Online Edge Coloring Algorithms via the Nibble Method*

Sayan Bhattacharya[1], Fabrizio Grandoni[2], and David Wajc[†3]

[1]University of Warwick.
[2]IDSIA, USI-SUPSI.
[3]Stanford University

Abstract

Nearly thirty years ago, Bar-Noy, Motwani and Naor [IPL'92] conjectured that an online $(1+o(1))\Delta$-edge-coloring algorithm exists for n-node graphs of maximum degree $\Delta = \omega(\log n)$. This conjecture remains open in general, though it was recently proven for bipartite graphs under *one-sided vertex arrivals* by Cohen et al. [FOCS'19]. In a similar vein, we study edge coloring under widely-studied relaxations of the online model.

Our main result is in the *random-order* online model. For this model, known results fall short of the Bar-Noy et al. conjecture, either in the degree bound [Aggarwal et al. FOCS'03], or number of colors used [Bahmani et al. SODA'10]. We achieve the best of both worlds, thus resolving the Bar-Noy et al. conjecture in the affirmative for this model.

Our second result is in the adversarial online (and dynamic) model with *recourse*. A recent algorithm of Duan et al. [SODA'19] yields a $(1+\epsilon)\Delta$-edge-coloring with $\text{poly}(\log n/\epsilon)$ recourse. We achieve the same with $\text{poly}(1/\epsilon)$ recourse, thus removing all dependence on n.

Underlying our results is one common offline algorithm, which we show how to implement in these two online models. Our algorithm, based on the Rödl Nibble Method, is an adaptation of the distributed algorithm of Dubhashi et al. [TCS'98]. The Nibble Method has proven successful for distributed edge coloring. We display its usefulness in the context of online algorithms.

1 Introduction

Edge coloring is the problem of assigning one of k colors to all edges of a simple graph, so that no two incident edges have the same color. The objective is to minimize the number of colors, k. The edge coloring problem goes back to the 19th century and studies of the four-color theorem [37, 35]. In 1916, König [29], in what many consider to be the birth of matching theory, proved that any bipartite graph of maximum degree Δ is colorable using Δ colors. (Clearly, no fewer colors suffice.) Nearly half a century later, Vizing [38] proved that any general graph is $(\Delta+1)$-edge-colorable. Vizing's proof is algorithmic, yielding such a coloring in polynomial time. This is likely optimal, as it is NP-hard to determine if a general graph is Δ-edge-colorable [25]. Algorithms for the edge coloring problem have been studied in several different models of computation, including offline, online, distributed, parallel, and dynamic models (see, e.g., [13, 11, 36, 9, 27, 34, 14, 10] and references therein.) In this work, we study the edge coloring problem in online settings.

Online edge coloring: Here, an adversary picks an n-node graph G of maximum degree Δ (the algorithm knows n and Δ, but not G), and then reveals the edges of G one at a time. Immediately after the arrival of an edge, the algorithm must irrevocably assign a color to it, with the objective of minimizing the final number of colors used. This problem was first studied nearly thirty years ago, by Bar-Noy, Motwani, and Naor [4]. They showed that the greedy algorithm, which returns a proper $(2\Delta-1)$-edge coloring, is worst-case optimal among online algorithms. This might seem to be the end of the story for this line of research. However, as pointed out by Bar-Noy et el. [4], their lower bound only holds for bounded-degree graphs, with some $\Delta = O(\log n)$. This then led them to conjecture that online $(1+o(1))\Delta$-edge-coloring is possible for graphs with $\Delta = \omega(\log n)$. This conjecture remains wide open.

Recently, an important progress was made towards proving the Bar-Noy et al. conjecture: Cohen et al. [11] showed how to obtain a $(1+o(1))\Delta$-edge coloring for bipartite graphs in an online setting under *node arrivals* (together with their edges). This is a relaxation of the online edge-arrival model. Thus, this latter result can be seen as an intermediate step towards the ultimate goal of proving the Bar-Noy et al. conjecture. In a similar spirit, we consider edge coloring in two well-studied relaxations of the online model, that act as intermediate steps towards the Bar-Noy et al. conjecture, and make substantial progress on the state-of-the-art results in both these settings.

(I) Random-order online edge coloring: Here, an

*The full version of the paper can be found at `https://arxiv.org/abs/2010.16376`.

†Work done while the author was at Carnegie Mellon University.

adversarially-chosen graph has its edges revealed to the algorithm in *uniformly random order*. Such random-order arrivals, which capture numerous stochastic arrival models, have been widely studied for many online problems. (See, e.g., [26, 31, 30, 33, 28] and the survey by Gupta and Singla [24] and references therein.) In the context of edge coloring, this model was studied by [1, 3]. Aggarwal et al. [1] were the first to show that high Δ suffices for near-ideal coloring in this model, giving a $(1 + o(1))\Delta$-edge-coloring algorithm for *multigraphs* with $\Delta = \omega(n^2)$. Bahmani et al. [3] then breached the greedy $2\Delta - 1$ barrier for simple graphs with polylogarithmic Δ, giving a 1.43Δ-edge coloring algorithm for $\Delta = \omega(\log n)$ (improved to 1.26Δ in their journal version). This leads to the following natural open question: can one obtain "the best of both worlds" w.r.t. [1, 3]? That is, can one obtain a $(1 + o(1))\Delta$-edge-coloring for graphs of maximum degree $\Delta = \omega(\log n)$ whose edges are presented in random order? Put another way, is the Bar-Noy et al. conjecture true for random-order edge arrivals? We answer this question in the affirmative.

THEOREM 1.1. *For some absolute constant $\gamma \in (0, 1)$, there exists an online algorithm that, when given a graph G of maximum degree $\Delta = \omega(\log n)$, whose edges are presented in random order, computes a proper $\left(\Delta + O\left(\Delta^{\gamma} \cdot \log^{1-\gamma} n\right)\right) = (1 + o(1))\Delta$-edge-coloring of G w.h.p.*

We complement this upper bound with a lower bound showing that, for some $\Delta = O(\log n)$, not only is it impossible to guarantee a $(1 + o(1))\Delta$-edge-coloring under random-order arrivals, but it is even impossible to use any fewer than $2\Delta - 1$ colors: see Appendix A.

We note that previous random-order online edge coloring algorithms [3, 1] required the graph to be Δ-regular. This assumption is without loss of generality in an offline setting, but it is unclear whether the same holds in the random-order online model. Our algorithm from Theorem 1.1, however, works on any graph (including non-regular ones): this is discussed in Section 3.1.

(II) Dynamic edge coloring with recourse: Another widely-studied relaxation of online algorithms is online algorithms with *recourse*. Here, an algorithm must make immediate choices upon each arrival, but is also allowed to make a small number of changes to its solution after each arrival (referred to as *recourse*). The concept of recourse helps us understand the robustness/sensitivity of (near-)optimal solutions. Accordingly, an influential line of research in the online algorithms community is devoted to studying the trade-offs between the solution quality and recourse for many well-known problems [21, 23, 6, 17, 20, 32, 8, 22].

Many results for bounded-recourse online algorithms hold in a more general, *dynamic* setting. In the context of edge coloring, the dynamic (oblivious) version of the problem with recourse is captured by the following scenario: The input graph G changes via a sequence of $\mathsf{poly}(n)$ *updates* chosen in advance by an adversary, where each update consists of an edge insertion or deletion in G. The maximum degree in G remains at most Δ throughout. The algorithm maintains a proper edge coloring, while changing the colors of some edges in G after each update (the number of such changes per update is the recourse of the algorithm). The challenge is to design an algorithm that simultaneously (a) maintains a proper coloring with few colors and (b) has small recourse.

