A 1.43-Competitive Online Graph Edge Coloring Algorithm In The Random Order Arrival Model

Bahman Bahmani[*] Aranyak Mehta[†] Rajeev Motwani[‡]

Abstract

A classic theorem by Vizing proves that if the maximum degree of a graph is Δ, then it is possible to color its edges, in polynomial time, using at most $\Delta+1$ colors. However, this algorithm is offline, i.e., it assumes the whole graph is known in advance. A natural question then is how well we can do in the online setting, where the edges of the graph are revealed one by one, and we need to color each edge as soon as it is added to the graph. Online edge coloring has an important application in fast switch scheduling. Here, a natural model is that edges arrive online, but in a random permutation. Even in the random permutations model, the best analysis for any algorithm is factor 2, which comes from the simple greedy algorithm (which is factor 2 even in the worst case online model). The algorithm of Aggarwal et al. [1] provides a 1+o(1) factor algorithm, but for the case of multigraphs, when $\Delta = \omega(n^2)$, where n is the number of vertices. In this paper, we show that for graphs with $\Delta = \omega(log\, n)$, it is possible to color the graph with $1.43\Delta + o(\Delta)$ colors in the online random order model. Our algorithm is inspired by a 1.6 factor distributed offline algorithm of Panconesi and Srinivasan [9], which we extend by reusing colors online in multiple rounds.

1 Introduction

An edge coloring of a graph is a coloring of its edges so that no two edges incident on each other get the same color. If the maximum degree of a graph is Δ then, obviously, every edge coloring needs at least Δ colors. A classic theorem by Vizing [10] proves that it is possible to edge-color a graph using at most $\Delta + 1$ colors. However, determining whether the edge coloring number is Δ or $\Delta + 1$ is known to be NP-complete for general graphs [6]. The proof of Vizing's theorem is constructive and actually gives a polynomial time algorithm to find an edge coloring using at most $\Delta + 1$ colors. For bipartite graphs there are fast algorithms to edge color using Δ colors [3]. However, these algorithms are offline, i.e., they assume that the graph is known in advance. A natural question then is how well we can do in the online setting, where the edges of the graph are revealed one by one, and we need to color each edge as soon as it is added to the graph.

Fast online edge coloring algorithms are not just of theoretical interest, but also important in high speed network switching. Bipartite edge coloring has a direct application in switch scheduling, see [1] and the references there for details. A natural model here is that the edges arrive online in a random permutation. This is the input model we work with in this paper. We will restrict our attention to bipartite graphs: this is without loss of generality (see an explanation in [9] for the reduction to bipartite graphs in the offline model. This can be modified for the online model as well). Also, the motivating application of switch scheduling is a bipartite graph edge coloring problem. We note that the random order arrival model can simply be considered as an algorithmic technique for fast offline approximation: randomly permute the edges and run a (simple and local) online algorithm.

1.1 Problem Definition Let $G = (B, T, E)$ be a bipartite graph. Throughout, we will call the vertices in B as bottom vertices, and vertices in T as top vertices. The vertices are known in advance, while the edges E are unknown. Edges arrive online in a random permutation of E. We have to color each edge as soon as it arrives, so as to get a valid edge coloring at the end of the algorithm. The objective is to do this using the smallest number of colors possible.

1.2 Prior work and Our Results Obviously, a simple greedy algorithm (Greedy), which colors each edge with the smallest color not already used by a neighboring edge, will color the graph with no more than $2\Delta-1$ colors. This is true in the worst case online model, with adversarial input order on E. But in fact, this is the best known analysis for Greedy even in the random order arrival model.

Bar-Noy et al. [2] prove that Greedy is optimal for both deterministic and randomized algorithms in the worst case order model. However, the examples they

*Stanford University, Supported by Stanford Graduate Fellowship, bahman@stanford.edu

†Google, Inc., Mountain View, CA, aranyak@google.com

‡Stanford University, rajeev@cs.stanford.edu

construct to prove the lower bounds are very sparse, i.e., $\Delta = O(\log n)$ for deterministic algorithms, and $\Delta = O(\sqrt{\log n})$ for randomized algorithms. Then, the natural question is can we do better than Greedy if the graph is "dense".

Aggarwal et al. [1] gave an online algorithm which colors a bipartite graph using $\Delta + o(\Delta)$ colors in the random order arrival model, when $\Delta = \omega(n^2)$. Thus, this achieves essentially optimal performance, but in an extremely dense *multigraph* (we note that this is shown to be sufficient for their application of switch scheduling).

Thus, no algorithm is known to perform better than factor 2 (using fewer than $2\Delta - 1$ colors) in the (random order arrival) online model, when the graph is denser than $\log n$, but sparser than n^2. We note a minor point that the algorithm in [1] involves some batch processing, so it is in fact not strictly online (but this batch processing is reasonable for the application of switch scheduling).

