University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
Z. Parsaeian, G. Schmid

Distributed Graph Algorithms
Sample Solution Exercise Sheet 2

Exercise 1: Defective Edge Coloring *(10 Points)*

Give a distributed algorithm that computes a d-defective $O((\Delta/d)^2)$-edge coloring in 1 communication round, for any $d \geq 2$.

- Describe the algorithm (e.g., what does each node or edge do to select a color?).
- Analyze the algorithm: prove that the resulting coloring is d-defective and uses $O((\Delta/d)^2)$ colors.

Sample Solution

We design a one-round distributed algorithm as follows. Each edge is conceptually split into two *half-edges*, one incident to each endpoint. Every vertex v independently colors each of its incident half-edges with a color chosen from a palette of $4 \cdot \Delta/d$ colors, such that each color is used on at most $d/4$ of its incident half-edges. (This can be done deterministically or by using a simple randomized scheme with high probability.)
Now, consider an edge $e = \{u, v\}$. Its final color is defined as the unordered pair $\{c_u(e), c_v(e)\}$, where $c_u(e)$ and $c_v(e)$ are the colors chosen by u and v for their respective half-edges of e. Thus, the total number of possible colors is

$$O\big((\Delta/d)^2\big).$$

Defect analysis. At each endpoint v, each half-edge color appears at most $d/4$ times. Therefore, for any edge $e = \{u, v\}$ with color $\{a, b\}$:

- e has at most $d/4$ adjacent edges at u that share the same first component a, and
- at most $d/4$ adjacent edges at v that share the same second component b.

Hence, each edge has at most $d/2 + d/2 = d$ adjacent edges of the same final color. Therefore, the resulting coloring is a d-defective edge coloring.
Summary. This algorithm completes in one communication round, produces a d-defective coloring, and uses

$$O\big((\Delta/d)^2\big)$$

colors.

Exercise 2: Coloring Oriented Graphs *(10 Points)*

Let $G = (V, E)$ be an oriented graph where each node has an out-degree of at most β.

- Describe a distributed algorithm to compute a proper vertex coloring of G with $O(\beta^2)$ colors.
- Prove that your algorithm is correct (i.e., the coloring is proper).
- What is the round complexity of your algorithm?

Sample Solution

The key insight is that a node only needs to avoid conflicts with its *out-neighbors*. We can therefore use a standard deterministic coloring algorithm, but define the "conflict graph" based on out-degree.

Algorithm Description

The algorithm is to simply run the well-known Linial's $O(\log^* n)$-round deterministic vertex coloring algorithm.
However, we must define what constitutes a "conflict" for this algorithm:

1. **Conflict Definition:** When a node v runs the algorithm, it only considers its set of **out-neighbors**, $N_{\text{out}}(v)$, as its conflict set.
2. **Algorithm Guarantee:** Linial's algorithm is guaranteed to produce a proper coloring C such that $C(v) \neq C(u)$ for all u in v's conflict set.
3. **Color Count:** The algorithm guarantees a coloring with $O(k^2)$ colors, where k is the maximum size of any node's conflict set.

In our problem, the maximum size of the conflict set is $k = \max_{v \in V} |N_{\text{out}}(v)|$. The problem states this is at most β. Therefore, the algorithm produces a coloring with $O(\beta^2)$ colors.

Proof of Correctness (Proper Coloring)

We must prove that for any directed edge $(u, v) \in E$, the resulting coloring C satisfies $C(u) \neq C(v)$.

1. Let $(u, v) \in E$ be an arbitrary directed edge in G.
2. By the definition of a directed edge, v is an out-neighbor of u.
3. Therefore, $v \in N_{\text{out}}(u)$.
4. Our algorithm (Step 1) defines the conflict set for node u to be $S_u = N_{\text{out}}(u)$.
5. Linial's algorithm (Step 2) guarantees that node u will select a color $C(u)$ that is different from the color of all nodes in its conflict set S_u.
6. Since $v \in S_u$, it is guaranteed that $C(u) \neq C(v)$.
7. As this holds for any arbitrary edge, the coloring is proper.

Round Complexity

The standard Linial's deterministic coloring algorithm, which we use as a "black box," is known to run in $O(\log^* n)$ rounds.