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

Distributed Graph Algorithms
Exercise Sheet 10

The following Theorem will be useful for your analysis during the exercises.
Useful Theorem (Chernoff Bound): Let $X_1, \ldots, X_n$ be independent Bernoulli random variables with $P(X_i = 1) = p_i$. Let $X = \sum_{i=1}^{n} X_i$ be their sum and $\mu = \mathbb{E}[X]$. Then for any $0 < \delta < 1$:

$$P(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2 \mu}{2}}$$

and

$$P(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2 \mu}{3}}$$

1 Maximal Matching in Near Linear Memory MPC

In this exercise, we want to find a randomized solution for the maximal matching problem. In the maximal matching problem, we aim to find a set M of edges that do not share the same endpoint, such that any other edge from the graph outside of this set, i.e., $E \setminus M$ shares an endpoint with at least one edge in M.

1. Given a graph $G = (V, E)$, let $E' \subseteq E$ be a set of edges of G chosen independently with probability p. Show that with probability at least $1 - e^{-n}$, for all subsets $I \subseteq V$, either $|E[I]| < 2n/p$ or $E[I] \cap E' \neq \emptyset$, where $E[I]$ denotes the set of edges in the subgraph induced by I.

2. **(Algorithm Design)** We will now design an algorithm for the MPC model where the available memory per machine is $s = n^{1+\epsilon}$ (for some constant $\epsilon > 0$). Let $M = \emptyset$ at the beginning of the algorithm, let $E_0 = E$ be the initial set of edges, and let E_t be the set of edges remaining in the graph after iteration t.

   The algorithm proceeds in iterations. In each iteration $t + 1$, we perform the following steps:

   (a) **Base Case:** If $|E_t| \leq s$, describe a simple $O(1)$ round MPC procedure to find a maximal matching on E_t.

   (b) **Sampling Step:** If $|E_t| > s$, we cannot follow the same procedure as step (a). We need to sample a subset of edges $E'_{t+1} \subseteq E_t$ first to proceed with. Define a sampling probability p in terms of s and $|E_t|$ such that the expected size of the sample fits within the memory of a single machine (i.e., $\mathbb{E}[|E'_{t+1}|] \leq s/10$).

   (c) **Filtering Step:** Suppose we compute a maximal matching M' on the sampled subgraph $G' = (V, E')$. To ensure that the final solution is a valid matching, we must remove all edges from the remaining graph that share an endpoint with any edge in M', to do so, we need to inform all the machines about the edges in M'. First prove show that M' can be stored on a single machine. Then, design an MPC procedure to broadcast the edges of M' to all machines storing parts of the graph, and prove that this filtering procedure can be implemented in $O(1)$ rounds.

   (d) **Convergence and Complexity Analysis:** Using the result from part (1), derive an upper bound on the size of the remaining edge set $|E_{t+1}|$ in terms of $|E_t|$, n, and s. *(Hint: Consider the induced subgraph of the unmatched vertices.)* Based on this recurrence, determine

the number of iterations k required to reduce the number of edges from an initial size of $|E_0| = n^{1+c}$ down to the base case size s. Finally, deduce the total round complexity of the maximal matching algorithm.

2 Maximal Independent Set in the Near-Linear Memory MPC

In this exercise, we analyze a sampling-based algorithm for the MIS problem in the MPC model. Let $G = (V, E)$ be a graph with n vertices and maximum degree Δ. Consider the MPC model where each machine has memory $s \geq \gamma \cdot n \log^2 n$ for some sufficiently large constant $\gamma > 0$.
Consider the following randomized procedure. **Sampling:** Let $U \subseteq V$ be a subset of vertices where each vertex of V is included independently in U with probability p. Let $k = |U|$ be the size of the sample. **Local MIS:** Let M_U be an MIS computed on the induced subgraph $G[U]$ using the Randomized Greedy algorithm[1]. **Pruning:** Let $V' \subseteq V$ be the set of vertices that are neither in M_U nor adjacent to any vertex in M_U. That is: $V' = V \setminus (M_U \cup N(M_U))$, where $N(M_U)$ denotes the set of neighbors of vertices in M_U.
Key Lemma: With high probability (w.h.p.), the maximum degree of the induced subgraph $G[V']$ satisfies the following:

$$\Delta(G[V']) \leq \frac{c \cdot n \log n}{k}$$

for some constant $c > 0$.

1. **Parameter Selection:** Our goal is to execute the **Local MIS** step on a single machine.
   (a) What is the expected number of edges in the induced subgraph $G[U]$ in terms of $|E|$ and p?
   (b) Determine the maximum value of the sampling probability p (in terms of s and $|E|$) such that the expected number of edges in $G[U]$ fits within the memory of one machine (i.e., $\mathbb{E}[|E(G[U])|] \leq s/10$).
2. **Degree Reduction Analysis:** We now analyze how much the maximum degree decreases after one iteration of this procedure.
   (a) Consider the worst-case density where $|E| \approx \frac{n\Delta}{2}$. Using the probability p you derived in part 1(b) and the memory lower bound $s \geq \gamma n \log^2 n$, calculate the expected size of the set U (i.e., $\mathbb{E}[k]$).
   (b) Prove that with high probability, the size of the sample is at least $k \geq \frac{c \cdot n \log n}{\sqrt{\Delta}}$ (assuming γ is sufficiently large).
   (c) Using the key Lemma and the result from part 2(b), show that the maximum degree of the remaining graph $G[V']$ satisfies:
   $$\Delta(G[V']) \leq \sqrt{\Delta}$$
3. **Algorithm & Complexity:** Based on the analysis above, design a recursive MPC algorithm to compute the MIS of the whole graph G.
   (a) State the algorithm steps, including the base case when the graph fits entirely in the memory of one machine.
   (b) Based on the degree reduction factor derived in part 2(c), how many iterations (phases) are required to reduce the maximum degree to a constant size?
   (c) Conclude that the algorithm runs in $O(\log \log \Delta)$ MPC rounds.

[1] Choose a random permutation π of the vertices V, effectively assigning each vertex v a unique random rank $r(v) \in \{1, \ldots, n\}$. Initialize the independent set $M = \emptyset$. Process the vertices in the increasing order of their ranks. For each vertex v, add v to M if and only if none of its neighbors $u \in N(v)$ with lower rank ($r(u) < r(v)$) are already in M.