Switch Scheduling via Randomized Edge Coloring *

Gagan Aggarwal[†] Rajeev Motwani[‡] Devavrat Shah[§] An Zhu[¶]

Abstract

The essence of an Internet router is an $n \times n$ switch which routes packets from input to output ports. Such a switch can be viewed as a bipartite graph with the input and output ports as the two vertex sets. Packets arriving at input port i and destined for output port j can be modeled as an edge from i to j. Current switch scheduling algorithms view the routing of packets at each time step as a selection of a bipartite matching. We take the view that the switch scheduling problem across a sequence *of time-steps is an instance of the edge coloring problem for a bipartite multigraph. Implementation considerations lead us to seek edge coloring algorithms for bipartite multigraphs that are* fast, decentralized, *and* online. *We present a randomized algorithm which has the desired properties, and uses only a near-optimal $\Delta + o(\Delta)$ colors on* dense *bipartite graphs arising in the context of switch scheduling. This algorithm extends to non-bipartite graphs as well. It leads to a novel switch scheduling algorithm which, for stochastic online edge arrivals, is* stable*, i.e., the queue length at each input port is bounded at all times. We note that this is the first* decentralized *switch scheduling algorithm that is also guaranteed to be stable.*

1 Introduction

We present the first known switch scheduling algorithm which is both stable and implementable, in the sense that our solution is *fast*, *decentralized* and uses minimal *inter-chip communication*. The key ingredient of our scheduler is a simple, randomized algorithm for near-optimal *edge coloring* of a *dense* multigraph. Before we can describe our results in greater detail, it will be necessary to provide some background.

Switch Scheduling. In the recent past, the input-queued (IQ) switch architecture [3] has become dominant in high-speed network switching. Figure 1 shows the logical structure of an $n \times n$ IQ switch with n input and n output ports. The main function of the switch is to route packets arriving at the input ports to the appropriate output ports. Packets arriving at input i and destined for output j are buffered in a *virtual output queue* denoted by VOQ_{ij}. Let $A_{ij}(t) \in \{0,1\}$ be a random variable indicating the arrival of a new packet in VOQ_{ij} at time t. We assume that $A_{ij}(t)$'s are i.i.d. Bernoulli[1] with parameter $\lambda_{ij} = \Pr[A_{ij}(t) = 1]$. The line-rates at all the ports are normalized to 1, i.e., each input can receive at most one packet per time slot and an output can transmit at most one packet per time slot. The incoming traffic is called *admissible* if $\forall j,\ \sum_{i=1}^{n} \lambda_{ij} < 1$ and $\forall i,\ \sum_{j=1}^{n} \lambda_{ij} < 1$. Let $\lambda^* = \max_{i,j}\{\sum_k \lambda_{ik}, \sum_l \lambda_{lj}\} < 1$.

In an input-queued switch, packets are switched from input to output ports via a crossbar fabric, which imposes the constraint that in any time slot, at most one packet may be removed from each input's VOQs and at most one packet may be sent to each output. Thus a *scheduling algorithm* is needed to find a schedule (a matching between input and output ports) to transfer packets at each time step. A good scheduling algorithm should have the following two properties:
(i) **Stability:** A scheduling algorithm is said to be *stable* if the length of each queue VOQ_{ij} stays bounded under admissible network traffic.
(ii) **Implementability:** Current hardware technology imposes some constraints on the scheduling algorithm. First, for high-speed switches (running at Gigabit/second speeds) the inter-arrival time of packets is of the order of nanoseconds. This implies that an algorithm cannot take more than a few on-chip operation-cycles per time step to come up with a transfer schedule, i.e., the algorithm should be simple and not involve complex computations. Second, due to

* All authors are affiliated with the Department of Computer Science at Stanford University.

† Supported in part by a Stanford Graduate Fellowship and NSF Grant EIA-0137761. E-mail `gagan@cs.stanford.edu`.

‡ Supported in part by NSF Grant IIS-0118173 and EIA-0137761, an SNRC Grant, and grants from Microsoft and Veritas. E-mail `rajeev@cs.stanford.edu`.

§ Supported in part by a grant from the Stanford Networking Research Center, NSF grant ANI-9985446 and AFOSR grant F49620-01-1-0365. E-mail: `devavrat@cs.stanford.edu`.

¶ Supported in part by a GRPW fellowship from Bell Labs, Lucent Technologies, and NSF Grant EIA-0137761. E-mail `anzhu@cs.stanford.edu`.

[1] All results in this paper can be easily generalized to any regenerative traffic model. We omit the details.

cost and hardware constraints, it is not possible to maintain the state of switch (e.g., queue-lengths) on chip along with the scheduling algorithm's logic module. This implies that information about the switch state needs to be communicated to the scheduling module from other chips. Again, cost considerations and time constraints imply that only a few bits of information per queue can be communicated to the scheduling algorithm. Essentially, this leads to a need for a decentralized algorithm that operates in an online fashion.

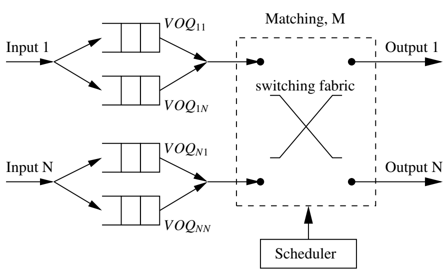

Figure 1. Logical structure of an input-queued packet switch

Previous Work on Switch Scheduling. McKeown, Anantharan, and Walrand [10] gave a *stable* switch scheduling algorithm which involves finding a *maximum weight matching* with respect to an appropriate weighted bipartite graph at each time step. This algorithm is impractical in that it requires $\Omega(n^3)$ work per time step and, in any case, implementing such a complex algorithm on chip is infeasible. Subsequently, several other stable algorithms, such as those of Tassiulas [15] and Giaccone, Prabhakar, and Shah [7], have been proposed. While simpler than maximum weight matching, they still require centralized control. In commercial switches, only simple scheduling algorithms such as iSLIP [11] are implemented. In the implemented version of iSLIP, only a constant number of cycles are used to produce a scheduling decision. Essentially, iSLIP computes a *maximal* matching at each step using a decentralized approach. Unfortunately iSLIP and its variants are *unstable*. It has been an open question to find an algorithm which is both stable and implementable.

Relating Edge Coloring and Switch Scheduling. Let $G = (U, V, E)$ be a bipartite multigraph, with $|U| = |V| = n$. We will refer to the n nodes in U as *inputs*, and the n nodes in V as *outputs*. Let Δ be the maximum degree in G, and define Δ_{ij} as the number of edges between input i and output j, for $1 \leq i, j \leq n$; since we allow multigraphs, Δ_{ij} may exceed 1. An edge coloring is an assignment of colors to the edges in E such that adjacent edges receive distinct colors. The minimum number of colors in any valid edge coloring, $\chi'(G)$, is called the *chromatic index* of the graph.

It is well-known that bipartite edge coloring has a direct application to switch scheduling, and to other similar scheduling problems arising in applications such as radio-hop networks [17] and optical networks [16]. Consider the bipartite multigraph between input and output ports, where each edge (i, j) denotes a distinct packet from VOQ_{ij}; note that the *multiplicity* of an edge is the number of packets in the queue VOQ_{ij}. A valid switch schedule at each time step connects each input i to at most one output j, such that no two inputs are connected the same output. Thus, the packets transferred in any one time step have to form a matching. Recall that the edges in any color class of an edge coloring form a matching. Thus, a switch scheduler can be viewed as performing an edge coloring of the multigraph – the color of an edge is the time slot at which the corresponding packet is transferred. The number of colors in the edge coloring is equivalent to the number of time slots taken to schedule the packet load by a corresponding switch schedule.