In recent years, the edge coloring problem has been extensively studied from a different, but highly related, perspective of *dynamic data structures* [5, 14, 7, 39]. Here, the goal is to maintain a proper edge coloring with few colors in a dynamic graph, taking little time after each edge update (insertion/deletion), where this time is referred to as *update time*. Note that the update time of any data structure for dynamic edge coloring upper bounds its recourse, since the data structure has to spend at least $\Omega(1)$ time per edge which changes its color after an update.

The state-of-the-art result for dynamic edge coloring with recourse follows from the work of Duan et al. [14]. In any dynamic graph with $\Delta = \Omega(\log^2 n/\epsilon^2)$, their algorithm maintains a proper $(1 + \epsilon)\Delta$-edge coloring with $\mathsf{poly}(\log n, 1/\epsilon)$ recourse. Given that other dynamic problems are known to admit super-constant recourse lower bounds (see, e.g., [17]), it is natural to ask if one can get a recourse bound for $(1+\epsilon)\Delta$-edge coloring that is independent of n. We answer this question in the affirmative.

THEOREM 1.2. *There is an algorithm that maintains a proper $(1 + \epsilon)\Delta$-edge-coloring w.h.p., with $\mathsf{poly}(1/\epsilon)$ expected recourse in dynamic graphs of maximum degree $\Delta = \Omega\left(\log n/\mathsf{poly}(\epsilon)\right)$.*

1.1 Our Techniques At the heart of both our results is one common algorithmic approach, inspired by the Rödl Nibble Method [2], as applied to distributed edge coloring by Dubhashi et al. [15]. This method and its variants have since found further uses in distributed settings [9, 16]. To the best of our knowledge, we are the first to export this method to online settings.

We analyze our basic algorithm, which is a variant of [15], in an offline model. We then show how to implement this algorithm in online and dynamic settings, from which we obtain our results. We now outline this basic algorithm, and the ideas needed to

implement it in the models we study. For simplicity, we focus on Δ-regular graphs in this section.

The High-Level Framework: The Nibble Method in the framework of edge coloring was first used in [15] in the distributed model. Let us sketch how their algorithm would work in the offline setting. The algorithm consists of multiple rounds. In each round, each vertex v selects a random ϵ fraction of its incident uncolored edges. Each sampled edge e chooses a tentative color u.a.r. among the colors in $[\Delta]$ not yet taken by incident edges (*palette* of e). We then assign the tentative color $c(e)$ to sampled edges e for which no incident edge e' picked the same tentative color $c(e)$, else we mark e as *failed*, and leave e uncolored. It turns out that each sampled edge fails at each round with probability $O(\epsilon)$. Crucially, picking ϵ appropriately results in a number of important parameters (degrees in the uncolored subgraph, palette sizes, etc') behaving in a predictable manner, and being sharply concentrated around their mean, w.h.p. In particular, this results in the uncolored subgraph's maximum degree decreasing w.h.p. at a rate of roughly $1-\epsilon$ per application of this subroutine, or *round*. Consequently, some $t_\epsilon = O(\log(1/\epsilon)/\epsilon)$ rounds leave an uncolored subgraph of maximum degree $\Delta' = \mathsf{poly}(\epsilon)\Delta$ w.h.p., which can then be greedily colored using a further $2\Delta' = \mathsf{poly}(\epsilon)\Delta$ colors. This approach therefore yields a proper $(1+\mathsf{poly}(\epsilon))\Delta$ edge coloring.

In part inspired by [9], we consider a slight modification of the above algorithm which is more convenient for our goals. In more detail, we make the following changes:
(1) We do not attempt to re-color an edge e which fails in a given round in future rounds, instead leaving e to be colored greedily in the final stage. Intuitively, ignoring these edges still results in a low-degree uncolored graph after t_ϵ rounds, since few edges incident to each vertex fail.
(2) Whenever an edge e picks a tentative color c, we remove c from the palettes of its incident edges even if e fails. Intuitively, this does not decrease the palette sizes much, again, since few edges incident to each vertex fail.
(3) We sample each edge independently with probability ϵ in each round.

Our modifications bring two main advantages. First, the analysis can be substantially simplified: rather than using a specialized concentration inequality of Grable [19], we mostly use Hoeffding bounds for negatively-associated variables. This allows us to provide a relatively concise, but complete analysis for subconstant values of ϵ and for non-regular graphs. Second, and importantly for us, it is easier to adapt the modified algorithm to the online settings that we study.

Random-Order Online Implementation: To obtain our results for random-order arrivals, we first observe that our edge-centric sampling of modification (3) allows us to use the randomness of edge arrivals to "sample edges for us". More formally, we implement the independent edge-sampling part of each round by considering an appropriate binomially-distributed prefix of the remaining edges (relying on our knowledge of the number of edges of the Δ-regular graph, $m = \frac{n\Delta}{2}$). This results in each remaining edge of the graph being sampled independently with probability ϵ.

For each round, we have each edge of the round sample a tentative color u.a.r. from its palette. In this online setting, however, we cannot always tell when an edge arrives whether it picked the same tentative color as its incident edges of the same round (since some of these arrive *later*). We therefore assign the tentative color $c(e)$ to sampled edges e for which no *previous* incident edge e' picked the same tentative color $c(e)$, else we mark e as *failed*. Modification (2) in the basic algorithm implies that this change still results in a feasible (partial) coloring. On the other hand, the uncolored subgraph "after" the rounds in this algorithm clearly has lower maximum degree than its counterpart in the basic algorithm, and so greedily coloring this subgraph requires fewer colors than the same stage of the basic algorithm. Finally, modification (1) of our basic algorithm, whereby we do not attempt to recolor a failed edge in "future rounds" (which would require knowledge of future arrivals), implies that we can greedily color every failed edge before the next edge arrives. So, by the analysis of our basic algorithm, we obtain Theorem 1.1 for Δ-regular graphs. In Section 3 we build on this approach to obtain our full result, for general graphs.

Low-Recourse Implementation: For our low-recourse dynamic implementation, we show how to maintain, after each update, a coloring drawn from the same distribution as that of the basic algorithm applied to the current graph. Modifications (1) and (3) of the basic algorithm imply that deciding the round in which we sample any edge can be done in advance (prior to any arrival), sampling this round number from the appropriately capped geometric distribution. The more delicate point is dealing with the choice of tentative colors. For example, when an edge is added, its tentative color is removed from the palettes of its incident edges of later rounds. Thus, some incident edges no longer have a tentative color which is u.a.r. drawn from their current palette. In Section 4 we show that a natural approach of correcting these distributions—sampling a new ten-

tative color if the previous one is no longer valid, and switching to a new color if one samples a newly-available color—results in bounded recourse. In particular, building on our analysis of the basic algorithm, we show that the expected number of edges of round $i+1$ whose tentative color changes due to the change of a tentative color of an edge in round $j \leq i$ is at most $O(\epsilon)$. Therefore, for any update, the number of tentative color changes is at most $(1+O(\epsilon))^{t_\epsilon} = \mathsf{poly}(1/\epsilon)$, from which we obtain Theorem 1.2. In Section 4 we formalize this approach and its analysis

1.2 Further Related Work Other than [15], most closely related to our approach are other distributed edge coloring algorithms using the Nibble Method and its variants and extensions [9, 16]. While these distributed algorithms improve on [15], they require crucially that edges be tentatively colored in multiple rounds—a design pattern which seems hard to implement in online settings. Another approach is suggested by the work of Cohen et al. [11] for bipartite one-sided vertex arrivals; using an online matching algorithm of Cohen and Wajc [12] which matches each edge with probability $\frac{1-o(1)}{\Delta}$, they color roughly one edge of each maximum-degree node per round, resulting in a $(1+o(1))\Delta$-edge-coloring. Unfortunately, for the models we study, no such matching algorithm is known—ruling out this approach.

