Streaming Edge Coloring with Asymptotically Optimal Colors

Mohammad Saneian
Northeastern University, Boston, MA, USA

Soheil Behnezhad
Northeastern University, Boston, MA, USA

---- **Abstract** ----

Given a graph G, an *edge-coloring* is an assignment of colors to *edges* of G such that any two edges sharing an endpoint receive different colors. By Vizing's celebrated theorem, any graph of maximum degree Δ needs at least Δ and at most $(\Delta + 1)$ colors to be properly edge colored. In this paper, we study edge colorings in the *streaming* setting. The edges arrive one by one in an *arbitrary* order. The algorithm takes a single pass over the input and must output a solution using a much smaller space than the input size. Since the output of edge coloring is as large as its input, the assigned colors should also be reported in a streaming fashion.

The streaming edge coloring problem has been studied in a series of works over the past few years. The main challenge is that the algorithm cannot "remember" all the color assignments that it returns. To ensure the validity of the solution, existing algorithms use many more colors than Vizing's bound. Namely, in n-vertex graphs, the state-of-the-art algorithm with $\widetilde{O}(ns)$ space[1] requires $O(\Delta^2/s + \Delta)$ colors. Note, in particular, that for an asymptotically optimal $O(\Delta)$ coloring, this algorithm requires $\Omega(n\Delta)$ space which is as large as the input. Whether such a coloring can be achieved with sublinear space has been left open.

In this paper, we answer this question in the affirmative. We present a randomized algorithm that returns an asymptotically optimal $O(\Delta)$ edge coloring using $\widetilde{O}(n\sqrt{\Delta})$ space. More generally, our algorithm returns a proper $O(\Delta^{1.5}/s + \Delta)$ edge coloring with $\widetilde{O}(ns)$ space, improving prior algorithms for the whole range of s.

1 Introduction

Motivated by its applications in processing massive graphs, the *graph streaming model* has gained significant attention over the past two decades. This model assumes that the input graph is too large to be stored in memory and is presented to the algorithm in a sequence of edges. Many graph problems have been studied in this model, including maximum matching [7, 36, 5, 28, 6], graph connectivity [2, 13], vertex coloring [10, 12, 9, 19, 21, 8], edge coloring [17, 22, 4], cut and spectral sparsifiers [3, 37, 11, 24, 38], and graph clustering [15, 16, 26, 1] among many others (this is by no means a comprehensive list). In this paper, we continue the line of work on the *edge coloring* problem in the streaming model. The goal of edge coloring is to assign colors to edges of a graph such that no two adjacent edges share the same color.

---

[1] Here and throughout the paper, $\widetilde{O}(f) = O(f \cdot \operatorname{poly} \log n)$.

Since the output of edge coloring is as large as its input, a streaming algorithm cannot return it in memory. Instead, the goal is to also return the solution in a streaming fashion. Doing so, the main challenge is that the algorithm cannot "remember" all the reported edge colors, yet has to ensure that any two incident edges receive different colors.

Edge coloring is a fundamental problem in graph theory and has many practical applications in areas such as scheduling, communication networks, and VLSI design. By a classic result of Vizing, any graph of maximum degree Δ needs at least Δ and at most $(\Delta + 1)$ colors to be properly edge colored.[2] While existing algorithms for finding a $(\Delta + 1)$ edge coloring are rather complicated, a $(2\Delta - 1)$ coloring can be found by a simple greedy algorithm that iterates over the edges and chooses an arbitrary available color for each. Unfortunately, even this simple greedy algorithm is hard to implement in the streaming setting. Recall that the algorithm cannot remember all the assigned edge colors in memory, hence it is unclear how to verify which color is available for the next edge that arrives.

Prior works.   The streaming edge coloring problem was first studied by Behnezhad, Derakhshan, Hajiaghayi, Knittel, and Saleh [17] who gave a randomized algorithm for $O(\Delta^2)$ edge-coloring with $\widetilde{O}(n)$ space, where n is the number of vertices. In a follow up work, Charikar and Liu [22] showed that, more generally, for any parameter $s \geq 1$ there is a randomized streaming algorithm that $O(\Delta^2/s + \Delta)$ edge-colors the graph using $\widetilde{O}(ns)$ space. Later, Ansari, Saneian, and Zarrabi-Zadeh [4] obtained the same bound but using a simple and clean *deterministic* algorithm. Note that for an asymptotically optimal $O(\Delta)$ edge coloring, the algorithms above require $O(n\Delta)$ space. This, unfortunately, is *not* sublinear in the input size as any graph of maximum degree Δ has at most $O(n\Delta)$ edges. Put differently, for the case of $O(\Delta)$ coloring, no improvement over the trivial algorithm that stores the whole graph in memory and then colors it is known. This state of affairs leaves an important question open:

Does there exist a streaming algorithm with sublinear space for $O(\Delta)$ edge coloring?

Our contribution.   In this paper, we answer the question above in the affiramtive. Our main result is the following algorithm:

▶ **Theorem 1.1.** *For any $s \geq 1$, there is a randomized streaming algorithm that with high probability reports a $O(\Delta^{1.5}/s + \Delta)$ edge-coloring under arbitrary edge arrivals using $\widetilde{O}(ns)$ space.*

Setting $s = \sqrt{\Delta}$, we obtain the following corollary, answering the question above:

▶ **Corollary 1.2.** *There is a randomized streaming algorithm that with high probability reports a $O(\Delta)$ edge-coloring under arbitrary edge arrivals using $\widetilde{O}(n\sqrt{\Delta})$ space.*

We show that the space-complexity can be further improved to $\widetilde{O}(n)$ under arbitrary *vertex* arrivals. Here instead of edges arriving in an arbitrary order, the vertices of the graph arrive one by one and when a vertex v arrives, all of its edges to the previous vertices are revealed.

---

[2] To see the lower bound, note that Δ colors are needed just to color the edges of a vertex with degree Δ.

▶ **Theorem 1.3.** *There is a randomized streaming algorithm that $O(\Delta)$ edge colors the graph under arbitrary vertex arrivals using $\widetilde{O}(n)$ space.*

1.1 Further Related Work

Streaming algorithms for edge coloring have also been studied under the extra assumption that the edges arrive in a random order. Behnezhad, Derakhshan, Hajiaghayi, Knittel, and Saleh [17] showed that there is a single pass $\widetilde{O}(n)$ space algorithm that obtains a 5.44Δ coloring. Charikar and Liu [22] later showed the number of colors can in fact be improved to $(1 + o(1))\Delta$ under random arrivals while keeping the memory $\widetilde{O}(n)$. Both of these algorithms rely heavily on the random-arrival assumption and do not have any implications for adversarial edge arrivals, which is the focus of this paper.

Online edge-coloring is another related problem. In this problem, the algorithm has no space constraints, but edges arrive one by one and each edge has to be colored upon arrival irrevocably. Note that the greedy algorithm can easily be implemented in the online setting; therefore much of the research has been focused on whether the greedy bound can be improved. For low-degree graphs, Bar-Noy, Motwani, and Naor [14] showed that the greedy algorithm is optimal for online edge coloring. However, there has been several improvements over the past few years for graphs of degree at least $\omega(\log n)$ starting from the work of Cohen, Peng, and Wajc [25]; see [25, 41, 20, 40] and the references therein. The best current bound for general edge arrivals is a beautiful $\frac{e}{e-1}\Delta$ coloring algorithm of Kulkarni, Liu, Sah, Sawhney, and Tarnawski [40]. Whether the number of colors can be improved to $(1 + o(1))\Delta$ for $\Delta = \omega(\log n)$ under arbitrary edge arrivals remains an important open problem.

Independent work. Two other works by Ghosh and Stoeckl [32] and Chechik, Mukhtar, and Zhan [23], which appeared concurrently with our paper, also consider the streaming edge coloring problem. The algorithm of Ghosh and Stoeckl [32], for any $1 \leq s \leq \sqrt{\Delta}$, obtains a (Δ^2/s^2)-edge-coloring using $\widetilde{O}(ns)$ space.[3] The algorithm of Chechik, Mukhtar, and Zhan [23] obtains a $\widetilde{O}(\Delta^{1.5})$-edge-coloring using $\widetilde{O}(n)$ space.

