6.8S99 Distributed Graph Algorithms September 5, 2014

Lecture 1

Lecturer: Mohsen Ghaffari *Scribe: Mira Radeva*

1 Main Topics and Models

The two main concepts covered in the class will be *locality* and *congestion*.

First, we describe a basic abstract model that we use in studying these two concepts.

The Model:

- the network is represented as a graph $G = (V, E)$, where $|V| = n$
- each node knows its neighbors
- the execution of an algorithm proceeds in synchronous rounds where in each round each node sends a message to each of its neighbors
- each node may know some approximation of a global property of G; for instance, nodes might know an approximation of n.

In the first half of the course, we assume that the message sizes are unbounded, and thus, each node can send all the information that it has to its neighbors. This model is called the *LOCAL* model. In the second half of the course, we assume that the each message size has at most B-bits, and where typically one assumes $B = O(\log n)$. The model with this restriction is called the *CONGEST* model.

2 On Locality in Graph Problems

In the rest of this lecture, and generally the first half of the course, our focus will be on the issue of *locality*. We start with a simple question:

- With how many colors can we color a graph within k rounds? Note that in k rounds, each node learns information only of its neighborhood of size k. Thus, a rephrasing of the question is, with how many colors can we color a graph if all that each node knows is only its k-neighborhood.

Problem Statement for Coloring: Each node $v \in V$ needs to be assigned a color $c_v \in \{1, \cdots, c\}$ such that for all edges $e = (u, v) \in E$, $c_v \neq c_u$.

For this lecture, we assume $\Delta = \Theta(1)$, where Δ is the maximum degree in the graph, and we seek coloring with $\Theta(1)$-many colors. We cover two main results:

1. **Cole-Vishkin [86]**: $\Theta(1)$-coloring for graphs with $\Delta = \Theta(1)$, in $\log^* n + O(1)$ rounds [1].
2. **Linial [89]**: $1/2 \log^* n - 1$ rounds are necessary.

3 The Cole-Vishkin Algorithm

3.1 Main Algorithm Idea

We start with a simpler setting where G is an arbitrary rooted tree, in which each node knows its parent, and each node has a unique ID $\in \{1, \cdots, n\}$. Note that the node ids induce a coloring with palette size n.

Outline The objective is to reduce the size of the palette from n to 3 in $\log^* n + O(1)$ rounds. We will reduce the size of the palette exponentially in each round, which will result in a running time of approximately $\log^* n$ to go from palette size n to a constant palette size. After that, we will use extra $\Theta(1)$ rounds to get to a 3-coloring.

Remark: Any tree can be colored in 2 colors. However, finding a 2-coloring distributedly might take $\Omega(n)$ rounds (see intuition for this at the end of the section).

Notation: Let c_v^i denote the color of node v after round i.

Algorithm: Each round of the algorithm is as follows:

- (Rule for non-root) if $v \neq r$, $c_v^{i+1} = (\ell, b_\ell)$, where ℓ and b_ℓ are defined as follows:
  - ℓ is the index[2] of the first bit in the binary representation of the color of v that differs from v's parent's color.
  - b_ℓ is the value of v's bit at index ℓ
- (Rule for root) if $v = r$, pick an arbitrary index ℓ, let b_ℓ be v's bit at index ℓ, and set $c_v^{i+1} = (\ell, b_\ell)$.

[1] Definition of $\log^* n$: $\log^* n = 0$ for $n \leq 1$, and $\log^* n = 1 + \log^*(\log n)$ for $n > 1$

[2] The direction from which the bits are counted does not matter, as long as it is consistent throughout the algorithm.

Lemma 1 *If the old coloring c_i is good, so is the new coloring c_{i+1}.*

Proof (Sketch) Consider two neighbors v and w; WLOG, assume w is the parent of v. Let $c_v^{i+1} = (x_v, y_v)$ and $c_w^{i+1} = (x_w, y_w)$. Note that x_v and x_w exist because c^i was a legal coloring. If $x_v \neq x_w$, the new coloring is good. Otherwise, by the choice of x_v, which is the index of the bit in which c_v^i and c_w^i differ, it must be true that $y_v \neq y_w$. That completes the proof. ■

Lemma 2 *If the old coloring uses m bits, then the new coloring can be represented using $\lceil \log m \rceil + 1$ bits.*

Therefore, applying this lemma iteratively, it is easy to see that the algorithm results in a $\Theta(1)$-coloring in $\log^* n + O(1)$ rounds.

3.2 Reducing the number of colors from $\Theta(1)$ to 3

We will show how to get from $c = \Theta(1) \geq 4$-coloring to a $c-1$ coloring in 2 rounds. In particular, we show how to get rid of color number c:

- (Round 1):
  - if $v = r$, $c_v^{i+1} \in \{1, 2, 3\}$ such that $c_v^i \neq c_v^{i+1}$.
  - if $v \neq r$, v adopts the color of its parent.
- (Round 2): if $c_v^i = c$, $c_v^{i+1} = \{1, 2, 3\}$, such that c_v^{i+1} is different from the color of all neighbors of v.

From the algorithm, we can see that after round 1, each node has neighbors colored in at most two colors. So, in round 2, a node has at least one of the colors $\{1, 2, 3\}$ to pick that is different from its neighbors' colors.

