975482.48.
5. Sepehr Assadi, Pankaj Kumar, and Parth Mittal. Brooks' theorem in graph streams: a single-pass semi-streaming algorithm for Δ-coloring. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 234–247, 2022.
6. Sepehr Assadi and Helia Yazdanyar. Simple sublinear algorithms for $(\Delta+1)$ vertex coloring via asymmetric palette sparsification. In *2025 Symposium on Simplicity in Algorithms (SOSA)*, pages 1–8. SIAM, 2025. doi:10.1137/1.9781611978315.1.
7. Soheil Behnezhad and Mohammad Saneian. Streaming edge coloring with asymptotically optimal colors. *arXiv preprint arXiv:2305.01714*, 2023. doi:10.48550/arXiv.2305.01714.
8. Suman K. Bera, Amit Chakrabarti, and Prantar Ghosh. Graph Coloring via Degeneracy in Streaming and Other Space-Conscious Models. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:21, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.11.
9. 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.
10. Sriram Bhyravarapu, Tim A. Hartmann, Subrahmanyam Kalyanasundaram, and I. Vinod Reddy. Conflict-free coloring: Graphs of bounded clique width and intersection graphs. In Paola Flocchini and Lucia Moura, editors, *Combinatorial Algorithms*, pages 92–106, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-79987-8_7.
11. Sriram Bhyravarapu and Subrahmanyam Kalyanasundaram. A tight bound for conflict-free coloring in terms of distance to cluster. *Discrete Mathematics*, 345(11):113058, 2022. doi:10.1016/j.disc.2022.113058.
12. Sriram Bhyravarapu, Subrahmanyam Kalyanasundaram, and Rogers Mathew. A short note on conflict-free coloring on closed neighborhoods of bounded degree graphs. *Journal of Graph Theory*, 97(4):553–556, 2021. doi:10.1002/JGT.22670.

13 Hans L. Bodlaender, Sudeshna Kolay, and Astrid Pieterse. Parameterized complexity of conflict-free graph coloring. *SIAM J. Discret. Math.*, 35(3):2003–2038, 2021. doi:10.1137/19M1307160.

14 Panagiotis Cheilaris. *Conflict-free coloring*. City University of New York, 2009.

15 Michał Dębski and Jakub Przybyło. Conflict-free chromatic number versus conflict-free chromatic index. *Journal of Graph Theory*, 99(3):349–358, 2022. doi:10.1002/JGT.22743.

16 Khaled M. Elbassioni and Nabil H. Mustafa. Conflict-free colorings of rectangles ranges. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, volume 3884 of *Lecture Notes in Computer Science*, pages 254–263. Springer, 2006. doi:10.1007/11672142_20.

17 Guy Even, Zvi Lotker, Dana Ron, and Shakhar Smorodinsky. Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM Journal on Computing*, 33(1):94–136, 2003. doi:10.1137/S0097539702431840.

18 Roman Glebov, Tibor Szabó, and Gábor Tardos. Conflict-free colouring of graphs. *Combinatorics, Probability and Computing*, 23(3):434–448, 2014. doi:10.1017/S0963548313000540.

19 Prasad Krishnan, Rogers Mathew, and Subrahmanyam Kalyanasundaram. Pliable index coding via conflict-free colorings of hypergraphs. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 214–219. IEEE, 2021. doi:10.1109/ISIT45174.2021.9518120.

20 E Kushilevitz and N Nisan. Communication complexity, cambridge univ. *Press, Cambridge, UK*, 1997.

21 Nissan Lev-Tov and David Peleg. Conflict-free coloring of unit disks. *Discrete Applied Mathematics*, 157(7):1521–1532, 2009. doi:10.1016/j.dam.2008.09.005.

22 Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.

23 János Pach and Gábor Tardos. Conflict-free colourings of graphs and hypergraphs. *Combinatorics, Probability and Computing*, 18(5):819–834, 2009. doi:10.1017/S0963548309990290.

24 Shakhar Smorodinsky. *Combinatorial problems in computational geometry*. PhD thesis, Tel-Aviv University, 2003.

25 Shakhar Smorodinsky. Conflict-free coloring and its applications. *Geometry – Intuitive, Discrete, and Convex: A Tribute to László Fejes Tóth*, pages 331–389, 2013.