Our Theorem 1.1 subsumes both of these results: for any $1 \leq s \leq \sqrt{\Delta}$, we achieve a $(\Delta^{1.5}/s)$-edge-coloring using $\widetilde{O}(ns)$ space. Letting $s = 1$, this gives an $O(\Delta^{1.5})$-edge-coloring with $\widetilde{O}(n)$ space matching the result of Chechik, Mukhtar, and Zhan [23]. Moreover, since $\Delta^{1.5}/s \leq \Delta^2/s^2$ for the whole range of $1 \leq s \leq \sqrt{\Delta}$, our color/space trade-off is never worse than that of Ghosh and Stoeckl [32], but is strictly better when s is small. For instance, letting $s = 1$, the algorithm of Ghosh and Stoeckl [32] requires $O(\Delta^2)$ colors whereas ours requires $O(\Delta^{1.5})$ colors.

Finally, we note that Ghosh and Stoeckl [32] also consider vertex arrival streaming algorithms achieving the same bound as our Theorem 1.3. They also consider online algorithms.

1.2 Preliminaries

Unless otherwise stated, we use $G = (V, E)$ to denote the input graph. We use $n := |V|$ and $m := |E|$ to respectively denote the number of vertices and edges in G. We use Δ to denote the maximum degree of the graph G. For any integer k, we use $[k]$ to denote the set $\{1, \ldots, k\}$.

---

[3] In their paper this is equivalently stated as an $O(\Delta t)$ coloring using $\widetilde{O}(n\sqrt{\Delta/t})$ space.

The graph streaming model. In the standard graph streaming model, edges of an arbitrary graph G arrive one by one in an arbitrary order. The algorithm has a space much smaller than the total number of edges, can take few – preferably just one – pass over the input, and should return the output. In the case of edge coloring, the output size is as large as the input, making it impossible to store the entire output and return it all at once. Therefore, we allow the algorithm to output the solution in a streaming manner as well. This model is also referred to as the "W-streaming" model in the literature [33, 27]. All of the algorithms presented in this paper take only a single pass over the input. We measure the space in the number of words, each consisting of $\Theta(\log n)$ bits.

Our algorithms for the general edge arrival model build on algorithms that we develop for two more restrictive arrival models of *general vertex arrivals* and *one-sided vertex arrivals in bipartite graphs*. We present the definition of these standard models below.

▶ **Definition 1.4** (vertex arrival model). *In this model, vertices of the input graph G arrive one by one according to some arbitrary permutation π. Upon arrival of a vertex, all of its edges to previous vertices in the permutation π arrive.*

▶ **Definition 1.5** (one-sided vertex arrivals in bipartite graphs). *In this model, the input graph G is assumed to be bipartite with vertex sets U and V. The "offline" vertices in V are present from the beginning, but the "online" vertices in U arrive one by one in an arbitrary order. Every time an online vertex u arrives, all of its edges to the offline vertices V are revealed.*

In our proofs, we use the following standard variant of the Chernoff bound.

▶ **Proposition 1.6** (Chernoff bound). *Let $X_1, ..., X_n$ be independent random variables in $[0, 1]$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. Then for all $\delta \geq 0$ and $\mu' \geq \mu$, $\Pr[X \geq (1 + \delta)\mu'] \leq \exp\left(-\frac{\delta^2}{2+\delta}\mu'\right)$.*

2 Overview of Techniques

In this section, we give an informal high-level overview of our algorithms.

As we discussed, the main challenge in solving the streaming edge-coloring problem is that the algorithm cannot "remember" all the colors that we assign to the edges as this takes too much space. This turns out to be a challenge particularly when the degrees evolve unevenly. To convey the key intuitions in this section, let us first focus on the one-sided vertex arrival model in bipartite graphs (Definition 1.5). We note that even in this restricted arrival model, the best known algorithm from the literature remains to be those of [17, 22, 4] which require $O(\Delta^2)$ colors with $O(n)$ space. Here, we describe how this can be improved to an asymptotically optimal $O(\Delta)$ coloring with only $\widetilde{O}(n)$ space. Our final algorithm of Theorem 1.1 in the more general edge-arrival model builds on this vertex-arrival algorithm.

Since all edges of an online vertex arrive at the same time, it is not hard to ensure they receive different colors. What is challenging is to do so while ensuring that all edges of an *offline* vertex receive different colors too. Towards this, we first describe an algorithm that uses $O(\Delta \log n)$ colors, $O(n\Delta)$ "pre-processing space," and $O(n)$ working space. We then show how this can be turned into an $O(\Delta)$ coloring algorithm that uses $\widetilde{O}(n)$ space overall.

With regards to the second reviewer's question, for the streaming model, we measure the space in terms of the number of words. We point this out in the preliminaries section. In the paragraph describing the algorithm that uses $O(\triangle \log n)$ colors the counters keep track of a number which is at most $K = \Theta(\Delta \log n)$. Thus it can be stored in $O(\log(K)) = O(\log n)$ bits of space. Since we set the size of words to be $\Theta(\log n)$, each counter can be stored in $O(1)$ space. Hence, to store counters for all the $O(n)$ vertices we need $O(n)$ space.

An algorithm with $O(\Delta \log n)$ colors but large space. Let $K = \Theta(\Delta \log n)$ be the number of colors we use. In the pre-processing step (i.e., before seeing any edges of the graph), for every offline vertex $v \in V$, we store a random permutation π_v of the colors $\{1, \ldots, K\}$ in memory which overall takes $\widetilde{O}(n\Delta)$ pre-processing space. Now suppose that the first online vertex u arrives. For each of its offline neighbors v_i, we consider the first random color $\pi_{v_i}(1)$ of v_i. Since the random permutations of offline vertices are independent, when $\Delta \geq 2$ we get the color chosen for v_i is different from $\pi_{v_j}(1)$ for all other neighbors v_j of u with probability $(1 - \frac{1}{K})^{\deg(u)} \geq (1 - \frac{1}{\Delta})^{\Delta} \geq 0.25$. Note that $(1 - \frac{1}{\Delta})^{\Delta}$ is a monotonically increasing and for $\Delta = 2$ the value is 0.25. If this happens, we assign color $\pi_{v_i}(1)$ to edge (u, v_i). If the first color of v_i is not unique, we discard it and reveal the next color $\pi_{v_i}(2)$ of v_i. Every round of this process successfully colors a constant fraction of the remaining uncolored edges of u. Thus it takes $O(\log n)$ rounds to color all edges of u w.h.p. On the other hand, since we have reserved $K = \Theta(\Delta \log n)$ colors for each offline vertex, we can afford to reveal up to $\Theta(\log n)$ colors for each of their edges, hence fully coloring the graph w.h.p. To implement this algorithm, we only need to maintain a counter d_v for every offline vertex v on how many colors of its random permutation we have revealed so far, which can be done with $O(n)$ total working memory.

Reducing space. To get rid of the huge pre-processing space of $\widetilde{O}(n\Delta)$ in the previous algorithm, we limit the amount of randomness needed. To do so, instead of choosing a fully random permutation of $\{1, \ldots, K\}$ for every offline vertex v, which requires $\Theta(nK) = \widetilde{\Theta}(n\Delta)$ space, we just pick a random number $r_v \in [K]$ and use the randomly *shifted* permutation $(r_v, r_v+1, \ldots, K, 1, \ldots, r_v-1)$. The advantage of doing so is that this shifted permutation can be stored using $O(1)$ space, by just storing the random number r_v in memory. Its downside is that lack of independence breaks the analysis above. For example, if $\pi_{v_i}(1) = \pi_{v_j}(1)$ then we also have $\pi_{v_i}(2) = \pi_{v_j}(2)$ and so cannot argue that every round colors a constant fraction of the edges of the online vertex u. To fix this, we take $t = \Theta(\log n)$ random numbers $r_v^1, \ldots, r_v^t$ for each offline vertex v and considering the t shifted permutations

$$[r_v^1, \ldots, \Delta, 1, \ldots, r_v^1 - 1], \ldots, [r_v^t, \ldots, \Delta, 1, \ldots, r_v^t - 1].$$

Now if the first color choice of v_i and v_j according to the first shifted permutation are the same, we consider the second shifted permutations, then the thirds, and so on and so forth. This gets rid of the dependence between the colors proposed for the edges of an online vertex, but not the edges of an offline vertex. Luckily, the latter is not needed for the analysis to go through and we can implement this algorithm with $\widetilde{O}(n)$ space overall.

