Course 2.18.1: Distributed Algorithms for Networks LOCAL Algorithms

Mikaël Rabie

1 Creating Complexities

1.1 $2\frac{1}{2}$-coloring - Slides 4-5

We assume that all nodes know the size n of the graph. Either nodes are colored with color 1 or 2, or use color 3 to ignore the coloring restriction. There exists a $O(\sqrt{n})$ algorithm that solves the problem:

- In one round, detect if you are in V_2 or not.
- If you are on an appended path, spend $\sqrt{n}$ rounds to know if the path you are in is of length at most $\sqrt{n}$ or not:
  - If it is the case, choose your color as your distance to the end of the path modulo 2
  - Otherwise, take color 3.
- If you are in V_2, first check in $\sqrt{n}$ rounds if your appended path is of length at most $\sqrt{n}$:
  - If it is the case, take color 3.
  - Otherwise, spend $\sqrt{n}$ more rounds to detect the longest subpath in V_2 you belong to where all nodes are in the same situation as yours. This subpath cannot be longer than $\sqrt{n}$, otherwise you would see more than n nodes. 2-color this subpath.
    Note that all nodes you need to see are at distance at most $2\sqrt{n}$ from you.

This problem must have complexity $\Omega(\sqrt{n})$. Indeed, consider a path of V_2 of length $\sqrt{n}$, where all appended paths are of length $\sqrt{n}$. Either one of those subpaths is 2-colored, and it needed $\sqrt{n}$ rounds to do it, either none of them are 2-colored, and the V_2 path (of length $\sqrt{n}$) is 2-colored, needing $\sqrt{n}$ rounds.

For a better understanding of the generalization of the problem, see [2].

2 Maximal Matching - Slides 9-10

The algorithm works in three steps:

1. Partition the edges of the graph into Δ rooted forests:

- Each node order its edges increasingly in regards to their neighbor's identifiers. It gives colors 1 to the first one, 2 to the second...
- for each edge, orient the edge towards the bigger identifier, choosing the color selected by the corresponding node.

For each color from 1 to Δ, we have a rooted forest:

- A node cannot have two incoming edges of the same color: Each incoming edge has its color chosen by the node it points toward, and each node gave a different color to each of its edges in the first step.
- We cannot have a cycle, as each edge goes toward an higher identifier

2. We compute a 3-coloring of each forest in $O(\log^* n)$ rounds. This can be done by applying Cole-Vishkin's algorithm with your parent in $\log^* n$ rounds, which leads to a 6-coloring. To remove a color, first each node takes the color of its parent (the root chooses a color different from its previous one). This step ensures that each node in the forest sees at most two different colors. Nodes of maximal color can always choose 1, 2 or 3. Repeating this process 3 times ensures that we reach a 3-coloring.

   Note that each node can do this computing for the Δ forests at the same time in parallel.

3. For Forest $f = 1$ to Δ do:

   - For Color $c = 1$ to 3 do:
     - For all node, add if possible an edge connecting it to one of its children of Color c in Forest f.

   We are sure that in the 3rd loop, no pairs of adjacent edges are added at the same time, as they are of the form "parent of color c to one of its children". Each node has at most one parent, and if you choose one of your children, as it has a different color from yours, it is not choosing itself one of its edges at the same time. Hence, a matching is computed.

   Moreover, the matching is maximal. Indeed, assume that the matching is not maximal, i.e. an edge uv (u being the parent) could have been added. The parent u was considered at some point in the loop. Either it chose another child (meaning that u is already matched), or it must have added uv, which is also a contradiction.

3 3-coloring Trees

3.1 Tree Shattering - Slides 12-13

The goal is to compute an MIS such that, after its removal, each connected component is of size 1 or 2. Just computing an MIS in trees is not helpful. For example, if you are given a tree T, and add a leaf pending to each node of T, selecting those pending nodes form an MIS, and its removal gives you back the initial tree.

The process used is called Rake and Compress:

1. $i = 0$

2. While we still have an unlabeled node:

- Identify leafs (i.e. nodes of degree 1) and nodes in induced path of length at least 3 (i.e. at least 3 consecutive nodes of degree 2 or less)
- Give them label i
- $i++$

One can prove that this process uses $O(\log n)$ steps (see [1] for a proof). We have the following properties on each level:

- Each node of level i has at most two neighbors of level $\geq i$.
- Each node has at most one neighbor of higher level.
- Each connected component of nodes in the same level is either a single node (with at most one higher level neighbor), or a path of length at least 3.

Let max be the maximal level. We build the MIS as follows:

1. For each level, in parallel, compute a 3-coloring (in $O(\log^* n)$ rounds).
2. $I = \emptyset$
3. From level $i = max$ to 0, for each node u of level i:
    - If u has a neighbor of higher level that is not in I, add u to I
    - For $c = 1$ to 3:
        - if u has color c and has no neighbor in I, add u to I

This process has complexity $O(\log^* n + \log n)$. We have computed an MIS, as each node was considered to be added sequentially. Moreover, the MIS has the following properties:

- for each edge uv, if u and v have different levels, $u \in I$ or $v \in I$.
- For each paths of length at least 3 with the same level, the nodes in I from this path form an MIS of that path.

This allows us to deduce that the MIS we computed as the desired properties.

From this MIS, computing the 3-coloring is direct. Nodes in I take color 1, isolated remaining nodes take color 2, and components of size 2 after the removal of I take color 2 and 3 (for example, 2 for the node of smallest identifier and 3 for the other one).

References

[1] Marthe Bonamy, Paul Ouvrard, Mikaël Rabie, Jukka Suomela, and Jara Uitto. Distributed recoloring. *arXiv preprint arXiv:1802.06742*, 2018.

[2] Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the local model. *SIAM Journal on Computing*, 48(1):33–69, 2019.