Note Title

6.S899 Lecture 1 09/05/2014

Welcome to

Distributed Graph Algorithms

URL: http://people.csail.mit.edu/ghaffari/DGA14/

Instructors: Mohsen Ghaffari, Stephan Holzer & Nancy Lynch

Place : Room 4-145
Time : Fridays 11:00 — 12:30

Mailing List: email ghaffari@mit.edu to get added to the list.

SOME ADMINISTRATIVE COMMENTS:

- ☆ There will be 5 problem sets; one every other week. Each PSET will contain only one or two problems & the answers will be typically short; each should fit within one page.

- ☆ We will need a scribe note for each lecture; try to sign up for them as soon as possible.

- ☆ We will have course projects, details will be announced later on during the semester.

- ☆ Make sure you subscribe to the mailing list by sending an email to ghaffari@mit.edu.

- ☆ This is a gradute class, so there will be many details that will be left for the students to work them out.

GENERAL OUTLINE

The course will focus on two fundamental issues in distributed graph algorithms:

(1) **Locality**: How well can we solve a given problem if each node in the graph just knows the k-neighborhood of itself; that is, the topology of the graph induced on the nodes that are at most k hops away.

(2) **Congestion**: How fast can we solve a given problem when the communication rate between each two adjacent nodes is bounded to some limit.

The course will start with the study of the first topic, in problems such as graph coloring and maximal independent sets, and we will then continue to the second topic and visit problems such as minimum spanning tree, shortest path, spanners, etc. During the whole course, the emphasis will be on the techniques rather than the end results, and we will cover most of the basic techniques of the area.

MODEL

During the first half of the course where we are studying the "locality" issue, we will assume the following abstract model, tited "LOCAL": the time is divided into synchronous lock-step rounds $1, 2, 3, \ldots$, and during each round, each node can send a message of unbounded size to each of its neighbors. In the second half of the course, when study congestion, the model will be the same except that each message size will be bounded to B-bits, where typically $B = \Theta(\log n)$, n being the number of the nodes in the graph.

REMARK 1: For the first half where we use the local model, we can ignore thinking about round by round communications using the following observation: Any k-round algorithm in the LOCAL model can be viewed as each node first learns its k-neighborhood through communicating with neighbors for k rounds, and then each node solves the graph problem locally based on the topology of its k-neighborhood and without any further communication with other nodes. One can use a simple induction based on k to formally prove that this transformation is without loss of generality.

REMARK 2: For most of the course we will assume that the nodes have unbounded computational power and they can solve any computational problem, even NP problems, in each round. This allows us to focus on the limitations that are rooted in locality or congestion, and are not due to computational limitations. However, we note that almost all the algorithms that we see will use only polynomial-time computations.

REMARK 3: Unless noted otherwise, we will assume that each node has a unique identifier (id) in $\{1, 2, \ldots, n^{\Theta(1)}\}$ and that different nodes have distinct ids.

DISTRIBUTED GRAPH COLORING

DEFINITION: Given an undirected graph $G=(V,E)$, a k-color coloring assigns a color $c_v \in \{1,2,\cdots,k\}$ to each node $v \in V$ such that for each two adjacent nodes v & u, we have $c_v \neq c_u$. That is, neighboring nodes get different colors.

Following remark 1 above, a k-coloring distributed graph algorithm in the LOCAL model with round complexity T is a (potentially probabilistic) function that maps k-neighborhoods to $\{1,2,\cdots,k\}$. That is, for each node $v \in V$, this function receives the topology of the k-neighborhood of v as input and it outputs a color $c_v \in \{1,2,\cdots,k\}$.

The question that we want to understand is as follows:

What is the best coloring — the one with the smallest pallete size k — that we can achieve using T-round distributed algorithms.

The answer will of course depend on the graph & our goal is to characterize its dependency on the basic parameters of the graph like the number of nodes $n = |V|$ and the maximum degree in the graph Δ.

For this lecture, we will start with a simple setting: $\Theta(1)$-coloring of graphs with $\Delta = \Theta(1)$. We first see a neat trick due to Cole & Vishkin that solves the problem in $O(\log^* n)$ rounds, and then we study an even nicer argument due to Linial that shows this bound to be optimal!