In this paper we provide an online algorithm which uses $1.43\Delta + o(\Delta)$ colors for graphs with $\Delta = \omega(log\, n)$. We briefly describe the algorithm here (a detailed description is provided in Section 2): The algorithm has a number of palettes P^i. It partitions the incoming edges into two types, "Early" and "Late", depending on the arrival time of the edge. For an early edge, the algorithm tries to color it with a random color from P^1. If it fails, then it tries to color it with a random color from P^2, and so on. After all the early edges have arrived, a subset R_b^i of colors from P^i have failed to be used by each bottom node b. The algorithm augments this set by injecting a set of new colors N^i. Then, for a late edge (b, t), the algorithm tries to color it using a random color chosen from $R_b^1 \cup N^1$. If it fails to do so, it will try to color the edge with a random color from $R_b^2 \cup N^2$, and so on. The basic idea is that each color from the main palettes gets a second chance (at each bottom node) before it is thrown away.

By extending the algorithm to allow b to try a color c more than two times, we may hope to use a smaller number of colors. An extreme case is with a single palette and an unlimited (i.e. sufficiently large) number of trials per color. We believe this may lead to a $\Delta + o(\Delta)$ algorithm. We describe this algorithm in Section 5. However, our current techniques do not completely extend to analyze that algorithm. In Section 5, we exactly point out the difficulty our current techniques face in analyzing the generalized algorithm.

1.3 Other Related Work

There is another, related sequence of results in the literature on distributed offline algorithms for edge coloring. The first such algorithm with a factor less than 2 was provided in [9] (call it PS), which uses $\frac{e}{e-1}\Delta \simeq 1.59\Delta$ colors. Our online algorithm is inspired by PS. We first extend PS to a distributed algorithm which uses only 1.43Δ colors. The extension works by making every phase of PS work in two rounds instead of one, by reusing colors rejected in the first round. Then we show how our online algorithm is an online version of this two round distributed algorithm. We note that Aggarwal et al. [1] mention in their introduction that PS can be made online – however no proof or suggestion is given as to how this can be done. Subsequent to [9] distributed algorithms were found [4, 7] using a nearly-optimal number of colors. However, it is unclear how to make those algorithms work in the online model. We also note the unpublished monograph [5], which includes a clear exposition of the concentration of measure techniques used in the analysis of these distributed algorithms.

2 The Algorithm

2.1 Online Algorithm

Let $r = e/(e+1)$. We partition the input edges into two types: Early and Late, as follows: any edge that arrives before time $rn\Delta$ is considered an Early edge, and all other edges are considered as late edges. The algorithm has a collection of $L = O(log\ n)$ main palettes $P^1, P^2, P^3, \ldots$, as well as L augmenting palettes $N^1, N^2, N^3, \ldots$, each with a distinct set of colors. Palette P^i has size $r\Delta/e^{i-1}$. Palette N^i has size x_i. We will determine the values of x_i's later. The algorithm also has a special palette, P^∞, with $O(log\ n)$ colors.

For each bottom vertex b and each color $c \in \cup_{i\geq 1} P^i$, we maintain one bit $touched_b(c)$, which determines if b has tried to color any of its edges with c. For each b and each $i \in [1, L]$, we also maintain a set R_b^i of colors tried by b from P^i, but rejected by a neighboring top vertex. The pseudocode of the online algorithm is in Figure 1.

Clearly, the algorithm never colors two edges incident on each other with the same color. We will prove that, by choosing the right values for x_i's, with high probability, the algorithm does not abort (in the last step), i.e., each edge gets colored from one of the palettes.

We analyze the online algorithm Online by analyzing a distributed offline version, algorithm Distributed-2-Round . We describe the distributed algorithm below.

2.2 Distributed-2-Round Algorithm

In the distributed algorithm, the graph is randomly partitioned into two subgraphs, called Early and Late, as follows: each bottom node randomly partitions its edges to two subsets of sizes $r\Delta$, considered as edges in the Early subgraph, and $(1-r)\Delta$, considered as edges in the Late

For all bottom vertices b and all colors c, initialize $touched_b(c) = 0$. For all bottom vertices b and all levels $i \in [1, L]$, set $R_b^i = \emptyset$.

When the s^{th} edge $e = (b, t)$ arrives in the online order:

If $s \leq rn\Delta$ (e is an early edge):

- Set $i = 0$
- While (e is not colored and $i < L$):
  - $i{+}{+}$
  - If there exists a color c in palette $P^i - R_b^i$ with $touched_b(c) = 0$:
    - pick a color c^* uniformly at random among all colors from palette P^i with $touched_b(c) = 0$. Set $touched_b(c^*) = 1$.
    - if t has not used c^* already, then color e with c^*, else set $R_b^i = R_b^i \cup \{c^*\}$, and $touched_b(c^*) = 0$.
- If e is not yet colored, color it greedily from P^∞. If no such color is available in P^∞, then abort.