Reducing colors via k-out sampling. The algorithm above is greedy in that we first reveal a random color for each edge of the online vertex u, greedily color those whose proposed colors are unique, then reveal the next batch of proposals. Instead of this greedy algorithm which takes up to $O(\log n)$ rounds, inevitably stretching the number of colors to $O(\Delta \log n)$, we first draw 3 random colors x_i^1, x_i^2, x_i^3 for each edge $e_i = (u, v_i)$ of the online vertex u and consider all these 3 colors *at the same time*. We show that with probability $1 - 1/\text{poly}(\Delta)$, it is possible to pick one of the colors x_i^1, x_i^2, x_i^3 for each edge e_i of u such that all edges of u receive different colors. Our proof of this theorem builds on a new lemma (Lemma 3.6) that we prove on the existence of perfect matchings in a *one-sided random k-out model*. This lemma, which might be of independent interest, says that if we have a random bipartite graph with vertex sets V and U, and every vertex in V is made adjacent to exactly k vertices in U chosen uniformly, then G has a perfect matching provided that $k \geq 3$ and $|U| > e|V|$.

While k-out sampling has been used recently in obtaining more efficient algorithms for distributed graph connectivity [34] and minimum cuts [31, 11], this is to our knowledge its first application in graph coloring.

From vertex-arrivals to edge-arrivals. We now overview how we go from the vertex arrival model to the more general edge arrival model. To convey the key intuitions, we focus only on the simpler special case of Theorem 1.1 where $s = 1$. That is, an algorithm that uses $O(n)$ space and returns an $O(\Delta^{1.5})$ edge coloring. Our first step is to generalize the algorithm above to work in the non-bipartite vertex arrival model (Definition 1.4). This step, in fact, follows more or less from a known random bipartization technique of the literature that we present in Appendix A. Our next, more challenging, step is to generalize the algorithm to a *batch* arrival setting, where at each step instead of all of the edges of a vertex, we see some $\Theta(\sqrt{\Delta})$ edges of it (without having any guarantee about when the next batch of this vertex arrives). Once we achieve this more general algorithm, we run an instance $\mathcal{A}$ of it. To feed our edges to this batch arrival algorithm $\mathcal{A}$, we keep storing edges in a set H. Whenever the number of edges in H reaches n, we look at the vertex v with the largest degree in H. If $\deg_H(v) \geq \sqrt{\Delta}$, we feed $\Theta(\sqrt{\Delta})$ of these edges of vertex v as the next batch to algorithm $\mathcal{A}$ and remove them from H. Otherwise, the maximum degree in H is less than $\sqrt{\Delta}$; in this case, we edge color all edges in H greedily using $O(\sqrt{\Delta})$ colors and remove them all from H. Every time that we color H greedily, we color n edges of the graph. Therefore this happens at most $m/n = O(\Delta)$ times, requiring a total of $O(\Delta^{1.5})$ colors. The proof of Theorem 1.1 for larger values of s requires a more involved white-box application of the vertex-arrival algorithm; see the edge arrival section in the full version of our paper [18] for the details of the algorithm.

3 Streaming Edge Coloring Under Vertex Arrivals

In this section, we start by proving Theorem 1.3, restated below, for the streaming edge-coloring under vertex arrivals. Our algorithm in the more general edge-arrival model builds on the vertex-arrival algorithm that we describe in this section.

▶ **Theorem 1.3** (restated). *There is a randomized streaming algorithm that $O(\Delta)$ edge colors the graph under arbitrary vertex arrivals using $\widetilde{O}(n)$ space.*

3.1 Basic Reductions

We start with two basic reductions that essentially reduce the edge coloring problem in general graphs under vertex arrivals to the same problem in bipartite graphs under one-sided vertex arrivals.

The following lemma asserts that instead of general graphs, we can focus on bipartite graphs. The idea is to randomly partition the vertex set V into two sets A and B, edge color the bipartite subgraph of G between A and B, and recurse on the induced subgraphs $G[A]$ and $G[B]$. Since the idea is standard and follows from known reductions, we present its proof in Appendix B.

▶ **Lemma 3.1.** *Suppose there is a streaming algorithm that edge colors any n-vertex bipartite graph of maximum degree Δ with $f(\Delta) \geq \Delta$ colors using $s(n, \Delta)$ words of space. Then there is a streaming algorithm that edge colors any general (i.e., not necessarily bipartite) n-vertex graph of maximum degree Δ using*

$$f(\Delta) + f(\Delta/2) + f(\Delta/4) + \ldots + f(1)$$

colors and $\widetilde{O}(n + s(n, \Delta))$ space. If the algorithm for bipartite graphs works under vertex arrivals (resp. edge arrivals), then the algorithm for general graphs also works under vertex arrivals (resp. edge arrivals).

We emphasize that to solve general graphs under vertex arrivals, we have to ensure that the bipartite algorithm that we feed into Lemma 3.1 works under two-sided vertex arrivals. The next reduction shows that any one-sided vertex arrival algorithm also solves two-sided vertex arrivals by losing a factor of 2 in the number of colors.

▷ **Claim 3.2.** Suppose there is a streaming algorithm A, that edge colors any n-vertex bipartite graph of maximum degree Δ under one-sided vertex arrivals using $f(\Delta)$ colors. Then this can be turned into an algorithm that $2f(\Delta)$ colors the bipartite graph under two-sided vertex arrivals using asymptotically the same space.

Proof. Let $G = (V, U, E)$ be the bipartite graph under two-sided vertex arrivals. We partition its edges into two subgraphs G_U and G_V. If a vertex in V arrives, we add its edges to G_V, and if a vertex in U arrives, we add its edges to G_U. This way, G_V (resp. G_U) will be a bipartite graph under one-sided vertex arrivals with the online part being vertex set V (resp. U). Since both G_U and G_V will have maximum degree upper bounded by Δ (as they are subgraphs of G), we can run one instance of A on G_U and one on G_V in parallel using disjoint colors. This only multiplies the number of colors by two and the space by two, finishing the proof. ◁

3.2 The Algorithm

The reductions of the previous section show that instead of general graphs, we can focus on bipartite graphs under one-sided vertex arrivals. The following lemma is our main contribution in the rest of Section 3.

▶ **Lemma 3.3.** *There is a streaming algorithm that edge colors any bipartite graph of maximum degree Δ, using $O(\Delta)$ colors under one-sided vertex arrivals and uses $O(n)$ space w.h.p. (The coloring is always valid and the space-bound holds with probability $1 - 1/\operatorname{poly}(n)$.)*

Let us first see how Lemma 3.3 proves Theorem 1.3 using the reductions of the previous section.

Proof of Theorem 1.3. First, we feed the algorithm of Lemma 3.3 to Claim 3.2 to obtain a streaming algorithm that for some constant C, $C(\Delta)$ colors a bipartite graph under two-sided vertex arrivals with $O(n)$ space. We then plug in this algorithm into Lemma 3.1 to obtain an algorithm for general graphs. The algorithm uses $\widetilde{O}(n)$ space and the number of its colors by Lemma 3.1 is

$$C\Delta + C(\Delta/2) + C(\Delta/4) + \ldots + C \le 2C\Delta = O(\Delta). \qquad \blacktriangleleft$$

So it only remains to prove Lemma 3.3. In this section, we present the algorithm formalized below as Algorithm 1. We analyze the space complexity of the algorithm in Section 3.3 and analyze its correctness and the number of used by it in Section 3.4. Finally, we show in Section 3.5 that the algorithm can be generalized to a batch arrival model as well.

■ **Algorithm 1.** Streaming edge coloring for one-sided vertex arrivals in bipartite graphs.

