Jessica Salguero, UIN: 937005006 **Homework 7** CSCE 629 April 17, 2025

Problem Statement

The Multiway Cut Problem is defined as follows: "given an undirected graph G = (V, E) and a set of terminal nodes s1, s2, · · · , sk V , find a minimum set of edges in E whose removal will make the k terminal nodes be mutually disconnected. (That is, if we remove that set of edges from the graph, every two terminal nodes will be disconnected from each other.)" It is known that when k = 2, the problem can be solved in polynomial time. Your task: Prove that when k = 3, the problem has a polynomial-time 2-approximation algorithm. (Hint: use "It is known that when k = 2, the problem can be solved in polynomial time." as known knowledge.)

Main Idea

With the Multiway Cut Problem, you have an undirected graph (G) and a set of terminals (sk). It is known that when k=2, it is possible to find a minimum set of edges (E) whose removal will disconnect the terminal nodes in polynomial time. In order to solve it for k=3, we can actually use the solution for k=2 as a part of the solution. To do this, we would turn separating 3 nodes into a series of separating two nodes from each other and combine the results.

To do this, it would require the use of a greedy algorithm. For every k=3 nodes, we would make 3 separate cuts: one separating s1 from s2 and s3, one separating s2 from s1 and s3, and one separating s3 from s1 and s2. Then, we use the size of each cut to determine which is ideal for our final solution. We sort the cuts by size, and take the union of the two smallest cuts as the approximate solution. Using the two smallest cuts is ideal because we are able to maintain disconnected nodes while preventing the removal of all edge nodes. The greedy algorithm approach also helps maintain efficiency while still providing good approximations.

Pseudocode

Algorithm 1 Hello, World!

1: **function** MY ALGORITHM(void)
2: Initialize global variables:
- G = (V, E); undirected graph
- tNodes = s1, s2, s3; set of terminal nodes
- Cuts = []; this will be used for keeping track of the possible cuts

3: For i in tNodes:
- currCut = remove i from tNodes
- tempG = graph with currCut as a merged node
- minCut = find the minimum set of edges between i and the merged node in tempG
- Add minCut set to Cuts

4: Once you are outside of the for loop:
- Identify two smallest cuts found in Cuts
- unionCuts = union of the two smallest cuts

5: return unionCuts
6: **end function**

Proof of Correctness

When attempting to solve the Multiway Cut problem for k=3, an efficient approach is to break down the larger problem into multiple iterations of the k=2 solution, which is already known to have a polynomial solution time. During each iteration, you merge two nodes into one, and then find minimum cuts. Since we already know that the k=2 version can be successfully solved in polynomial time, this can aid in maintaining algorithm efficiency. Additionally, since we are consistently choosing the smallest cuts, this will aid in maintaining a 2-approximation.

Once we have the three cuts, we pick the two smallest cuts and return their union as the final solution; the union of these two cuts will still disconnect all the terminal nodes while maintaining that the final solution is at most twice the optimal solution since we remove the largest cut from the sum.

Time Complexity

It is already known and specified that when k=2, this Multiway Cut Problem can be solved in polynomial time. This algorithm attempts to solve the k=3 scenario by splitting the larger problem into 3 iterations of the k=2 solution. Additionally, we also know that this algorithm is 2-approximation because we use the two smallest cuts, which maintains the efficiency of the approximation; it will not be more than twice the optimal solution. This means that overall, the algorithm should successfully estimate a solution and produce a polynomal time 2-approximation.