Returning to previous algorithms in our models, we note that the algorithm of Duan et al. [14], which uses an augmenting path based approach, has an inherent polylogarithmic recourse. On the other hand, the approaches of [1, 3] for random-order arrivals seem challenging to extend to dynamic recourse-bounded algorithms. Moreover, in the random-order online model, it is unclear how to provably achieve a $(1+o(1))\Delta$-edge-coloring for simple graphs with $\Delta = \omega(\log n)$ by using the ideas in those papers. In this work we show that the Nibble Method, and in particular, a variant of the algorithm of [15], allows us to obtain such desired results in *both* these online models.

1.3 Full Version of the Paper Due to space constraints, many technical proofs are omitted from this extended abstract. The full version of the paper can be found at: `https://arxiv.org/abs/2010.16376`.

2 The Basic Algorithm

In this section, we describe our basic algorithm for near-regular graphs in the *static setting.* and state the key theorem needed for its analysis. We defer a more detailed analysis to the full version of the paper. Our online and dynamic algorithms in subsequent sections will be built on top of this basic algorithm.

The input to the algorithm is a graph $G = (V, E)$ with $|V| = n$ nodes, where the degree of each node lies in the interval $[(1-\epsilon^2)\Delta, (1+\epsilon^2)\Delta]$. The parameter ϵ satisfies the following condition:

$$1/10^4 \geq \epsilon \geq 10 \cdot (\ln n/\Delta)^{1/6}. \tag{2.1}$$

Note that such ϵ exist if $\Delta = \Omega(\log n)$ is large enough. The algorithm runs in two *phases*, as follows.

Phase One. In phase one, the algorithm properly colors a subset of edges of G using $(1+\epsilon^2)\Delta$ colors, while leaving an uncolored subgraph of small maximum degree. This phase consists of $t_\epsilon - 1$ *rounds* $\{1, \ldots, t_\epsilon - 1\}$, for

$$t_\epsilon := \lfloor \ln(1/\epsilon)/(2K\epsilon) \rfloor, \text{ and } K = 48. \tag{2.2}$$

Each round $i \in [t_\epsilon - 1]$ operates on a subgraph $G_i := (V, E_i)$ of the input graph (with $E_1 = E$), identifies a subset of edges $S_i \subseteq E_i$, picks a *tentative* color $c(e) \in [(1+\epsilon^2)\Delta] \cup \{\text{null}\}$ for each edge $e \in S_i$, and returns the remaining set of edges $E_{i+1} = E_i \setminus S_i$ for the next round. Thus, we have: $E = E_1 \supseteq E_2 \supseteq \cdots \supseteq E_{t_\epsilon}$. We now describe how a given round $i \in [t_\epsilon - 1]$ works. We start by defining a couple of important notations.

- (a) For all $v \in V$, let $P_i(v) := \{\chi \in [(1+\epsilon^2)\Delta] : \chi \neq c(u,v) \text{ for all } (u,v) \in \bigcup_{j<i} S_j\}$ denote the *palette* of the node v for round i. A color $\chi \in [(1+\epsilon^2)\Delta]$ belongs to $P_i(v)$ iff no edge incident on v has tentatively picked the color χ in previous rounds $j < i$. (b) Similarly, for all $(u,v) \in E_i$, let $P_i(u,v) := P_i(u) \cap P_i(v)$ denote the *palette* of the edge (u,v) for round i.

In round i, we first sample each edge $e \in E_i$ *independently* with probability ϵ. Let $S_i \subseteq E_i$ be the set of sampled edges. Next, each edge $e \in S_i$ with $P_i(e) \neq \emptyset$ tentatively picks a color $c(e)$ from its palette $P_i(e)$ uniformly and independently at random. We say that an edge $e \in S_i$ *failed* in round i iff either (a) $P_i(e) = \emptyset$ (in this case we set $c(e) :=$ null), or (b) among the edges $N(e) \subseteq E$ that are adjacent to e, there is some edge $e' \in S_i$ that tentatively picked the same color (i.e., $c(e) = c(e')$). Let $F_i \subseteq S_i$ denote the set of failed edges in round i. The remaining sampled edges $e \in S_i \setminus F_i$ are called *successful* in round i. Each such edge $e \in S_i \setminus F_i$ is *assigned* the color $c(e)$ it tentatively picked in round i. Before terminating the current round, we set $E_{i+1} := E_i \setminus S_i$ and $G_{i+1} := (V, E_{i+1})$. We remark that the color tentatively sampled by a failed edge e cannot be used by the edges incident to e in subsequent rounds. This will prove useful both for our analysis and

when implementing this algorithm in other models in subsequent sections.

Phase Two. Finally, in phase two, we greedily color all edges that were not successful in phase one. That is, letting $G_F := (V, \cup_i F_i)$ be the subgraph consisting of all the edges that failed in phase one, and $G_{t_\epsilon} := (V, E_{t_\epsilon})$ be the subgraph consisting of all the edges that were never sampled in phase one, we color the edges of $G_{t_\epsilon} \cup G_F$ greedily, using a new palette of $2\Delta(G_{t_\epsilon} \cup G_F) - 1$ colors. Here $\Delta(H)$ denotes the maximum degree in any graph H.

Algorithm 1 The Basic Algorithm

$E_1 \leftarrow E$ and $G_1 \leftarrow (V, E_1)$
$\Delta' \leftarrow (1+\epsilon^2)\Delta$ ▷ Initial palette size
for $i = 1, 2, \ldots, t_\epsilon - 1$ **do**
  $S_i \leftarrow \emptyset$
  for each $e \in E_i$ independently **do**
    with probability ϵ, add e to S_i.
    let $P_i(e) \leftarrow [\Delta'] \setminus \{c(e') \mid e' \in N(e) \cap \bigcup_{j<i} S_j\}$
    if $P_i(e) \neq \emptyset$, sample $c(e) \sim_R P_i(e)$
    else set $c(e) \leftarrow$ null ▷ Tentative color of e
  let $F_i^n \leftarrow \{e \in S_i \mid c(e) = \text{null}\}$
  let $F_i^c \leftarrow \{e \in S_i \mid c(e) = c(f) \text{ for } f \in N(e) \cap S_i\}$
  let $F_i \leftarrow F_i^n \cup F_i^c$ ▷ The failed edges
  color each edge $e \in S_i \setminus F_i$ using color $c(e)$
  $E_{i+1} \leftarrow E_i \setminus S_i$ and $G_{i+1} \leftarrow (V, E_{i+1})$
let $G_F := (V, \bigcup_i F_i)$
color $G_{t_\epsilon} \cup G_F$ greedily using colors $\Delta' + 1, \Delta' + 2, \ldots, \Delta' + 2\Delta(G_{t_\epsilon} \cup G_F) - 1$

The algorithm's pseudocode is given in Algorithm 1. We now turn to discussing its analysis.

OBSERVATION 2.1. *Algorithm 1 outputs a proper* $((1+\epsilon^2)\Delta + 2\Delta(G_{t_\epsilon} \cup G_F) - 1)$*-edge-coloring of the input graph* $G = (V, E)$.

Proof. First observe that the algorithm computes a valid partial coloring in Phase One. Indeed, any $e \in S_i \setminus F_i$ selects a color $c(e) \in P_i(e) \subseteq [(1+\epsilon^2)\Delta]$, and the definition of $P_i(e)$ and F_i guarantees that no other edge $e' \in N(e)$ in any round of Phase One can be colored with $c(e)$. The claim follows by observing that in Phase One we use only colors from $[(1+\epsilon^2)\Delta]$, while in Phase Two the greedy algorithm uses a disjoint set of at most $2\Delta(G_{t_\epsilon} \cup G_F) - 1$ extra colors. ∎

The key property of the basic algorithm is captured in the following theorem.

