Lecture 28: Randomized Algorithms: Minimum Cut

Anup Rao

June 7, 2019

Randomized algorithms are often faster than their deterministic counterparts, as well as simpler.

Min-Cut

We are given an undirected graph and want to partition the vertices into two non-empty sets A, B, such that the number of edges that cross from A to B is minimized. Consider the following simple algorithm:

> **Input:** Undirected graph G
> **Result:** A partition A, B
> Repeatedly do the following as long as the graph has more than 2 vertices: pick a uniformly edge that connects two distinct vertices and merge them. Output the partition that corresponds to the two vertices that are left at the end.

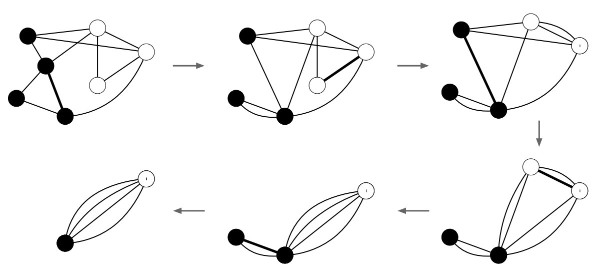

Figure 1: An execution of the randomized algorithm for Min-cut. This execution finds the cut indicated by black vertices.

We shall show that this algorithm find the minimum cut with non-negligible probability. Suppose the min-cut has k edges. Then the algorithm finds this min-cut if and only if none of these k edges are picked to do a merge by the algorithm.

Observe that every vertex must have at least k neighbors, or the vertex by itself would give a smaller cut. This means that the graph has at least $nk/2$ edges. The probability that we pick one of the k edges of the min-cut for the merge is at most $k/(nk/2) \leq 2/n$. Assuming that one of these edges is not picked, then again we must have that every vertex in the new graph has degree at least k, or we would get a smaller min-cut in the original graph. Continuing in this way, we get that the probability that the k edges of the min-cut are never picked is at least

$$\begin{aligned}
&\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{4}\right)\left(1-\frac{2}{3}\right)\\
&=\left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{2}{4}\right)\left(\frac{1}{3}\right)\\
&=\frac{2}{n(n-1)}.
\end{aligned}$$

This is a small probability, but imagine we just repeat the above algorithm t times and then output the best cut that we find. Then the probability that every run of the algorithm does not find the min-cut is at most $\left(1-\frac{2}{n(n-1)}\right)^t \leq e^{-\frac{2t}{n(n-1)}}$. If we set $t \gg n(n-1)$, this probability is extremely small.