If $s > rn\Delta$ (e is a late edge):

- Set $i = 0$
- While (e is not colored and $i < L$):
  - $i{+}{+}$
  - If there exists a color c in $R^i \cup N^i$ with $touched_b(c) = 0$:
    - pick a color c^* uniformly at random among all colors from palette $R_b^i \cup N_i$ with $touched_b(c) = 0$. Set $touched_b(c^*) = 1$.
    - if t has not used c^* already then color e with c^*.
- If e is not yet colored, color it greedily from P^∞. If no such color is available in P^∞, then abort.

Figure 1: The Online Algorithm

subgraph. Here, $r = e/(1 + e)$ is such that $1 - r = r/e$. We alternately refer to early and late edges, respectively, as round 1 and round 2 edges. The algorithm works in a number of phases on each of these subgraphs. For the Early subgraph, it just runs the PS algorithm (with random color proposals), using palettes P^i, where $|P^i| = r\Delta/e^{i-1}$. Afterwards, for each bottom node b, and each phase i, we will have a subset of R_b^i of colors in P^i which were proposed by but rejected from b in the first subgraph. We also have an augmenting palette N^i for each phase. Then, for each edge (b, t) in the Late subgraph, the bottom node b proposes a random color from where $R_b^1 \cup N^1$. If it gets rejected by t, then b will try to color the edge using $R_b^2 \cup N^2$, and so on. Eventually, after $O(log\ n)$ phases, we will run a brute force method to color any remaining edge using a final special palette P^∞, just as in [9].

Remark 1. *This distributed algorithm is inspired by a distributed algorithm due to Panconesi and Srinivasan [9], which achieves an edge coloring using* $1.59\Delta + o(\Delta)$ *colors in the offline distributed model. This algorithm can be seen as a special case of our distributed algorithm with* $r = 1$ *(instead of* $r = \frac{e}{1+e} < 1$*), i.e., there is only one round per phase. The additional insight in our distributed algorithm is to start with fewer colors than the degree, and reuse the rejected colors in the second subgraph (or for Late edges). The degrees seen by subsequent phases decreases slower than in PS, but each phase uses fewer colors than the degree. Overall, the tradeoff is in our favor.*

Remark 2. *The equivalence between our distributed and online algorithms is as follows: an early edge in the online algorithm corresponds with an edge in the early subgraph in the distributed algorithm. Similarly, late edges in the online setting correspond to the edges in the late subgraph in the distributed case. This equivalence is discussed in detail in section 4.*

Remark 3. *We can think of a natural extension of the two round distributed algorithm to multiple rounds. If the analysis would proceed as naturally expected from our* 2 *round analysis, then additional rounds will reduce the number of colors used, with a* $\log\log n$ *round algorithm using an essentially optimal* $\Delta + o(\Delta)$ *number of colors. However, there are technical roadblocks which prevent us from going beyond two rounds. For further details refer to Section 5.*

3 Analysis of the Distributed Algorithm

We first make a simple observation based on Chernoff bounds:

PROPOSITION 3.1. *With high probability, every bottom and top node has $\sim r\Delta$ early edges, and $\sim (1-r)\Delta$ late edges.*

Let us fix the set of early edges and late edges, and assume that every vertex has $r\Delta$ early edges and $(1-r)\Delta$ late edges. The analysis below assumes that we have fixed the set of early and late edges (i.e. the two subgraphs).

3.1 Analysis of Early Edges In this subsection, we analyze the performance of the first round of the algorithm (i.e. early subgraph). First, we give two definitions:

DEFINITION 1. *For a bottom vertex b and a phase i ($1 \le i \le L$), let R_b^i be the set of colors which b proposes in round 1 of phase i, but get rejected by the corresponding top vertexes. For a top vertex t and a phase i ($1 \le i \le L$), let E_t^i be the set of those colors in P^i which are not proposed to t in round 1 of phase i, i.e., no round 1 edge (b,t) proposes the color.*

The first round of the algorithm is identical to the algorithm in [9]. The following proposition follows from Chernoff bounds and the method of bounded average differences (the analysis is identical to that in [9] and [5]), and an induction on the phases. Throughout the paper, sharp concentration and with high probability will mean $1/poly(n)$ error.

PROPOSITION 3.2.