Our algorithm performs online edge coloring. Each edge is assigned a color as soon as it arrives; i.e., the color of an edge does not depend on future edge arrivals. This minimizes the amount of state maintained, and reduces the delay at the switch. However, the same color may be assigned to edges arriving at different time steps. Thus, we have to wait till all edges assigned a single color have arrived before transferring them together in a single time step; in other words, we perform batch-scheduling. We assign colors to packets *as they arrive* over a span of $B = O(n^2)$ time steps (which we will call a batch), while storing them in virtual output queues. Our edge coloring algorithm uses C_B colors. Then, for the next C_B time slots (which is also the duration of the next batch), the packets from the previous batch are scheduled according to their assigned colors (with an appropriate mapping of colors to time slots), while simultaneously coloring and storing the packets arriving in the present batch. Thus, on average, a packet gets delayed by the duration of a batch. Instead of coloring packets *as they arrive*, we can also use an offline edge coloring algorithm to color the packets after accumulating all the packets in a batch. Assuming computing speed barely keeps up with the arrival rate of packets, the delay involved in this case would be more than twice the duration of a single batch. Let G be the graph obtained by accumulating all the packets which arrive during a batch. We will show that this switch scheduling is stable iff the edge coloring algorithm used as a subroutine colors a bipartite graph of maximum degree Δ using at most $\Delta + o(\Delta)$ colors. Thereafter, we will present and analyze an edge coloring algorithm which achieves this goal.

Previous Work on Edge Coloring. For the case of general graphs, determining $\chi'(G)$ is NP-complete [6]. Vizing's Theorem [18] states that any graph with maximum degree Δ can be edge colored with at most $\Delta + 1$ colors and the result is constructive. It is obvious that at least Δ colors are needed in all cases. The algorithm due to Vizing is complex and not suitable for switch scheduling. For *bipartite* multigraphs, there exist edge coloring algorithms using Δ colors in $O(|E|\log\Delta) = O(n\Delta\log\Delta)$ time [4]. All such algorithms are inherently centralized and take $\omega(1)$ time per packet, rendering them ineffective for switch scheduling. Panconesi and Srinivasan [13] presented an offline but distributed algorithm which edge colors a graph using 1.58Δ colors. While this algorithm can be adapted to online edge coloring, the number of colors used is too high to yield a stable scheduler, which must use at most $\Delta + o(\Delta)$ colors.

Grable and Panconesi [8] proposed an edge coloring algorithm using $(1+\varepsilon)\Delta$ colors, for any constant $\varepsilon > 0$, but operating in $O(\log\log n)$ rounds, provided Δ is smaller than any positive power of n. When the degree of the graph cannot be bounded from above, they obtain similar guarantees with an algorithm running in $O(\log\Delta\log\log n)$ rounds. Dubhashi, Grable and Panconesi [9] proposed an alternate algorithm that uses $\Delta + \Delta/\log^s n$ colors, whenever $\Delta = \Omega(\log^k n)$, for constants $s,k > 0$, but operating in $O(\log n)$ rounds. Both algorithms operate in $\omega(1)$ rounds, and hence require $\omega(1)$ time per packet, which makes them virtually impossible to implement in a network switch. Further, the guarantee provided by the first algorithm of using at most $(1+\varepsilon)\Delta$ colors is too weak to guarantee stability for all admissible traffic.

Thus, none of the existing edge coloring algorithms can color a graph with $\Delta + o(\Delta)$ colors, using $O(1)$ operations per edge, which is necessary to achieve a fast, stable, and implementable switch scheduler.

Main Results and Roadmap. We present a fast, decentralized, randomized edge coloring algorithm for bipartite multigraphs, based on which we obtain a new stable switch scheduling algorithm. Our analysis establishes that the edge coloring algorithm uses at most $\Delta + o(\Delta)$ colors with high probability, provided the input graph is *dense*, i.e., it has $\Delta = \omega(n^2)$. While this may seem like a strong condition, note that it is a reasonable model for the heavy load generated in Internet routers and may indeed be the only interesting case in that scenario. The condition $\Delta = \omega(n^2)$ implies that we do batch-scheduling for batches of size $O(n^2)$, which imposes an average delay of $O(n^2)$ per packet. We would like to note that the best average delay bounds known for switch scheduling algorithm for an input-queued switch is $\Theta(n^2)$ [14]. Thus, in the application to switch scheduling, the requirement that $\Delta = \omega(n^2)$ is not restrictive. Further, since our algorithm is of independent interest for edge coloring, we relax this condition as stated in Theorem 4 and discussed in Section 6.

Our algorithm is online in the context of switch scheduling — instead of assuming an adversarial arrival order, the edge arrival order is assumed to obey certain stochastic models. We note that our algorithm can be applied to adversarial arrival patterns as well; packets in a batch are accumulated and the edge order is randomized before applying our algorithm. Our randomized algorithm has the additional feature of being decentralized – the choice of the color for an edge $e = (i,j)$ only involves examining the colors given to edges incident to nodes i and j only, which in turn implies a running time of $O(1)$ per vertex per time step.

The rest of the paper is organized as follows. In Section 2, we describe the switch scheduling and edge coloring algorithms, and state their performance guarantees which will be derived in the rest of the paper. Section 3 provides details of the proof of stability of the switch scheduling algorithm obtained by using our edge coloring algorithm as a subroutine. Section 4 outlines the analysis of the edge coloring algorithm, with some of the more intricate proofs being deferred to the appendix. Then we show that the edge coloring algorithm generalizes to the case of non-bipartite graphs as well; we sketch the analysis in Section 5. In Section 6, we show that constraints on the degree of the graph can be relaxed while maintaining the performance guarantee of the edge-coloring algorithm. Finally, we conclude with some general remarks in Section 7.

2 The Switch Scheduling Algorithm

We first describe a switch scheduling algorithm, called SWITCH-SCHED, which uses an online edge coloring algorithm, called EDGE-COLOR, to obtain a schedule of packet transfers. This algorithm performs *batch-scheduling* of packets. Let t_k denote the time at which the k^{th} batch ends. The packets arriving at any time t, for $t_{k-1} < t \le t_k$, are colored by the online edge coloring subroutine EDGE-COLOR. By a suitable mapping of colors to time slots, the colors assigned to the packets define a transfer schedule and the packets are transferred according to this schedule during the interval $(t_k, t_{k+1}]$, while the packets in the $(k+1)^{st}$ batch are accumulated.

Algorithm SWITCH-SCHED(EDGE-COLOR)

1. Let $t_0 = 0$ and $t_1 = n^2$, where n is the number of input ports in the switch. Initialize the current batch number $k = 1$, and current time $t = 1$.
2. Repeat forever:
   (a) While the current time $t \le t_k$ (the current batch has not ended), do the following:
      (i) If in the current time step t, a packet arrives at input i destined for output j, add an edge (i,j) in the bipartite graph G_k, and color this edge using the online edge coloring algorithm EDGE-COLOR.

(ii) Transfer all packets corresponding to edges colored t in the graph G_{k-1}. (Note that these edges will form a matching and thus result in a valid schedule.)

(iii) Go to the next time step, $t \leftarrow t+1$.

(b) Let $\Delta(G_k)$ be the maximum degree of graph G_k formed by the packets arriving in the k^{th} batch, and let $\kappa(G_k)$ be the number of colors used by EDGE-COLOR to color G_k. Define $t_{k+1} = t_k + \kappa(G_k)$, and do a one-to-one mapping of the colors of G_k to time slots in $(t_k, t_{k+1}]$.

As noted before, an offline edge coloring algorithm can also be used, in which case the edge coloring would be performed at the end of each batch, once G_k is known fully. This would cost an additional latency equal to the running time of the offline edge coloring algorithm.

Assuming that we have a suitable edge coloring algorithm, the following theorem guarantees the stability of Algorithm SWITCH-SCHED. Note that $\Delta(G_k)$, the maximum number of packets accumulated in one batch at any input port, represents the maximum queue size.

