Randomized min-cut

FullName: Christodoulides Kyriakos **AM:** 2016030025
PLH421 - Randomized Algorithms
University of Crete

March 13, 2021

Randomized min-cut

The idea of this algorithm is given a graph, find the smallest cut (smallest number of edges that disconnects the graph into two components. To acheive that, we follow the below steps, until our graph has two vertices left.

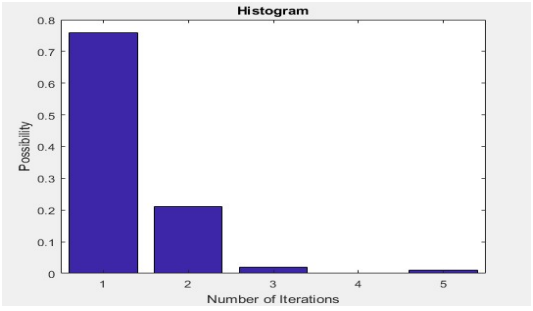
1. Construction of random adjacency matrix
2. While there are more than 2 vertices
   - Pick a random edge (u, v) in the contracted graph. κινήσεις.
   - Merge (or contract) u and v into a single vertex (update the contracted graph) and retain all the other edges in the graph.
   - Remove self-loops
3. Return cut represented by two vertices.

Following these steps, we are doing n-2 iterations, and after each iteration we choose uniformly at random an edge to contract it, so it decreases the total number of edges in the graph by 1. This algorithm has a possibilty to fail. It fails if the random edge that we pick $\in$ in min-cut. As the graph is being excecuted, it doesnt produce any cuts, and if we choose a edge that $\in$ in min-cut we 'll miss a cut. So in a succesful run, at first the probability is quite small, only 2/n(because the return min-cut is represeted by two vertices), but near the end of execution, when the graph has only three vertices, we have a 2/3 chance of failure. The average possibility of a succesful run is the multiplication of the possibility not to pick an edge that $\in$ in min-cut until we have 2 vertices. So in a graph with n nodes:

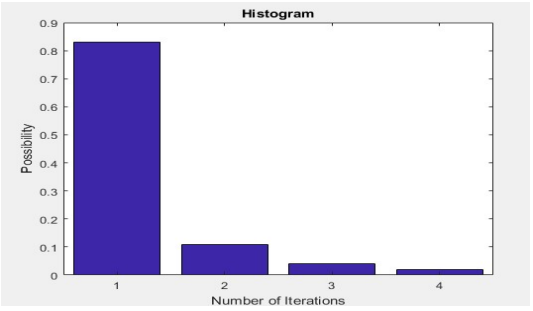
$$\prod_{i=1}^{n-2}(1-\frac{2}{n-i+1}) = \frac{2}{n(n-1)}$$

This algorithm has a single sided error on "yes" instances, which means Such an algorithm is always correct on "no" instances but erros with some probability on "yes" instances. Moreover in each experiment there is independancy , so running multiple times the experiment will reduce the error propability exponentially.

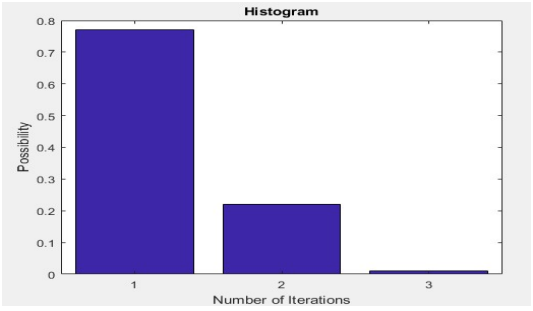
Simulations



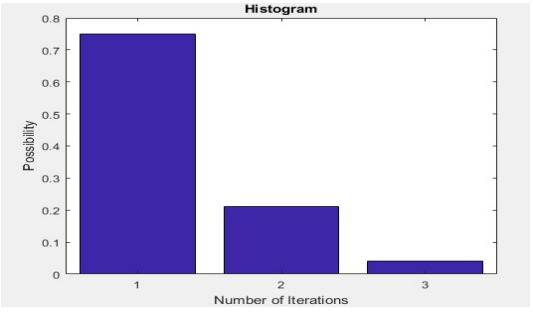
(a) nodes = 50, mincutweight = 10



(b) nodes = 100, mincutweight = 20



(c) nodes = 500, mincutweight = 100



(d) nodes = 1000, mincutweight = 200

From the above simulation of the algorithm we notice that in every graph the possibility of a succesful run from the first iteration is about 70-80%. Similarly 10-20% in the second iteration, and about 5% for the third. For the rest, as the number of iterations increases the possibility tend to 0. That's because it's almost impossible to have consecutively fails of the algorithm, as we said and before, each run is independant, so the probability to have more iterations, decreases exponentially.