1. *For every (top or bottom) node and any phase i, the number of incident first round edges in phase i is sharply concentrated around its mean, which is $\simeq r\Delta/e^{i-1}$.*
2. *For every top node t, $|E_t^i|$ is sharply concentrated around its mean which is $\simeq r\Delta/e^i$.*
3. *For every bottom node b, $|R_b^i|$ is sharply concentrated around its mean, which is $\simeq r\Delta/e^i$.*

For the next proposition (as well as later on in the paper), we will need a small technical lemma proved in [9], which we mention here:

LEMMA 3.1. *If $0 \le p \le 1$, $q = 1-p$, and s is a positive integer, then $\sum_{r=0}^{s} \binom{s}{r} p^r q^{s-r} \frac{1}{r+1} = \frac{1-q^{s+1}}{(s+1)p}$.*

PROPOSITION 3.3. *If b is a bottom vertex, and t is a top vertex with a late edge to b, then for any phase i, belonging to R_b^i is (essentially) positively correlated with belonging to E_t^i, i.e. for any color $c \in P^i$:*

$$Pr[c \in R_b^i | \; c \notin E_t^i] \le (1+o(1))Pr[c \in R_b^i]$$

Proof. The intuition behind the proof is that if $c \notin E_t$, then at least one of the nodes proposing to t in the first round proposes c to t. This increases the chance for any of the nodes proposing to t in the first round to have c as their proposal to t. In turn, this decreases the chance that any of these nodes will propose c on any of their other edges, particularly their edges to the nodes receiving a proposal from b in the first round. Hence, the nodes receiving a proposal from b in the first round have a smaller chance of receiving the same proposal (of color c) from nodes other than b. Thus, b's proposal for color c has a higher chance of getting accepted.

Formally, since we are fixing the early and late subgraphs, assume Z is the set of top nodes with a (phase i) early edge to b, and W is the set of bottom nodes with a (phase i) early edge to t. Note that, by assumption, $b \notin W$ and $t \notin Z$. Also, note that $|Z| \simeq |W| \simeq |P^i| = r\Delta/e^{i-1}$. Let's denote $r\Delta/e^{i-1}$ by d. Ignoring the higher order terms, we have:

$$\begin{aligned} Pr[c \in R_b^i] &= \sum_{z \in Z} Pr[c \in R_b^i | \; b \xrightarrow{c} z] Pr[b \xrightarrow{c} z] \\ &= \frac{1}{d} \sum_{z \in Z} Pr[c \in R_b^i | \; b \xrightarrow{c} z] \end{aligned}$$

where by $b \xrightarrow{c} z$, we mean that b proposes c to z. But, by conditioning on the number of early neighbors of z proposing c to it, we have:

$$\begin{aligned} Pr[c \in R_b^i | \; b \xrightarrow{c} z] &= \sum_{k=0}^{d-1} \binom{d-1}{k} (\frac{1}{d})^k (1-\frac{1}{d})^{d-1-k} \frac{k}{k+1} \\ &= (1-\frac{1}{d})^d \end{aligned}$$

Therefore,

$$Pr[c \in R_b^i] = (1-1/d)^d \tag{3.1}$$

Similarly,

$$\begin{aligned} &Pr[c \in R_b^i | \; c \notin E_t^i] \\ &\quad = \sum_{z \in Z} Pr[c \in R_b^i | \; b \xrightarrow{c} z, \; c \notin E_t^i] Pr[b \xrightarrow{c} z | \; c \notin E_t^i] \\ &\quad = \frac{1}{d} \sum_{z \in Z} Pr[c \in R_b^i | \; b \xrightarrow{c} z, \; c \notin E_t^i] \end{aligned}$$

But, $c \notin E_t^i$ means that a non-empty subset of W, denoted by W_c, proposes c to t. Thus:

$$Pr[c \in R_b^i |\ b \xrightarrow{c} z,\ c \notin E_t^i] =$$

$$\sum_{\varnothing \neq W_c \subseteq W} Pr[c \in R_b^i |\ b \xrightarrow{c} z,\ W_c \xrightarrow{c} t].Pr[W_c \xrightarrow{c} t |\ c \notin E_t^i]$$

But,

$$Pr[c \in R_b^i |\ b \xrightarrow{c} z,\ W_c \xrightarrow{c} t] =$$

$$\sum_{k=0}^{d'-1} \binom{d'-1}{k} (\frac{1}{d'})^k (1 - \frac{1}{d'})^{d'-1-k} \frac{k}{k+1} = (1 - \frac{1}{d'})^{d'}$$

where $d' \leq d$ is the number of phase i early neighbors of z which are not in W_c. Notice that the nodes in W_c can not propose c to z, because they propose c to t.

Therefore, since:

$$(1 - \frac{1}{d'})^{d'} \leq (1 - \frac{1}{d})^d$$

we have:

$$Pr[c \in R_b^i |\ b \xrightarrow{c} z,\ W_c \xrightarrow{c} t] \leq (1 - \frac{1}{d})^d$$

and hence:

$$Pr[c \in R_b^i |\ b \xrightarrow{c} z,\ c \notin E_t^i] \leq (1 - \frac{1}{d})^d$$

which, together with equation 3.1, completes the proof.

