DECENTRALIZED VERTEX COLORING

Overview

- We will discuss about two randomized algorithms
- The aim is to color vertices of a graph with maximum degree d in $d + 1$ colors
- Each vertex cannot be adjacent to another vertex with the same color

Part 1: Algorithm A

- Claim: this thing ends in expected $O(n \log d)$ recolorings ($\Delta = d$)
- Conflicted means that it has a neighbor which is of the same color as itself

Persistent Decentralized Coloring $(G = (V, E))$

1. Initially, every vertex v chooses a color $\chi_0(v)$ at random from $\{1, 2, \ldots, \Delta + 1\}$.
2. At each time t, a vertex v is chosen uniformly at random among all conflicted vertices.
3. While v is conflicted, it keeps changing to a random color in $\{1, 2, \ldots, \Delta + 1\}$.
4. Steps 2 and 3 repeat until there are no conflicted vertices.

Part 1: Algorithm A

- First thing we need to notice is that no vertices are being chosen in step 2 twice
- So, the algorithm is equivalent to, we pick a random permutation, and consider these vertices one by one

Part 1: Algorithm A

- Let's define some notations
- Let $adj(v)$ be the set of vertices adjacent to v
- Let p be the permutation that we consider the vertices in
- So, we want to prove that expected number of recolors $\mathrm{E_v}\left(E_p(No.\ of\ recolors\ of\ v)\right)$ over all vertices v and all permutations p is $O(\log d)$

Part 1: Algorithm A

- Let fix a vertex v and the permutation p
- Let B be the set of vertices that appear before v in $adj(v)$
- Let A be the set of vertices that appear after v in $adj(v)$
- Vertices in A have never been recolored yet
- Let K be the number of colors that appear in the initial coloring of vertices in A
- Then number of vertices that v cannot choose is $\leq |B| + K$

Part 1: Algorithm A

- Imagine a clique with vertices $A + B + \{v\}$
- The vertices are colored the same way they are colored in G, the original graph
- This new graph is $d + 1$ vertices
- Consider the same permutation we are considering just now
- Then the number of colors we cannot choose for v is exactly $|B| + K$
- So, we realize that expected number of recolors for vertex v in this new graph is not less than that in the original graph

Part 1: Algorithm A

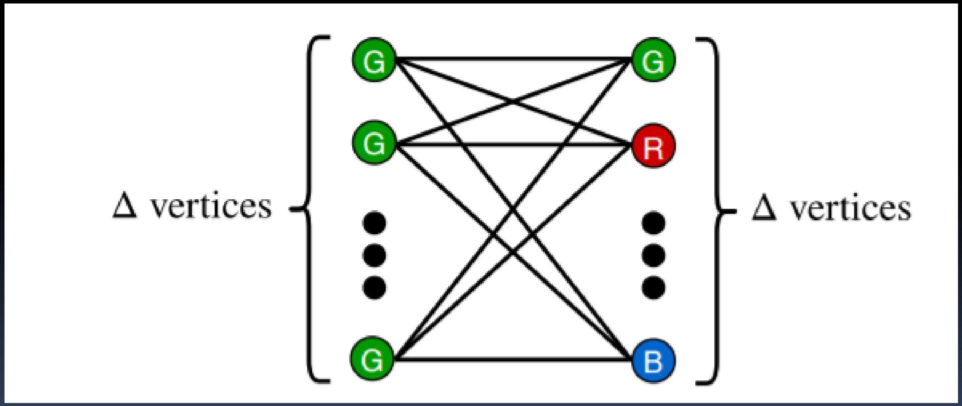
- In other words, a clique is the worst case of this problem
- we only have to prove that the expected number of colorings of a vertex in a clique of size $d + 1$ is $O(\log d)$
- As both the initial coloring and the order we color the vertices is random, the above statement is equivalent to, expected number of colorings of a vertex in a clique of size n is $O(\log n)$

Part 1: Algorithm A

- Imagine there are r different colors on the graph currently and we face a conflicted node
- Then, we will take expected $\frac{n}{n-r}$ recolorings until the node stop being conflicted, and the number of different colors on the graph becomes $r+1$
- Thus the expected number of recolorings is $\frac{n}{1}+\frac{n}{2}+\frac{n}{3}+\cdots+\frac{n}{n}=O(n\log n)$

Part 1: Algorithm A (extra)

- If the initial vertices are not being colored randomly, then we need $O(nd)$ recolorings
- Hack:



Part 2: Algorithm B

- Claim: this thing ends in expected $O(nd)$ recolorings ($\Delta = d$)

<u>Decentralized Coloring</u> $(G = (V, E))$

1. Initially, every vertex v chooses a color $\chi_0(v)$ ~~at random~~ from $\{1, 2, \ldots, \Delta + 1\}$.
2. At each time t, a vertex v is chosen ~~uniformly at random~~ among all conflicted vertices.
3. v changes its color to a random color in $\{1, 2, \ldots, \Delta + 1\}$.
4. Steps 2 and 3 repeat until there are no conflicted vertices.

Part 2: Algorithm B

- Different to Algorithm A in several ways
  - Step 1: Initial color not random
  - Step 2: Choice of conflicted vertices is not random
  - Step 3: We don't wait until the vertex becomes non-conflicted to move on to next iteration

Decentralized Coloring $(G = (V, E))$

1. Initially, every vertex v chooses a color $\chi_0(v)$ ~~at random~~ from $\{1, 2, \ldots, \Delta + 1\}$.
2. At each time t, a vertex v is chosen ~~uniformly at random~~ among all conflicted vertices.
3. v changes its color to a random color in $\{1, 2, \ldots, \Delta + 1\}$.
4. Steps 2 and 3 repeat until there are no conflicted vertices.

Part 2: Algorithm B

- We will use a different approach
- If we can define some potential function f($state$) such that
  - its expected value $\mathrm{E}(f(state))$ decreases by at least 1 after each recoloring when the state is not a valid coloring
  - $f(state) = 0$ iff the state is a valid coloring
- Answer is maximum possible value of $f(state)$

Part 2: Algorithm B

- Let $g(state)$ be the number of connected components of the same color in the graph
- Let $f(state) = (d+1) \times (n - g(state))$
- Then
  - $f(state) = 0$ iff $g(state) = n$ which means the coloring is valid
  - $f(state) \leq n(d+1)$
- It suffices to prove that $E(g(state))$ increases by at least $1/(d+1)$ after each recoloring

Part 2: Algorithm B

- Let's investigate how g changes over a recoloring
- Lets imagine v is the vertex being recolored
- Let c_v be then color of v before recoloring
- For each color c, imagine v is removed and let $p(c)$ be the number of components adjacent to v of color c
- If the new color is c_v', then let
  - $\mathrm{h}(\mathrm{c}_\mathrm{v}') = g(new\ state) - g(old\ state) = (1 - p(c_v')) - (1 - p(c_v)) = p(c_v) - p(c_v')$
- Very intuitive to understand this formula: the component of color c_v is broken while the components of color c_v' is merged into one.

Part 2: Algorithm B

- $p(c_v)$ is at least 1 because v is conflicted
- Sum of $p(c_v')$ is at most d because v has at most d neighbours
- So we have enough tools to analyse the change in $g(state)$ now
- $E\left(h(c_v')\right) = \frac{\sum_{c=1}^{d+1} p(c_v)-p(c)}{d+1} \geq \frac{(d+1)-d}{d+1} \geq \frac{1}{d+1}$
- So easy

Extra

- The authors conjectured that if we let step 1 and step 2 of Algorithm B be randomized, it takes $O(n \log d)$ recolorings only