Parameter: $c := 2.72$
1. For each offline vertex v draw 3 *distinct* random numbers r_v^1, r_v^2 and r_v^3 from $[c\Delta]$ uniformly and store them in memory.[a]
2. Additionally, for each offline vertex v, we store a counter $\deg_v$ in memory to keep track of its degree as the edges arrive.
3. Upon arrival of an online vertex u:
   a. For each edge $e_i = (u, v_i)$ consider the following three colors x_i^1, x_i^2, x_i^3:

   $$x_i^1 := \left((r_{v_i}^1 + \deg_{v_i}) \bmod c\Delta\right),$$
   $$x_i^2 := \left((r_{v_i}^2 + \deg_{v_i}) \bmod c\Delta\right) + c\Delta,$$
   $$x_i^3 := \left((r_{v_i}^3 + \deg_{v_i}) \bmod c\Delta\right) + 2c\Delta.$$

   b. If there is an assignment of colors to edges of u such that each edge e_i receives a color from $\{x_i^1, x_i^2, x_i^3\}$ and all the edges of u receive different colors, then we assign these colors, stream them out, and remove edges of u from memory. Otherwise, we add all edges of u to set S which we store in memory.
   (In Claim 3.4 we show u's edges are successfully colored with probability $1 - O(1/\Delta^5)$.)
4. It only remains to color the edges in S. Note that since S is a subgraph of G, its maximum degree is no larger than Δ. Thus we can edge-color it using Δ fresh colors via existing offline edge-coloring algorithms for bipartite graphs; see e.g. [30].

---

[a] For brevity, here and in the rest of the analysis we assume that $c\Delta$ is an integer. If it is not, $c\Delta$ must be replaced by $\lceil c\Delta \rceil$.

3.3 Space Complexity

In Algorithm 1, for every vertex, we keep its degree and three random numbers which together can all be stored with $O(n)$ words. The only non-trivial part of the space that we need to bound is the size of the set S of the edges that we store in memory and color at the end. Our main result in this section is to show that the set S has size $O(n)$ w.h.p. We start by bounding the expected size of S in Claim 3.4 using a connection to k-out subgraphs (formalized in Lemma 3.6). We then prove a high probability bound on the size of S in Claim 3.7.

▷ **Claim 3.4.** Take any online vertex u. The probability that we store the edges of u in Step 3b of Algorithm 1 is at most $O(1/\Delta^5)$. This, by linearity of expectation, implies that

$$\mathbb{E}|S| \leq O(n/\Delta^4).$$

Proof. Our main idea to prove this claim is to use the maximum matching problem on an appropriately defined "color graph" defined below. When an online vertex u arrives in the input we construct the bipartite graph H_u in the following way.

▶ **Definition 3.5** (the color graph). *For any online vertex u, we define the color graph $H_u = (X, Y, E)$ as follows. The set X corresponds to the edges of u, i.e., for each edge (u, v_i), we have a vertex corresponding to v_i in X. For each color in the range $[c\Delta]$ we have a corresponding vertex in set Y. Each vertex in X has exactly three edges to Y which are to the three colors:*

$$y_i^1 := \big((r_{v_i}^1 + \deg_{v_i}) \bmod c\Delta\big),$$
$$y_i^2 := \big((r_{v_i}^2 + \deg_{v_i}) \bmod c\Delta\big),$$
$$y_i^3 := \big((r_{v_i}^3 + \deg_{v_i}) \bmod c\Delta\big),$$

where v_i is any offline vertex adjacent to u. Note that since $r_{v_i}^1, r_{v_i}^2, r_{v_i}^3$ are distinct in Algorithm 1, the three colors y_i^1, y_i^2, y_i^3 will be distinct as well.

We first describe that a perfect matching in H_u corresponds to a valid edge-coloring of all edges of u. Let M be a perfect matching in H_u. Suppose that the node corresponding to edge e_i is matched in M to color y_i^j (which is distinct for all e_i's). In this case, we assign color x_i^j to e_i where x_j^i is as defined in Algorithm 1. To see why all edges of u receive different colors note that

$$y_i^j = x_i^j \bmod c\Delta.$$

In addition, all matched y_i^j are unique because M is a matching. Therefore, if we find a perfect matching in the color graph H_u we can easily color edges incident to u using this matching.

Thus, it remains to show that H_u does indeed have a perfect matching (with large enough probability). We do so using the following Lemma 3.6 on existence of perfect matchings in a so-called k-out model. Lemma 3.6 proves that if each vertex of one side of a bipartite graph is made adjacent to 3 random vertices on the other side, then the graph has a perfect matching provided that the other side is sufficiently large. We note that by a classic result of Frieze [29] from 1986, if vertices in *both* sides of the bipartite graph pick k random edges, then the graph has a perfect matching w.h.p. if and only if $k \geq 2$. However, this does not hold when only one side of the graph pick random edges, which is the focus of the following Lemma 3.6.

▶ **Lemma 3.6.** *Consider a random bipartite graph G with vertex sets V and U where each vertex $v \in V$ is adjacent to exactly 3 distinct vertices in U picked uniformly (from all subsets of size 3 of U) and independently from the choice of the rest of vertices in V. If $|V| \leq n$ and $|U| = cn$ for fixed $c > e$, the graph G has a perfect matching with probability at least $1 - O(1/n^5)$.*

Proof. First, we argue that we can w.l.o.g. assume that $|V| = n$. Since if we have $|V| < n$ we can add some dummy vertices to side V and draw random neighbors for them to U. Any perfect matching in this new graph gives us a perfect matching in the original graph by removing its dummy edges. Therefore the probability of finding a perfect matching in this graph is no smaller than the case with $|V| = n$.

We let $c = (1 + \varepsilon)e$ for some fixed $\varepsilon > 0$. We assume that n is larger than any needed constant (possibly a function of ε). Note that if n is fixed, then by adjusting the constants we can ensure $1 - O(1/n^5) = 0$, making the probablistic statement of the lemma trivial.

By Hall's theorem, G has a perfect matching iff for every subset $S \subseteq V$ it holds that $|N_G(S)| \geq |S|$. This condition deterministically holds if $|S| \leq 3$ as every vertex in V has degree exactly 3. Therefore it suffices to show that it holds with high probability for all sets of size at least 4. Let us fix $4 \leq k \leq n$. The probability that all the edges of a vertex $v \in V$ go to a fixed subset U of size at most $k - 1$ in U is exactly $\binom{k-1}{3} / \binom{|U|}{3}$. Therefore, we have

$$\Pr[\text{exists } S \subseteq V : |S| = k \text{ and } |N(S)| < k] \leq \binom{n}{k} \cdot \binom{|U|}{k-1} \cdot \left(\frac{\binom{k-1}{3}}{\binom{|U|}{3}} \right)^k$$

$$\leq \left(\frac{ne}{k} \right)^k \cdot \left(\frac{cne}{k-1} \right)^{k-1} \cdot \left(\frac{(k-1)(k-2)(k-3)}{cn(cn-1)(cn-2)} \right)^k$$

$$\text{(since } \binom{n}{k} \leq \left(\frac{ne}{k} \right)^k \text{ for all } n \text{ and } k\text{)}$$

$$\leq \left(\frac{ne}{k} \right)^k \cdot \left(\frac{cne}{k-1} \right)^{k-1} \cdot \left(\frac{k-1}{cn/(1+0.1\varepsilon)} \right)^{3k}$$

(Here we use $cn/(1+0.1\varepsilon) \leq cn - 2$ which holds for n larger than some constant.)

$$= \frac{(1+0.1\varepsilon)^{3k} e^{2k-1} (k-1)^{2k+1}}{c^{2k+1} \cdot k^k \cdot n^{k+1}}$$

$$\leq \left(\frac{(1+0.5\varepsilon)e}{c} \right)^{2k-1} \cdot \left(\frac{k}{n} \right)^{k+1}$$

(Here we use $(1+0.1\varepsilon)^2 \leq 1+0.5\varepsilon$ for small enough ε)

$$\leq \left(\frac{1+0.5\varepsilon}{1+\varepsilon} \right)^{2k-1} \left(\frac{k}{n} \right)^{k+1}. \qquad \text{(Since } c = (1+\varepsilon)e.\text{)}$$

