CS 180 Homework 5

March 15, 2015

1 Mobile Computing

Design a polynomial-time algorithm for the following problem: Given the positions of a set of clients and a set of base stations, decide whether every client can be connected simultaneously to a base station, subject to the range and load constraints.

The problem can be solved like the bipartite matching problem, by first creating a source that connects to all client nodes with a capacity of 1, and a terminal sink that connects to all base station nodes again with a capacity of 1.

We then create edges between each client and base station if the distance between the two is within the given range. Each of these edges will also have a capacity of 1.

The final step is to run the Ford-Fulkerson maximum flow algorithm on the graph we created. If any client is not connected to a base station by the end of the algorithm, we return false, otherwise we return true, indicating that every client can be connected to a base station.

2 Simple Maximum Flow

(a) Find a maximum flow; how many planeloads of people can we move from MDL to LAX?

Based on the output of a python implementation of Ford-Fulkerson, the maximum flow from MDL to LAX is 18.

(b) Find a minimum cut for this network, and give its capacity.

A minimum cut of the network is given by splitting the network into just MDL, and every other aiport, yielding a capacity of 18.

(c) Is the minimum cut unique? (Is there any other cut with the same capacity?)

The cut is unique in that there is no other cut with the same capacity.

3 Min Flow / Max Cut

(a) Give a polynomial time algorithm that finds the minimum possible flow on G.

The problem of finding the minimum flow can be reduced to the maximum flow problem as follows.

First one must find any flow through the network that satisfies the directions and lower bounds, which can be done by depth-first search or any other search from the source s to the sink t.

One can then create a new graph G' such that each edge has capacity of $flow(e) - l(e)$, where $flow(e)$ denotes the edge flow resulting from the inital flow we discovered through the search. We can then use Ford-Fulkerson to solve the maximum flow problem that results, the output of which will be the minimum flow through the network.

(b) Consider any minimum flow f. Is it true that it corresponds to some maximum cut ? That is, for networks like this, is there a Min Flow / Max Cut theorem? If so, prove it; if not, give a counterexample.

There is in fact a Min-Flow / Max-Cut theorem and it can be proven similarly to the Max-Flow / Min-Cut theorem. Furthermore, given that the minimum flow problem with lower bounds can be simply reduced to the maximum flow problem, it follows that a minimum cut from the transformed G' graph above corresponds to a maximum cut in G, which leads to the Min-Flow / Max-Cut theorem by reduction.

4 Directed Hamiltonian Path

Show that the Directed Hamiltonian Path problem is NP-complete. Show that this problem is in NP, then show that it is NP-hard by reduction from some other problem.

First, it can be shown that the Directed Hamiltonian Path problem is in NP by showing that it is possibly to verify the problem in NP time. This can by done by looking at a hypothetical verification function that nondeterminstically creates all possible permutations of nodes of the graph that are not u and v. The function will then check if each permutation is a directed edge in the graph, and return true if yes, and false otherwise.

The next step of proving NP-Completeness is to show that the problem is NP-hard, which can be shown by reduction from the undirected Hamiltonian Path problem, that is Hamiltonian Path $\leq_p$ Directed Hamiltonian Path. This can be done simply by showing that a undirected graph G' corresponding to G has a Hamiltonian Path with Hamiltonian Path and then verifying that edges are

directed as in the input graph G. Therefore, HAMILTONIAN PATH $\leq_p$ DIRECTED HAMILTONIAN PATH.