Low Memory Algorithms for Online Edge Coloring

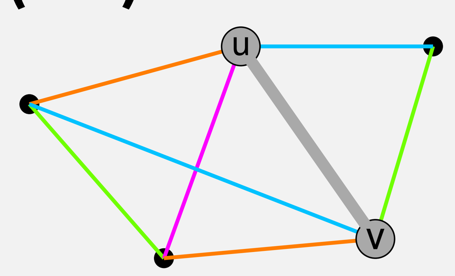
Manuel Stoeckl (Dartmouth), *joint work with* Prantar Ghosh (DIMACS)

Link to paper

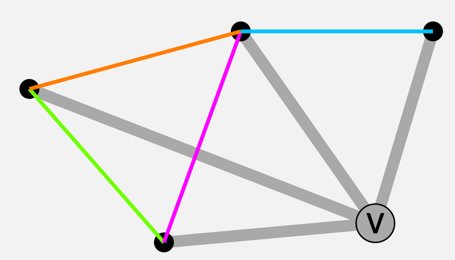
1. Online edge coloring

- Algorithm receives edges that together form n vertex graph of max degree Δ
- When each edge (or group of edges) arrives, must assign a color (or colors)

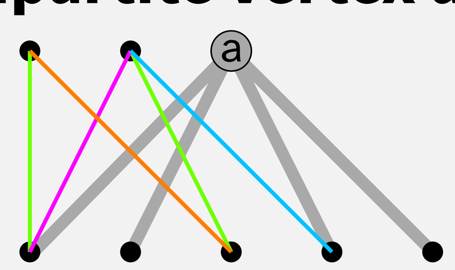
Edge arrival (EA)



Vertex arrival (VA)



One-sided bipartite vertex arrival (1VA)



Goal: use as few colors as possible

2. State of the art for edge arrivals

No memory constraints:

- $\left(\frac{e}{e-1} + o(1)\right)\Delta$ colors: *Kulkarni, Liu, Sah, Sawhney, Tarnawski 2022*

With $o(n\Delta)$ bits of space:

- $O\left(\Delta^2/s + \Delta\right)$ colors in $\tilde{O}(ns)$ space: *Ansari, Saneian, Zarrabi-Zadeh 2022*

W-streaming:

- $O\left(\Delta^{1.5}/s + \Delta\right)$ in $\tilde{O}(ns)$ space, simple graphs: *Saneian, Behnezhad 2023*
- $\tilde{O}\left(\Delta^{1.5}\right)$ in $\tilde{O}(n)$ space, multigraphs: *Checkik, Mukhtar, Zhang 2023*

3. Open problems

- Space usage of "intuition" algorithms
- Is online $O\left(\Delta^{1.5}\right)$-edge coloring in edge arrival streams possible with $\tilde{O}(n)$ space, to match W-streaming?
- EA or VA space lower bounds for $\beta\Delta$-edge coloring when $\beta \geq 2$

4. Reducing VA to 1VA model

- Well known randomized construction
- Deterministic construction using high rate-distance product codes

5. Intuition: unproven 2Δ color 1VA algorithm on $A \sqcup B$

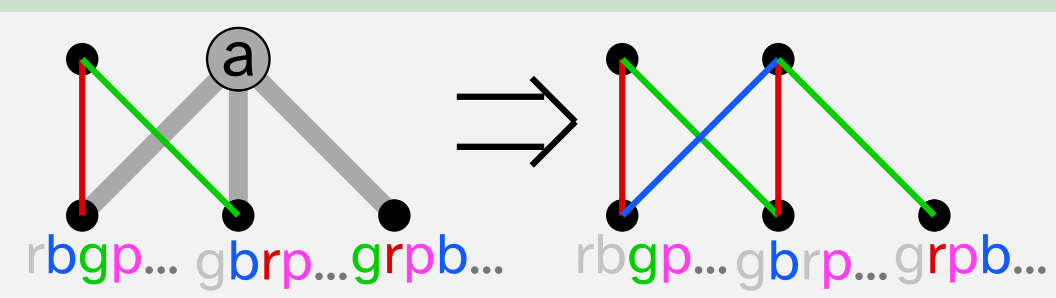
Init

- For all $b \in B$, $\sigma_b \leftarrow$ random permutation on $[2\Delta]$, $h_b \leftarrow 1$, $F_b \leftarrow \emptyset$

Process$(a \in A)$

- $S \leftarrow \emptyset$
- For $\{a, b\}$ incident on a, in random order
  - While $F_b \subseteq S$: add $\sigma_b[h_b]$ to F_b, increment h_b
  - Assign random color $c \in F_b \setminus S$ to $\{a, b\}$
  - Add c to S, remove c from F_b

6. Vertex arrival example



7. Making a practical 1VA algorithm for $O(\Delta)$ edge coloring using $\tilde{O}(n)$ space

For easier analysis:

- Use more colors
- Discard all unused colors

Multigraph:

- A longer proof

Deterministic (exponential time)

- Acquire blocks of $O(\log n)$ colors at a time, discard when half used
- Find colors using perfect matching
- Use fixed set of "good" permutations

Compact advice or fewer random bits:

- Use $(\epsilon, \log n)$-wise independent permutation families

8. Intuition: unproven 2Δ color edge arrival algorithm

Init

- For all $v \in V$, $\sigma_v \leftarrow$ random permutation on $[2\Delta]$, $h_v \leftarrow 1$, $F_v \leftarrow \emptyset$