THEOREM 2.1. $\Delta(G_{t_\epsilon} \cup G_F) = O\left(\epsilon^{1/(3K)}\Delta\right)$ *w.h.p.*

COROLLARY 2.1. *The basic algorithm* $\left(\Delta + O\left(\epsilon^{1/(3K)}\Delta\right)\right)$ *edge colors* G*, w.h.p.*

Proof. Follows from Theorem 2.1 and Observation 2.1. ∎

In some sense, the arguments behind the proof of Theorem 2.1 were already apparent in the work of Dubhashi et al. [15]. Consequently, we defer a complete and self-contained proof of this theorem to the full version of the paper. For now, we turn to exploring implications of this theorem and Algorithm 1 to online edge coloring.

3 Random-Order Online Algorithm

In this section we present algorithms which (essentially) implement Algorithm 1 in the random-order online model. We start with a warm-up case, where the input graph is near-regular, and we know the value of m (the number of edges in the final graph).

3.1 Warm-up: Near-Regular Graphs with Known m

One subroutine we rely on is the ability to use the stream's randomness to simulate independent sampling of edges. For completeness, we provide a proof of the following simple fact in the full version of the paper.

FACT 3.1. *Consider a universe* U *of* n *elements, and let* $p \in [0,1]$*. Let* $U_k \subseteq U$ *denote the first* k *elements in a random-order stream of* U*, and let* $X \sim Bin(n,p)$ *be a binomial random variable with parameters* n *and* p*. Then the random set* U_X *contains every element in* U *independently with probability* p*.*

Using Fact 3.1, we simulate (a variant of) Algorithm 1 with parameter ϵ under random-order edge arrivals in a graph $G = (V, E)$ with m edges and n nodes, where the degree of each node lies in the interval $(1 \pm \epsilon^2)\Delta$, and $\Delta = \omega(\log n)$. The algorithm knows n, Δ and m (but not G).

Warm-up Algorithm: Set $\epsilon := 10 \cdot (\ln n/\Delta)^{1/6}$ (see (2.1)). For round $i = 1, \ldots, t_\epsilon - 1$, sample an independent random variable $X_i \sim Bin(m - \sum_{j<i} X_j, \epsilon)$, and let S_i be the set of edges in G whose positions in the random-order stream lie in the interval $(\sum_{j\leq i} X_j, \sum_{j \leq i} X_j]$. As with Algorithm 1, each edge $e \in S_i$, upon its arrival, samples a tentative color

$$c(e) \sim_R P_i(e) := [(1+\epsilon^2)\Delta] \setminus \{c(e') \mid e' \in N(e) \cap S_j,\ j < i\},$$

where we set $c(e) \leftarrow$ *null* if $P_i(e) = \emptyset$. Unlike in Algorithm 1, in this online setting the algorithm cannot know whether the color $c(e)$ conflicts with the tentative

color of a neighboring edges $e' \in N(e) \cap S_i$ that arrives in the same round i, but *after* e in the stream. Hence, we color each edge $e \in S_i$ with its tentative color $c(e)$, unless $c(e) = null$ or some *previously-arrived* neighboring edge $e' \in N(e) \cap S_i$ also picked color $c(e') = c(e)$. In the latter case, we instead color e greedily with the first available color $j > (1+\epsilon^2)\Delta$. We let F'_i be the edges in S_i which are colored greedily.

As we show, this online algorithm inherits the performance of the basic Algorithm 1.

THEOREM 3.1. *For some absolute constant $\gamma \in (0,1)$, the warm-up algorithm described above yields a proper $\left(\Delta + O\left(\Delta^{\gamma} \cdot \log^{1-\gamma} n\right)\right) = (1+o(1))\Delta$-edge coloring of G w.h.p.*

Proof. This algorithm outputs a valid edge coloring, as it colors every edge (due to the greedy stage) and never assigns an edge a color used by an incident edge. It remains to bound its performance.

For any $i \geq 0$, Let E_i be the set of edges whose positions in the random-order stream lie in the interval $(\sum_{j<i} X_j, m]$. By Fact 3.1, the set of edges S_i is a random subset of E_i which contains each edge in E_i independently with probability ϵ. A simple induction on i shows that the sets S_i and E_i share the same distributions as their counterparts in Algorithm 1. Next, denote by $F_i \supseteq F'_i$ the set of edges $e \in S_i$ for which $c(e) \in \{null\} \cup \{c(e') \mid e' \in N(e) \cap S_i\}$. Since each edge $e \in S_i$ picks a color uniformly at random from the set of colors not picked by any of its neighboring edges in previous rounds (including the edges in F_j for all $j < i$), a simple induction on i shows that the random variables F_i and $c(e)$ in this algorithm are distributed exactly as their counterparts in Algorithm 1. Consequently, the upper bounds on $\Delta(\bigcup_i F_i) \geq \Delta(\bigcup_i F'_i)$ and $\Delta(G_{t_\epsilon})$ of Algorithm 1 hold for this online algorithm as well. Therefore, the greedy (online) algorithm colors the uncolored edges in $G_{t_\epsilon} \cup G_F$ using at most $2\cdot\Delta(G_{t_\epsilon} \cup G_F) - 1 = O(\epsilon^{1/(3K)}\Delta)$ colors w.h.p., by Theorem 2.1 and our choice of $\epsilon = 10 \cdot (\ln n/\Delta)^{1/6}$, as in (2.1). As we use $(1+\epsilon^2)\Delta$ distinct colors for all other edges, this online algorithm uses $\Delta + O\left(\epsilon^{1/(3K)}\Delta\right)$ colors overall w.h.p. Since $\Delta = \omega(\log n)$ and K is an absolute constant (see (2.2)), the theorem follows from our choice of ϵ. ∎

Assuming near-regularity, and known m. The assumption of near-regularity used by the above algorithm is common in the literature. Indeed, all prior random-order online edge-coloring algorithms assume perfect regularity [1, 3]. As pointed out in those papers, this assumption is without loss of generality in the offline model, where we can add dummy edges to make the graph regular. In a random-order online setting, this is problematic, however, as these dummy edges should be interspersed among the real edges to create a regular graph *presented in random order*. This last point seems impossible without prior knowledge of vertices' final degrees, and the number of edges, m, which we assume prior knowledge of. In the next section we show how to remove the assumption of near-regularity, as well as knowledge of m, while retaining the asymptotic performance of Theorem 3.1.

3.2 General Graphs We now present and analyze our random-order online edge coloring algorithm for general graphs $G = (V, E)$ with n nodes, m edges and maximum degree $\Delta = \omega(\log n)$. In particular, we do not assume that all nodes have degree close to Δ. The algorithm knows n, Δ; but does *not* know m nor G. Let $e_1, \ldots, e_m$ be the random stream of edges, $G^{(k)}$ be the subgraph induced by $e_1, \ldots, e_k$, and $d^{(k)}(v)$ be the degree of node v in $G^{(k)}$. Our key insight is to observe the first few edges in the input stream until some node reaches degree $\epsilon\Delta$. This is sufficient to infer (approximately) the value of m and the degree of each node in G. At the same time, we can afford to color such initial set of edges greedily.

In more detail, our algorithm consists of 3 main steps. In Step (I), we observe the first T edges until some node v reaches the degree $d^{(T)}(v) = \epsilon\Delta$ (or we reach the end of the stream). This first set of edges is colored greedily using the first available color. Let Δ_1 be the largest color used in Step (I). The following technical lemma follows from a standard application of Chernoff bounds over sums of negatively-associated variables (proof appears in the full version of the paper).