A Proof of Lemma 7

In any given iteration of the "while" loop, the "for" loop in Step 5 runs at most n times. The while loop in Step 3 runs at most $T = \lceil 16e \ln^2 r \rceil$ times. Thus it is easy to see that the algorithm terminates in $O(nT) = O(n \ln^2 r)$ steps.

Note that the vertex set V of $\mathcal{H}$ can be stored in $O(n)$ space. The algorithm does not need to store the edges regardless of the structure of the hypergraph $\mathcal{H}$. The algorithm does a randomized coloring of the vertices in V, which is explained in steps 5-8. This can be done in $O(n)$ workspace. Thus, the total space required is $O(n)$.

Consider an execution of Algorithm 1. Let A_v denote the event of v not receiving any non-zero color.

$$\begin{aligned}\Pr[A_v] = (1-p)^T, \text{ where } p \text{ is the probability for Head and } p &= \frac{1}{8e \ln r} \\ = \left(1 - \frac{1}{8e \ln r}\right)^T &\leq e^{-\frac{T}{8e \ln r}} \leq \frac{1}{r^2} \qquad (\because T = \lceil 16e \ln^2 r \rceil)\end{aligned}$$

Let A denote the event of some vertex in V not receiving a non-zero color. Then, $\Pr(A) = \Pr(\bigcup_{v \in V} A_v) \leq \sum_{v \in V} \Pr(A_v) \leq n \cdot \frac{1}{r^2} \leq r \cdot \frac{1}{r^2} (\because n \leq r) = \frac{1}{r}$. Thus, we have $\Pr(A) \leq \frac{1}{r}$.

Let $\bar{A}$ denote the complement of event A. For a hyperedge $E \in \mathcal{E}$, let B_E denote the event of E not seeing a uniquely colored vertex. And, let B denote the event of some hyperedge $E \in \mathcal{E}$ not seeing a unique color. That is, B denotes the event of the coloring function not being a conflict-free coloring of $\mathcal{H}$. From the definition of B, $B = \bigcup_{E \in \mathcal{E}} B_E$. Let $\bar{B}$ denote the complement of the event B. Then we know that,

$$\Pr(\bar{A} \cap \bar{B}) = \Pr(\bar{A}) \cdot \Pr(\bar{B}|\bar{A}) \geq (1 - \frac{1}{r}) \cdot \Pr(\bar{B}|\bar{A}) \tag{1}$$

We are left with the task of estimating $P(\bar{B}|\bar{A})$. We have,

$$\Pr(\bar{B}|\bar{A}) = 1 - \Pr(B|\bar{A}) = 1 - \Pr(\bigcup_{E \in \mathcal{E}} B_E|\bar{A}) \geq 1 - \sum_{E \in \mathcal{E}} \Pr(B_E|\bar{A}) \tag{2}$$

To estimate $\Pr(B_E|\bar{A})$, we use the following lemma from [23].

▶ **Lemma 15** (Lemma 3.1 in [23]). *Let V be a set of elements and let $E \subseteq V$ with $|E| \geq 2t-1$ for a positive integer t. Let us color each element of V independently, according to the geometric distribution with parameter p. Then the probability that no element of E receives a unique color (one that is not received by any other vertex of E) is at most $2(etp)^t$.*

With respect to the execution of Algorithm 1, given that the event $\bar{A}$ has happened (that is, the algorithm has assigned a non-zero color for every vertex $v \in V$), the coloring c returned by the algorithm is a coloring based on a geometric distribution with parameter $p = \frac{1}{8e \ln r}$. Thus, Lemma 15 gives an upper bound to $P(B_E|\bar{A})$. Applying Lemma 15 with $p = \frac{1}{8e \ln r}$ and $t = 2 \ln r$, we have

$$\Pr(B_E|\bar{A}) \leq 2 \left(e \cdot 2 \ln r \cdot \frac{1}{8e \ln r} \right)^{2 \ln r} \leq \frac{2}{r^{2.773}} \tag{3}$$

From Equations (2) and (3), we get

$$\Pr(\bar{B}|\bar{A}) \geq 1 - |\mathcal{E}| \cdot \frac{2}{r^{2.773}} = 1 - \frac{4}{r^{1.773}} \qquad \text{(Because } |\mathcal{E}| \leq 2n \leq 2r\text{)} \tag{4}$$

From Equations (4) and (1), we have

$$\Pr(\bar{A} \cap \bar{B}) \geq \left(1 - \frac{1}{r}\right) \left(1 - \frac{4}{r^{1.773}}\right) \geq \left(1 - \frac{1}{r}\right)^2 > \left(1 - \frac{2}{r}\right) \tag{5}$$

Here, the second inequality follows from the fact that $1 - \frac{4}{r^{1.773}} > 1 - \frac{1}{r}$ when $r \geq 7$. This completes the proof of the lemma.