To finish the proof, note that as discussed, for $k \leq 3$ the Hall's guarantee holds deterministically. For $k = 4$, the inequality above is upper bounded by $(k/n)^{k+1} = O(1/n^5)$ and thus all subsets of size $k = 4$ in V have at least 4 neighbors in U with probability at least $1 - O(1/n^5)$. For each choice of $5 \leq k \leq n^{0.1}$, the inequality above is upper bounded by $(k/n)^{k+1} \leq n^{-0.9 \times 6} = n^{-5.4}$; a union bound over all $O(n^{0.1})$ such choices of k gives that with probability at least $1 - O(n^{-5.3})$, any set of size $5 \leq k \leq n^{0.1}$ in V has at least k neighbors in U as well. Finally, for the case where $n^{0.1} < k \leq n$, the inequality above is upper bounded by

$$\left(\frac{1+0.5\varepsilon}{1+\varepsilon} \right)^{2k-1} \left(\frac{k}{n} \right)^{k+1} \leq \left(\frac{1+0.5\varepsilon}{1+\varepsilon} \right)^{2k-1} = 2^{-O_\varepsilon(k)} = 2^{-O_\varepsilon(n^{0.1})} \ll O(1/n^6).$$

Thus, again by a union bound over all such choices of k, we get that with probability at least $1 - O(1/n^5)$, any set of size $n^{0.1} < k \leq n$ in V has at least k neighbors in U as well. Putting together all of these cases of k, we get that with probability at least $1 - O(1/n^5)$, Hall's condition holds and G has a perfect matching. ◄

Now observe that the color graph H_u meets the conditions of Lemma 3.6. There are at most Δ edges incident to u so $|X| \leq \Delta$. On the other hand $|Y| \geq c\Delta$. Moreover, for each edge $e_i = (u, v_i)$ its corresponding vertex in X is made adjacent to the vertices corresponding to colors y_i^1, y_i^2, y_i^3 as specified in Definition 3.5. Since the $r_{v_i}^j$'s are distinct and uniform, each vertex in X is made adjacent to 3 distinct uniform vertices in Y in graph H_u and so we can apply Lemma 3.6. This implies that H_u has a perfect matching with probability at least $1 - O(1/\Delta^5)$. This means that we store the edges of each online vertex in S with probability at most $O(1/\Delta^5)$, concluding the proof of Claim 3.4. ◁

It is worth noting that to find the maximum matching in H_u we use the algorithm of [35] that has space complexity of $O(|V| + |E|)$ which is $O(\Delta)$ here. We can delete this part of the memory before seeing the next online vertex.

While Claim 3.4 bounds the expected space of Algorithm 1, our next Claim 3.7 bounds the space by $O(n)$ w.h.p.

▷ **Claim 3.7.** Algorithm 1 uses $O(n)$ space with probability $1 - 1/\operatorname{poly}(n)$.

Proof. For an online vertex u, define X_u to be the indicator random variable where $X_u = 1$ iff we store edges of u in S. Additionally, define $X = \sum_{u \in U} X_u$. We know that

$$\Pr[|S| > O(n)] \leq \Pr[\Delta \cdot X > O(n)] = \Pr[X > O(n/\Delta)].$$

We know that $\mathbb{E}[X] \leq n \cdot O(1/\Delta^5)$ by Claim 3.4. However, since the X_u's are *not* independent we can not use Chernoff bound to provide a concentration bound on their sum. To see why, take two online vertices u_1, u_2 that share the same set of offline neighbors and come one after another in the input. If $X_{u_1} = 1$ then also $X_{u_2} = 1$ since the colors used for H_{u_2} are exactly the same as the colors used for H_{u_1} except they are shifted by one.

To bound X our plan is to divide all X_u's into groups such that in each group all the random variables are independent and then apply the Chernoff bound on each group separately. We know that if two online vertices u and w do not share any offline vertex as their neighbor then X_u and X_w are independent. This is because the colors specified in Step 3a of Algorithm 1 are all a function of the randomness on the set of neighbors an online vertex has. So if these sets are disjoint then X_u and X_w are independent.

Let us define the dependency graph between these variables as follows: The vertex set of this dependency graph is $\{X_u : u \in U\}$ and there is an edge between X_u, X_w with $u, w \in U$ iff u and w share an offline neighbor. The maximum degree in this graph is at most Δ^2 since each vertex $u \in U$ has at most Δ offline neighbors that each has at most Δ online neighbors.

Since any graph of maximum degree Δ' can be *vertex* colored using at most $\Delta' + 1$ colors via a simple sequential greedy algorithm, the dependency graph can be vertex colored via at most $\Delta^2 + 1$ colors. Doing so, note that vertices in each color class become independent random variables by definition of the dependency graph. Therefore, we can apply the Chernoff bound on the sum in each color class. With a slight abuse of notation, let $X_1, ..., X_t$ be a group of variables in a color class. Define $\mu = \mathbb{E}[X_1 + X_2 + ... + X_t]$. Since $\mathbb{E}[X_i] = O(1/\Delta^5)$ (by Claim 3.4) by linearity of expectation we get that for some constant C,

$$\mu = O(t/\Delta^5) \leq C \cdot t/\Delta^5. \tag{1}$$

Applying the Chernoff bound we get that for any $\delta > 0$ and $\mu' \geq \mu$,

$$\Pr[X_1 + X_2 + ... + X_t \geq (1+\delta)\mu'] \leq \exp\left(\frac{-\delta^2}{2+\delta}\mu'\right). \tag{2}$$

Let us call this color class *small* if $t \leq n/\Delta^3$ and *large* otherwise. For small groups, we do not need to prove any concentration bound, since a total of $(\Delta^2 + 1)(n/\Delta^3) = O(n/\Delta)$ vertices belong to small groups and each stores Δ edges in S, giving a deterministic upper bound of $O(n)$ on the number of such edges in S. It thus remains to analyze large groups only.

To deal with large groups we consider two cases for Δ.

Case 1: $\Delta = O((n/\log n)^{1/4})$. Letting $\delta = \Delta^4$ in Equation (2), we get:

$$\Pr\left[\sum_{i=1}^{t} X_i \geq (1+\Delta^4)\mu'\right] \leq \exp\left(\frac{-\Delta^8}{2+\Delta^4}\mu'\right).$$

Let $\mu' = C \cdot t/\Delta^5$. Note that $\mu \leq C \cdot t/\Delta^5 = \mu'$ and $t \geq \frac{n}{\Delta^3}$ since we are only considering large groups we get,

$$\exp\left(\frac{-\Delta^8}{2+\Delta^4}\mu'\right) \leq \exp\left(\frac{-\Delta^8}{2+\Delta^4} \cdot \frac{Cn}{\Delta^8}\right) = \exp\left(\frac{-Cn}{2+\Delta^4}\right).$$

Since we have $\Delta \leq O\left(\frac{n}{\log n}\right)^{1/4}$ the last term can be upper bounded by n^{-10}. Take $t_1, t_2, ..., t_k$ to be the size of the groups. We know that $k \leq n$; thus by a union bound over k, the probability that there is one group that deviates from its mean with a multiplicative factor of more than Δ^4 is at most n^{-9}. Therefore, with probability at least $1 - n^{-9}$, all groups deviate from their mean with a multiplicative factor of less than Δ^4. In this case, noting that $\sum_i t_i = n$, we get

$$X \leq \Delta^4 \cdot \sum_i \frac{C \cdot t_i}{\Delta^5} = O(n/\Delta).$$

Recalling that X is the total number of online vertices that store their edges in S, we get that the total number of edges in S is upper bounded by $\Delta \cdot O(n/\Delta) = O(n)$ w.h.p.

Case 2: $\Delta = \Omega((n/\log n)^{1/4})$. Notice that the probability of storing any u edges in S is $n^{-5/4+o(1)}$. By union bounding over all n online vertices, none will be stored in S with a probability of $O(n^{-0.25+o(1)})$. $\lhd$

3.4 Correctness and the Number of Colors

In this section, we discuss why any two adjacent edges receive different colors in our algorithm. We can ignore the edges colored in Step 4 of Algorithm 1 as they use a totally new set of colors from the rest of the edges. So we focus on the rest of the edges in the remainder of this section.