Theorem 1. *Suppose that Algorithm* EDGE-COLOR *can edge-color a bipartite multigraph with maximum degree* Δ *using at most* $\Delta + o(\Delta)$ *colors with high probability. Then, Algorithm* SWITCH-SCHED(EDGE-COLOR) *is stable, i.e., under Bernoulli i.i.d. arrival process with* $\lambda^* < 1$*, we have* $\Delta(G_k) < \infty$ *with a high probability for all* $k \in \mathbb{N}$*.*

The conclusion of Theorem 1, that queue-sizes are finite with a high probability, remains true under a large class of arrival processes; for example, it is true for any regenerative arrival process. We defer the proof of Theorem 1 to Section 3.

It remains to specify the online edge coloring algorithm, EDGE-COLOR. Consider first a GREEDY algorithm for online edge coloring of a bipartite multigraph G. Assume, without loss of generality, that G is Δ-regular; otherwise, add dummy edges to make the graph regular (see also Cole et al. [4]). We start with an initial pool of colors $\{1,\ldots,A\}$. As the edges arrive in some arbitrary order, we color each edge with the smallest *free* color, where a free color is one which has not been used at any adjacent edge. Clearly, GREEDY succeeds if and only if $A = 2\Delta - 1$. In fact, Bar-Noy, Motwani, and Naor [2] show that no online algorithm (deterministic or randomized) can do better for sparse graphs. Our goal is to reduce the number of colors used to $\Delta + o(\Delta)$ so as to guarantee stability. We next describe a natural randomized variant of this algorithm.

Algorithm RAND-COLOR

1. Let $1,\ldots,\Delta$ be the initial pool of available colors.
2. At each input, independently select a random permutation of the Δ incident edges.
3. **For** $s = 1$ to Δ **do**
   For each input i:
   (a) Pick the s^{th} edge e in the random permutation associated with input i.
   (b) Select a color for e uniformly at random from its current set of *free* colors.
   (c) If no *free* color is available for e, leave the edge uncolored. This edge will be colored in Step 4.
4. Let r be the maximum number of uncolored edges at any vertex. Color the remaining edges using the GREEDY algorithm with $2r-1$ fresh colors.

Under adversarial arrival patterns, the randomization done in Step 2 is necessary, as discussed in Appendix A. Since packet arrivals at a switch follow Bernoulli i.i.d. processes, when RAND-COLOR is used as a subroutine for SWITCH-SCHED, we do not need to perform Step 2. We can use the actual arrival order itself, since that order is itself random enough for the performance guarantees of RAND-COLOR to hold. Furthermore, we can perform the GREEDY coloring performed in Step 4 in Step 3(c) itself by assigning the edge e the minimum *available* color from a fresh set of colors, say $1', 2' \ldots, i', \ldots$. Thus, algorithm RAND-COLOR can operate online as a subroutine within algorithm SWITCH-SCHED, and can color edges as they arrive.

Theorem 2 characterizes the performance of RAND-COLOR; its proof is sketched in Section 4. Note that throughout this paper, "high probability" is used to denote probability at least $1 - o(\frac{1}{n^2\Delta^2})$, unless specified otherwise.

Theorem 2. *For a bipartite multigraph G with n vertices and maximum degree* $\Delta = \omega(n^2)$*, Algorithm* RAND-COLOR *uses at most* $\Delta + o(\Delta)$ *colors with a high probability.*

For the case of general (non-bipartite) graphs, a similar result holds. The straightforward extension of RAND-COLOR to general graphs, denoted RAND-COLOR-G, is as follows: Start with Δ colors for each vertex. At each round, each vertex (instead of just the *input* vertices) picks an edge at random from the remaining edges, and colors it with a random free color. This process continues until all the edges have been considered. The edges that are not colored at the end are colored using the simple GREEDY algorithm. The proof technique for Theorem 3 below is similar to that for Theorem 2, as described in Section 5.

Theorem 3. *For a general multigraph G with n vertices and maximum degree* $\Delta = \omega(n^2)$*, Algorithm* RAND-COLOR-G *uses at most* $\Delta + o(\Delta)$ *colors with a high probability.*

In contrast to the condition $\Delta = \omega(n^2)$, consider the following two conditions:

$$\Delta_{ij} = \Omega(\sqrt{\Delta}), \quad \forall\, i,j \text{ s.t. } \Delta_{ij} \neq 0 \qquad (1)$$
$$\Delta = \omega(\log^4 n) \qquad (2)$$

We state the following theorem and discuss it briefly in Section 6.

Theorem 4. *The algorithm* RAND-COLOR *(resp.* RAND-COLOR-G*) edge-colors a bipartite graph G (resp. general graph G) satisfying conditions (1) and (2) with* $\Delta + o(\Delta)$ *colors with a high probability.*

3 Proof of Theorem 1: Analysis of SWITCH-SCHED

Recall that G_k is the bipartite multigraph formed in k^{th} batch, $\Delta(G_k)$ is the max-degree of G_k and $\kappa(G_k)$ is the number of colors used by Algorithm EDGE-COLOR to color G_k.

Note that $\Delta(G_k)$ forms an irreducible Markov chain (on space $\mathbb{N}$). We will show that for some $\varepsilon > 0$ and constant C,

$$E[\Delta(G_{k+1}) - \Delta(G_k) | \Delta(G_k) > C] \leq -\varepsilon\Delta(G_k) \qquad (3)$$

Then, from (3) above, by applying Foster's criteria [1], we will obtain that with high probability,

$$E[\Delta(G_k)] < \infty \Rightarrow \Delta(G_k) < \infty,$$

thus proving Theorem 1.

We will prove (3) now. Let $0 < \delta < (1 - \lambda^*)$, and let ε be such that $\lambda^*(1+\delta)^2 \leq (1-\varepsilon)$. Then,

1. From the assumption on the performance of algorithm EDGE-COLOR, we know that there exists a constant C (depending only on n), s.t. if $\Delta(G_k) > C$, then
$$\kappa(G_k) \leq \Delta(G_k) + o(\Delta(G_k)) \leq (1+\delta)\Delta(G_k). \qquad (4)$$

2. Next we bound $\Delta(G_{k+1})$. Let $\Delta(G_k) > C$ as above. Given $\kappa(G_k)$, the graph G_{k+1} is made by packets arriving during the time period of length $\kappa(G_k)$. Since the arrival process of packets arriving at any input or destined for any output is a Bernoulli i.i.d. process of rate at most λ^*, with probability $1 - o\left(\frac{1}{n^2(\kappa^2(G_k))}\right)$,
$$\begin{aligned}\Delta(G_{k+1}) &\leq \lambda^*\kappa(G_k) + O(\sqrt{\lambda^*\kappa(G_k)\log(n\kappa(G_k))}) \\ &\leq (1+\delta)\lambda^*\kappa(G_k) \qquad (5)\end{aligned}$$
Furthermore, since at most one packet can arrive for any input in one time step, and there are n inputs, we always have
$$\begin{aligned}\Delta(G_{k+1}) &\leq n\kappa(G_k) \\ &\leq 2n\Delta(G_k) \qquad (6)\end{aligned}$$
The last inequality follows from the fact that $\kappa(G_k) \leq 2\Delta(G_k)$.

3. From (4), (5) and (6) we obtain
$$\begin{aligned}&E[\Delta(G_{k+1}) - \Delta(G_k)|\Delta(G_k) > C] \\ &\leq ((1+\delta)^2\lambda^* - 1)\Delta(G_k) + o\left(\frac{1}{n^2\Delta^2(G_k)}\right)2n\Delta(G_k) \\ &\leq -\varepsilon\Delta(G_k) + o\left(\frac{1}{n\Delta(G_k)}\right) \\ &\approx -\varepsilon\Delta(G_k)\end{aligned}$$
This proves (3) and hence Theorem 1. □

4 Proof of Theorem 2: Analysis of RAND-COLOR

We start with an outline of the proof of Theorem 2; the details are in Sections 4.2, 4.3 and 4.4, followed by a discussion in Section 7.

