6.5240 Sublinear Time Algorithms September 16, 2024

Lecture 4

Lecturer: Ronitt Rubinfeld *Scribe: Shashata Sawmya*

Today in the class we learned about a sublinear time algorithm for graph coloring.

1 Graph Coloring

Definition 1 *A "proper c-coloring" of graph $G = (V, E)$ assigns C_v from a "palette" of size c to each $v \in V$ such that $\forall (u, v) \in E$, $C_u \neq C_v$*

- Graph coloring is a NP-Hard problem in general
- There are some special classes of graphs where they are easier. For example: trees and bipartite graphs.
- One important case solvable in linear time is $c \geq \Delta + 1$ coloring when maximum degree of graph is Δ.

Algorithm 1 A greedy algorithm for finding a $(\Delta + 1)$-coloring

```
for each v ∈ V do
    Assign C_v different from all C_u for u ∈ N(v)
end for
```

Running time of Algorithm 1 is $O(m)$. But the question is can we do better than this?

Algorithm 2 GreedyListColoring

```
for each vertex v ∈ V do
    L(v) ← {1, ..., Δ + 1}
end for
for each vertex v ∈ V (in an arbitrary order) do
    if L(v) = ∅ then
        return fail
    else
        Assign any color c_v ∈ L(v) to v
        Remove c_v from L(u) for all neighbors u of v
    end if
end for
```

Let's analyze the running time of GreedyListColoring

$$\begin{aligned}\text{Running time} &= \sum_v \left(\text{time to find color } c_v \text{ in } L(v) + \sum_{u \in N_v} \text{time to remove color } c_v \text{ from } L(u) \right) \\ &= O(m)\end{aligned}$$

It looks like we are not improving. To make our lives a little bit easier, let's consider a graph query model for Graph $G = (V, E)$ that supports the following queries:

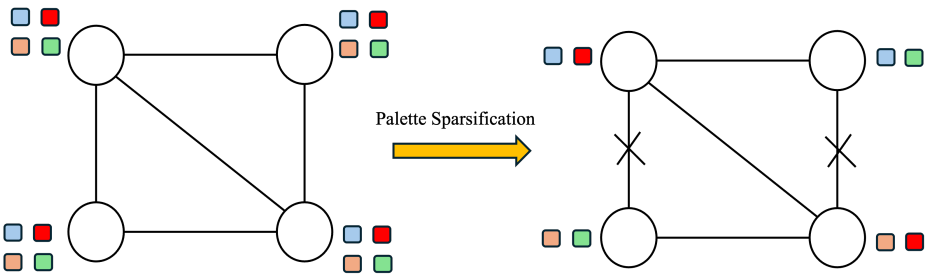

Figure 1: A simple example of palette sparsification

- **Degree queries:** Given $u \in V$, what is $\deg(u)$?
- **Pair queries:** Given $u, v \in V$, is it true that $(u, v) \in E$?
- **Neighbor queries:** Given $u \in V$ and $k \in \mathbb{N}$, what is the k-th neighbor of u?

2 Palette Sparsification

Till now, we haven't been able to improve the running time beyond $O(m)$. To alleviate this, we introduce a new technique known as Palette Sparsification.

Definition 2 *$\forall$ nodes v, sample $k = O(log\ n)$ colors from $\{1, 2, ..., \Delta + 1\}$ to get the list $L(v)$*

Here $L(v)$ is the sparser palette of vertex v. We state a lemma without providing a proof.

Lemma 3 *With high probability, a graph $G = (V, E)$ can be colored such that $c_v \in L(v)$ for all $v \in V$ via the GreedyListColoring algorithm.*

A theorem follows from this lemma with the addition of $\Delta \leq \sqrt{n}$ case. The purpose of introducing Palette Sparsification is to establish a framework that enables the proof of this theorem.

Theorem 4 *One can find a $(\Delta + 1)$-coloring of an n-vertex graph in $\tilde{O}(n\sqrt{n})$ time.*

Theorem 4 is the main result that we will prove soon. The key idea is that using palette sparsification, we can ignore some edges as they won't create coloring conflicts between the edge vertices. Figure 1 shows an example where, after sparsification, we can ignore the crossed-out edges as it won't cause other neighbors to lose colors from their palette. We throw out all edges (u, v) such that $L(v) \cap L(u) = \emptyset$. We will see shortly that G_{sparse} doesn't have a lot of edges. We define $E_{\text{sparse}} = \{(u, v) \in E : L(u) \cap L(v) = \emptyset\}$.