For an online vertex u, its edges by Step 3b all receive different colors (and if this is not possible, we store u's edges in S which we discussed can be ignored earlier). So it remains to prove that there are no conflicts for the offline vertices.

For an offline vertex v, firstly, note that for any edge e_i:

$$x_i^1 \in [0, c\Delta), \qquad x_i^2 \in [c\Delta, 2c\Delta), \qquad x_i^3 \in [2c\Delta, 3c\Delta).$$

Since these ranges are disjoint, for two edges to receive the same color they must be in the same range.

Note that when the algorithm is running $\deg_v$ is increasing by one after we see an edge of v. Let $x_i^1 = x_j^1$ for two edges of v where e_i arrives before e_j. Define α to be the value of $\deg_v$ when the edge e_i arrives and β to be the value of $\deg_v$ when the edge e_j arrives. Then we get that,

$$\left((r_v^1 + \alpha) \bmod c\Delta\right) = \left((r_v^1 + \beta) \bmod c\Delta\right),$$

which is equivalent to $\alpha - \beta \bmod c\Delta = 0$. Since $|\alpha - \beta| \leq \Delta$ and $c > 1$, this is a contradiction and thus we cannot have $x_i^1 \neq x_j^1$. Same can be applied to the second and the third ranges of colors. Therefore, offline vertices also recieve distinct colors on all their edges and the coloring our algorithm finds is a valid edge coloring.

Finally, it can be verified from Algorithm 1 that the number of colors used is $3c\Delta + \Delta = O(\Delta)$. This section, put together with the previous section concludes the proof of Lemma 3.3 and, as discussed, this finishes the proof of Theorem 1.3.

3.5 Generalization to Batch Arrivals

Our Algorithm 1 considers the vertex arrival model, i.e., all edges of an online vertex arrive at the same time. Here, we consider a more general *batch arrival* model that interpolates between edge arrivals and vertex arrivals. We show that our Algorithm 1 can be generalized to this model.

▶ **Definition 3.8** (one-sided batch arrival model). *In this model, we have a bipartite graph $G = (U, V, E)$ with online vertices U and offline vertices V. For a parameter k of the problem, edges arrive in batches of size k where all edges in the same batch are incident to the same online vertex u. The edges of a single online vertex v can arrive in up to Δ/k different batches. Importantly, there is no guarantee that different batches of the same online vertex arrive consecutively (otherwise the model would be equivalent to the vertex arrival model). We use $B_i(u)$ to denote the i'th batch of online vertex u.*

We prove that Algorithm 1 leads to the following bound in the one-sided batch arrival model. Note that setting $k = \Delta$ recovers Theorem 1.3, so this is only more general.

▶ **Lemma 3.9.** *There is a streaming algorithm that in the one-sided batch arrival model with batches of size k, uses $O(n)$ space and w.h.p. reports a proper $O(\Delta^2/k)$ edge coloring.*

Proof. Note that the only part of Algorithm 1 where we use the assumption that all edges of the online vertices arrive at the same is in Step 3b. There, the offline vertex of each edge e_i of u proposes three colors $\{x_i^1, x_i^2, x_i^3\}$ for e_i. Committing to choose a color from $\{x_i^1, x_i^2, x_i^3\}$ for each edge e_i already ensures there will be no conflicts among the edges of the offline vertices. On the other hand, our Lemma 3.6 guarantees that for each online vertex u, with probability $1 - O(1/\Delta^5)$, there is a proper coloring of all of its edges using their proposed colors. However, to find this proper coloring, it is important to have all edges of the online vertex at once. If, as in our case in this lemma, the edges of the online vertex arrive in batches, then we cannot ensure that edges in two different batches of the same online vertex u receive different colors. To fix this, first we only properly color the edges of each batch of the online vertex using their proposed colors. If this assigns color c to edge e, the final color that we report for e is (c, i) where i denotes which batch of u this edge e belongs to. Since there are at most Δ/k batches for each vertex, the total number of colors needed with this approach is $O(\Delta \cdot \Delta/k) = O(\Delta^2/k)$. Finally, note that the space remains $O(n)$ since the only additional information that we need is the number of batches that have arrived for each vertex which can be stored with $O(n)$ counters, one for each online vertex. ◀

4 Edge-Arrivals: $O(\Delta)$ Edge-Coloring with $\widetilde{O}(n\sqrt{\Delta})$ Space

In this section, we prove Theorem 1.1 for the special case where $s = \sqrt{\Delta}$. That is, we obtain an asymptotically optimal $O(\Delta)$ edge-coloring using $\widetilde{O}(n\sqrt{\Delta})$ space.

To solve the edge arrival model, we present a reduction to the one-sided batch arrival model of Definition 3.8. Throughout this section, we only consider the batch arrival model for batches of size $k := \lceil\sqrt{\Delta}\rceil$.

▷ **Claim 4.1.** Suppose that there is a streaming algorithm $\mathcal{A}$ that edge colors a bipartite graph of maximum degree Δ using $O(\Delta)$ colors under one-sided batch arrivals with batches of size $k = \Theta(\sqrt{\Delta})$ using space $\widetilde{O}(n\sqrt{\Delta})$. Then there is a streaming edge coloring $\mathcal{A}'$ algorithm that $O(\Delta)$ edge colors any general graph of maximum degree Δ under edge arrivals using $\widetilde{O}(n\sqrt{\Delta})$ space.

Proof. Let us first assume that graph G is bipartite with vertex sets U and V but its edges arrive in an arbitrary order. We start with a buffer $T = \emptyset$ and add any edge that arrives in the stream to T. Whenever there is some vertex u that has at least k edges in T, we remove those edges of u from T and feed them into the batch-arrival algorithm $\mathcal{A}$. This way, each batch has exactly k edges all adjacent to the same vertex. However, this vertex could belong to either of U and V. That is, this is not one-sided arrivals but rather two-sided

arrivals. The reduction to one-sided arrivals simply follows from Claim 3.2 at the expense of multiplying the final number of colors by two. Finally, if the stream ends and there are any remaining edges in T, we color them all using Δ colors via offline algorithms. Note that since each vertex in T has degree at most k at any point, the space of this reduction is only $O(nk) = O(n\sqrt{\Delta})$ as desired.

From the reduction above, we get that under edge arrivals, there is an algorithm $\mathcal{A}''$ that edge-colors the graph using $C(\Delta)$ colors for some constant C and $\widetilde{O}(n\sqrt{\Delta})$ space provided that the graph is bipartite. We now apply Lemma 3.1. This gives an algorithm for general graphs under edge-arrivals that uses $\widetilde{O}(n\sqrt{\Delta})$ space and the number of colors that it uses is at most

$$C\Delta + C\Delta/2 + C\Delta/4 + \ldots + C \leq 2C\Delta = O(\Delta). \qquad \blacktriangleleft$$

Our goal for the rest of this section is to prove the following algorithm which plugged to Claim 4.1 proves Theorem 1.1 for $s = \sqrt{\Delta}$:

▶ **Lemma 4.2.** *There is a streaming algorithm that edge-colors any bipartite graph of maximum degree Δ under one-sided batch-arrivals (Definition 3.8) with batches of size $k = \Theta(\sqrt{\Delta})$ using $O(\Delta)$ colors and $\widetilde{O}(n\sqrt{\Delta})$ space.*

Proof. One may wonder whether we can now directly apply the naive algorithm of Lemma 3.9 in order to prove Lemma 4.2. This is not doable because Lemma 3.9 would require $O(\Delta^2/k) = O(\Delta^{1.5})$ colors which is much larger than our desired $O(\Delta)$ colors. So we need more ideas.

First, we assume that $\Delta \geq 300 \log^2 n$. If not, the whole graph has at most $O(n\Delta) = \widetilde{O}(n)$ edges, so we can store them all and run an offline Δ edge coloring algorithm.

