December 3, 2015

CS 577 - Homework 10
Sejal Chauhan, Vinothkumar Siddharth, Mihir Shete

1 Graded written problem

Part a: The Travelling Turkey Problem (TTP-3) as mentioned in the written problem has the following constraints:

- Each city is visited at least once
- Number of distinct routes has to be the a small as possible
- Each city is visited no more than 3 times
- Path starts and ends at the same city (DC)

A city is said to be *visited* if the turkey arrives at the city from some other city. So if the turkey starts from DC, then DC wont be considered as visited till it comes back to DC from some other city. Also we note that the turkey can visit any city from every other city, so we will assume that we have a complete graph with weights along each edge to denote the effort to from city on one end of the edge to the other. To satisfy the constraint that each city can be *visited* no more than 3 times our solution graph should not have a vertex with degree more than 3. Now since we have to visit every city once in a complete graph with the least number of routes and since each route is represented by an edge, we can say that the Minimum Spanning Tree of the graph with the constraint that no node in the tree has degree more than 3 is the desired solution if the weight of the edges for such a tree is smallest. More formally, we can describe the problem as:

Given a complete weighted graph, find a spanning tree in which every vertex has degree at most 3 and which has minimum cost among all possible spanning trees.

Valid solutions to the above problem can be denoted by strings of length proportional to the number of cities. It is also trivial to see that the validity and objective value of such a solution string can be checked in polynomial time. Hence the above formulization fits in the framework of NP-optimization problems as descibed in the lecture notes.

Part b: polynomial-time reduce the Circuit-TSP-optimization to TTP-3-optimization

Some terms we will use while doing the reductions:
TTP-3: Travelling Turkey Problem where each city can be visited no more than 3 times
TTP-2: Travelling Turkey Problem where each city can be visited no more than 2 times

We will prove that Circuit-TSP can be reduced to TTP-3 in 2 steps. First we will reduce Path-TSP to TTP-2, and from the equivalence in the Practice problem we wil claim that since Path-TSP is equivalent to Circuit-TSP we can reduce Circuit-TSP to TTP-2. In the next step we will reduce TTP-2 to TTP-3.

Path-TSP-optimization $\leq$ ***TTP-2-optimization***

Given an instance of Path-TSP with graph G. Run TTP-2 on the graph G, which returns a MST of G of minimal cost such that degree of no vertex in the tree is more than 2. (Note that MST will always exist in the complete graph) The path in G is represented by the MST and it starts and ends in the leaf nodes of the MST. This is true because we will have exactly 2 leaf nodes in the MST, leaf nodes have degee of one and they will represent the start and end of our path. We cannot have third leaf node as it will imply that atleast one vertex in MST has a degree of 3. We claim that P is a minimal-cost path in G or no path exists in G that visits each vertex exactly once.

Note that in the method used to construct P, we start from one of the leaf nodes of the MST and cover all the nodes till we reach the other leaf. Also note that since MST covers all the vertices we will be visiting each vertex once when we travel from one leaf to the other. There will be exactly one path from the leaf to the other beacause apart from the leaf all other vertices in MST will have degree 2 (The cannot have degree 1 because they will be leaves then and we cannot have more than 2 leaves). The cost of this path will be equivalent to the cost of MST and since our MST has the minimum cost, our path will also have the minimum possible cost.

Now, in the other direction, note that for any TSP path X in G, starting at some vertex s and ending at some vertex t, we can construct a TTP-2 MST Y in G by considering any non-terminal vertex as root.

Finally, since the MST to Path conversion can be done in linear time, the reduction can be computed in polynomial time.

TTP-2-optimization $\leq$ ***TTP-3-optimization***

Given that we to run an instance of TTP-2-optimization with graph G, construct a copy of G, G'. If G has n vertices then add n new vertices to graph G'. Now, connect each new vertex in G' with just one vertex in G' which was obtained from G with an edge of weight 0. Now, run TTP-3 on G' to obtain an MST M_3. Contruct a MST M_2 of G by removing all the newly added vertices in G' and their associated edges from M_3. We now claim than M_2 is the MST in G with minimum cost.

Note that in the method used to construct M_2, we removed edges of cost 0 from M_3, so M_2 has the same cost as M_3. Further note that since M_3 visits every vertex of G, and M_2 visits every vertex M_3 visits other than the newly added vertices in G', which do not exist in G, M_2 visits every vertex of G. Also, note that the newly added vertices have a degree of 1 and so they are the leaves of M_3 and so on removing them and their associated edges we will get a tree only. Finally, since this construction did not depend on M_3 being minimal, this shows for every TTP-3 circuit in G' we can construct a TTP-2 circuit in G of equivalent cost. Now, in the other direction, note that for any TTP-2 circuit X in G, we can construct a TTP-3 circuit Y in G' of equivalent cost by adding n new vertices and connecting them to one vertex in G with an edge of cost 0. This equivalence shows that if no TTP-3 circuit exists in G', then no TTP-2 cicuit exists in G, otherwise, the minimal cost TTP-3 circuit in G' is the same cost as the minimal cost TTP-2 circuit in G, completing the reduction.

Finally, since constructing the graph G' and removing edges from the minimal cost MST M_3 in G' can be done in linear time, the reduction can be computed in polynomial time.

Part c: Formulate a decision version of the Turkey Tour Problem, and polynomial-time reduce it to and from the optimization version

TTP-3-decision problem can be formulated as follows: Given a complete weighted graph, find if a spanning tree exists in which every vertex has degree at most 3 and which has the cost of atmost C.

TTP-3-optimization $\leq$ ***TTP-3-decision***

Given the decision oracle we need to return a MST with minimal cost. Given a graph G and cost C the decision oracle will just report if MST is possible with cost atmost C. We will proceed with the reduction in following steps:

- start with cost as sum of all edges and do a binary search using the decision oracle to find the minimum cost C_{min}
- now iterate over each edge in the connected graph and remove it to form graph G', repeat the above step for G' to find C'_{min}, if C'_{min}<C_{min} the removed edge has to be included in the MST for the optimization problem.
- we can stop when we have $n-1$ edges.

If sum of all edges is S then the above logic is bounded by $m * S * \log S$

TTP-3-decision $\leq$ ***TTP-3-optimization***

Given the optimization oracle which will return the MST with least cost. Once we run the optimization oracle and have the MST we calculate its cost, say its is C_{min}. If C is the input cost in the decision problem we should return *no* if C<C_{min} and *yes* otherwise