4.1 Proof Outline for Theorem 2

The algorithm runs in Δ stages. At each stage, n edges, one at each input, are considered for coloring. Thus, as time progresses, the number of edges in the graph keeps growing. We will show that the graph grows "regularly", in the sense that the intermediate graphs are "similar" to the final graph. Thus, the vertices are well-informed about the graph structure when they pick colors for edges incident to them even in the earlier stages, which leaves sufficiently many free colors for the later stages. For the purpose of analysis, we divide these Δ stages into $\frac{1}{\varepsilon}$ epochs, each of length $\varepsilon\Delta$, where $\varepsilon = \frac{\log^2(n\Delta)}{\sqrt{\Delta}}$. The analysis considers the algorithm's behavior over each of these epochs.

Consider a fixed input-output pair (i,j). Let $L_i(t)$ be the colors used by edges incident to input i by time t, and $R_j(t)$ be colors used by edges incident to output j by time t. Let $C_{ij}(t)$ denote the number of common colors of i and j, i.e., the colors used by both input i and output j at time t. Thus, $C_{ij}(t) = |L_i(t) \cap R_j(t)|$. At any stage t, the number of edges incident to input i is t. Let $E_k(a,b)$ denote the number of edges between input-output pair (a,b) at the end of the k^{th} epoch, i.e., at the end of stage $t = k\varepsilon$, and let $E_k(.,b)$ represent the number of edges incident to output b at the end of the k^{th} epoch. Note that $E_k(.,b) = \sum_{a=1}^{n} E_k(a,b)$.

The following lemma guarantees that at all stages, the number of selected edges incident to each node is highly concentrated around the mean value.

Lemma 1. *At the end of the* k^{th} *epoch, for any input-output pair* (a,b)*,*

$$\Pr\left[|E_k(a,b) - k\varepsilon\Delta_{ab}| \geq \sqrt{10\log(n\Delta)k\varepsilon\Delta_{ab}}\right] \leq \frac{1}{n^5\Delta^5}. \qquad (7)$$

In addition,

$$\Pr\left[|E_k(.,b) - k\varepsilon\Delta| \geq \sqrt{10\log(n\Delta)k\varepsilon\Delta}\right] \leq \frac{1}{n^5\Delta^5}. \qquad (8)$$

Note that $k\varepsilon\Delta_{ab}$ is the mean value of $E_k(a,b)$, while $k\varepsilon\Delta$ is the mean value of $E_k(.,b)$. The proof of this lemma follows from a straightforward application of Chernoff bounds. We omit the proof due to lack of space.

For a given fixed input-output pair (i,j), define $C(k) \stackrel{\triangle}{=} C_{ij}(k\varepsilon)$. The following lemma lets us bound the number of common colors in terms of a recurrence. We give the proof of this lemma in Section 4.2.

Lemma 2. *The increase in the number of common colors for an input-output pair (i,j) in the $(k+1)^{st}$ epoch is*

$$|C(k+1)-C(k)| \quad \geq \quad p_{ij}\varepsilon\Delta + \varepsilon\Delta(1-p_{ij})\alpha\left(\frac{(k\varepsilon\Delta - C(k))}{\Delta - k\varepsilon\Delta}\right) \pm O\left(\sqrt{k\varepsilon\Delta\log(n\Delta)}\right),$$

with probability $1-o(\frac{1}{n^5\Delta^5})$, where $p_{ij} \stackrel{\triangle}{=} \frac{\Delta_{ij}}{\Delta}$, and $\alpha \in (0,1)$ is a constant.

In the expression on the right side of the inequality, the first term gives the increase in the number of common colors due to edges of type (i,j) added in the $(k+1)^{st}$ epoch, the second term gives the increase due to new edges incident to i or j but not on both, while the third term is the error term.

Note that at any given stage, since we begin with the same palette of colors at each node, if the number of common colors for an input-output pair (i,j) is large, then there are more *free* colors available for an edge (i,j) arriving in the later stages. Conversely, if the number of common colors between any pair of input-output pairs is at least $\Delta - c$ when the algorithm halts, then there are at most c uncolored edges at each vertex. Thus, one needs to show that when the algorithm halts at Step 3, the number of common colors between any two vertices is large. We state the following lemma whose proof can be found in Section 4.3.

Lemma 3. *When* RAND-COLOR *halts in Step 3, the number of edges remaining to be colored at any vertex is $o(\Delta)$ with a high probability.*

Lemma 3 implies that using the GREEDY algorithm in Step 4, these remaining edges can be colored using an additional $o(\Delta)$ fresh colors. Thus, the algorithm uses a total of $\Delta + o(\Delta)$ colors with a high probability. This completes the proof of Theorem 2.

4.2 Proof of Lemma 2

Our goal is to obtain a lower bound on the increase in the number of common colors between a pair of input-output nodes (i,j) in the $(k+1)^{st}$ epoch, given the number of common colors $C(k)$ at the end of k^{th} epoch. The common colors between (i,j) increase when: **(a)** an edge between (i,j) is added; **(b)** an edge (i,j'), $j' \neq j$, is added and it chooses a color already used by j before; or, **(c)** an edge (i',j), $i' \neq i$, is added and it chooses a color already used by i before.

The number of edges added at input i in the epoch is $\varepsilon\Delta \stackrel{\triangle}{=} \delta$. The number of edges added at output j in the $(k+1)^{st}$ epoch will be $\delta \pm O(\sqrt{\log(n\Delta)(k+1)\varepsilon\Delta})$ (from Lemma 1). Further, $\varepsilon\Delta_{ij} \pm O(\sqrt{\log(n\Delta)(k+1)\varepsilon\Delta_{ij}})$ edges are added between input i and output j. Thus, in the $(k+1)^{st}$ epoch: **(a)** edges of type (i,j) number $\delta_{ij} \stackrel{\triangle}{=} \varepsilon\Delta_{ij}$; **(b)** edges incident to input i number δ; and, **(c)** edges incident to output j number δ; all with error of at most $O(\sqrt{\log(n\Delta)\Delta})$ (as $k+1 \leq 1/\varepsilon$) with a high probability. We denote this error in the number of trials as:

$$E_1 \quad = \quad O(\sqrt{\log(n\Delta)\Delta}). \tag{9}$$

We give the basic argument neglecting this error and later show that it does not violate our claims.

We now estimate the increase in C_{ij} during an epoch — the δ_{ij} edges of type (i,j) increase the number of common colors by the same number. The remaining $\delta - \delta_{ij}$ edges incident on i and an equal number of edges incident on j may or may not increase the number of common colors. Given $C(k) = C_{ij}(k\varepsilon)$, we would like to obtain a lower bound on the probability, $p_C(k)$, of each of these remaining edges increasing the number of common colors in the $(k+1)^{st}$ epoch. Since the colors picked by these edges are independent Bernoulli trials, in order to apply Chernoff bounds, we just need a lower bound on the mean increase in C_{ij} due to trials in the $(k+1)^{st}$ epoch.

Let $p_C^{in}(k)$ denote the probability that an edge arriving at input i (not incident to j) increases the number of common colors between input i and output j. This probability can vary depending on the output to which the edge is incident. We let $p_C^{in}(k)$ be the average probability of increasing $C(k)$ over all choices of the output. Let the edge be incident to output $o_l (o_l \neq j)$, for $1 \leq l \leq n$ with probability p_l^o. We would like to note that by Lemma 1, the value of p_l^o is the same for all epochs (with possible error of order $1/\sqrt{\Delta}$ which is negligible with respect to the error bound we are trying to prove).

Similarly, let edges arriving at output j be incident to input i_m ($i_m \neq i$), for $1 \leq m \leq n$, with probability p_m^i. Define $p_C^{out}(k)$ as the average probability for increasing common colors by edges arriving at output j. Let $p_C(k) = p_C^{in}(k) + p_C^{out}(k)$. We claim the following lower bound on $p_C(k)$. Some intuition for this lemma is provided in Section 4.4; the detailed proof can be found in Appendix B.

Lemma 4. $p_C(k) \geq \alpha\frac{k\delta - C(k)}{\Delta - k\delta}$.

