CS-E3190 Principles of Algorithmic Techniques

10. Distributed Algorithms – Tutorial Exercise

We consider the LOCAL model of distributed computing. We are given a graph $G = (V, E)$, with $|V| = n$, and $|E| = m$ that corresponds to both the input graph of our problems and the communication network. Each node corresponds to a computational unit and the units are able to send messages to each other if there is an edge between them. We assume that

1. Each node is assigned a unique identifier from set $\{1, 2, \ldots, n^c\}$ for a constant c.
2. We have synchronous communication rounds.
3. We have unlimited message sizes and unlimited computation per node.

Each node must announce their own part of the solution and once the solution is announced, it cannot be revoked. The complexity of an algorithm is the number of communication rounds until every node has announced its output.

1. **Coloring with $O(\Delta)$ colors.** Give a randomized distributed coloring algorithm that runs in $O(\log n)$ rounds with high probability and is allowed to use $O(\Delta)$ colors.

   Solution. Every node has at most Δ neighbors. Say node i is *bad* if it has the same color as at least one of its neighbors under some assignment of colors. We say a color is *free* for i if no neighbor of i uses this color. Fix some integer $K \geq 2$.

   Algorithm 1: Randomized coloring

   ```
   Each node selects one of (K + 1)Δ colors uniformly at random;
   Each node communicates its color to its neighbors;
   while node i is bad do
       Change i's color to one of i's free colors uniformly at random;
       Communicate i's color to neighbors;
   end
   ```

   Correctness. The algorithm terminates only when all nodes have different colors than their neighbors, so a valid $((K + 1)\Delta)$-coloring is always returned.

   Runtime. One can use the union bound to prove the runtime with high probability. Note that a node that is not bad will never become bad again, because while its neighbors may change color they only use free colors. Consider a node i. The probability of i remaining bad after one round depends on the colors of i and its neighbors. The probability of i being bad after one round can be bounded by

   $$\mathrm{P}[i \text{ is bad after one round}] \leq \Delta \mathrm{P}[i \text{ conflicts with j}] \leq \Delta \frac{1}{K\Delta} = \frac{1}{K}.$$

   The first inequality follows from the union bound; j denotes some neighbor of i. The second bound follows from realizing that the two nodes i and j have at least $K\Delta$ free colors each, which in the worst case are entirely overlapping.

   The above bound can be used to bound the runtime with high probability. Fix another integer $c \geq 2$. Let $B(i)$ denote the event that node i is bad after $c \log_K(n)$ iterations.

Then the probability that the algorithm has not terminated by $c \log_K(n)$ iterations is bounded by

$$\begin{aligned}\mathbf{P}[\text{at least one bad node after } c\log_K n \text{ iters}] &= \mathbf{P}[\bigcup_{i \in V} B(i)] \\ &\leq \sum_{i \in V} \mathbf{P}[B(i)] \\ &\leq n \left(\frac{1}{K}\right)^{c \log_K n} = n^{-c+1} \leq \frac{1}{n^{c-1}},\end{aligned}$$

for a $c \geq 2$ that we chose freely. This proves the algorithm terminates within $O(\log n)$ rounds with high probability.

2. **Coloring trees.** Suppose that your input graph G is a tree. Use the following fact to argue that there cannot exist a deterministic distributed algorithm that colors a tree with $o(\Delta/\log\Delta)$ colors in $o(\log_\Delta n)$ rounds.

   Fact 1 (Bollobás '78)**.** *There is an infinite family of n-node graphs with girth $\Omega(\log_\Delta n)$ and chromatic number $\Omega(\Delta/\log\Delta)$.*

   Hint: Suppose such an algorithm existed and arrive at a contradiction.

   Solution. Let $G = (V, E)$ be an instance of the family of n-node graphs with girth $\Omega(\log_\Delta n)$ and chromatic number $\Omega(\Delta/\log\Delta)$. A graph with girth g has no cycles of fewer than g edges. Consider an arbitrary node i and its neighbors $N^1(i)$. Let $N^2(i)$ be the neighbors of i and their neighbors, and so on. Then, any neighborhood $N^k(i)$ with $k < g$ on graph G is a tree.

   Suppose that there exists a deterministic distributed algorithm that colors a tree with $o(\Delta/\log\Delta)$ colors in time $o(\log_\Delta n)$ time. The runtime of $t = o(\log_\Delta n)$ means that every node i is colored using only local information from at most $N^t(i)$. In fact, the algorithm is a mapping from $N^t(i)$ neighborhoods to colors. Consider running the algorithm on our graph G. The algorithm only considers neighborhoods $N^t(i)$ with t less than the girth of the graph. As a result, the algorithm runs on sub-graphs that are trees, and hence returns a proper $o(\Delta/\log\Delta)$-coloring for the overall graph.

   This is a contradiction: graph G supposedly has chromatic number $\Omega(\Delta/\log\Delta)$ but the existence of the algorithm would imply that graph G has a $o(\Delta/\log\Delta)$-coloring.

3. **Luby's with high probability.** Show that the MIS algorithm from the lecture works with high probability in $O(\log n)$ rounds. In this task, you can omit the correctness proof.

   Solution. Recall that from the lectures we know that R number of edges are removed in a phase and

   $$\mathbf{E}[R] \geq m_i/72,$$

   where m_i is the number of edges in the graph in the beginning of phase i. The equation above can be restated as

   $$\mathbf{E}[m_{i+1}] \leq \frac{71}{72} m_i.$$

   After $k := (c+2)\log n$ phases, where c is some constant, it holds that

   $$\mathbf{E}[m_k] \leq \left(\frac{71}{72}\right)^k m = \left(\frac{1}{72/71}\right)^k m = \frac{m}{n^{c+2}} \leq \frac{1}{n^c},$$

where $m \leq n^2$ is the number of edges in the original input graph. Applying Markov's inequality for the random variable m_k, i.e., the number of edges left after k phases, yields

$$\mathrm{P}[m_k \geq a] \leq \frac{\mathrm{E}[m_k]}{a}.$$

By choosing $a = 1$, we arrive at

$$\mathrm{P}[m_k \geq 1] \leq \frac{1}{n^c}$$
$$\mathrm{P}[m_k = 0] \geq 1 - \frac{1}{n^c},$$

which reads that after k phases, the graph is empty with high probability.

4. **Challenge Problem: Coloring trees with 3 colors.** Design a distributed algorithm that finds a 3-coloring in time $O(\log n)$ on trees. This is a brain teaser; no model solutions will be provided. If you have an idea on how to solve it and want feedback, send your solution to [rustam.latypov@aalto.fi].