LEMMA 3.1. *Let $\epsilon \leq \frac{1}{2}$, and let $\alpha > 0$ be a constant, and assume $\Delta \geq \frac{24(\alpha+3)\ln n}{\epsilon^8}$. Then, with probability at least $1 - O(n^{-\alpha})$, the following properties hold:*[1]

1. *$T = \epsilon \cdot m(1 \pm \epsilon^2)$.*
2. *$d^{(T)}(v) = \epsilon \cdot d(v) \pm 2\epsilon^3\Delta$ for every node v.*
3. *Let $m' := T/(\epsilon(1+\epsilon^2))$. Conditioned on $m' \leq m$, every node v has $d(v) - d^{(m')}(v) \leq 2\epsilon^2\Delta$.*

Henceforth, we assume that all the high-probability events in Lemma 3.1 actually occur (otherwise the algorithm fails). In Step (II), we color the next $m' - T$ edges $R := \{e_{T+1}, \ldots, e_{m'}\}$ using colors larger than Δ_1, as described below.

Let $G_R = (V, R)$ denote the subgraph of G induced by the edges in R. Before processing the $(T+1)^{th}$ update e_{T+1}, we virtually expand G_R by adding *dummy*

[1] We let $c = a \pm b$ denote $c \in [a-b, a+b]$.

nodes W and *dummy edges* D in the following manner. For each node $v \in V$, create Δ dummy nodes $v_1, v_2, \ldots, v_\Delta$ which form a Δ-clique via dummy edges, and add extra dummy edges from v to $\max\{0, \Delta - (1/\epsilon - 1) \cdot d^{(T)}(v)\}$ of these dummy nodes $\{v_1, \ldots, v_\Delta\}$. Let H be the resulting graph. Note that at this point we only know the dummy edges in H, as the edges in G_R will arrive in future.

Let $\mathcal{A}$ denote the warm-up online algorithm from Section 3.1. In Step (II), we run this online algorithm $\mathcal{A}$ with parameter 2ϵ on H, where the edges of H are presented to $\mathcal{A}$ in *random order*. More precisely, initializing $j = T + 1$, $D' = D$ and $R' = R$, we perform the following operations for $|R| + |D|$ iterations.

- With probability $|D'|/(|R'| + |D'|)$, we sample a random edge e_d from D', and feed the edge e_d to the online algorithm $\mathcal{A}$. We then set $D' = D' \setminus \{e_d\}$ before going to the next iteration.
- With remaining probability, we feed the edge e_j to $\mathcal{A}$ and color e_j with the color $\chi(e_j) + \Delta_1$, where $\chi(e)$ is the color chosen by $\mathcal{A}$ for e_j. Then we let $R' = R' \setminus \{e_j\}$ and increase j by one.

Let Δ_2 be the largest color chosen in Step (II).

Finally, in Step (III), we color the remaining edges $e_{m'+1}, \ldots, e_m$ greedily with the first available color $j > \Delta_2$. Let Δ_3 be the largest color used at the end of Step (III).

We next analyze the above algorithm (assuming the occurrence of the high probability events from Lemma 3.1). Obviously this algorithm computes a feasible coloring. By definition, Step (I) uses $\Delta_1 \leq 2\epsilon\Delta$ colors. Analogously, by Item 3 of Lemma 3.1, the number of colors used in Step (III) is at most $\Delta_3 - \Delta_2 = O(\epsilon^2\Delta)$. It remains to upper bound the number of colors $\Delta_2 - \Delta_1$ used in the second step. To this end, we note that Lemma 3.1 implies that H is near-regular. More precisely, we have the following bound, whose proof is deferred to the full version of the paper.

LEMMA 3.2. *The graph H satisfies $d_H(v) = \Delta(1 \pm 4\epsilon^2)$ for all $v \in V(H)$, w.h.p.*

It is easy to see that Step (II) implements the warm-up algorithm on H, as the edges of H are fed to this algorithm in a uniform random order. Thus, by Theorem 3.1, w.h.p. the number of colors used in Step (II) is at most $\Delta_2 - \Delta_1 \leq \Delta + O\left(\Delta^\gamma \cdot \log^{1-\gamma} n\right)$ for a constant $\gamma \in (0, 1)$. By choosing ϵ small enough so that $\epsilon\Delta \leq \Delta^\gamma \cdot \log^{1-\gamma} n$, we immediately obtain Theorem 1.1.

4 Low-Recourse Dynamic Algorithm

In this section, we give an implementation of Algorithm 1 in a dynamic setting with low recourse. Before describing our algorithm, we explain why in the dynamic setting we can focus our attention on near-regular graphs, as required of inputs to Algorithm 1.

(Near-)Regularizing Gadget. Consider a dynamic input graph $G = (V, E)$ on n nodes, where the degree of each node remains at most Δ all the time. In this dynamic setting, we describe a procedure to maintain a super-graph $G' = (V', E')$ of $G = (V, E)$, with $V' \supseteq V$ and $E' \supseteq E$, such that: (a) each node in G' always has degree either Δ or $\Delta - 1$, and (b) each update (edge insertion/deletion) in G results in a constant number of updates in G'. The node-set V' of G' consists of the nodes $v \in V$ of the input graph, plus Δ *dummy nodes* $v_1, v_2, \ldots, v_\Delta$ for each $v \in V$.

Initially, when G is empty,[2] the edge-set of G' is defined as follows. For each node $v \in V$ in the input graph, the set of nodes $\{v, v_1, \ldots, v_\Delta\}$ induces a $(\Delta + 1)$-clique of *dummy edges* in the supergraph G'. In other words, there is a *dummy edge* (v, v_i) for all $v \in V, i \in [\Delta]$; and a *dummy edge* (v_i, v_j) for all $v \in V, i \in [\Delta], j \in [\Delta], i \neq j$. At this point in time, the edge-set E' of G' consists only of the dummy edges.

Subsequently, whenever an edge $e = (u, v)$ in added to (resp., removed from) G during an update, we perform the following steps. We first add e to (resp., remove e from) G'. Next, let i and j be the smallest indices in $[\Delta]$ for which the dummy edges (u, u_i) and (v, v_j) currently exist (resp., do not exist) in G': we remove (resp., add) these two dummy edges (u, u_i) and (v, v_j).

Note that every update in G generated by an oblivious adversary results in three updates (likewise generated by an oblivious adversary) in G'. Since any k-edge-coloring of G' trivially induces a k-edge-coloring of G, our goal will be to maintain a $(1+\epsilon)\Delta$-edge-coloring of G'.

Accordingly, w.l.o.g., henceforth we will assume that the input graph G given to us at preprocessing is near-regular (specifically, each of its nodes has degree either Δ or $\Delta - 1$), and that the input graph G remains near-regular throughout the sequence of updates.

Our dynamic algorithm: At each time t (i.e., after the t-th update), our dynamic algorithm strives to assign tentative colors $c^{(t)}(e)$ to all edges in the current input graph, denoted by $G^{(t)} = (V, E^{(t)})$, as in Algorithm 1. It likewise defines failures in the same way as in Algorithm 1. Finally, it maintains a coloring of the failed and otherwise uncolored edges using a simple $O(\Delta)$-edge-coloring, constant-recourse *dynamic* algorithm, which we denote by SIMPLECOLOR

[2] It is easy to extend our gadget to the setting where the input graph G is not empty at preprocessing.

(e.g., [7, 39]).