Consider the subgraphs $H_1, H_2, \ldots, H_k$ where

$$H_i := \bigcup_{u \in U} B_i(u),$$

recalling that $B_i(u)$ is the i'th batch of online vertex u. In words, H_i is the collection of the i'th batches of all online vertices. Observe that the arrival order of the edges of each subgraph H_i follow the vertex arrival model. That is, for each online vertex v, all of its edges arrive at the same time. Note also that the maximum degree on the online side of each subgraph H_i is at most k. Now, if we also had the same upper bound of k on the degrees on the offline side of each subgraph H_i, then we could run $\Delta/k = O(\sqrt{\Delta})$ instances of Algorithm 1, which works under vertex arrivals, for each of the k subgraphs H_i all in parallel, using disjoint color palettes. This way, the total number of colors used would be $\frac{\Delta}{k} \cdot O(k) = O(\Delta)$ and the space would be $\frac{\Delta}{k} \cdot O(n) = O(n\sqrt{\Delta})$. Unfortunately, however, the offline vertices in each H_i may have degree as large as Δ. This is because all edges of an offline vertex might be in the first batch of their corresponding online vertices, making the maximum degree in H_1 equal to Δ rather than k. This is the main obstacle that we overcome in the remainder the proof.

Let us for each online vertex u choose a random number $b_u \in [k]$ at the beginning of the algorithm uniformly and independently at random. Now for the i'th batch of u define its *permuted batch number* $\rho_u(i)$ to be

$$\rho_u(i) := (i + b_u) \bmod k.$$

Doing so, we now redefine H_i for $i \in [k]$ as

$$H_i := \bigcup_{u \in U} B_{\rho_u^{-1}(i)}(u).$$

In words, H_i is now the graph including, for each online vertex u, its edges in its j'th batch such that $\rho_u(j) = i$. Since the total number of batches of each online vertex u is at most $\Delta/k \leq k$, all of u's batches receive different batch numbers. It remains to bound the degree of offline vertices in each H_i. The following lemma asserts that with high probability this is in fact $O(k)$.

▶ **Lemma 4.3.** *All H_i's have maximum degree at most $2k$ with probability $1 - 1/\operatorname{poly}(n)$.*

Due to space constraints, we defer the proof of this lemma to the appendix.

Now that both the offline and online vertices of each H_i have degree at most $2k$, we can follow the approach outline before to achieve a $O(\Delta)$ coloring with $O(n\sqrt{\Delta})$ space. The final algorithm is formalized below as Algorithm 2.

🟧 **Algorithm 2.** A streaming edge coloring for bipartite graphs under one-sided batch arrivals with batches of size $k = \Theta(\sqrt{\Delta})$. The algorithm uses $O(\Delta)$ colors and $O(n\sqrt{\Delta})$ space. This is the algorithm used for Lemma 4.2 which plugged into Claim 4.1 proves Theorem 1.1 for $s = \sqrt{\Delta}$.

Parameter: $k := \sqrt{\Delta}$, $c := 300$ (Lemma 4.3)
1. If $\Delta \leq c \cdot \log^2 n$ we read all the edges from the stream and color them with Δ colors, so assume for the rest of the algorithm that $\Delta > c \cdot \log^2 n$.
2. As the edges of G arrive, we decompose them into k subgraphs $H_1, \ldots, H_k$ and run k instances of the vertex-arrival Algorithm 1 in parallel on these subgraphs using disjoint colors in these instances, and an overall space of $O(nk)$. Each graph H_i will have maximum degree k, so overall we use $O(k \cdot k) = O(\Delta)$ colors.
3. For each online vertex u choose a random number $b_u \in [k]$ uniformly and independently.
4. For each online vertex u store a counter I_u. This will keep track of the number of *batches* of u seen at any point. Initially we have $I_u \leftarrow 0$.
5. Upon arrival of a batch of size k for an online vertex u:
   a. $I_u \leftarrow I_u + 1$.
   b. Add the edges of u to graph H_x where

   $$x := (I_u + b_u) \bmod k.$$

   c. These will be the only edges of u in graph H_x, thus this is indeed a bipartite graph under one-sided vertex arrivals and so we can run the instance of Algorithm 1 on graph H_x to color the edges of u in it.

Space-complexity. Since we are running $k = O(\sqrt{\Delta})$ instances of Algorithm 1 in Algorithm 2, the space needed for keeping all these instances is also $O(n\sqrt{\Delta})$ as each has a space of $O(n)$ words by Claim 3.7. Moreover, note that the size of T never exceeds nk or else there must be a vertex of degree at least k in it. We also use at most $O(n)$ space for the counters and random numbers (specified in Step 3 and 4). So in total, the space that Algorithm 2 uses is $O(nk) = O(n\sqrt{\Delta})$.

Number of colors. For the number of colors used by Algorithm 2, note that we run k instances of Algorithm 1, and in each instance the maximum degree is w.h.p. $2k$ by Lemma 4.3. Since Algorithm 1 uses linear colors in the maximum degree, the total number of colors is thus $O(k \cdot k) = O(\Delta)$ w.h.p.

This completes the proof of Lemma 4.2. ◀

As discussed, plugging Lemma 4.2 into Claim 4.1 completes the proof of Theorem 1.1 for $s = \sqrt{\Delta}$, concluding this section. For the final algorithm that does $O(\Delta^{1.5}/s + \Delta)$ Edge-coloring with $\widetilde{O}(ns)$ space see the full version of our paper at [18].

References

1   Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2237–2246, 2015. URL: http://proceedings.mlr.press/v37/ahn15.html.

2   Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467, 2012. doi:10.1137/1.9781611973099.40.

3   Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14, 2012. doi:10.1145/2213556.2213560.

4   Mohammad Ansari, Mohammad Saneian, and Hamid Zarrabi-Zadeh. Simple streaming algorithms for edge coloring. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, pages 8:1–8:4, 2022. doi:10.4230/LIPICS.ESA.2022.8.

5   Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 708–742, 2022. doi:10.1137/1.9781611977073.32.

6   Sepehr Assadi and Soheil Behnezhad. Beating two-thirds for random-order streaming matching. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, pages 19:1–19:13, 2021. doi:10.4230/LIPICS.ICALP.2021.19.

7   Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li. On regularity lemma and barriers in streaming and dynamic matching. In *Proccedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2023, to appear*, 2023.

8   Sepehr Assadi, Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Coloring in graph streams via deterministic and adversarially robust algorithms. *CoRR*, abs/2212.10641, 2022. doi:10.48550/arXiv.2212.10641.

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

10   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, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 767–786, 2019. doi:10.1137/1.9781611975482.48.

11   Sepehr Assadi and Aditi Dudeja. A simple semi-streaming algorithm for global minimum cuts. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 172–180. SIAM, 2021.

12   Sepehr Assadi, Pankaj Kumar, and Parth Mittal. Brooks' theorem in graph streams: a single-pass semi-streaming algorithm for δ-coloring. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 234–247, 2022. doi:10.1145/3519935.3520005.

13   Sepehr Assadi and Janani Sundaresan. (Noisy) gap cycle counting strikes back: Random order streaming lower bounds for connected components and beyond. In *Proccedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2023, to appear*, 2023.

14   Amotz Bar-Noy, Rajeev Motwani, and Joseph Naor. The greedy algorithm is optimal for on-line edge coloring. *Inf. Process. Lett.*, 44(5):251–253, 1992. `doi:10.1016/0020-0190(92)90209-E`.

15   Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Almost 3-approximate correlation clustering in constant rounds. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 720–731, 2022. `doi:10.1109/FOCS54457.2022.00074`.

16   Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Single-pass streaming algorithms for correlation clustering. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, to appear*, 2023.

17   Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Marina Knittel, and Hamed Saleh. Streaming and massively parallel algorithms for edge coloring. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, pages 15:1–15:14, 2019. `doi:10.4230/LIPICS.ESA.2019.15`.

18   Soheil Behnezhad and Mohammad Saneian. Streaming edge coloring with asymptotically optimal colors. *CoRR*, abs/2305.01714, 2023. `doi:10.48550/arXiv.2305.01714`.

19   Suman K. Bera, Amit Chakrabarti, and Prantar Ghosh. 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)*, pages 11:1–11:21, 2020. `doi:10.4230/LIPICS.ICALP.2020.11`.

20   Sayan Bhattacharya, Fabrizio Grandoni, and David Wajc. Online edge coloring algorithms via the nibble method. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2830–2842. SIAM, 2021. `doi:10.1137/1.9781611976465.168`.

