Principles of Distributed Computing	04/12, 2017

Exercise 8

Lecturer: Mohsen Ghaffari

1 Sublinear-Time Approximation of Maximum Matching

Consider a graph $G = (V, E)$. Recall that a *matching* is a set of edges $M \subseteq E$ such that no two of the edges in M share an end-point. A fractional matching is the corresponding natural relaxation, where we assign to each edge $e \in E$ a value $x_e \in [0, 1]$ such that the summation of the edge-values in each node is at most 1, that is, for each node $v \in V$, we have $\sum_{e \in E(v)} x_e \leq 1$, where $E(v)$ denotes the set of edges incident on v. We define $y(v) = \sum_{e \in E(v)} x_e$ as the value of node v in the given fractional matching. The *size* of a fractional matching is defined as $\sum_{e \in E} x_e$, and we have $\sum_{e \in E} x_e = (\sum_{v \in V} y(v))/2$ (why?). We call a fractional matching *almost-maximal* if for each edge $e \in E$, there is one of its endpoints $v \in e$ such that $y(v) = \sum_{e' \in E(v)} x_{e'} \geq \frac{1}{1+\epsilon}$.

Exercise

(1a) In the class, we saw that any maximal matching has size at least 1/2 of the maximum matching. Prove that the size $\sum_{e \in E} x_e = (\sum_{v \in V} y(v))/2$ of any almost-maximal fractional matching is at least $\frac{1}{2(1+\epsilon)}$ of the size of maximum matching.

Thus, the above item indicates that almost-maximal fractional matchings also provide a reasonable approximation of the size of the maximum matching. But computing an almost-maximal fractional matching is much easier. We next see a LOCAL algorithm for that.

LOCAL-Algorithm for Almost-Maximal Fractional Matching: Initially, set $x_e = 1/\Delta$ for each edge $e \in E$. Then, for $\log_{1+\epsilon} \Delta$ iterations, in each iteration, we do as follows:

- For each vertex v such that $y(v) = \sum_{e \in E(v)} x_e \geq \frac{1}{1+\epsilon}$, we freeze all of its incident edges.
- For each unfrozen edge e, set $x_e \leftarrow x_e \cdot (1 + \epsilon)$.

Exercise

(1b) Prove that the process always maintains a fractional matching, meaning that we always have $\sum_{e \in E(v)} x_e \leq 1$.

(1c) Prove that at the end, we have an almost-maximal fractional matching, meaning that for each edge $e \in E$, there is one of its endpoints $v \in e$ such that $\sum_{e \in E(v)} x_e \geq \frac{1}{1+\epsilon}$.

Now that we have a simple LOCAL-algorithm for almost-maximal fractional matching, we use it to obtain a centralized algorithm for approximating the maximum matching. To estimate the size of maximum matching, we pick a set S of $k = \frac{20\Delta \log 1/\delta}{\epsilon^2}$ nodes at random (sampled with replacement). Here, δ is some certainty parameter $\delta \in [0, 0.25]$. For each sampled node $v \in S$, we run the above LOCAL-algorithm around v, hence allowing us to learn $y(v)$.

Exercise

(1d) Define a linear function $f : \mathbb{R} \to \mathbb{R}$ such that when applied on the sample average $\sum_{v \in S} y(v)/|S|$, the resulting value $f(\sum_{v \in S} y(v)/|S|)$ is an unbiased estimator of $\sum_{e \in E} x_e = (\sum_{v \in V} y(v))/2$. That is,

$$\mathbb{E}_S[f(\sum_{v \in S} y(v)/|S|)] = \sum_{e \in E} x_e.$$

(1e) What is the query complexity of our sublinear-time approximation algorithm?

(1f) Prove that the estimator that you defined in (1d) gives a $(2 + 3\epsilon)$-approximation of the maximum matching size, with probability at least $1 - \delta$.