For the rest of this section, the superscript (t) on any given notation will indicate the state of the concerned object after the t^{th} update in the input graph.[3] Intuitively, the outcome of our dynamic algorithm just after the t^{th} update will be the same as the outcome we would get if we ran Algorithm 1 in the static setting with $G^{(t)}$ as input (see Lemma 4.1). We will refer to an *unordered* pair of nodes (u, v), with $u, v \in V$, as a *potential edge*. Thus, there are $\binom{|V|}{2}$ many potential edges, and the set of potential edges do not change during the sequence of updates. In contrast, we will refer to an edge $e \in E^{(t)}$ as a *current edge* in the input graph after the t^{th} update.

Preprocessing: We start by assigning a *round* $i(u, v) \in [t_\epsilon]$ to each potential edge (u, v), where these $i(u, v)$ values are i.i.d. samples from the capped geometric distribution $CappedGeo(\epsilon, t_\epsilon)$ with success probability ϵ and at most t_ϵ attempts.[4] For each $j \in [t_\epsilon]$, we let $S_j := \{(u, v) \mid i(u, v) = j\}$ denote the set of all *potential* edges that are assigned to round j. Furthermore, for each $t \geq 0$, let $S_j^{(t)} := S_j \cap E^{(t)}$ denote the set of *current* edges (u, v) after the t^{th} update which have $i(u, v) = j$.

In future, throughout the sequence of updates, the random variables $\{i(u, v)\}$ will determine which sets of current edges get sampled in which round (see Step 6 of Algorithm 1). Specifically, consider any potential edge (u, v), with $i(u, v) = j$, and any two nonnegative integers $t \neq t'$ such that $(u, v) \in E^{(t)} \cap E^{(t')}$. Then the edge (u, v) will get sampled in the same round j in both $G^{(t)}$ and $G^{(t')}$. Thus, a given edge gets assigned to the same round across all the updates.

After drawing the random variables $\{i(u, v)\}$ for all potential edges as described above, we implement Algorithm 1 on the input $G^{(0)} = (V, E^{(0)})$ given to us at the preprocessing phase.

Handling an update: For any $t \geq 1$, consider the t^{th} update which changes the input graph G from $G^{(t-1)} = (V, E^{(t-1)})$ to $G^{(t)} = (V, E^{(t)})$. Our dynamic algorithm handles this update by computing an edge-coloring for $G^{(t)}$ in three steps, as described below.

Step I: We perform the following operations in increasing order of $i = 1, 2, \ldots, t_\epsilon - 1$:

- For every *potential* edge $e \in S_i$, we first define its palette

$$P_i^{(t)}(e) := [\Delta(1+\epsilon^2)] \setminus \left\{ c^{(t)}(e') \;\middle|\; e' \in N^{(t)}(e) \cap \bigcup_{j<i} S_j^{(t)} \right\}.$$

  Next, we call Algorithm 2 to update the tentative color $c^{(t)}(e)$ of e. Note that as in Algorithm 1, if $P_i^{(t)}(e) = \emptyset$, then Algorithm 2 sets $c^{(t)}(e) \leftarrow null$.

Step II: For every round $i \in [t_\epsilon - 1]$, we now define the set of *failed* current edges $F_i^{(t)}$. Specifically, the set $F_i^{(t)}$ consists of all the edges $e \in S_i^{(t)}$ such that:

$$c^{(t)}(e) \in \{\text{null}\} \cup \{c^{(t)}(e') : e' \in N^{(t)}(e) \cap S_i^{(t)}\}.$$

Let $F^{(t)} := \bigcup_{i=1}^{t_\epsilon - 1} F_i^{(t)}$ denote the set of all failed current edges across all rounds. Every current edge $e \in \left(\bigcup_{i=1}^{t_\epsilon - 1} S_i^{(t)}\right) \setminus F^{(t)}$ gets colored with its tentative color $c^{(t)}(e)$.

Step III: Finally, let $G_U^{(t)} := G[F^{(t)} \cup S_{t_\epsilon}^{(t)}]$ be the subgraph of $G^{(t)}$ consisting of all the edges that are not colored using their tentative colors in Step II above. We use SimpleColor to maintain an $O(\Delta(G_U^{(t)}))$ edge coloring of $G_U^{(t)}$. In more detail, after each update, denoting by $A \bigoplus B := (A \setminus B) \cup (B \setminus A)$ the symmetric difference, we think of the graph $G_U^{(t)} := G[F^{(t)} \cup S_{t_\epsilon}^{(t)}]$ as having undergone $|F^{(t)} \bigoplus F^{(t-1)}| + |S_{t_\epsilon}^{(t)} \bigoplus S_{t_\epsilon}^{(t-1)}|$ updates, which we feed to algorithm SimpleColor.

This concludes the description of our dynamic algorithm.

Algorithm 2 TentativelyColor(e)

```
1: if c^(t-1)(e) ∈ P_i^(t-1)(e) \ P_i^(t)(e) then
2:     if P_i^(t) ≠ ∅, then sample c^(t)(e) ∼_R P_i^(t)(e),
3:     else set c^(t)(e) ← null
4: else
5:     if P_i^(t) ≠ ∅, then sample c ∼_R P_i^(t)(e),
6:     else set c ← null
7:     if c = null or c ∈ P_i^(t)(e) \ P_i^(t-1)(e) then
8:         c^(t)(e) ← c
9:     else
10:        c^(t)(e) ← c^(t-1)(e) ▷ Keep the previous color
```

Analysis: Looking back at Algorithm 2, a moment's thought reveals that if the tentative color $c^{(t-1)}(e)$ was chosen uniformly at random from the set $P_i^{(t-1)}(e)$, then the tentative color $c^{(t)}(e)$ is also chosen uniformly at

[3] The reader should not confuse time t in the dynamic setting with the last round t_ϵ in phase one of Algorithm 1.

[4] For a random variable $X \sim CappedGeo(\epsilon, t_\epsilon)$, we have $\Pr[X = k] = \begin{cases} \epsilon \cdot (1-\epsilon)^{k-1} & k \in \{1, \ldots, (t_\epsilon - 1)\} \\ (1-\epsilon)^{t_\epsilon - 1} & k = t_\epsilon. \end{cases}$

random from the set $P_i^{(t)}(e)$. This, however, is far from being sufficient for our purpose. In particular, we need a formal (and much stronger) guarantee stated below.

LEMMA 4.1. *For each time $t \geq 0$, the joint distribution $\{c^{(t)}(e)\}_{e \in E^{(t)}}$ of colors sampled by the dynamic algorithm is distributed identically to $\{c(e)\}_e$ of Algorithm 1 when applied to graph $G^{(t)}$.*

We defer the proof of Lemma 4.1 to the full version of the paper. This lemma, together with our analysis of Algorithm 1, immediately leads us to the following corollary.

COROLLARY 4.1. *For ϵ as in (2.1) and K as in (2.2), the above dynamic algorithm $\Delta(1 + O(\epsilon^{(1/3K)}))$-edge-colors G at any time t, w.h.p.*

Bounding Recourse: We now fix some $t \geq 1$, and bound the expected recourse our algorithm has to pay while handling the t^{th} update. Say that an edge $e \in E^{(t)} \cap E^{(t-1)}$ is *dirty* iff $i(e) \leq t_\epsilon - 1$ and it changes its tentative color $c(e)$ due to the t^{th} update. Let D_j denote the set of dirty edges e assigned to round $i(e) = j$. We will use the symbol $D_{<i} = \cup_{j<i} D_j$ to denote all the dirty edges at rounds $j < i$. Let $D = D_{<t_\epsilon}$ denote the set of dirty edges across all rounds.

