Principles of Distributed Computing 03/14, 2018

Exercise 4

Lecturer: Mohsen Ghaffari

1 Color Reduction in Vertex-Coloring

Exercise

(1a) Design a single-round algorithm that transforms any given k-coloring of a graph with maximum degree Δ into a k'-coloring for $k' = k - \lceil \frac{k}{2(\Delta+1)} \rceil$, assuming $k' \geq \Delta + 1$.

We assume $k \geq \Delta + 2$ (as otherwise we cannot in general reduce the number of colors). We put the colors of the given k-coloring into $\lfloor \frac{k}{\Delta+2} \rfloor$ buckets, each of size $\Delta + 2$ except for the last one which may have size between $\Delta + 2$ to $2\Delta + 3$. Within each bucket we can in one round reduce the number of colors by 1, using the method of Lemma 5. Since this can be done for all buckets in parallel, in total the number of colors can be reduced by $\lfloor \frac{k}{\Delta+2} \rfloor \geq \lceil \frac{k}{2(\Delta+1)} \rceil$ in 1 round.

(1b) Use repetitions of this single-round algorithm, in combination with the $O(\log^* n)$-round $O(\Delta^2 \log \Delta)$-vertex-coloring that we saw in class, to obtain an $O(\Delta \log \Delta + \log^* n)$-round $(\Delta + 1)$-coloring algorithm.

First, we run the $O(\log^* n)$ algorithm that gives us a $(C\Delta^2 \log \Delta)$-coloring for some constant C. Then we repeat the algorithm from (a) as often as possible, that is, until $k \leq \Delta + 1$. Observe that the number of colors drops in each iteration by at least a factor $\frac{k'}{k} = \frac{1}{1-\frac{1}{2\Delta+4}}$. The number of iterations thus is bounded by

$$\log_{\frac{1}{1-\frac{1}{2\Delta+4}}} (C\Delta^2 \log \Delta) = \frac{\log(C\Delta^2 \log \Delta)}{\log(2\Delta+4) - \log(2\Delta+3)} \leq (2\Delta+4) \cdot \log(C\Delta^2 \log \Delta)$$

$$\leq (2\Delta+4) \cdot \log(C\Delta^2 \log \Delta) = O(\Delta) \cdot (O(\log \Delta) + \log\log \Delta) = O(\Delta \log \Delta),$$

where we use that $\frac{x}{y} \leq \log y - \log(y - x)$ for all real numbers $0 \leq x < y$.

2 SuperImposed Codes

Here, we use the concept of cover free families to obtain an encoding that allows us to recover information after superimposition. That is, we will be able to decode even if k of the codewords are *superimposed* and we only have the resulting bit-wise OR.

Exercise

(2a) Concretely, we want a function $Enc : \{0,1\}^{\log n} \to \{0,1\}^{\log m}$ — that encodes n possibilities using $\log m$-bit strings — such that the following property is satisfied: $\forall S \neq S' \subseteq \{1, ..., n\}$ such that $|S| \leq k$ and $|S'| \leq k$, we have that $\vee_{i \in S} Enc(i) \neq \vee_{i \in S'} Enc(i)$. Here $\vee$ denotes the bit-wise OR operation. Present such an encoding function, with a small m, that depends on n and k.

The solution to this exercise is obtained by bending the description of the task into the terms of Lemma 3 from the lecture notes. We use a $(2k-1)$-cover-free family for the encryption, where the bitstring of length m is interpreted as being a set where the i^{th} bit is 1 if element i

is in the set. Note that the bitwise OR is the same as the union of the corresponding sets. The encryption then satisfies $Enc(j) \not\subseteq \bigcup_{i \in S \cup S' \setminus \{j\}} Enc(i)$ for all $j \in [n]$ (since $|(S \cup S') \setminus \{j\}| = k - 1$), which implies $\bigcup_{i \in S} Enc(i) \neq \bigcup_{i \in S'} Enc(i)$. Applying Lemma 3, we get that there exists such a family with ground set of size $m = O((2k-1)^2 \log n) = O(k^2 \log n)$.