Let $\mu = (1-p_{ij})\varepsilon\Delta p_C(k)$ be the mean increase in the number of common colors due to the independent Bernoulli trials corresponding to the edges incident to i or j (but not of type (i,j)). Then Lemma 4 implies that

$$\mu \geq (1-p_{ij})\varepsilon\Delta\alpha\frac{k\delta - C(k)}{\Delta - k\delta}.$$

By Chernoff bounds, we obtain that with probability at least $1-O(\frac{1}{n^5\Delta^5})$, the increase in common colors between pair (i,j) due to such edges is at least $\mu - O(\sqrt{\log(n\Delta)\mu})$. Thus, gathering the net increase in the number of common colors in the $(k+1)^{st}$ epoch, we get

$$\begin{aligned} |C(k+1)-C(k)| \quad &\geq \quad \delta_{ij} + (1-p_{ij})\delta\alpha\frac{k\delta - C(k)}{\Delta - k\delta} \\ &\quad - O(\sqrt{\log(n\Delta)\delta}). \end{aligned} \tag{10}$$

Note that in (10) above, the error term is $O(\sqrt{\log(n\Delta)\delta})$. Our aim is to show that the overall error is bounded by $E \triangleq O(\sqrt{\log(n\Delta)\Delta})$. Thus the error in (10) is negligible compared to E.

In estimating the change in the number of common colors during an epoch, we had neglected some error terms. We now show that they contribute an error of at most E.

- **Number of trials:** This was error E_1 as in (9), which is $O(E)$.
- **Error in** $p_C(k)$**:** In the expression for $p_C(k)$, $\alpha\frac{k\delta - C(k)}{\Delta - k\delta}$, the numerator could have possible deviation of $O(\sqrt{\log(n\Delta)\Delta})$, while the denominator is always $\Omega(\varepsilon\Delta)$. This leads to a possible error in $p_C(k)$ of $O(\frac{\sqrt{\log(n\Delta)\Delta}}{\varepsilon\Delta})$. Over $\varepsilon\Delta$ trials, this can contribute an error of $O\sqrt{\log(n\Delta)\Delta}$, which is again $O(E)$.
- **Change in** $p_C(k)$ **during an epoch:** During the $(k+1)^{st}$ epoch, $p_C(k)$ was considered to be fixed. But in reality, the expression $\frac{k\delta - C(k)}{\Delta - k\delta}$ will change during the course of an epoch. The denominator always decreases, thereby increasing $p_C(k)$, and can thus be neglected for a lower bound. However, the numerator can decrease. It decreases iff $C(k+1) - C(k) \geq \varepsilon\Delta$, in which case $p_C(.)$ is large enough. Hence, such a change in $p_C(k)$ does not violate our claims about the increase in C_{ij} during the $(k+1)^{st}$ epoch.

From (10) and the preceding discussion,

$$\begin{aligned} |C(k+1)-C(k)| \quad &\geq \quad \delta_{ij} + (1-p_{ij})\delta\alpha\frac{k\delta - C(k)}{\Delta - k\delta} \\ &\quad - O(\sqrt{\log(n\Delta)\Delta}). \end{aligned} \tag{11}$$

Note that the estimate in the number of trials and the estimate for $p_C(k)$ fail with probability $O\left(\frac{1}{n^5\Delta^5}\right)$. Thus, by union bound, the result (11) holds with probability $1-O\left(\frac{1}{n^5\Delta^5}\right)$. This proves Lemma 2. □

4.3 Proof of Lemma 3

From Lemma 2, for a particular pair (i,j),

$$C(k+1)-C(k) \geq \delta_{ij} + (1-p_{ij})\delta\alpha\frac{k\delta - C(k)}{\Delta - k\delta} - O(\sqrt{\log(n\Delta)\Delta})$$

with probability at least $1-\frac{1}{n^5\Delta^5}$. From the union bound, we can conclude that for n^2 pairs of input-outputs and all $\frac{1}{\varepsilon}$ epochs, the desired change will happen with probability at least $1-\frac{1}{n^3\Delta^3}$. The only problem is that the error term adds up over $\frac{1}{\varepsilon}$ epochs to give a weak $O(\Delta)$ error bound. To remedy this, we use the Azuma-Hoeffding inequality for martingales [12] to establish that the total error is $o(\Delta)$.

Let Z denote the number of common colors between a pair (i,j) of interest at the end of Step 3 in RAND-COLOR. We want to show that Z is concentrated sharply around its mean. Define the conditional expectation of Z with respect to information $C(k)$, as $Z_k = E[Z|C(k)]$. By definition, the Z_k's form a martingale. We would like to bound the difference $|Z_{k+1} - Z_k|$. For simplicity, assume that the $C(k)$'s are evolving exactly as their high probability lower bound and Z is the mean for this lower bound. From Lemma 2, we note that $|C(k+1)-C(k)|$ deviates from its mean by at most $O(\sqrt{\Delta\log(n\Delta)})$ with probability $1-o(\frac{1}{n^5\Delta^5})$. Thus, $C(k+1)$ is within $O(\sqrt{\Delta\log(n\Delta)})$ of $E[C(k+1)|C(k)]$, the expected value of $C(k+1)$ given $C(k)$, with probability $1-o(\frac{1}{n^5\Delta^5})$. The form of $p_C(k)$ in Lemma 3 suggests that if $C(k+1)$ is far from the mean, then $p_C(k)$ will be such that it will push $C(k+1)$ back towards mean. This implies that Z_k, the expected value of Z given $C(k)$, and Z_{k+1}, the expected value of Z given $C(k+1)$ will differ by at most $\sqrt{\Delta\log(n\Delta)}$ with probability $1-o(\frac{1}{n^5\Delta^5})$. With the remaining probability, it can change at most by $O(\Delta)$ and hence in expectation it contributes a change of $O(\frac{1}{n^5\Delta^4})$. Thus, we see that $|Z_{k+1} - Z_k| \leq \sqrt{\Delta\log(n\Delta)}$. Then by applying Azuma-Hoeffding inequality for martingales [12], we get

$$\Pr[|Z - E[Z]| \geq \lambda] \leq \exp\left(-\frac{\lambda^2}{\frac{2\Delta\log(n\Delta)}{\varepsilon}}\right).$$

Setting $\lambda = 5\log(n\Delta)\sqrt{\Delta}$, we obtain that

$$\Pr\left[|Z - E[Z]| \geq 5\log(n\Delta)\sqrt{\Delta}\right] \quad \leq \quad \exp(-5\log(n\Delta)) = \frac{1}{n^5\Delta^5}. \tag{12}$$

Thus Z, the number of common colors at the end of Step 3, is within $O(\log(n\Delta)\sqrt{\Delta})$ of its mean with a high probability. Now we move on to estimating the mean of Z.

From Lemma 2, and assuming that $C(k)$ is evolving exactly as its high probability lower bound,

$$\frac{C(k+1)-C(k)}{\varepsilon\Delta} \quad = \quad p_{ij} + (1-p_{ij})\alpha\left(\frac{k\delta - C(k)}{\Delta - k\delta}\right)$$

with an error of $O(\sqrt{\Delta\log(n\Delta)})$. On the scale of $\varepsilon\Delta$, this error does not show up. Let $x(s) = C(s\Delta/\varepsilon)/\Delta$ for $s \in [0,1]$. Then, we can rewrite the above expression as

$$\frac{x(s+\varepsilon) - x(s)}{\varepsilon} \quad = \quad p_{ij} + (1-p_{ij})\alpha\frac{s - x(s)}{1-s}.$$

Since $\varepsilon = \log^2(n\Delta)\Delta^{-1/2} \to 0$ as $\Delta \to \infty$, we get

$$\frac{\mathrm{d}x(s)}{\mathrm{d}s} = p_{ij} + (1-p_{ij})\alpha\frac{s-x(s)}{1-s}.$$

With $x(0) = 0$ as the initial condition, we get the following solution of this differential equation.

$$x(s) = 1 - \frac{p_{ij} - \alpha(1-p_{ij})}{1-\alpha(1-p_{ij})}(1-s) - \frac{1-p_{ij}}{1-\alpha(1-p_{ij})}(1-s)^{\alpha(1-p_{ij})}.$$

Thus, $x(1) = 1$. This evaluation of the discrete differential equation happens with granularity $\varepsilon\Delta$. Therefore, the mean of Z will be at least $x(1)\Delta - \varepsilon\Delta$.