LEMMA 4.2. *The recourse of the dynamic algorithm to handle the t^{th} update is $O(1 + |D|)$.*

Proof. Let e^* denote the edge being inserted/deleted during the t^{th} update. The additive $+1$ term in the claimed recourse bound of $O(1 + |D|)$ comes from the edge e^*. To simplify notations, we will assume $c^{(t-1)}(e^*) = c^{(t)}(e^*)$ for the rest of this proof. Any other edge $e \in E^{(t-1)} \cap E^{(t)}$ changes its final color during the t^{th} update only if: $i(e) < t_\epsilon$, and {either (1) the edge e changes its tentative color, or (2) the edge e switches from being successful to failed (or vice versa) without changing its tentative color}. In case (1), we clearly have $e \in D$. In case (2), the edge e must have at least one neighboring edge $e_d \in (D \cup \{e^*\}) \cap N(e)$ such that $c^{(t-1)}(e) = c^{(t)}(e) \in \{c^{(t-1)}(e_d), c^{(t)}(e_d)\}$. We *charge* the 1 unit of recourse the algorithm has to pay for e to any one such edge e_d. A moment's thought reveals that each edge $(u, v) \in D \cup \{e^*\}$ can receive at most 4 units of charge in this scheme, one for each ordered pair $\{u, v\} \times \{c^{(t-1)}(u, v), c^{(t)}(u, v)\}$. Hence, the algorithm's recourse is at most $O(1 + |D|)$. ∎

Below, we state the key lemma that contains the technical meat of our argument. Since the argument requires some nontrivial properties of Algorithm 1, we defer the proof of Lemma 4.3 to the full version of the paper.

LEMMA 4.3. *For every round $i \in [t_\epsilon - 1]$, we have: $\mathsf{E}[|D_i|] \leq 6\epsilon + 6\epsilon \cdot \mathsf{E}[|D_{<i}|]$.*

COROLLARY 4.2. *Our dynamic algorithm has an expected recourse of $O(1/\epsilon^{(3/K)})$ per update.*

Proof. Since $|D_{<1}| = 0$ and $D = D_{<t_\epsilon}$, from Lemma 4.3 we derive that:

$$\mathsf{E}[|D|] \leq \sum_{i=0}^{t_\epsilon - 1} 6\epsilon \cdot (1 + 6\epsilon)^i \leq (1 + 6\epsilon)^{t_\epsilon} \leq \exp(6\epsilon \cdot t_\epsilon)$$
$$\leq \exp((3/K) \cdot \ln(1/\epsilon)) = 1/\epsilon^{(3/K)}.$$

The last inequality in the derivation above holds because of (2.2). The corollary now follows from Lemma 4.3. ∎

From (2.1), (2.2), Corollary 4.1 and Corollary 4.2, we obtain the following theorem.

THEOREM 4.1. *There exists two absolute constants $\gamma, \gamma' \in (0, 1)$ such that for all $\epsilon \in \left[(\log n/\Delta)^{\gamma'}, \gamma\right]$ we can maintain, w.h.p., a proper $(1+\epsilon)\Delta$-edge-coloring in a dynamic graph G of maximum degree $\Delta = \omega(\log n)$, with $O(\mathsf{poly}(1/\epsilon))$ expected recourse per update.*

5 Conclusions and Open Questions

We presented one common approach for tackling edge coloring in two widely-studied relaxations of the online model of computation, making progress on (and in one case, resolving) the conjecture of Bar-Noy et al. [4] for these models. We conclude with a few interesting research directions.

Adversarial Online Arrivals. The most natural question is whether the Bar-Noy et al. conjecture holds in the strictest, adversarial edge-arrival model. This question still seems out of reach. One algorithmic approach which suggests itself is to extend the ideas in [11]. This would require some form of online dependent rounding for fractional matching under edge arrivals, generalizing [12]. Alternatively, it is not implausible that the Bar-Noy et al. conjecture is false under adversarial edge arrivals, despite being true for vertex arrivals [11]. Such a refutation of this conjecture would mirror a similar separation between these arrival models recently proven for online matching [18].

Knowledge of Δ. All our algorithms assume knowledge of the maximum degree Δ. This assumption is common to all prior best algorithms in the models we study [3, 1, 14].[5] In fact, Cohen et al. [11] showed that

[5] Duan et al. [14] run a logarithmic number of algorithms for unknown Δ, and switch between their colorings whenever Δ changes. This results in *fast update time*, but high recourse.

not knowing Δ in their online model results in a strictly harder problem, for which no better than $\frac{e}{e-1}\Delta$-edge-coloring algorithm exists, for any (unknown) Δ. Is the same separation between known and unknown Δ true for the models studied in this paper?

Acknowledgments

We thank Janardhan Kulkarni for many helpful discussions. The work of Sayan Bahttacharya was supported by Engineering and Physical Sciences Research Council, UK (EPSRC) Grant EP/S03353X/1. The work of Fabrizio Grandoni was supported in part by the SNF Excellence Grant 200020B_182865/1. The work of David Wajc was supported in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship.

References

[1] Gagan Aggarwal, Rajeev Motwani, Devavrat Shah, and An Zhu. Switch scheduling via randomized edge coloring. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*, pages 502–512, 2003.

[2] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[3] Bahman Bahmani, Aranyak Mehta, and Rajeev Motwani. Online graph edge-coloring in the random-order arrival model. *Theory of Computing (conference version appeared in SODA 2010)*, 8(1):567–595, 2012.

[4] Amotz Bar-Noy, Rajeev Motwani, and Joseph Naor. The greedy algorithm is optimal for on-line edge coloring. *Information Processing Letters (IPL)*, 44(5):251–253, 1992.

[5] Leonid Barenboim and Tzalik Maimon. Fully-dynamic graph algorithms with sublinear time inspired by distributed computing. *Procedia Computer Science*, 108:89–98, 2017.

[6] Aaron Bernstein, Jacob Holm, and Eva Rotenberg. Online bipartite matching with amortized replacements. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 947–959, 2018.

[7] Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. Dynamic algorithms for graph coloring. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–20, 2018.

[8] Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 384–393, 2014.

[9] Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. The complexity of distributed edge coloring with small palettes. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2633–2652, 2018.

[10] Moses Charikar and Paul Liu. Improved algorithms for edge colouring in the w-streaming model. In *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)*, page To appear, 2021.

[11] Ilan Reuven Cohen, Binghui Peng, and David Wajc. Tight bounds for online edge coloring. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*, pages 1–25, 2019.

[12] Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 960–979, 2018.

[13] Richard Cole, Kirstin Ost, and Stefan Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Combinatorica*, 21(1):5–12, 2001.

[14] Ran Duan, Haoqing He, and Tianyi Zhang. Dynamic edge coloring with improved approximation. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1937–1945, 2019.

[15] Devdatt Dubhashi, David A Grable, and Alessandro Panconesi. Near-optimal, distributed edge colouring via the nibble method. *Theoretical Computer Science*, 203(2):225–252, 1998.

[16] Michael Elkin, Seth Pettie, and Hsin-Hao Su. (2δ—1)-edge-coloring is much easier than maximal matching in the distributed setting. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 355–370, 2014.

[17] Björn Feldkord, Matthias Feldotto, Anupam Gupta, Guru Guruganesh, Amit Kumar, Sören Riechers, and David Wajc. Fully-dynamic bin packing with little repacking. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 51:1–51:24, 2018.

[18] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*, pages 26–37, 2019.

[19] David A. Grable. A large deviation inequality for functions of independent, multi-way choices. *Combinatorics Probability and Computing*, 7(1):57–63, 1998.

[20] Albert Gu, Anupam Gupta, and Amit Kumar. The power of deferral: maintaining a constant-competitive steiner tree online. In *ACM Symposium on Theory of Computing (STOC)*, pages 525–534, 2013.

[21] Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 537–550, 2017.

[22] Anupam Gupta and Amit Kumar. Online steiner tree with deletions. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages

455–467, 2014.

[23] Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining assignments online: Matching, scheduling, and flows. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 468–479, 2014.