Process$(\{x, y\})$

- While $F_x \cap F_y = \emptyset$:
  - add $\sigma_x[h_x]$ to F_x and increment h_x
  - add $\sigma_y[h_y]$ to F_y and increment h_y
- Assign random color $c \in F_x \cap F_y$ to $\{x, y\}$
- Remove c from F_x and F_y

9. Making a practical EA algorithm for $\tilde{O}(\Delta)$ edge coloring using $\tilde{O}\left(n\sqrt{\Delta}\right)$ space

For easier analysis:

- Periodically replace each F_v with fresh set of $O\left(\sqrt{\Delta \log n}\right)$ colors
- For randomized algorithm/static input, will have $F_x \cap F_y \neq \emptyset$ w.h.p.

Multigraph ($\times O(\log \Delta)$ more colors):

- Base design breaks on frequently repeated edges; they are easy to detect
- Process repeat edges in sketch with greater tolerance for repetition

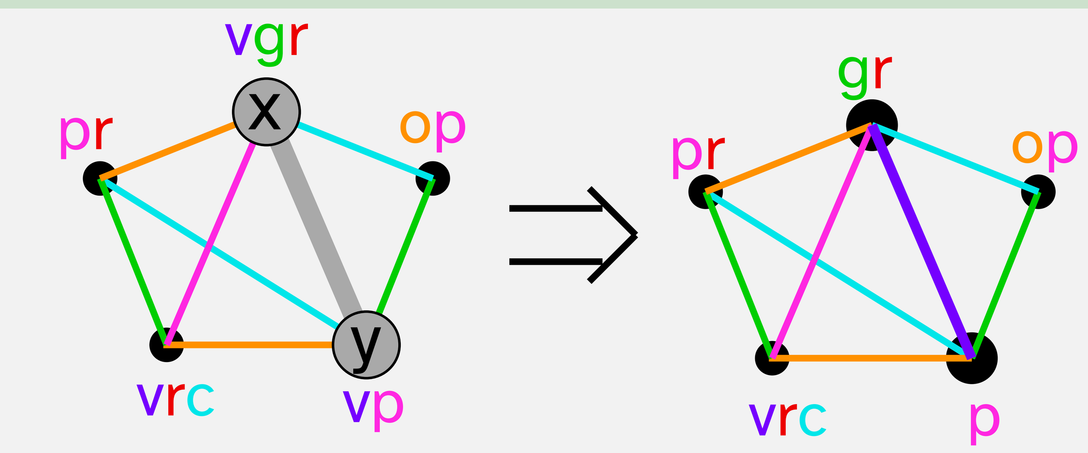
Deterministic (exp time, $\times O(\log \Delta)$ more colors)

- For certain "good" permutations, algorithm would always work – if we could guess the right color in $F_x \cap F_y$
- Picking arbitrary color from $F_x \cap F_y$ succeeds $\geq 1/3$ of the time
- Chain together $O(\log \Delta)$ instances – $O(n)$ edges left over

Compact advice or fewer random bits:

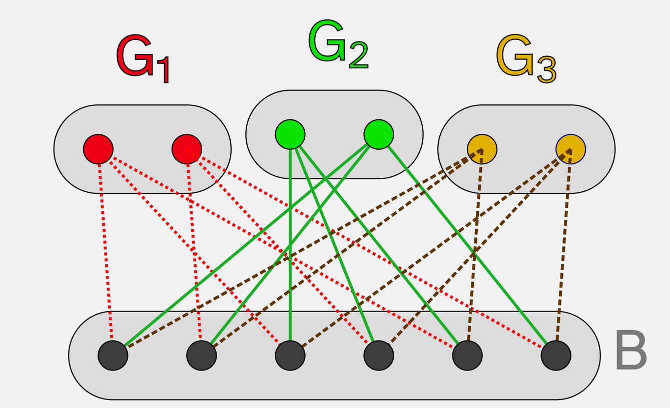
- Use $\left(\epsilon, \sqrt{\Delta \log n}\right)$-wise independent permutation families

10. Edge-arrival example



11. $\Omega\left((2-\beta)^3 n\right)$ space needed for deterministic $\beta\Delta$ coloring, for $\beta < 2$

Proof:



- Hard instance: union of Δ lopsided regular bipartite graphs $G_1, \ldots, G_\Delta$, presented one by one
- Each time algorithm processes a graph G_i, for each $v \in B$, it marks a fresh set $S_v^{(i)}$ of colors possibly used – must be disjoint from $S_v^{(j)}$, for $j < i$
- If $\sum_{v \in B} \left|S_v^{(i)}\right|$ is small, only very few inputs list colorable using colors from $\left(S_v^{(i)}\right)_{v \in B}$
- If algorithm has too few states, there must exist a next possible G_i value where $\sum_{v \in B} \left|S_v^{(i)}\right| \geq \beta n$
- After $\Delta - 1$ iterations, not enough colors left for all possible G_Δ

12. Space/color tradeoffs

Combining multiple vertices into one super-vertex gives space/color tradeoff*

* This only works for multigraphs.

- $\tilde{O}\left(\Delta^2/s^2 + \Delta\right)$ colors for EA in $\tilde{O}(ns)$ space