21   Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Adversarially robust coloring for graph streams. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, pages 37:1–37:23, 2022. `doi:10.4230/LIPICS.ITCS.2022.37`.

22   Moses Charikar and Paul Liu. Improved algorithms for edge colouring in the w-streaming model. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 181–183, 2021. `doi:10.1137/1.9781611976496.20`.

23   Shiri Chechik, Doron Mukhtar, and Tianyi Zhang. Streaming Edge Coloring with Subquadratic Palette Size. *CoRR*, abs/2305.07090, 2023. `arXiv:2305.07090`, `doi:10.48550/arXiv.2305.07090`.

24   Yu Chen, Sanjeev Khanna, and Huan Li. On weighted graph sparsification by linear sketching. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 474–485, 2022. `doi:10.1109/FOCS54457.2022.00052`.

25   Ilan Reuven Cohen, Binghui Peng, and David Wajc. Tight bounds for online edge coloring. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1–25. IEEE Computer Society, 2019. `doi:10.1109/FOCS.2019.00010`.

26   Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 2069–2078, 2021. URL: `http://proceedings.mlr.press/v139/cohen-addad21b.html`.

27   Camil Demetrescu, Irene Finocchi, and Andrea Ribichini. Trading off space for passes in graph streaming problems. *ACM Trans. Algorithms*, 6(1), December 2010. `doi:10.1145/1644015.1644021`.

28   Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, pages 531–543, 2004. `doi:10.1007/978-3-540-27836-8_46`.

29   Alan M. Frieze. Maximum matchings in a class of random graphs. *J. Comb. Theory, Ser. B*, 40(2):196–212, 1986. `doi:10.1016/0095-8956(86)90077-8`.

30   Harold N Gabow. Algorithms for edge-coloring graphs. *Technical Report*, 1985.

31   Mohsen Ghaffari, Krzysztof Nowicki, and Mikkel Thorup. Faster algorithms for edge connectivity via random 2-out contractions. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1260–1279. SIAM, 2020. `doi:10.1137/1.9781611975994.77`.

32   Prantar Ghosh and Manuel Stoeckl. Low-memory algorithms for online and w-streaming edge coloring. *CoRR*, abs/2304.12285, 2023.

33   Christian Glazik, Jan Schiemann, and Anand Srivastav. Finding euler tours in one pass in the w-streaming model with o(n log(n)) RAM. *CoRR*, abs/1710.04091, 2017. `arXiv:1710.04091`.

34   Jacob Holm, Valerie King, Mikkel Thorup, Or Zamir, and Uri Zwick. Random k-out subgraph leaves only o(n/k) inter-component edges. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 896–909. IEEE Computer Society, 2019. `doi:10.1109/FOCS.2019.00058`.

35   John E. Hopcroft and Richard M. Karp. A n5/2 algorithm for maximum matchings in bipartite. In *12th Annual Symposium on Switching and Automata Theory (SWAT 1971)*, pages 122–125, 1971.

36   Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1874–1893, 2021. `doi:10.1137/1.9781611976465.112`.

37   Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570, 2014. `doi:10.1109/FOCS.2014.66`.

38   Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1814–1833, 2020. `doi:10.1137/1.9781611975994.111`.

39   Howard J. Karloff and David B. Shmoys. Efficient parallel algorithms for edge coloring problems. *J. Algorithms*, 8(1):39–52, 1987. `doi:10.1016/0196-6774(87)90026-5`.

40   Janardhan Kulkarni, Yang P. Liu, Ashwin Sah, Mehtaab Sawhney, and Jakub Tarnawski. Online edge coloring via tree recurrences and correlation decay. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 104–116. ACM, 2022. `doi:10.1145/3519935.3519986`.

41   Amin Saberi and David Wajc. The greedy algorithm is not optimal for on-line edge coloring. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 109:1–109:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPICS.ICALP.2021.109`.

A   Reduction To Bipartite Graphs

In this section, we present the proof of Lemma 3.1 using a random bipartization idea that has been used extensively in the literature of edge colorings. To our knowledge, this idea was first used by Karloff and Shmoys [39] in the context of parallel algorithms and by by Behnezhad et al [17] in the context of streaming edge coloring. We emphasize that we claim no novelty in the algorithm of this section and merely present it here for the sake of self-containment.

Proof of Lemma 3.1. If $\Delta \leq 10 \log n$, then we store the whole graph in memory using $\widetilde{O}(n)$ space and edge color the graph via $(\Delta + 1)$ colors using offline algorithms. So assume that $\Delta > 10 \log n$.

Let V be the vertex-set of the graph. We partition V into two subsets A and B by putting each vertex randomly in either A or B chosen uniformly and independently from the rest of the vertices. Now consider the bipartite subgraph G' of G including any edge (u, v) of G that has one endpoint in A and one endpoint in B. Note that conditioned on $v \in A$, each neighbor of v in the original graph belongs to B independently with probability $1/2$. This means that the expected degree of v in G' is $\deg_G(v)/2$. By a simple application of the Chernoff bound and recalling that $\Delta > 10 \log n$, we get that G' has maximum degree $\Delta/2 + O(\sqrt{\Delta \log n}) = (1/2 + o(1))\Delta$ with probability $1 - 1/n^{10}$. We can thus run the bipartite edge coloring algorithm on G'. Notice, however, that this leaves the edges that go from A to A or from B to B uncolored. We recursively apply the same algorithm. That is, we recursively partition A and B into two subsets each. This results in a subgraph of maximum degree $\Delta/4 + O(\sqrt{\Delta \log n})$ w.h.p. We continue this for $O(\log \Delta)$ steps until the resulting subgraph has maximum degree smaller than $O(\log n)$, at which point we store the whole subgraph in memory and color it using offline algorithms. Overall, the total number of colors used by the algorithm is

$$\sum_{i=1}^{\log \Delta} f\left(\left(\frac{1}{2^i} + o(1)\right)\Delta\right) \leq \sum_{i=1}^{\log \Delta} f\left(\frac{\Delta}{2^{i-1}}\right).$$

Note that this partitioning of the vertices can be done at the beginning of the stream before any edges arrive. It only suffices to store, for each vertex, which of the two random subsets it belongs to at each of the $O(\log \Delta)$ levels. So this only requires an overhead space of $O(n \log \Delta)$. Additionally, if the edges of the original graph arrive under vertex arrivals, then so do the edges of the random bipartite graphs.   ◀

B   Deferred Proofs

▶ **Lemma 4.3** (restated).   *All H_i's have maximum degree at most $2k$ with probability* $1 - 1/\operatorname{poly}(n)$.

Proof. Due to symmetry, let us focus on bounding the degree of a given offline vertex v in H_1. Let neighbors of v be $u_1, u_2, ..., u_l$ where each u_i is an online vertex. Define X_i to be the indicator random variable where $X_i = 1$ iff the edge (v, u_i) receives a batch number of 1. The degree of v in H_1 can be written as

$$X = \sum_{i=1}^{l} X_i.$$

Since $l \leq \Delta$ and also $\mathbb{E}[X_i] = \frac{1}{k}$, by linearity of expectation, we get

$$\mathbb{E}[X] = \sum_{i=1}^{l} \mathbb{E}[X_i] \leq \frac{\Delta}{k} \leq k.$$

Observe that all b_{u_i}'s are chosen at the beginning of the algorithm and uniformly at random. Therefore the batch number of any given edge being 1 has a probability of $\frac{1}{k}$. Since all b_{u_i}'s are independent of each other, also all X_i's will be independent. Thus, we can apply the Chernoff bound for $\delta = 1$ and having $\Delta \geq 300 \log^2 n$ (as assumed at the start of the proof of Lemma 4.2) we get,

$$\Pr[\text{degree } v \text{ in } H_1 \geq 2k] = \Pr\left[\sum_{i=1}^{l} X_i \geq 2k\right] \leq \exp\left(\frac{-k}{3}\right) \leq n^{-10}$$

By a union bound over all n choices of v and over k choices of H_i, we get

$$\Pr[\text{exists } v \in V \text{ and } i \in [k] \text{ such that degree } v \text{ in } H_i \geq 2k] \leq n^{-8}. \qquad \blacktriangleleft$$