The next corollary follows directly from Proposition 3.3

COROLLARY 3.1. *For any late edge (b, t), the mean of $|R_b^{(i)} \cap E_t^{(i)}|$ is (up to lower order terms) at least $r\Delta/e^{i+1}$*

PROPOSITION 3.4. *For any late edge (b, t), $|R_b^{(i)} \cap E_t^{(i)}|$ is sharply concentrated around its mean.*

COROLLARY 3.2. *For any late edge (b, t), $|R_b^{(i)} \cap E_t^{(i)}|$ is with high probability at least $r\Delta/e^{i+1}$ (up to lower order terms).*

This concludes the analysis of the first round. We will next proceed to analyzing the second round of the algorithm, which is our main contribution.

3.2 Analysis of Late Edges At the end of the first round, we have a situation in which every bottom vertex has got some proposed colors (from each of P^i's) accepted and some rejected. From Proposition 3.2, we know that with high probability, $\forall\ b, i :\ |R_b^i| \simeq r\Delta/e^i$ and $\forall\ t, i :\ |E_t^i| \simeq r\Delta/e^i$.

Each vertex has $(1 - r)\Delta = r\Delta/e$ round 2 edges incident on it. Refereing to the description of the algorithm in subsection 2.2, we see that in round 2, each bottom vertex b will *reuse* the colors in R_b^i, as well as colors in augmenting palettes N^i, to propose colors for its round 2 edges. However, the process by which a proposed color is accepted or rejected in round 2 differs from round 1. Suppose that a bottom vertex b proposes a color $c \in R_b^i \cup N^i$ for a round 2 edge (b, t). If $c \notin E_t^i \cup N^i$, then this means that c was already used to color some round 1 edge (b', t), where $b' \neq b$. So, this proposal is rejected outright. If $c \in E_t^i \cup N^i$, then whether the proposal for (b, t) is accepted or not depends on the number of other round 2 edges (b', t) also proposing c (since t accepts only one of them uniformly at random).

We are interested in knowing the outcome of the round 2 process, i.e., how many colors get accepted at a top vertex and at a bottom vertex. But one can see from the above discussion that the outcome depends crucially on how the sets R_b^i and E_t^i intersect. We abstract this process out in the following section, as a balls-in-bins process with restricted choice-sets for the players and restricted set of usable bins. This balls and bins process also is of independent interest.

3.3 A Restricted-balls-in-bins process Consider the balls-in-bins process defined in Figure 2.

REMARK 4. *It should be clear that the above balls-in-bins process captures the process in the distributed algorithm in round 2 of any phase j, as seen from the point of view of one top vertex t and its round 2 neighboring bottom vertices, $b_1, b_2, ..., b_{r\Delta/e^j}$. The set C of bins is the entire palette of $r\Delta/e^{j-1}$ colors for the relevant phase, the set E is the set E_t^j, and the R_i are the sets $R_{b_i}^j$.*

We wish to find the number of empty bins at the end of the process and the probability that a particular ball was accepted. Clearly, the answers depend on the intersection patterns of the sets E and R_i ($i \in [m]$). For instance:

- If $R_i \cap E = \emptyset\ \forall\ i$, then clearly every ball not in N is rejected outright and every bin in E remains empty.

- There is a set C of k bins, and a set N of x bins. There is a subset $E \subseteq C$ of βk bins. At the beginning of the process, the bins in E and N are empty, and the bins in $C \backslash E$ are already full.

- There are $m = |E|+|N| = \beta k + x$ balls. For each ball i there is a subset $R_i \subseteq C$, with $|R_i| = |E|$. Ball i is thrown into a bin chosen uniformly at random from $R_i \cup N$.

- If a ball is thrown into a bin in $C \backslash E$ (an already full bin), then it is rejected outright. For each bin in $E \cup N$ which receives at least one ball, exactly one of the balls is accepted, uniformly at random, and the rest rejected, and the bin is called full. A bin in $E \cup N$ which doesn't receive any balls remains empty.

Figure 2: A restricted balls-in-bins problem.

- If $R_i = E \ \forall\ i$, then the process is identical to an unrestricted balls-in-bins process over the m bins in $E \cup N$ and the m balls. Thus we know that $\sim m/e$ bins will remain empty, and by symmetry, each ball will have a probability of $\sim 1/e$ to be rejected.

- An asymmetric example: Consider the case when some small subset P of the balls have R_i with a large intersection with E and the remaining majority of the balls have R_i with a small intersection with E. Then the balls in P will have a high probability of being accepted, while other balls have a small probability of being accepted. Also, a large fraction of the bins in E will remain empty.

The examples above show that the outcome of the process depends on two things: (1) How do the different R_i intersect with E, and (2) How do the different R_i intersect with each other. In the light of Proposition 3.3, we will make an assumption on (1), that for each i, R_i and E are positively correlated, that is, choosing a member c of C uniformly at random, $Pr[c \in R_i,\ c \in E] \geq Pr[c \in R_i]Pr[c \in E]$. Clearly, this is equivalent to the constraint $|R_i \cap E| \geq \beta^2 k \quad (\forall\ 1 \leq i \leq m)$. Note that we make no assumption whatsoever on (2), i.e., on the intersection between two sets R_i and R_j. Once we have positive correlation between the R_i and E, we can prove a strong bound on the outcome, as described in the next section.