B Proof of Lemma 13

To prove Lemma 13, first we prove the following lemma and then use this lemma.

▶ **Lemma 16.** *Let $(u, v) \in E(G)$. Then $\Pr[(u, v) \in E(H)] \leq \frac{2t^2}{(1+\epsilon)\Delta}$.*

Proof. Recall that for all $w \in V(G)$, $L(w)$ denotes the pallete of colors for the vertex w and $|L(w)| = t$. The colors are sampled uniformly and independently at random with replacement from the set of colors $[(1+\epsilon)\Delta]$. Let E_1 be the event $(L(u) \cap L(s(v))) \neq \emptyset$ and let E_2 be the event $(L(s(u)) \cap L(v)) \neq \emptyset$. Now, we have $\Pr[(u, v) \in E(H)] = \Pr[E_1 \vee E_2] \leq \Pr[E_1] + \Pr[E_2]$. We now calculate $\Pr[E_1]$. Let $L(s(v)) = \{r_1, \ldots, r_t\}$. Recall that $L(s(v))$ is a multiset.

$$\Pr[E_1] = \Pr\left[\bigcup_{i\in[t]} r_i \in L(u)\right] \leq \sum_{i\in[t]} \Pr(r_i \in L(u)) = \sum_{i\in[t]} \frac{t}{(1+\epsilon)\Delta} = \frac{t^2}{(1+\epsilon)\Delta}$$

Similarly, $\Pr[E_2] \leq \frac{t^2}{(1+\epsilon)\Delta}$. Thus, we get $\Pr[(u,v) \in E(H)] \leq \Pr[E_1] + \Pr[E_2] \leq \frac{2t^2}{(1+\epsilon)\Delta}$. ◀

Below we state Markov's Inequality.

▶ **Proposition 17** (Markov's inequality). *If X is a nonnegative random variable and $a > 0$, then $\Pr[X \geq a] \leq \frac{E(X)}{a}$.*

Now, we are ready to prove Lemma 13.

Proof of Lemma 13. Let $\mathbb{X}$ be the random variable that denotes $|E(H)|$. For each $v \in V(H)$, let $\mathbb{X}_v$ denote the degree of vertex v in H. For each $(v,u) \in E(G)$, $\mathbb{X}_{vu}$ is the indicator random variable defined as follows: $\mathbb{X}_{vu} = 1$ if and only if $(v,u) \in E(H)$. Then, for any $v \in V(G)$, $\mathbb{X}_v = \sum_{u\in N_G(v)} \mathbb{X}_{vu}$, and $E[\mathbb{X}_v]$ is calculated as follows.

$$E[\mathbb{X}_v] = \sum_{u\in N_G(v)} E[\mathbb{X}_{vu}] \leq \Delta \frac{2t^2}{(1+\epsilon)\Delta} = \frac{2t^2}{(1+\epsilon)} \tag{6}$$

Here, the second inequality follows from Lemma 16. Notice that $\mathbb{X} = \frac{1}{2}\sum_{v\in V(G)} \mathbb{X}_v$. Thus, by Equation (6)

$$E[\mathbb{X}] = \frac{1}{2}\sum_{v\in V(G)} E[\mathbb{X}_v] \leq \frac{n}{2} \cdot \frac{2t^2}{(1+\epsilon)} \leq \frac{16n\ln^2 n}{\epsilon^2}.$$

By Markov's inequality, $\Pr[\mathbb{X} > \frac{1600n\ln^2 n}{\epsilon^2}] \leq \frac{1}{100}$. Thus, $|E(H)| \leq \frac{1600}{\epsilon^2} n\ln^2 n$ with probability at least 0.99. ◀

