CS301: Assignment #5

Muhammed Orhun Gale
morhun@sabanciuniv.edu

Sabanci University — June 5, 2022

1 Decision Problem

Fundamentally, given problem aims to check that if there exist a itinerary from given source vertex to every other vertex and if there exist find the shortest one. Therefore, for the optimization problem the input is a graph and sets of vertices for each vertex which must be included to shortest itineraries for a source vertex. However, in order to evaluate NP-completeness of a problem the problem must be decision problem and in this sense given problem can be described as a decision problem as follows:

Input:

Graph $G = (V, E, W)$ where V is set of vertices, E is set of edges and W is a function that maps each edge to a non-negative real number.

C is set of source vertices and S_c is set of vertices which must be included to the shortest itineraries of the source node $c \in C$.

A non-negative integer k which describes limits the length of the shortest path.

Output:

Yes: If there is a path from the source node c to v such that cost is less than k

No: If there does not exist a path from c to v that costs less than k

The decision problem that is described above basically says that if there exist a path from a given source vertex s to any other vertex v the output would be "yes" since it highlights that there exist a shortest path. By solving this problem via iterating k (decreasing in this case) one can find the shortest path too. In order to achieve this, the k can be defined as "infinity" in order to highlight that there does not exist a path.

2 NP-Completeness

In order to prove that given problem is NP-complete, one must show that problem is a member of NP and solving it is as hard as solving a problem in NP.

Then argue that given problem is NP-complete problem.

2.1 Membership

Start with nondeterministicaly guessing a candidate solution whose size is polynomial and verify it. Let this witness be city v. First check that every city in the set S_c appears in the itinerary and also check that every city is included in the path exactly one time which of both take $\Theta(|V|)$ time to verify. Then compute the total cost of the itinerary for all paths from v to other cities which would take $O(|E|)$ for only

one city and would take total $O(|V|.|E|)$ in worst case. After that verify that length of found paths are smaller than the non-negative number k which would take $O(1)$ time. After that, repeat this procedure for all cities in the set S which would be at most $\Theta(|V|)$. Therefore, verifying a polynomial size solution this problem takes total $O(|V|)$*($O(|V|)$ + $O(|V|.|E|)$ + $O(1)$) = $O(|V|^2.|E|)$ which shows that this problem is a member of NP problems since it can be verified in polynomial time.

2.2 Hardness

In order prove that given problem is as hard as a problem in NP, one can reduce the "Traveling Salesman Problem" (TSP) to the given problem. In TSP given a graph $G = (V, E)$ one must visit all vertices form starting a vertex v and get back to (stops at) the same vertex c. For the given problem (problem that it is tried to find itineraries), the agent tries to visit all nodes given in the S_c list along with the other vertex (I_c) that creates the shortest itinerary found starting from the source vertex c. Therefore, via setting the given graph as S_c + I_c and setting the starting vertex as the source vertex c but setting the ending vertex as the target vertex v the problem can be solved. Also note that in both problems all vertices in the given graph must be visited only once. Then,

First as the arbitrary instance of TSP and for the specific instance of the given problem set S_c + I_c as the graph which can be at max $O(|V|)$ and set the "k" parameter of the TSP which indicates the number of visited cities as the parameter k of the given problem.

After that it can be seen that since all of the vertices visited only once, a path would be found to every other city and as the output **YES** would be returned which is also the case in the TSP since all of the vertices are visited. If there exist a path (which is k in this case) which is smaller than the found path, the given problem outputs **NO** which is again the case for TSP since if k city did not visited it should return **NO**. Therefore, the reduction holds the "if and only if" statement which means that reduction is correct.

Reduction from TSP to the given problem takes polynomial time since in order to reduce the problem only the ending vertex is set as the target vertex (all other vertices) and checking found itineraries are smaller than the k parameter. Therefore, it can be said that the problem is as hard as TSP which is a NP problem which eventually shows that given problem is NP-Complete.