Combining our knowledge about the mean of Z and the concentration of Z around its mean, we get that with high probability,

$$Z \geq \Delta - O(\varepsilon\Delta) - O(\sqrt{\log(n\Delta)\Delta}) = \Delta - o(\Delta). \quad (13)$$

The last equality follows from the fact the $\varepsilon = o(1)$. Then, by applying the union bound to the n^2 input-output pairs, we assert that all of them will obey the above lower bound with high probability too. This proves that the proposed randomized algorithm for coloring a bipartite graph will use at most $\Delta + O(\log^2(n\Delta)\sqrt{\Delta})$ colors with probability $1 - o(\frac{1}{n\Delta})$. □

4.4 Intuition for Lemma 4

We sketch the main idea behind the proof of Lemma 4 via an example. When i (respectively j) does not choose an edge for j (respectively i), it chooses an edge of type (i, j'), with $j' \neq j$ (respectively (i', j), with $i' \neq i$). Conditioned on the selection of the edge (i, j') by i, $p_C^{in}(k)$ depends on colors not used by i and j', and on how many of these colors are used by j. Denote the set of colors used by j, but not used by i and j' as $A(j;\bar{i};\bar{j'})$, and the colors not used by i and j as $A(\bar{i};\bar{j'})$. Then, conditioned on the selection of edge (i, j'), we have $p_C^{in}(k) = A(j;\bar{i};\bar{j'})/A(\bar{i};\bar{j'})$. Similarly, conditioned on the selection of edge (i', j), we have $p_C^{out}(k) = A(i,\bar{i'};\bar{j})/A(\bar{i'};\bar{j})$. Recall that δ is the number of edges picked by an input in each epoch. Let us ignore lower order terms for the moment. Then, $k\delta$ is the number of colors which have already been used at any vertex by the end of the k^{th} epoch. Thus, $A(\bar{i},\bar{j'}) = A(\bar{i'},\bar{j}) = \Delta - 2k\delta + C(k) \leq \Delta - k\delta$, which is precisely the term in the denominator of the right side of the inequality in Lemma 4.

Now consider the term $k\delta - C(k)$ in the numerator. Here, $k\delta$ is the total number of colors used at output j (respectively at input i), while $C(k)$ colors have been used both at i and j. Thus, $A(j;\bar{i}) = A(i,\bar{j}) = k\delta - C(k)$. Out of these $k\delta - C(k)$ colors, we have to subtract the colors used at j' (respectively at i') in order to calculate $A(j;\bar{i};\bar{j'})$ (respectively $A(i;\bar{i'};\bar{j})$). We establish Lemma 4 by showing that $D_{ij} = A(j;\bar{i};\bar{j'}) + A(i,\bar{i'};\bar{j})$ is always larger than a constant fraction of $k\delta - C(k)$. This is achieved by showing that there is a positive "drift" for $\frac{D_{ij}}{k\delta - C(k)}$ if it is smaller than some constant. The reason for the positive "drift" (or a conservation law) is as follows. If $A(j;\bar{i};\bar{j'})$ and $A(i;\bar{i'};\bar{j})$ are both really small, then the color chosen for the edge (i, j') will be such that it will, with high probability, increase the pool $A(\bar{j};i;\bar{i'})$. Similarly, the color chosen for edge (i', j) will increase the pool $A(\bar{i};j;\bar{j'})$ with high probability. Thus, in each epoch, sufficiently many such events will occur giving us lower bound in Lemma 4.

This idea can be extended to the case of general graphs (not necessarily bipartite) by choosing D_{ij} appropriately, taking into account all possible edges which could be selected by input i and output j. Next we sketch the proof of Theorem 3, which addresses the case of general graphs.

5 Proof Sketch for Theorem 3

To prove Theorem 3, we need to prove lemmas similar to the ones shown for the proof of Theorem 2. We would like to note that unlike algorithm RAND-COLOR, algorithm RAND-COLOR-G runs for only $\Delta/2$ stages, since in the case of general graphs, an edge gets selected in a stage if either of its end-points gets selected; thus the probability of an edge getting selected in any stage is twice that in RAND-COLOR.

As in the proof of Theorem 2, we have to bound the number of common colors C_{ij} for any pair of vertices i and j at the end of Step 3. Lemma 1 is about the regularity of the graph's growth and it holds for a general graph too under algorithm RAND-COLOR-G. A result analogous to Lemma 3 would imply Theorem 3. The crux of the proof of Lemma 3 is Lemma 2. For the proof of Lemma 2, we consider pairs of input-output vertices, but the fact that the graph is bipartite is not used in the proof. Hence the same argument can be applied to any arbitrary pair of vertices. Thus, a result analogous to Lemma 2 is true for the case of general graphs too, which in turn implies Lemma 3 for this case. The details of the proof will be given in the full version of the paper.

6 Proof sketch of Theorem 4

Theorem 4 follows from the proofs of Theorems 2 and 3 when we inspect the reason behind the constraint $\Delta = \omega(n^2)$. We would like to summarize these reasons as follows (which are mainly in the proof of Lemma 3).

(a) Constraint on ε: The error term introduced in Lemma 1 is $O(\sqrt{\log(n\Delta)\Delta})$. The epoch length is $O(\varepsilon\Delta)$. For the error to be of a lower order than the mean, we need $\varepsilon\Delta = \omega(\sqrt{\log(n\Delta)\Delta})$, or $\varepsilon =$

$\omega(\frac{\sqrt{\log(n\Delta)}}{\sqrt{\Delta}})$. In addition, we need $\varepsilon = o(1)$ in order for equation (13) to hold.

(b) Relation between Δ and n: For using the Law of Large Numbers, we need $\varepsilon\Delta_{ij} = \omega(1)$. For this, it suffices to have $\Delta_{ij} = \Omega(\sqrt{\Delta})$. If $\Delta_{ij} = o(\sqrt{\Delta})$, then we ignore these edges in the analysis of Step 3, and let them contribute to the uncolored edges remaining at the end of Step 3. Clearly, if $\Delta_{ij} = 0$ whenever $\Delta_{ij} = o(\sqrt{\Delta})$ as in condition 1 of Theorem 4, no such edges will be left over. Otherwise, these edges contribute no more than $n\Delta_{ij} = o(n\sqrt{\Delta})$ to the degree of each vertex at the end of Step 3. Thus, the constraint $n = o(\sqrt{\Delta})$ or alternatively, $\Delta = \omega(n^2)$ is sufficient to ensure the max-degree of the graph colored greedily in Step 4 is $o(\Delta)$.

Thus, the conditions (1) and (2) satisfy the constraints imposed by the proofs as noted in (a) and (b) above. Hence, the proofs of Theorem 2 and Theorem 3 can be extended to get a proof of Theorem 4. This completes the sketch of the proof. □

7 General Remarks

Tightness of Theorems 2 and 3. The proof of Lemma 3 suggests an error of order $O(\log^2(n\Delta)\sqrt{\Delta})$. But neglecting all edges (i,j) with $\Delta_{ij} = o(\sqrt{\Delta})$ leads to an error bound of $o(n\sqrt{\Delta})$. Hence, the number of colors used by RAND-COLOR is $\Delta + \max\{O(\log^2 n\Delta)\sqrt{\Delta}, o(n\sqrt{\Delta})\}$.