[24] Anupam Gupta and Sahil Singla. Random-order models. *arXiv preprint arXiv:2002.12159*, 2020.

[25] Ian Holyer. The np-completeness of edge-coloring. *SIAM Journal on Computing (SICOMP)*, 10(4):718–720, 1981.

[26] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 587–596, 2011.

[27] Howard J. Karloff and David B. Shmoys. Efficient parallel algorithms for edge coloring problems. *J. Algorithms*, 8(1):39–52, 1987.

[28] Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 303–312, 2014.

[29] Dénes König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77(4):453–465, 1916.

[30] Nitish Korula, Vahab Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM Journal on Computing (SICOMP)*, 47(3):1056–1086, 2018.

[31] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 597–606, 2011.

[32] Nicole Megow, Martin Skutella, José Verschae, and Andreas Wiese. The power of recourse for online MST and TSP. In *Proceedings of 39th International Colloquium on Automata, Languages, and Programming, (ICALP)*, volume 7391, pages 689–700.

[33] Adam Meyerson. Online facility location. In *Proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS)*, pages 426–431, 2001.

[34] Rajeev Motwani, Joseph (Seffi) Naor, and Moni Naor. The probabilistic method yields deterministic parallel algorithms. *Journal of Computer and System Sciences*, 49(3):478–516, 1994.

[35] Julius Petersen. Sur le théoreme de tait. *L'intermédiaire des Mathématiciens*, 5:225–227, 1898.

[36] Hsin-Hao Su and Hoa T. Vu. Towards the locality of vizing's theorem. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 355–364, 2019.

[37] P.G. Tait. Remarks on the colourings of maps. *Proc. R. Soc. Edinburgh*, 10:729, 1880.

[38] Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Diskret analiz*, 3:25–30, 1964.

[39] David Wajc. Rounding dynamic matchings against an adaptive adversary. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 194–207, 2020.

Appendix

A A Lower Bound for Random Order Online Edge Coloring

Bar-Noy et al. [4] gave a simple lower bound for edge coloring under adversarial arrivals. Specifically, they showed a family of graphs $\mathcal{F}$ with maximum degree $\Delta = O(\sqrt{\log n})$ for which any randomized online algorithm $\mathcal{A}$ colors some graphs in $\mathcal{F}$ with $2\Delta - 1$ colors with constant probability. Extending these ideas slightly, we show that the same holds even if the arrival order is randomized.

LEMMA A.1. *There exists a distribution over n-node graphs $\mathcal{G}$ of maximum degree $\Delta = \Omega(\sqrt{\log n})$, for which any online edge coloring algorithm $\mathcal{A}$ must, with constant probability, use $2\Delta-1$ colors on a graph $G \sim \mathcal{G}$ presented in random order.*

Proof. Consider a star on $\Delta - 1$ leaves. If Algorithm $\mathcal{A}$ uses $2\Delta - 2$ or fewer colors, then it may color any such star's edges with $\binom{2\Delta-2}{\Delta-1}$ possible subsets of colors. If Δ such stars' edges are colored using the same subset $S \subseteq [2\Delta - 2]$ of $\Delta - 1$ colors and some node v neighbors the roots of these Δ stars, then the algorithm fails, as it is forced to use Δ colors outside of S for the edges of v, for a total of $2\Delta - 1$ distinct colors. We show a random graph $\mathcal{G}$ for which this bad event happens with constant probability, even when the edges are presented in random order.

Our graph consists of independent copies of the following random graph, $\mathcal{H}$. The graph $\mathcal{H}$ contains $\beta := 2\Delta \cdot \binom{2\Delta-2}{\Delta-1} \cdot \binom{2\Delta-1}{\Delta} \leq 4^{O(\Delta)}$ stars with $\Delta - 1$ leaves, and one node v which neighbors the centers of Δ randomly-chosen such stars. For any star, the probability that all $\Delta - 1$ edges of the star arrive before any of the Δ edges of v arrive is $\frac{(\Delta-1)!\Delta!}{(2\Delta-1)!} = 1/\binom{2\Delta-1}{\Delta}$. Therefore, by linearity of expectation, if we denote by X the fraction of stars in $\mathcal{H}$ whose edges arrive before all edges of v, we have that $\mu := \mathsf{E}[X] = 1/\binom{2\Delta-1}{\Delta}$. By Markov's inequality applied to the non-negative variable $Y := 1 - X$, whose expectation is $\mathsf{E}[Y] = 1 - \mu$, we have that

$$\Pr\left[X \leq \frac{\mu}{2}\right] = \Pr\left[Y \geq 1 - \frac{\mu}{2}\right] \leq \frac{1-\mu}{1-\frac{\mu}{2}} = 1 - \frac{\mu}{2-\mu} \leq 1 - \frac{\mu}{2}. \tag{A.1}$$

Now, if $X \geq \frac{\mu}{2} = 1/(2 \cdot \binom{2\Delta-1}{\Delta})$, then at least

$\beta \cdot \frac{\mu}{2} = \Delta \cdot \binom{2\Delta-2}{\Delta-1}$ of the stars of $H \sim \mathcal{H}$ have all their edges arrive before any edge of v arrives. By pigeonhole principle, some Δ of these stars are colored with a common set of $\Delta - 1$ colors, $S \subset [2\Delta - 1]$. If v neighbors the roots of Δ such stars whose edges are colored with the colors in S, then Algorithm $\mathcal{A}$ fails, as it must color the graph using $2\Delta - 1$ colors, as argued above. Therefore, conditioned on $X \geq \frac{\mu}{2}$, Algorithm $\mathcal{A}$ fails when coloring $H \sim \mathcal{H}$ with probability at least

(A.2)
$$\Pr\left[\mathcal{A} \text{ fails on } H \sim \mathcal{H} | X \geq \frac{\mu}{2}\right] \geq 1/\binom{\beta}{\Delta} \geq 4^{-O(\Delta^2)}.$$

Consequently, combining (A.1) and (A.2), and using $\mu = 1/\binom{2\Delta-1}{\Delta} = 4^{-O(\Delta)}$, we find that the unconditional probability of Algorithm $\mathcal{A}$ not failing due to H is at most

$$\begin{aligned}
&\Pr\left[\mathcal{A} \text{ does not fail on } H \sim \mathcal{H}\right] \\
&\leq 1 - \Pr\left[\mathcal{A} \text{ fails on } H \sim \mathcal{H} | X \geq \frac{\mu}{2}\right] \cdot \Pr\left[X \geq \frac{\mu}{2}\right] \\
&\leq 1 - 4^{-O(\Delta^2)} \cdot \frac{\mu}{2} \\
&= 1 - 4^{-O(\Delta^2)}.
\end{aligned}$$

As stated above, the random graph $\mathcal{G}$ we consider consists of some γ independent copies of $\mathcal{H}$. For independent copies of $\mathcal{H}$, the above upper bound on the probability of $\mathcal{A}$ not failing on a copy $H \sim \mathcal{H}$ holds independently of other copies' realization and coloring by $\mathcal{A}$. Therefore, letting $\mathcal{G}$ consist of some sufficiently large $\gamma := 4^{\Theta(\Delta^2)}$ independent copies of $\mathcal{H}$, we have that

$$\Pr\left[\mathcal{A} \text{ does not fail on } G\right] \leq \left(1 - 4^{-O(\Delta^2)}\right)^{\gamma} \leq \frac{1}{e}.$$

Therefore, Algorithm $\mathcal{A}$ fails on G with constant probability. The lemma follows by noting that G consists of some $n = \gamma \cdot (\beta + 1) = 4^{\Theta(\Delta^2)}$ nodes, and therefore $\Delta = \Omega(\sqrt{\log n})$. ∎