Therefore, we can get from a palette of size $c = \Theta(1)$ to a palette of size 3 in a constant number of rounds.

Note on 2-coloring for trees: As mentioned earlier, a tree can always be colored in 2 colors (alternate colors between layers of the tree). Consider a tree which is just a line of n nodes. Since colors need to alternate, it is necessary for the two ends of the line to communicate through a chain of messages in order to ensure correct coloring. Such a causal dependency necessitates a linear number of rounds.

3.3 Extension to General Graphs with $\Delta = \Theta(1)$

Idea: Treat each neighbor in the graph as a parent. Each node has at most $\Theta(1)$ parents.

The algorithm in Section 3.1 can be modified in such a way that a node's color is not a pair anymore, but a sequence of Δ fields as follows:

$((\ell_1, b_{\ell_1}), (\ell_3, b_{\ell_2}), \cdots, (\ell_\Delta, b_{\ell_\Delta}))$, where each field corresponds to a neighbor of the node (numbered arbitrarily), and the ℓ's and b_ℓ's are defined as before.

To show an equivalent of Lemma 1 above, consider two neighbors v and w (w is a parent of v) and repeat the same arguments as in Lemma 1 with respect to the field (ℓ_w, b_{ℓ_w}) of the coloring of v.

To show an equivalent of Lemma 2 above, note that each field in the coloring of a node is again $\lceil \log m \rceil + 1$ bits, where m the the number of bits required to encode the old coloring of the node. Therefore, the color of each node is at most $\Delta(\lceil \log m \rceil + 1)$ bits. This results in a dependency on Δ of the palette size (final palette size of $\Delta \log \Delta$ bits), which is OK because we assume $\Delta = \Theta(1)$.

Exercise for the Next Lecture: Think about how to get a coloring with a palette size of $O(\Delta^2 \log n)$ in 1 round?

4 Linial's Lower Bound

Consider a ring of n nodes, where each node can distinguish between its left and right neighbor. We want to show that any 3-coloring of the ring requires $1/2 \log^* n - 1$ rounds[3]. The proof described here is based on the version presented by Laurinharju and Suomela [Brief announcement: linial's lower bound made easy, PODC 2014], which is a streamlined version of Linial's argument.

Suppose an algorithm uses $T \ll n$ rounds to complete the coloring of the ring. After T rounds, each node v in the ring has information about nodes T hops away to the left and T hops away to the right. We can express this as a vector of node ID's: $(x_1, x_2, \cdots, x_{2T+1})$ (node v's own ID in in the middle of this vector). Therefore, the algorithm for each node can be thought of as a mapping: $(x_1, x_2, \cdots, x_{2T+1}) \to c \in \{1, 2, 3\}$.

Notation: A is a k-ary c-coloring if:

- (P1) For all $1 \leq x_1 < x_2 < \cdots < x_k \leq n$, $A(x_1, \cdots, x_k) \in \{1, \cdots, c\}$.
- (P2) For all $1 \leq x_1 < x_2 < \cdots < x_k < x_{k+1} \leq n$, $A(x_1, \cdots, x_k) \neq A(x_2, \cdots, x_{k+1})$.

We show that for any k-ary 3-coloring, $k \geq \log^* n - 1$.

Lemma 3 *For any 1-ary c-coloring, $c \geq n$.*

Lemma 4 *For any k-ary c-coloring A, there is a $(k-1)$-ary 2^c-coloring B*

Proof (Sketch) Given A, we need to define $B(x_1, \cdots, x_{k-1})$. Let

$$B(x_1, \cdots, x_{k-1}) = \{A(x_1, \cdots, x_k) | x_{k-1} < x_k \leq n\}.$$

[3]It is easy to see that the lower bound remains $1/2 \log^* n - O(1)$ for any coloring with constant colors.

Note that there are at most 2^c possible values for $B(x_1, \cdots, x_k)$, i.e., the number of the subsets of $\{1, \ldots, c\}$. This shows that the coloring B has palette size 2^c-coloring (property P1 of the definition above).

We now prove property P2. For the sake of contradiction, suppose that there exist $1 \leq x_1 < x_2 < \cdots < x_k \leq n$ such that $B(x_1, \cdots, x_{k-1}) = B(x_2, \cdots, x_k)$. Let $\alpha = A(x_1, \cdots, x_k)$. By the definition of B, we know that $\alpha \in B(x_1, \cdots, x_{k-1})$. Since $B(x_1, \cdots, x_{k-1}) = B(x_2, \cdots, x_k)$, it must be true that $\alpha \in B(x_2, \cdots, x_k)$. Therefore, because of the way we defined B, we know there exists $x_{k+1} > x_k$ such that $\alpha \in A(x_2, \cdots, x_{k+1})$. However, now we have $A(x_1, \cdots, x_k) = A(x_1, \cdots, x_{k+1})$, which is in contradiction with property P2 of the coloring A. This completes the proof. ■

Using Lemma 4 iteratively, we can build the following sequence: k-ary 3-coloring $\rightarrow$ $(k-1)$-ary 2^3-coloring $\rightarrow$ $(k-2)$-ary 2^{2^3}-coloring $\rightarrow \cdots \rightarrow$ 1-ary (tower of height $k+1$)-coloring. Now from Lemma 3, we know that tower of height $k+1$ of 2 must be at least n, which proves that $k \geq \log^* n - 1$.