Note that if each node in the graph G is connected to $O(1)$ other nodes, then we need the constraint $\Delta = \omega(\log^4 n)$ for our results to hold. Bar-Noy et al.'s [2] lower bound of $2\Delta - 1$ for online edge coloring of graphs with $\Delta = O(\log n)$ implies our result is almost tight for such graphs. In general, if the maximum degree of any node in the graph is $\Phi \leq n$, then conditions (1) and (2) in Theorem 4 are satisfied whenever $\Delta = \Omega(\max\{\Phi^2, \log^4 n\})$. We note that since at least one node is connected to at least Φ other nodes, we always have $\Delta \geq \Phi$ in this case.

References

[1] S. Asmussen. *Applied Probability and Queues.* New York: Wiley, 1987.

[2] A. Bar-Noy, R. Motwani and J. Naor. *The Greedy Algorithm is Optimal for On-Line Edge Coloring.* Information Processing Letters, 44 (1992), pages 251-253.

[3] *Cisco 12000 Gigabit Switch Router.* Product overview, `www.cisco.com`, Feb. 2000.

[4] R. Cole, K. Ost and S. Schirra. *Edge Coloring Bipartite Multigraphs in $O(E \log D)$ Time.* Combinatorica 21(1):5–12, 2001.

[5] J. Dai and B. Prabhakar. *The Throughput of Data Switches With and Without Speedup.* Proceedings of INFOCOM 2000.

[6] M.R. Garey and D.S. Johnson. *Computers and intractability — a Guide to the Theory of NP-completeness.* W.H. Freeman, 1979.

[7] P. Giaccone, B. Prabhakar and D. Shah. *Towards Simple, High-Performance Schedulers for High-aggregate bandwidth.* Proceedings of INFOCOM 2002.

[8] D. Grable and A. Panconesi. *Nearly Optimal Distributed Edge Coloring in $O(\log\log n)$ Rounds.* `http://citeseer.nj.nec.com/140887.html`.

[9] D. Dubhashi, D. Grable and A. Panconesi. *Near-Optimal, Distributed Edge Coloring via the Nibble Method.* Proceedings of ESA 1995.

[10] N. McKeown, V. Anantharan and J. Walrand. *Achieving 100% Throughput in an Input-Queued Switch.* Proceedings of INFOCOM 1996.

[11] N. McKeown. *Scheduling algorithms for Input-Queued Switches.* PhD Thesis, UC Berkeley, 1995.

[12] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[13] A. Panconesi and A. Srinivasan. *Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding bounds.* Proceedings of PODC, 1992, pages 251-262.

[14] D. Shah and M. Kopikare. *Delay Bounds for Approximate Maximum Weight Matching Algorithms for Input-Queued Switches.* Proceedings of INFOCOM 2002.

[15] L. Tassiulas. *Linear Complexity Algorithms for Maximum Throughput in Radio Networks and Input Queued Switches.* Proceedings of INFOCOM 1998.

[16] A. Rasala and G. Wilfong. *Strictly non-blocking WDM cross-connects.* Proceedings of SODA 2002.

[17] L. Tassiulas and A. Ephremides. *Stability Properties of Constrained Queueing Systems and Scheduling for Maximum Throughput in Multihop Radio Networks.* IEEE Transactions on Automatic Control, Vol. 37, No. 12, pp. 1936-1949, December 1992.

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

Appendix

A Discussion of Step 2 in RAND-COLOR

Here we show that it is necessary to either assume that the edge arrival order is random or to explicitly randomize the order of edge arrivals. We present an example which shows that if the edges are presented in an adversarial order, then $\Delta + \Omega(\Delta)$ colors are needed even for dense graphs.

Consider a bipartite multigraph G with four vertices and $\Delta_{ij} = \Delta/2$, for $1 \leq i, j \leq 2$. Let $A = (1+\varepsilon)\Delta$. The edges are presented in the following order: In the first $\Delta/2$ stages, all $(1,1)$ and $(2,2)$ edges are presented. The algorithm chooses some $\Delta/2$ of the A colors for the edges $(1,1)$. Without loss of generality, let these be $1,\ldots,\Delta/2$. Also, the algorithm chooses $\Delta/2$ of the A colors for the edges $(2,2)$. If the two sets of $\Delta/2$ colors are chosen independently, the probability that any color chosen for the $(2,2)$ edges is from $\{1,\ldots,\Delta/2\}$ is $1/2(1+\varepsilon)$. Then, with high probability, out of the $\Delta/2$ colors chosen for $(2,2)$ edges, roughly $\Delta/4(1+\varepsilon) \pm O(\sqrt{\Delta})$ colors are also used for the $(1,1)$ edges. The $\Delta/2$ edges of type $(1,2)$ cannot share any color with either the $(1,1)$ or the $(2,2)$ edges. Hence, at least $\Delta/4(1+\varepsilon) \pm O(\sqrt{\Delta})$ extra colors will be required. Thus, the total number of colors required will at least be $1.25\Delta - O(\sqrt{\Delta})$.

B Proof of Lemma 4

Recall that the epoch of consideration is $k+1$. Let us define $A(x)$ to be the set of colors used by input x and $A(x^c)$ to be the set of colors not used by input x at the end of k^{th} epoch. Similarly, define the sets $B(y)$ and $B(y^c)$ for output y. For outputs j, j' and input i define

$$r_1(j,i;j') = \frac{|(B(j) \cap A(i^c)) \backslash B(j')|}{|B(j) \cap A(i^c)|},$$

and for inputs i, i' and output j define

$$r_2(i,j;i') = \frac{|(A(i) \cap B(j^c)) \backslash A(i')|}{|A(i) \cap B(j^c)|}.$$

When an edge (i, o_l), $o_l \neq j$ is added at input i, it will choose a color used by output j (but not by i and o_l) with probability

$$\begin{aligned} p_{in}(i,o_l) &= \frac{|(B(j) \cap A(i^c)) \backslash B(o_l)|}{|A(i^c) \backslash B(o_l)|} \\ &= r_1(j,i;o_l) \frac{|A(i^c) \cap B(j)|}{|A(i^c) \backslash B(o_l)|} \quad (14) \\ &\geq r_1(j,i;o_l) \frac{|A(i^c) \cap B(j)|}{\Delta - k\delta} \quad (15) \end{aligned}$$

The last inequality follows from the fact that

$$|A(i^c) \backslash B(o_l)| \leq |A(i^c)| = \Delta - k\delta.$$

Input i chooses an edge directed towards output o_l with probability p_l^i. Hence,

$$\begin{aligned} p_C^{in} &= \sum_l p_{in}(i,o_l) p_l^i \\ &\geq \frac{|A(i^c) \cap B(j)|}{\Delta - k\delta} \sum_l p_l^i r_1(j,i;o_l) \\ &= \frac{|A(i^c) \cap B(j)|}{\Delta - k\delta} r_1(i) \quad (16) \end{aligned}$$

where $r_1(i) \stackrel{\triangle}{=} \sum_l p_l^i r_1(j,i;o_l)$. Similarly, an edge incident to output j is the edge i_m, j with probability p_m^o. This edge picks a color used by input i (and not by j and i_m) with probability

$$\begin{aligned} p_C^{out} &\geq \frac{|A(i^c) \cap B(j)|}{\Delta - k\delta} \sum_m p_m^o r_2(i,j;i_m) \\ &= \frac{|A(i^c) \cap B(j)|}{\Delta - k\delta} r_2(j) \quad (17) \end{aligned}$$

where $r_2(j) \stackrel{\triangle}{=} \sum_m p_m^o r_2(i,j;i_m)$.

Thus we need to bound $r_1(i)$ and $r_2(j)$ in order to get a bound on p_C^{in} and p_C^{out}. Let us first consider how $r_1(j,i;j')$ changes due to the addition of new edges.

Before proceeding further, we introduce the following notation for simplicity. Assuming that we are considering a fixed pair (i,j), let $r_1^x = r_1(j,i;x)$, and $r_2^y = r_2(i,j;y)$. Also let $r_1(j,i;j') = \frac{a}{b}$, where a and b are integers. On an average, one edge gets added at a node in any stage. Note that only the edges incident to i, j and j' can change $r_1(j,i;j')$. We consider these edges one by one.