C Streaming algorithm for CFON coloring using $O(\sqrt{n})$ colors

Cheilaris proved that, for a graph G on n vertices, $\chi_{ON}(G) \leq 2\sqrt{n}$ [14]. Pach and Tardos improved this to $\chi_{ON}(G) \leq \sqrt{2n}$ [23]. The above upper bounds are asymptotically tight due to Example 4. Cheilaris's result in Proposition 3 uses a deterministic algorithm to do a coloring of the vertices as follows. When a vertex is present in at least $\sqrt{n}$ neighborhoods, then we color that vertex with a new distinct color. Note that this vertex will be a unique color for at least $\sqrt{n}$ neighbourhoods. Next, we consider another vertex that is present in at least $\sqrt{n}$ neighborhoods with no vertex in this neighborhood colored so far. Then, we color that vertex with a new distinct color and continue this process. At the end of this process, we have used at most $\sqrt{n}$ colors because there are only n neighborhoods. Also, at the end of this process, each vertex will be part of at most $\sqrt{n}$ neighborhoods with no vertex in these neighborhoods colored so far. Then we use Proposition 1 to color the remaining vertices using at most $\sqrt{n}$ new colors. Clearly, this algorithm requires to see the entire input to do the coloring.

In this section, we give a streaming algorithm to do a CFON coloring of G using $2\sqrt{n}$ colors. We prove the following result.

▶ **Theorem 18.** *There is a deterministic single-pass $O(n\sqrt{n})$-space streaming algorithm that, given a graph G on n vertices, finds a CFON coloring using at most $2\sqrt{n}$ colors.*

Algorithm 5 Algorithm for CFON coloring of a graph G on n vertices using $2\sqrt{n}$ colors.

Input: Stream of edges of G, where $V(G) = [n]$.
Output: coloring function $f : V(G) \to [\lceil 2\sqrt{n} \rceil]$.

Set $\mathcal{V} := V(G)$;
Set $L(v) := \emptyset$, for all $v \in V(G)$;
Set $i := 1$;
Set marked$[v]$:= false, for all $v \in V(G)$;
`/* marked[v] denotes if` $N_G(v)$ `has a unique colored member or not */`
while *an edge* $e = (u, v)$ *arrives in the stream* **do**
  if v *is colored* **then**
    marked$[u]$:= true;
    $L(u) := \emptyset$;
    `/*` v `is a uniquely colored vertex in` $N_G(u)$. `*/`
  if u *is colored* **then**
    marked$[v]$:= true ;
    $L(v) := \emptyset$;
    `/*` u `is a uniquely colored vertex in` $N_G(v)$. `*/`
  `/* Below, we update the lists` $L(u)$ `and` $L(v)$ `based on whether` u `and` v `are unmarked or not. */`
  if *marked*$[v]$ = *false* **then**
    $L(v) := L(v) \cup \{u\}$;
  if *marked*$[u]$ = *false* **then**
    $L(u) := L(u) \cup \{v\}$;
  if v *is not colored and* $|\{w \ : \ v \in L(w)\}| \geq \sqrt{n}$ **then**
    `/* Below, we color` v `*/`
    $f(v) := i$;
    $i := i + 1$;
    $\mathcal{V} := \mathcal{V} \setminus \{v\}$;
    $L(z) := \emptyset$ and *marked*$[z]$:= true, for all z such that $v \in L(z)$;
    `/*` v `is a uniquely colored vertex in` $N_G(z)$. `*/`
  if u *is not colored and* $|\{w \ : \ u \in L(w)\}| \geq \sqrt{n}$ **then**
    `/* Below, we color` u `*/`
    $f(u) := i$;
    $i := i + 1$;
    $\mathcal{V} := \mathcal{V} \setminus \{u\}$;
    $L(z) := \emptyset$ and *marked*$[z]$:= true, for all z such that $v \in L(z)$;
    `/*` u `is a uniquely colored vertex in` $N_G(z)$. `*/`
Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be the hypergraph, where $\mathcal{E} = \{L(v) : v \in V(G),\ marked[v] =$ false$\}$. Note that the maximum degree of any vertex in $\mathcal{H}$ is at most $\sqrt{n} - 1$. Apply the greedy coloring algorithm given by Proposition 1 to do a CF coloring of $\mathcal{H}$ using $\sqrt{n}$ colors from the set $\{i + 1, \ldots, i + \sqrt{n}\}$.
Output f.