Claim 5 *With high probability, $|E_{sparse}| = O(n \log^2 n)$*

Proof We begin by fixing a vertex $u \in V$. Let $L(u) = \{1, 2, \ldots, k\}$ represent the palette of available colors for u. Now, for each vertex $v \in N(u)$ and each color $i \in \{1, 2, \ldots, k\}$, define an indicator variable $X_{v,i}$ as follows:

$$X_{v,i} = \begin{cases} 1, & \text{if } i \in L(v), \\ 0, & \text{otherwise.} \end{cases}$$

We now define the total number of edges that involve color i in the subgraph G_{sparse}. Let

$$X = \sum_{i=1}^{k} \sum_{v \in N(u)} X_{v,i},$$

where X represents the number of edges involving color i in the sparsified graph G_{sparse}. Since the inclusion of any edge in E_{sparse} depends on whether two adjacent vertices share a common color in their respective lists, X acts as an upper bound on $\deg(u)$ in G_{sparse}.

Next, we compute the expected value of $X_{v,i}$, which is the probability that color i is present in $L(v)$. Since $L(v)$ consists of k colors chosen uniformly from $\Delta + 1$ possible colors, we have:

$$\mathbb{E}[X_{v,i}] = \frac{k}{\Delta + 1}.$$

Now, by the linearity of expectation, the expected value of X becomes:

$$\mathbb{E}[X] = \sum_{i=1}^{k} \sum_{v \in N(u)} \mathbb{E}[X_{v,i}] = k \cdot \Delta \cdot \frac{k}{\Delta + 1}.$$

Since $\frac{k}{\Delta+1} \leq \frac{k}{\Delta}$, we obtain:

$$\mathbb{E}[X] \leq k^2.$$

Given that $k = \Theta(\log n)$, it follows that $\mathbb{E}[\deg(u)] \leq O(\log^2 n)$.

Finally, since the expected degree of each vertex u in the sparsified graph is $O(\log^2 n)$, and there are n vertices, the total number of edges in G_{sparse} is at most $O(n \log^2 n)$. Therefore, we conclude that $|E_{\text{sparse}}| = O(n \log^2 n)$ (One can show "with high probability" using Chernoff bounds). ■

3 A $\tilde{O}(\frac{n^2}{\Delta})$-time palette coloring algorithm

Assuming an earlier lemma we provided without proof (Lemma 3), we now give a sublinear time algorithm for fnding a $(\Delta + 1)$-coloring of a graph.

Algorithm 3 A sublinear time algorithm for $(\Delta + 1)$ graph coloring

Input: A graph $G = (V, E)$ with $n = |V|$ vertices and maximum degree Δ.
Output: A valid $(\Delta + 1)$-coloring of G.

1. **Construct the Palette for Each Vertex:**
   - For each vertex $v \in V$, sample $k = \Theta(\log n)$ colors from the palette $\{1, 2, \ldots, \Delta + 1\}$ to create a list $L(v)$ of available colors for vertex v.
2. **Construct the Sparse Subgraph $G_{\text{sparse}} = (V, E_{\text{sparse}})$:**
   - For each color $c \in \{1, 2, \ldots, \Delta + 1\}$, create the set $X_c = \{v \in V : c \in L(v)\}$, which consists of all vertices that have color c in their palette.
   - For each pair of vertices $v_i, v_j \in X_c$, where $v_i \neq v_j$:
     - Query if $(v_i, v_j) \in E$. If true, add the edge (v_i, v_j) to the edge set E_{sparse} of the sparse subgraph.
3. **Apply GreedyListColoring on the sparse subgraph G_{sparse}**

Step 1 in algorithm 3 requires $O(n\log n)$ time. From 5 we get the running time of step 3 is $O(n\log^2 n)$. Creating the set X_c for each color c requires $O(n\log n)$ time. Let us analyze the running

time of finding E_{sparse} in step 2. For each color c,

$$\mathbb{E}\left[\binom{X_c}{2}\right] \leq \mathbb{E}[\sum_{\substack{u,v\in V\\ u\neq v}} \mathbf{1}_{u,v \text{ both pick color } c}] = \binom{n}{2}\left(\frac{k}{\Delta+1}\right)^2 = O\left(\frac{n^2\log^2 n}{\Delta^2}\right).$$

Therefore, the running time to query all pairs in each X_c is at most,

$$(\Delta+1)\cdot O\left(\frac{n^2\log^2 n}{\Delta^2}\right) = \tilde{O}\left(\frac{n^2}{\Delta}\right)$$

This can be used to prove theorem 4. To prove this theorem, we will split the argument into two cases: small-degree and large-degree vertices. For small-degree vertices, the coloring will be straightforward due to the limited number of colors required. For large-degree ones, we will rely on Lemma 3 and Claim 5 to handle the list coloring and ensure that all vertices can be colored appropriately.