Under this assumption, we bound the expected number of empty bins at the end of the process as well as the probability that a given ball is rejected. We can also prove a sharp concentration result for the number of empty bins, but we will prove the sharp concentration results directly for the distributed algorithm in subsection 2.2.

Proposition 3.5. *If for each i, R_i and E are positively correlated then the probability of any specific ball to get accepted is minimized when all the R_i's have the same intersection with E, of size $\beta|E| = \beta^2 k$. This minimum probability of acceptance is $\frac{x+\beta|E|}{x+|E|}(1-1/e)$.*

Proof. Let acc_i denote the event $\{$*ball i gets accepted*$\}$ and let F_{ic} denote the event $\{$*ball i is thrown in bin c*$\}$. We wish to find a lower bound on $Pr[acc_i]$ for each i. For convenience, let $A_i = R_i \cup N$ and $E' = E \cup N$. For each color $c \in A_i$ (for an arbitrary, but fixed i), define $q_c = |\{1 \leq j \leq m|\ c \in A_j\}|$. We have:

$$Pr[acc_i] = \sum_{c \in A_i} Pr[F_{ic},\ acc_i] \tag{3.2}$$

and

$$Pr[F_{ic},\ acc_i] = Pr[F_{ic}]Pr[acc_i|\ F_{ic}] = Pr[acc_i|\ F_{ic}]/m$$

But, if $c \notin E'$, $Pr[acc_i|\ F_{ic}] = 0$, and if $c \in E'$, then:

$$Pr[acc_i|\ F_{ic}] = \sum_{l=0}^{q_c-1} \binom{q_c-1}{l} (\frac{1}{m})^l (1-\frac{1}{m})^{q_c-1-l} \frac{1}{l+1} = \frac{1-(1-\frac{1}{m})^{q_c}}{q_c/m}$$

where the last equality is by using Lemma 3.1. Thus

$$Pr[F_{ic}, acc_i] = \begin{cases} 0 & \text{if } c \notin E', \\ \frac{1-(1-\frac{1}{m})^{q_c}}{q_c} & \text{if } c \in E'. \end{cases}$$

and hence, from (3.2)

$$Pr[acc_i] = \sum_{c \in A_i \cap E'} \frac{1-(1-\frac{1}{m})^{q_c}}{q_c}$$

Now, if $c \in N$, then $q_c = m$. But also note that $f(x) = (1-(1-1/m)^x)/x$ is a decreasing function over $x \in [m]$. Thus, the probability is minimized when all $q_c = m$:

$$Pr[acc_i] \geq \sum_{c \in R_i \cap E} \frac{1-(1-\frac{1}{m})^m}{m} + \sum_{c \in N} \frac{1-(1-\frac{1}{m})^m}{m} = |R_i \cap E|.\frac{1-(1-\frac{1}{m})^m}{m} + |N|\frac{1-(1-\frac{1}{m})^m}{m}$$

$$\geq (1-1/e)\frac{\beta|E|+x}{|E|+x} \tag{3.3}$$

where the last inequality is using the positive correlation assumption. Notice that in order for the above inequalities to be tight, we need $|R_i \cap E| = \beta|E|$, and also $q_c = m \forall c \in R_i \cap E$, i.e. $|R_i \cap E| \subseteq |R_j \cap E| \quad \forall 1 \leq j \leq m$. Since i is an arbitrary number in $[m]$, the proposition is proved.

PROPOSITION 3.6. *The maximum number of empty bins (in expectation) occurs when all R_i's have the same intersection with E, of size $\beta|E| = \beta^2 k$. This number is $(1 - \beta + \beta/e)|E| + x/e$*

Proof. If X is the number of non-empty bins in $E \cup N$ at the end of the process, and $X_i = 1_{acc_i}$ (with acc_i being the event that ball i is accepted, as before), then $X = \sum_{i=1}^{m} X_i$, and hence $E[X] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} Pr[acc_i] \geq (1 - 1/e)x + \beta(1 - 1/e)|E|$. Thus $E[number\ of\ empty\ bins] \leq (1 - \beta + \beta/e)|E| + x/e$. Also, equality happens in exactly the same case as in the previous proposition.