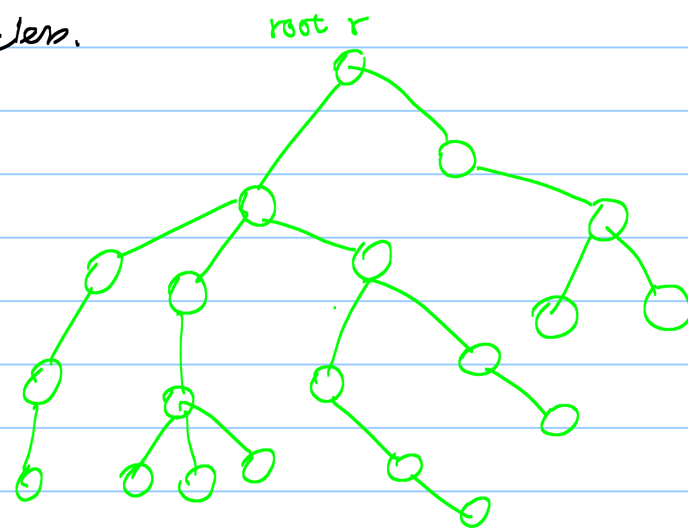
Recall that $\log^* n$ determines how many times do we need to take logs iteratively till we get a number ≤ 1. Formally

$$\log^* n = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + \log^*(\log n) & \text{if } n > 1 \end{cases}$$

COLE-VISHKIN [86]

"Deterministic coin tossing with applications to optimal parallel list ranking"

To explain the technique, it is most instructive to start from a simple setting, coloring of a rooted tree, where each node knows its parent and its children. Note that here Δ can be arbitrarily large but the technique will work nevertheless.



Note : the graph clearly has a 2-coloring; just color the nodes of odd levels 1 (blue) & the nodes of even levels 2 (red). Here, level means distance from the root r.

Cole-Vishkin gives us a $\log^* n + O(1)$ round 3-coloring algorithm. We start with the main part, that is a $\log^* n$ round 6-coloring, and then we see how to reduce the number of colors to 3 by a simple color-reduction procedure.

Let us use the notation c_v^i to denote the color of node v after i rounds.

For the initial coloring, for each node $v \in V$, we let $c_v^0 = id_v$, that is, the starting colors are the ids of the nodes.

To define c_v^{i+1}, we do as follows:

- If v is the root, pick $c_v^{i+1} \in \{1,2\}$ such that $c_v^{i+1} \neq c_v^i$.
- If v is not the root: suppose v's parent is node w. Let ℓ be the index of the most significant bit in which c_v^i & c_w^i differ & let b be the ℓ-th bit of the id_v.

  Set $c_v^{i+1} = (\ell, b)$.

Lem 1: If c^i is a (legal) coloring, so is c^{i+1}.

Lem 2: $\forall v \in V$, c_v^0 has $\lceil \log n \rceil$ bits, c_v^1 has $\lceil \log \log n \rceil + 1$ bits, c_v^2 has $\lceil \log(\lceil \log \log n \rceil + 1) \rceil + 1$ bits, & so on.

Corollary: For $i^* = \log^* n$, $\forall v \in V$, $c_v^{i^*}$ has 3 bits. Thus, c^{i^*+1} is a 6-coloring.

Color Reduction:

We know see how to transform the 6-coloring to a 3-coloring, in $O(1)$ rounds. We first remove color 6 in 2 rounds, then color 5 in 2 rounds, and then color 4 and end up with colors just $\{1,2,3\}$.

To remove color 6, in the first round, the root picks a color in $\{1,2,3\}$ different than its previous color and each non-root adopts the color of its parent. Then, the children of each node have the same color, so the neighbors of each node occupy at most 2 colors. Each color-6 node picks a color in $\{1,2,3\}$ that is not occupied by its neighbors.

We then remove colors 5 & 4 using the same trick.

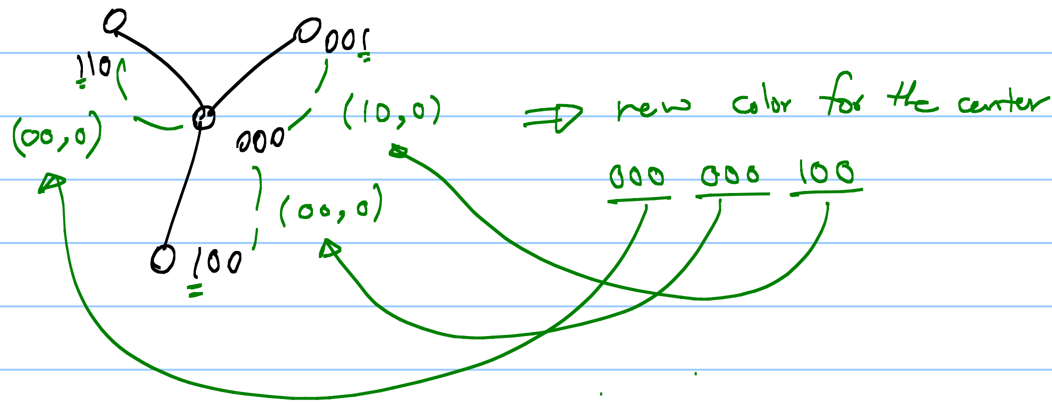
This gets us to a 3-coloring in $\log^* n + O(1)$ rounds

Excercise: Prove that 2-coloring the tree can require $\Omega(n)$ rounds.

