Bryan Arnold

CSE 3500

Homework 10

4/26/17

1 NP Completeness

To begin, we must prove that Strategic Advertising is in NP. Given a set of k nodes, you can check whether at least one is on path P_i. Given all the paths from 1 to i, you can check whether a given set of advertisements cover all the paths. This would take O(knt) time, making the verification of the problem doable in polynomial time. Thus, SA is in NP.

Next, we can show that VC <= pSA.

Suppose we are given an undirected graph G and some number k. Given an instance of vertex cover, we can determine if SA is NP complete by showing that VC will only be true if and only if SA advertisements go over every possible path. Construct a new graph G1 by arbitrarily directing each edge of G and let there be a path P_i for each edge in G1. G1 has no more than k advertisements if and only if G has a vertex cover of size at most k as well. Let us say G1 has a set of advertisements S, meaning it meets at least one end of each edge, thus making it a vertex cover for G. Now, suppose G has a vertex cover of size no more than k. The instance of the vertex cover meets each path P_i, making it a good set of advertisements. Showing that both vertex cover can be an SA, and an SA can be a vertex cover, SA can be reduced to VC (VC <= pSA). Thus, SA is NP Complete.

2 The Hamilton Path Problem

First, we need to show that HAMPATH is in NP. Given a set of edges and vertices, you can go through each edge/vertex, marking that it has been visited once. This can be done in O(n) times, n = number of vertices, since you're only going from vertex to vertex and checking whether each node has been visited once. Since this can be done in polynomial time, it can be said that HAMPATH is in NP, just as HAMCYCLE is.

Now, we show that HAMCYCLE <= pHAMPATH.

Given an instance of a HAMCYCLE, this already gives us a path that goes through each vertex in each graph G. This makes a valid path for us to easily create a HAMPATH instance from. Simply given the HAMCYCLE instance, any HAMPATH instance can be created by adding a single edge between the starting vertex and the ending vertex of the graph G. This would create a HAMPATH, making a reduction from HAMCYCLE to HAMPATH (HAMCYCLE <= pHAMPATH). Thus, HAMPATH is NP Complete.

3 NP Completeness

First, we need to prove that this problem is in NP. Given a tree T and set of vertices that are connected in the tree as described in the problem, you only need check whether each pair is still connected in the tree. Iterate throughout the entire tree and check whether a path exists between each node pair from 1 to i. This can be done in polynomial time (O(n)?), thus making it verifiable in polynomial time. So, this problem is in NP.

Now, we show for problem X: Vertex Cover <= pX.

Given an instance of vertex cover, we want to check whether each pair of nodes is connected to one each other, from 1 to i in the tree. Vertex cover is only true if the vertices/nodes in a graph/tree are connected by paths. So, when vertex cover fails for each pair of nodes, we know that problem X is satisfied. Now, we need to construct tree T to be a tree with each pair of nodes being connected by a path, with height of 1. This allows vertex cover to iterate over each pair to see if they're connected after edge deletion. Second, we must change the problem since it is optimization. We want to look for the smallest number of deletions to separate each node pair, so instead change the problem to determine whether the node pairs are connected given a maximum number of deletions of edges (M). By going through the tree, checking vertex cover for each node is false to make problem X true, and changing the problem to a decidability problem, we can say there is a reducibility from vertex cover to X (Vertex Cover <= pX). This, the problem X is NP Complete.

4 NP Completeness

a) First, we must show that Resource Reservation problem is in NP. Given a set of k processes, you can iterate throughout each resource, checking whether it has been requested more than once. This can be done in O(n), since you check each resource once for times it has been requested, therefore the answer for this problem can be verified in polynomial time. So, this problem is in NP.

   Now, we can show that Independent Set <= pResource Reservation.

   Given an instance of independent set, a graph G and number k are needed for an independent set. For the edges, they will be denoted as the resources and the nodes are the processes. If a node s wants a resource, the resource is the edge connected to s. If there are k processes and the requested resources are all different for these processes, then the k nodes are for an independent set. This is a valid thing to check for between the two problems because any edge between the nodes is a resource that each node requested. If there are a set of k resources, then there are k processes that make up a set of requests different sets of resources. Both sides can be reduced into one another in a problem, making them equivalent. So, given an instance on IS, if it is true, so will RR. This can also be done in polynomial time, making the reduction true (IS <= pResource Reservation). Thus, RR is NP Complete.

b) For the special case of k = 2 in the previous problem, this can be done as well. Try all O(n^2) pairs of processes in the problem and see if the pairs use different resources. Using the same IS and RR reduction logic and this slight differentiation for k = 2, this can be done in polynomial time. Thus, when k = 2, the special case is NP Complete.

c) Didn't need to do.

d) In part a, the reduction ensures that the instance of resource reservation where each resource was requested can encompass multiple nodes containing an edge. But, since an edge can only connect two nodes, this was already proven by the process of part a. This is already covered by how part a was described, also making it NP Complete.