Proof. The pseudocode of our algorithm is given in Algorithm 5. We explain the algorithm here. We use $\mathcal{V}$ to denote the set of all so-far uncolored vertices. Initially, as all the vertices in $V(G)$ are uncolored, we initialize $\mathcal{V} := V(G)$. We "mark" a vertex v if it sees some unique

coloring in $N_G(v)$. Initially, all the vertices are unmarked. For an unmarked vertex v, we shall use the list $L(v)$ to denote the set of neighbors of v that we have seen so far. From the moment v is marked, we maintain $L(v)$ to be an empty set as the "marked" status of v implies it is seeing a uniquely colored vertex in its open neighborhood. When a new edge (u, v) arrives in the stream, the algorithm executes Steps 6 to 27 which are explained below. Note that in these steps while coloring vertices we do not use the same color twice. If u (or v) is already colored then we mark v, (resp., u) as u (resp., v) will act as the uniquely colored vertex in the open neighborhood of v (resp., u). As mentioned above, as soon as a vertex is marked, its list (in this case, $L(u)$ or $L(v)$ or both) is reset to the empty set. These steps are executed in Lines 6 to 11. We update the list of u (or v), if it is unmarked, in Lines 12 to 15. If v is not colored and is present in at least $\sqrt{n}$ lists, then we assign a new color to v. We remove it from $\mathcal{V}$ which maintains the set of all uncolored vertices so far. We mark all vertices z that have v in $L(z)$, as v will act as the uniquely colored neighbor of such a vertex z. As mentioned before, we immediately reset $L(z)$ to the empty set. These steps are implemented in Lines 16 to 21. Now, in a similar fashion, if u is not colored and is present in at least $\sqrt{n}$ lists, we execute similar steps for the vertex u in Lines 22 to 27.

We claim that we color at most $\sqrt{n}$ vertices in total during the various iterations of the **while** loop (Lines 6 to 27). Every time we color a new vertex in Line 18 or Line 24, we update the status of least $\sqrt{n}$ new vertices from unmarked to marked and set their lists to $\emptyset$. Since there are only n vertices in total and since we do not unmark a vertex that is already marked, our claim holds. Next, we claim that every vertex that is marked sees some color exactly once in its open neighborhood. Every time a vertex is marked in Lines 10, 21, or 27, it has some colored vertex in its open neighborhood (see the comments below these lines). Since no two vertices receive the same color inside the **while** loop, our claim holds. To summarise, (i) we have used at most $\sqrt{n}$ colors so far, and (ii) every marked vertex is seeing some color exactly once in its open neighborhood.

When we exit the **while** loop, it is only the unmarked vertices that are yet to see a unique color in their open neighborhood. We take care of them in Line 28. In this step, we construct a hypergraph $\mathcal{H}$ whose vertex set is the set of so far uncolored vertices (maintained by $\mathcal{V}$) and whose edge set $\mathcal{E}$ is all the lists $L(v)$ for which v is unmarked. In other words, $\mathcal{E}$ is the set of all lists $L(v)$ that are not empty sets. Clearly, such lists $L(v)$ contain only uncolored vertices (that is, vertices in $\mathcal{V}$); else v would have been marked and $L(v)$ would have been reset to $\emptyset$. Further, for any unmarked vertex v, $L(v) = N_G(v)$ as our algorithm added every neighbor of v into $L(v)$ as and when they were unveiled (see Lines 13 and 15). Thus, any conflict-free coloring of $\mathcal{H}$ using a set of new colors would result in every unmarked vertex seeing a unique color in its open neighborhood in G. We claim that the maximum degree of any vertex in $\mathcal{H}$ (that is, the maximum number of hyperedges any vertex in $\mathcal{V}$ is present in) is at most $\sqrt{n} - 1$ and then use Proposition 1 to do a greedy conflict-free coloring of $\mathcal{H}$ using at most $\sqrt{n}$ new colors. We prove the claim by contradiction. Suppose there existed a vertex $v \in \mathcal{V}$ which was present in $\sqrt{n}$ or more hyperedges (or lists) $L(w)$ in $\mathcal{H}$. But, then such a vertex v would satisfy the conditions of the **if** statement in Line 16. Line 20 would then lead to the contradictory situation that v is not a member of $\mathcal{V}$. This proves the claim.

We have thus shown that the algorithm does give a CFON coloring of G using at most $2\sqrt{n}$ colors. It is easy to see that it is a single-pass, deterministic algorithm. Finally, the amount of memory required is the space required to store the lists $L(u)$ for all u. Notice that each time during the execution of the algorithm each vertex v is present in at most $\sqrt{n}$ lists. This implies that the space complexity is $O(n\sqrt{n})$. ◀