3.4 Analysis of Late Edges Continued So far, we have not yet determined the values of x_i's. But, now we can do that. x_i is basically the number of new colors that we need in the second round of phase i, to be sure that we have enough colors to make proposals for the second round edges in that phase. In other words, defining $deg_i^U := x_i + |R_b^i|$, we desire that $deg_i^U \geq deg_2^i$ for all i (where deg_2^i is the degree of an arbitrary node in the i^{th} phase of second round). We start with $deg_2^1 = (1 - r)\Delta = r\Delta/e$ round 2 edges in the first phase, and we know $|R_b^1| \simeq r\Delta/e$ whp. Thus, we do not need any new colors in that phase, i.e. $x_1 = 0$, and $deg_1^U = r\Delta/e$. We know from the above process that:

$$\frac{deg_2^{i+1}}{deg_2^i} \leq q_i = \frac{x_i/e + (1 - 1/e + 1/e^2)|E_t^i|}{x_i + |E_t^i|}$$

$$= \frac{deg_i^U/e + (1 - 1/e)^2|E_t^i|}{deg_i^U}$$

Therefore,

$$deg_2^{(i+1)} \leq \frac{deg_i^U/e + (1 - 1/e)^2|E_t^i|}{deg_i^U}.deg_2^i$$

$$\leq \frac{1}{e}deg_i^U + (1 - \frac{1}{e})^2|E_t^i|$$

Therefore, from this equation, we note that defining:

$$deg_{i+1}^U = \frac{1}{e}deg_i^U + (1 - \frac{1}{e})^2|E_t^i|$$

where $|E_t^i| = r\Delta/e^i$, we always have $deg_2^i \leq deg_i^U$. So, we can solve this recursion (with $deg_1^U = r\Delta/e$) and then define x_i as: $x_i = deg_i^U - |E_t^i|$. This will determine the size of the augmenting palette for each phase.

Finally, we give with high probability (whp) bounds for the number of colors accepted at each bottom and top node after round 2 of the distributed edge coloring algorithm.

PROPOSITION 3.7. *The number of colors rejected at a top vertex in the second round of phase i is whp smaller than deg_{i+1}^U.*

Proof. The proof is by induction on i. For better readability, we will only give the idea and the sketch of the proof, and omit the calculations. The idea is that, considering a top vertex t, we first prove, using a simple application of the method of bounded differences (MOBD)[5], that conditioned on the outcome of the algorithm up to the second round of phase i (i.e. particularly on m, the cardinality of the set $\{b_1, \ldots, b_m\}$of bottom nodes proposing to t in the second round of phase i, as well as on sets $R_{b_j}^i$ ($\forall 1 \leq j \leq m$) and E_t^i, the number of distinct colors rejected at t in the second round of phase i is sharply concentrated around its (conditional) expectation. That is, if X is the number of distinct colors rejected at t in the second round, and F_1 is the outcome of the algorithm up to this point upon which we are conditioning, then we have $Pr[|X - E[X|\ F_1]| \geq \lambda|\ F_1] \leq 2exp(-\lambda^2/m)$.

But, then we recall the inductive hypothesis on the whp results about F_1: $m \leq deg_i^U$, $R_{b_j}^i \simeq r\Delta/e^j$, $E_t^i \simeq r\Delta/e^i$, and also $|R_{b_j}^i \cap E_t^i| \gtrsim r\Delta/e^{i+1}$. (notice that the basis of the induction corresponds to the whp results for the algorithm's outcome on the first round subgraph, which is true). So, assuming that F_1 is one of the outcomes that occur whp, we can use the exact same balls-in-bins process as in section 3.3 to prove that $E[X|\ R_1] \leq deg_{i+1}^U$. Therefore, we know that for each of the high probability outcomes of F_1, the number of colors rejected at t in the second round of phase i is sharply concentrated around a number smaller than deg_{i+1}^U (notice that the λ in the sharp concentration inequality above can indeed be as large as $\Theta(\sqrt{\Delta log\, n})$ which is what we need). Therefore, almost always (i.e. whp) X is smaller than deg_{i+1}^U, which concludes the proof.

Similarly, we can prove:

PROPOSITION 3.8. *The number of proposals rejected from a bottom vertex in the second round is whp smaller than deg_{i+1}^U.*

3.5 Final Analysis From proposition 3.2, we know whp in each phase the round 1 edges at each node drop at a rate of $1/e$. Also, whp in phase i the second round edges at each node drop with a rate of at least q_i. Notice that each whp result can go wrong with a probability $1/poly(n)$. But, we have in total $O(log\ n)$ such results (one for each phase of each round). So, by a simple union bound, we know that whp all of those results hold, in which case the total number of colors that we will need is at most:

$$\sum_{i\geq 1}|P^i| + \sum_{i\geq 1} x_i + O(log\ n)$$

But, $x_i = deg_i^U - |E_l^i|$ and $|P^{i+1}| = |E_l^i| = r\Delta/e^i$. Thus, the above summation is equal to:

$$r\Delta + \sum_{i\geq 1} deg_i^U + O(log\ n)$$

But, $deg_{i+1}^U = deg_i^U/e + (1-1/e)^2 r\Delta/e^i$. Summing up all these equalities (for all $i \geq 1$), and defining $S = \sum_{i\geq 1} deg_i^U$, we have:

$$S - deg_1^U = S/e + (1-1/e)^2 \sum_{i\geq 1} r\Delta/e^i$$

from which we can calculate $S = r\Delta/e + \Delta e/(e^2-1) = 0.425\Delta$. Thus, the total number of colors used is: $1.425\Delta + O(log\ n) \leq 1.43\Delta + o(\Delta)$. We record this result in the following theorem:

THEOREM 3.1. *The two-round algorithm colors the edges of a graph with maximum degree $\Delta = \omega(log\ n)$, whp using at most $1.43\Delta + o(\Delta)$ colors.*

4 Analysis of the Online Algorithm

In this section, we show how to adapt the analysis of the distributed algorithm to the online algorithm. First, we point out the equivalence of our distributed and online algorithms. The Early edges in the online algorithm correspond to the edges in the Early subgraph of the distributed algorithm, and the Late edges in the online algorithm correspond to the edges in the Late subgraph of the distributed algorithm.