Edge incident to output j: Let this edge is incident to input i_m. This happens with probability p_m^o. Such an edge has to pick a color from the set of colors not used by j and i_m. Thus, it can be from one of the following set of colors:

(a) Colors already used by i, i.e. $A(i) \cap B(j^c) \cap A(i_m^c)$:
In this case, the number of common colors between (i,j) increases, while the ratio $r_1(j,i;j')$ is not affected. This happens with probability proportional to $|A(i) \cap B(j^c) \cap A(i_m^c)| = r_2(i,j;i_m)|A(i) \cap B(j^c)|$. At the end of k^{th} epoch,

$$|A(i) \cap B(j^c)| \approx k\varepsilon\Delta - C(k) = k\delta - C(k),$$

where $\delta = \varepsilon\Delta$ as before. Hence, this probability is proportional to

$$p_A = r_2(i,j;i_m)(k\delta - C(k)).$$

(b) Colors already used by j' and not by i, i.e. $A(i^c) \cap B(j') \cap B(j^c) \cap A(i_m^c)$:
The numerator of $r_1(j,i;j')$ is $|A(i^c) \cap B(j) \backslash B(j')|$, which remains same in this case. But the denominator $|A(i^c) \cap B(j)|$ increases by 1. Hence $r_1(j,i;j')$ changes from $\frac{a}{b}$ to $\frac{a}{b+1}$. Thus it decreases by $\frac{a}{b} - \frac{a}{b+1} \approx \frac{a}{b^2}$. This happens with probability proportional to

$$p_B \leq r_1(j,i;j')|A(i) \cap B(j^c)| = r_1(j,i;j')(k\delta - C(k)).$$

(c) Colors not used by i and not used by j', i.e. $A(i^c) \cap B(j'^c) \cap B(j^c) \cap A(i_m^c)$:
In this case, the numerator $|A(i^c) \cap B(j) - B(j')|$ of $r_1(j,i;j')$ increases by 1. Further, the denominator $|A(i^c) \cap B(j)|$ increases by 1 too. That is, $r_1(j,i;j')$ changes from $\frac{a}{b}$ to $\frac{a+1}{b+1}$. Thus it increases by $\frac{a+1}{b+1} - \frac{a}{b} \approx \frac{b-a}{b^2}$. This happens with probability proportional to

$$p_D \geq |B(j^c) \cap A(i_m^c)| - p_A - p_B.$$

Now $|B(j^c) \cap A(i_m^c)| \approx \Delta - 2k\delta + C_{i_m j}(k)$. One can show that if $C(k)$ are evolving "nicely" (as in Lemma 2), then $\Delta - 2k\delta + C_{i_m j}(k) \geq k\delta - C(k)$ (the details of the proof will be given in the full paper). This in turn implies that

$$p_D \, geq(k\delta - C(k) - p_A - p_B.$$

To summarize, given the edge (i_m, j) is added, the expected change in $r_1^{j'}$ is bounded below by

$$E[\partial r_1^{j'} | i_m] \quad \geq \quad (k\delta - C(k)) \left((1 - r_1^{j'} - r_2^{i_m}) \frac{(b-a)}{b^2} - r_1^{j'} \frac{a}{b^2} \right). \quad (18)$$

Edge incident to input i: Let the edge added at input i be (i, o_l). Then the following three possibilities arise:

(a) This edge picks a color from the set $A(i^c) \cap B(j) \cap B(j'^c)$. In this case, $r_1(j,i;j')$ changes from $\frac{a}{b}$ to $\frac{a-1}{b-1}$ and hence it decreases by $\approx \frac{b-a}{b^2}$. The upper bound on the probability of this event is proportional to

$$p_E = r_1(j,i;j')|A(i^c) \cap B(j)| \approx r_1(j,i;j')(k\delta - C(k)).$$

(b) This edge picks a color from $A(i^c) \cap B(j) \cap B(j')$. In this case, $r_1(j,i;j')$ increases.

(c) This edge picks a color from $A(i^c) \cap B(j^c)$. In this case, $r_1(j,i;j')$ remains the same.

Hence the expected change due to an edge added on i is :

$$E[\partial r_1^{j'}] \quad \geq \quad -(k\delta - C(k)) r_1^{j'} \frac{(b-a)}{b^2} \quad (19)$$

Edge incident to output j': Let the edge added to output j' be incident to input i_q, $1 \leq q \leq n$. This edge can result in a change in $r_1(j,i;j')$ only if it picks a color from the set $A(i^c) \cap B(j) \cap B(j'^c)$. In this case, $r_1(j,i;j')$ will change from $\frac{a}{b}$ to $\frac{a-1}{b}$; i.e. it decreases by $\frac{1}{b}$. The upper bound on the probability of this event happening is proportional to $r_1(j,i;j')|A(i^c) \cap B(j)|$. Thus the expected change due to addition of edge on j' is bounded below by:

$$E[\partial r_1^{j'}] \quad \geq \quad -(k\delta - C(k)) r_1^{j'} \frac{1}{b} \quad (20)$$

From (18), (19) and (20), and since (i_m, j) is chosen with probability p_m^o, we get

$$E[\partial r_1^{j'}] \quad \geq \quad (k\delta - C(k))((1 - r_1^{j'} - \sum_m p_m^o r_2^{i_m}) \frac{b-a}{b^2} - r_1^{j'} \frac{a}{b^2} - r_1^{j'} \frac{b-a}{b^2} - r_1^{j'} \frac{1}{b}). \quad (21)$$

Since $\frac{b-a}{b^2} = \frac{1}{b}(1 - a/b) = \frac{1}{b}(1 - r_1^{j'})$, (21) reduces to

$$\begin{aligned} E[\partial r_1^{j'}] \quad &\geq \quad \frac{(k\delta - C(k))}{b}((1 - r_1^{j'} - r_2(j))(1 - r_1^{j'}) - (r_1^{j'})^2 \\ &\quad - r_1^{j'}(1 - r_1^{j'}) - r_1^{j'}) \\ &= \quad \frac{(k\delta - C(k))}{b} \left(1 - 4r_1^{j'} - r_2(j) + (r_1^{j'})^2 + r_1^{j'} r_2(j) \right) \\ &\geq \quad \frac{(k\delta - C(k))}{b} \left(1 - 4r_1^{j'} - r_2(j) \right) \quad (22) \end{aligned}$$

Input i picks an edge incident to output o_l with probability p_l^i, $1 \leq l \leq n$. In equation (22), j' is a place-holder for such o_ls. Hence from (22) and the fact that $r_1(i) = \sum_l p_l^i r_1^{o_l}$, we get

$$E[\partial r_1(i)] \quad \geq \quad K_1 (1 - 4r_1(i) - r_2(j)) \quad (23)$$

where $C_1 = \frac{(k\delta - C(k))}{b}$. By an argument similar to the above argument, we get

$$E[\partial r_2(j)] \quad \geq \quad K_2 (1 - 4r_2(j) - r_1(i)) \quad (24)$$

From (23) and (24), we obtain

$$E[\partial(r_1(i) + r_2(j))] \quad \geq \quad K^* (2 - 5(r_1(i) + r_2(j))) \; (25)$$

where, $K^* = \min\{K_1, K_2\} > 0$.

Equation (25) implies that for $r_1(i) + r_2(j) < 2/5$, $r_1(i) + r_2(j)$ has a positive drift in expectation. This is independent of the value of K^* as long as $K^* > 0$, which is guaranteed by definition. This implies that there exists a constant $\alpha > 0$ such that $r_1(i) + r_2(j) > \alpha$ at all times with high probability by Strong Law of Large Numbers. One can use other concentration inequalities to get stronger results.

From (16), (17) and above, we obtain that

$$p_C(k) \quad \geq \quad \alpha \frac{|A(i^c) \cap B(j)|}{\Delta - k\delta} = \alpha \frac{k\delta - C(k)}{\Delta - k\delta}. \quad (26)$$

This completes the proof of Lemma 4. □