Proof We consider two cases based on the value of the maximum degree Δ in the graph $G = (V, E)$.

Case 1: If $\Delta \leq \sqrt{n}$, then we can directly use the *GreedyListColoring* algorithm. The time complexity of *GreedyListColoring* is bounded by $O(|E|)$. Since $|E| \leq n\Delta$, this gives a time complexity of:

$$O(|E|) \leq O(n\Delta).$$

Because $\Delta \leq \sqrt{n}$ in this case, we have:

$$O(n\Delta) \leq O(n\sqrt{n}),$$

Case 2: If $\Delta > \sqrt{n}$, we apply the palette sparsification technique. As discussed earlier, the time complexity for palette sparsification is $\tilde{O}\left(\frac{n^2}{\Delta}\right)$. Given that $\Delta > \sqrt{n}$, we have:

$$\tilde{O}\left(\frac{n^2}{\Delta}\right) \leq \tilde{O}\left(\frac{n^2}{\sqrt{n}}\right) = \tilde{O}(n^{3/2}),$$

Thus, the overall time complexity of the algorithm is $\tilde{O}(n^{3/2})$. ■

4 A weaker palette sparsification

We relax lemma 3 to allow choosing from 2Δ colors instead of $(\Delta+1)$ colors to show the "with high probability" part of lemma 3. A proof surely exists for the $(\Delta+1)$ case, but we relax the constraints to simplify the proof. Thus the updated palette sparsification problem is:

$\forall$ *nodes* v*, sample* $k = O(\log n)$ *colors from* $\{1, 2, ..., \Delta+1\}$ *to get the list* $L(v)$.

Note, other parts in the proof of Theorem 4 remain the same, hence giving an $\tilde{O}(n^{3/2})$ time algorithm for finding a 2Δ-coloring of an n-vertex graph.

Lemma 6 *With high probability, a graph* $G = (V, E)$ *can be colored via GreedyListColoring with* $\Delta+1$ *replaced by* 2Δ *colors such that* $c_v \in L(v)$ *for all* $v \in V$*.*

The key idea of proving Lemma 6 is to run the palette-coloring algorithm with palette size $s = 2\Delta$. Output fail if we run out of colors.

Proof Consider a vertex v and suppose that a color $c \in \{1, 2, \ldots, 2\Delta\}$ is *good* if $c \in L(v)$ and if c has not been assigned to any of v's previously colored neighbors. If the list $L(v)$ contains at least

one good color c, then v can be successfully colored. Since $L(v)$ is generated independently from the choices made for the neighbors of v, we can imagine choosing $L(v)$ "on the fly" as we process v.

Since each vertex v has at most Δ neighbors, we can compute the probability that none of the colors in $L(v)$ are good. This probability is given by:

$$\Pr[L(v) \text{ contains no good color}] \leq \frac{\binom{\Delta}{k}}{\binom{2\Delta}{k}} = \frac{\Delta(\Delta-1)\dots(\Delta-k+1)}{(2\Delta)(2\Delta-1)\dots(2\Delta-k+1)} < \left(\frac{1}{2}\right)^k.$$

Since $k = \Theta(\log n)$, we have:

$$\Pr[L(v) \text{ contains no good color}] < \left(\frac{1}{2^{\Theta(\log n)}}\right) = \frac{1}{n^\alpha}$$

for some constant α.

Using the union bound over $\forall$ v $\in V$, we now calculate the probability that there exists at least one vertex v such that $L(v)$ has no good color:

$$\Pr[\text{there exists a vertex } v \text{ such that } L(v) \text{ has no good color}] \leq \frac{1}{n^{\alpha'}},$$

for some constant α' (e.g., $\alpha' = 3$). Hence, with high probability, no vertex will fail to find a valid color from its list. It follows that G will have a proper list coloring with high probability. ■