Extension to graphs with $\Delta = \Theta(1)$

Simply think of each neighbor as one parent & concatenate the indices & bits of difference with them.

Example



Lem 1: The new coloring is legal.

Proof: For neighbors v & u where u is the j-th neighbor of v, consider the j-th field of the new color of v & that of u. If the difference indices are the same, the bits at those indices will be different (why?).

Lem 2: If in one round the number of bits used to represent the colors is m, after the next round, the number is at most $\Delta(\lceil \log m \rceil + 1)$.

Corollary: After $\log^* n + O(1)$ rounds, we get a $\Delta^{O(\Delta)} = \Theta(1)$ coloring.

REMARK: The current dependency on Δ is terrible.

In the next lecture, we will see much better colorings.

As a fun exercise for that lecture, think about how to get an $O(\Delta^2 \log n)$ coloring in one round.

LINIAL's $\log^* n$ LOWER BOUND [84]

Note: We discuss the simplified exposition due to Laurinharju & Suomela [2014], although at it's heart, it is the same argument as that of Linial.

Consider a cycle of n nodes and suppose that nodes have ids 1 to n. We show that any 3-coloring distributed algorithm must use at least $\frac{1}{2}\log^* n - 1$ rounds. We here describe the argument for deterministic algorithms. Naor [92] shows how to extend this argument to randomized algorithms as well.

Suppose that we have a T-round distributed algorithm on the cycle, for $T \ll n$. Note that in T rounds, what each node sees is essentially a path of length $2T+1$ centered on itself. Thus, after T rounds of communication, what a node learns is the ids of the $2T+1$ nodes that are in its T-neighborhood and their ordering. After that, the node has to output a color in $\{1,2,3\}$. Hence, any T-round algorithm is a mapping from vectors of $2T+1$ distinct ids to $\{1,2,3\}$.

Now note that if two nodes are neighbors, their T-neighborhood vectors overlap in $2T$ elements, and differ on the last elements on the two sides. That is, one sees $x_1, x_2, \dots, x_{2T+1}$ & the other sees $x_2, \dots, x_{2T+2}$. For a legal coloring these two vectors thus have to be mapped to different colors. Hence, if we use the notation $A(x_1, \dots, x_{2T+1})$ to indicate the coloring assigned to the node when it sees the id-vector $x_1, \dots, x_{2T+1}$, we must have

(1) $A(x_1, \dots, x_{2T+1}) \in \{1, 2, 3\}$

(2) $A(x_1, \dots, x_{2T+1}) \neq A(x_2, \dots, x_{2T+2})$

Let us say A is a k-ary c-coloring if the following two conditions hold:

(P_1) $\forall\, 1 \leq x_1 < x_2 < \dots < x_k \leq n$, $A(x_1, \dots, x_k) \in \{1, 2, \dots, c\}$

(P_2) $\forall\, 1 \leq x_1 < x_2 < \dots < x_{k+1} \leq n$, $A(x_1, \dots, x_k) \neq A(x_2, \dots, x_{k+1})$.

To prove the theorem, we show that for any k-ary 3-coloring, we must have $k \geq \log^* n - 1$.

Lem 1: For any 1-ary c-coloring, $c \geq n$.

Lem 2: If A is a k-ary c-coloring, then there exists a $(k-1)$-ary 2^c-coloring.

Note that Lem 2 intuitively says that we can trade the quality of a coloring for speed.

Proof of Lem 2:

Define $B(x_1, \ldots, x_{k-1}) = \{ A(x_1, \ldots, x_k) \mid x_k > x_{k-1} \}$

That is, the set of all colors given to all id-increasing extensions of vector $x_1, \ldots, x_{k-1}$, in the coloring A.

B is a 2^c coloring. To show that it has the property (P_2), for the sake of contradiction, suppose $\exists$ $1 \leq x_1 < x_2 < \cdots < x_k \leq n$ such that $B(x_1, \ldots, x_{k-1}) = B(x_2, \ldots, x_k)$. Let $\alpha = A(x_1, \ldots, x_k)$. Then, $\exists$ x_{k+1} such that $\alpha = A(x_2, \ldots, x_{k+1})$. But this is a contradiction $\times$.

Corollary: Applying Lem 2 recursively, we go from any K-ary 3-coloring to a 1-ary $2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$ ($K+1$ times).
Then applying Lem 1 shows that $k \geq \log^* n - 1$.

REMARK: This lower bound can also be presented as an application of Ramsey's theorem for hypergraphs. Simply consider all hyperedges of cardinality $k = 2T+1$. The algorithm will give a 3 coloring of these hyperedges. If $k = o(\log^* n)$, then there will be a monochromatic clique of $k+1$ nodes. That is, all the $K+1$ hyperedges formed by picking k nodes out of these $k+1$ nodes have the same color. Then, there is $(v_1, \ldots, v_k)$ and $(v_2, \ldots, v_{k+1})$ that have the same color. $\times$