Also, colors get proposed for edges identically in both algorithms – chosen at random, without replacement from the same set of colors: the palette $P^{(i)}$ in round 1 and the rejected colors plus the augmenting palette in round 2 (for the distributed algorithm), and from colors with *touched* = 0 for early arrivals or rejected colors and augmenting palette for late arrivals (for the online algorithm).

Finally, the decision on whether a proposed color is accepted or rejected is also taken identically in the two algorithms: In the distributed algorithm, top nodes pick one edge out of every color class (in each round). In the online algorithm, an edge's proposal is accepted iff it is the first arriving edge incident on the top node which proposed that color. These two decisions can be seen as having identical distributions, due to the random order in which the edges arrive. This is because conditioning on the set $\{b_1, \ldots, b_m\}$ proposing to a top node t in some point during the run of algorithm and the subset S of b_i's whose proposal to t gets accepted at that point, we get no information about the relative ordering of the nodes which go to the next phase, namely $\{b_i\}_{i=1}^m - S$ (the only information that we get out of knowing S is that the each of the edges going to the next phase arrives after a corresponding edge in S).

Also, note that in the distributed algorithm, the top nodes use fresh randomness to make accept and reject decisions in the different rounds and different phases. In the online algorithm, on the other hand, the random permutations seen by the different palettes are highly dependent – indeed, they all come from the same global permutation of edge arrivals. But, notice that in the analysis of the distributed algorithm, we only used a union bound over different palettes' error probabilities to get the final whp result. Thus, we don't care about the dependencies between different phases.

5 Open Problem: Extending the algorithms to multiple rounds

In our main sections we have described and analyzed an algorithm which works in two rounds per phase (distributed), and equivalently, an online algorithm in which each color can be tried two times by a bottom vertex. We now describe an extension to R rounds, and equivalently, R trials per color per bottom vertex.

Each bottom node randomly partitions its incident edges into R subsets. For each phase, we have an initial palette, for the first round. The palettes of later rounds are composed of an augmenting set of new colors plus whatever colors which got rejected from the bottom node in any of the previous rounds. The balls and bins process gives the damping factor for each round as well as the number of new colors necessary for each round. This will give us the total number of colors that we need. We conjecture that with arbitrarily large constant number of rounds one can get $(1+\epsilon)\Delta$ colorings, and that $O(loglog\ n)$ rounds gives $(1+o(1))\Delta$ colorings.

References

[1] Gagan Aggarwal, Rajeev Motwani, Devavrat Shah, An Zhu: Switch Scheduling via Randomized Edge Coloring. FOCS 2003

[2] Amotz Bar-Noy, Rajeev Motwani, Joseph (Seffi) Naor: The Greedy Algorithm is Optimal for On-Line Edge Coloring. Information Processing Letters, 44 (1992), pages 251-253.

[3] Richard Cole, Kirstin Ost, and Stefan Schirra: Edge-Coloring Bipartite Multigraphs in $O(ElogD)$ Time. Combinatorica 21(1):5–12, 2001.

[4] Devdatt Dubhashi, David A. Grable, Alessandro Panconesi: Near- Optimal, Distributed Edge Coloring via the Nibble Method. Proceedings of ESA 1995.

[5] Davdatt P. Dubhashi, Alessandro Panconesi: Concentration of Measure for the Analysis of Randomized Algorithms, 2005. Available at: http://www.dsi.uniroma1.it/˜ale/Papers/master.pdf

[6] Ian Holyer: The NP-Completeness of Edge-Colouring. SIAM J. Comput. 10, 718-720, 1981.

[7] David A. Grable, Alessandro Panconesi: Nearly optimal distributed edge coloring in $O(loglogn)$ rounds. Random Struct. Algorithms 10(3): 385-405 (1997)

[8] Rajeev Motwani and Prabhakar Raghavan: Randomized Algorithms, Cambridge University Press, 1995.

[9] Alessandro Panconesi, Aravind Srinivasan: Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding Bounds. SIAM J. Comput. 26(2): 350-368 (1997)

[10] Vadim G. Vizing: On an Estimate of the Chromatic Class of a p-graph. Metody Diskret. Analiz. 3 (1964), pages 25-30.