Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 03

Lecture 14

Lecture 14 : 2-Approximation Algorithm for Metric TSP

Welcome. So, in the last lecture we have seen that the travelling salesman problem the symmetric version does not have any good or meaningful approximation of algorithm in polynomial time. So, today we will see a very natural special case of travelling salesman problem which is called metric travelling salesman problem which admits constant factor approximation algorithms. So, today's topic is metric travelling salesman problem. So, what is metric travelling salesman problem? We assume in this problem that the distance between cities satisfy metric property. what are they? The first one is symmetry that for all $i, j \in [n], i \neq j$ the distance from i to j is same as distance from j to i and triangle inequality.

for all $i, j, k \in [n]$ i, j, k are different cities that means, no two of them are same then the distance between i and j is less than equal to distance between i and k plus distance between j and k. This holds for any triple triplet of cities i j k. Now, in travelling salesman problem we are looking to find a minimum cost or minimum weight Hamiltonian cycle. There is a cousin of Hamiltonian cycle which is called Eulerian cycle which is easy to compute.

So, what is Eulerian cycle? So Eulerian cycle of a graph is a cycle that visits every age exactly once. ok. In particular that Eulerian cycle will visit all vertices also, but maybe multiple times. So, here is an important result which is quite easy to prove you can take it as a homework. has an Eulerian cycle if and only if the degree of every vertex is even.

Moreover, an Eulerian cycle if it exists can be computed in polynomial time. Not only that you see the number any Eulerian cycle is and the weight of any Eulerian cycle is an upper bound on the weight of any travelling salesman tour because it visits all edges. It is the its weight is sum of the weights of all edges. So, the idea of this approximation algorithm is to have a subgraph of it subgraph of the given complete graph G which is spanning that means, all vertices are there and connected and connected and it is Eulerian. So, idea.

find an Eulerian connected spanning subgraph, spanning means it ah has all the vertices of G. of the input graph, input graph is a complete graph. Now, the goal is to find an Eulerian connected spanning subgraph, which has as small edges as possible, which has small number of edges ok. Now, what will happen if I find if I am able to find an Eulerian connected spanning subgraph which has small number of edges a small means weight total weight because the graph is weighted. then I will have an Eulerian tour which I can compute in polynomial time and then I will bypass I will shortcut the I will bypass the repeated visits and then thereby converting the Eulerian cycle to a travelling salesman tour.

So, if I find that Eulerian connected spanning after finding that subgraph H let us call it H, then we compute an Eulerian cycle of H let us call it C Eulerian cycle C of H this subgraph in polynomial time. Now, then from C which can be seen as a sequence of vertices in the order of the Eulerian cycle viewed as a sequence of vertices. We delete every occurrence of every vertex except the first occurrence. let the resulting cycle be C prime, then by triangle inequality cost of C' is less than equal to cost of C.

This trick is called short circuiting, this is called short circuiting. ok. To see this, let $i_1, i_2, \ldots, i_k$ be a sequence of vertices in C and $i_2, \ldots, i_{k-1}$ got deleted because of short circuiting. Pictorially here is i_1 then i_2 this was the part of the cycle or C of the Hamiltonian tour this is i_3.

i_k. Now, I delete this edges, this vertices got deleted and the subsequent edges and I add this edge. You see the cost of the edge with from $i_1, i_2, \ldots, i_k$ is less than the sum of the cost of the deleted edges, this is because triangle inequality. From triangle inequality $c_{i_1 i_k} \leq c_{i_1 i_2} + \ldots + c_{i_{k-1} i_k}$ and reduced by the total cost of this edges.

Now, because of triangle inequality and hence short cutting short circuiting the tool cannot increase the cannot increase the total cost of the tool. So, the so, if I have so, the question boils down to finding a good subgraph which is Eulerian and spanning and connected. So, the first idea is to have a spanning tree. So, algorithm 1.

So, we will describe two a spanning tree T of the input graph. So, let T be the spanning tree, but spanning tree or any tree is never Eulerian because any tree always has at least one vertex of degree 1. So, how to make how to convert this T as a spanning as a Eulerian graph? So, we add a parallel edge to every edge of T with same cost. So, let H be the resulting graph. resulting subgraph.

Let us see it pictorially suppose this tree T is look looks like suppose this is the tree look

like what I do is add parallel edge to every edge. So, H will look like first let us draw tree and add a parallel edge to every edge of same width. Now you observe that the degree of every vertex i in H is exactly 2 times degree of that vertex i times t because the number of edges has simply doubled. Moreover H is connected since T is connected and H is spanning.

all vertices ok. So, what is the total cost of the edges of T edges of H? C of H let us define it as or cost of H. you see which is the sum of the edges of H sum of their costs which is 2 times sum of the edges the H set of the tree $C(H)$ which is 2 times cost of So, and the ALG is less than equal to $C(H)$ cost of the Eulerian tour H which is the sum of the weights of the edges of H. Now, to minimize this which is $2c(T)$. Now to minimize this we should pick that spanning tree which has minimum weight. So, we instead of starting with any arbitrary spanning tree we compute a minimum spanning tree ok.

We use standard any standard greedy algorithm for example, Prim's algorithm. or Kruskal's algorithm to compute a minimum spanning tree. T of G ok, but how does this C of T cost of minimum spanning tree compare with opt. Now, here is a crucial observation claim that the cost of any optimal travelling salesman tour is at most or at least the cost of any minimum spanning tree. To see this let O be any optimal travelling salesman tour delete any edge from O, this will be a Hamiltonian path relate any edge E.

So, observe that O is a Hamiltonian path because we can I can delete any edge from a cycle and still all the vertices in the cycle remains connected. It visits all vertices of the graph and hence O minus is a Hamiltonian path and thus meaning a spanning tree. Thus we have cost of O because all costs are positive. that we always assume cost of O is less than equal to cost of O minus E which is the cost of any minimum spanning tree ok. So, which is here we are picking T if T is a minimum spanning tree.

So, this is equal to cost of T ok. So, this will be other way this inequalities cost of O is greater than equal to cost of C minus. So, I am dropping an edge with positive weight and this is a this is a spanning tree which is whose cost will be greater than equal to cost of minimum spanning tree. So, now, we have we can show the approximation guarantee ALG is less than equal to twice cost of T where T is a minimum spanning tree which is less than equal to C of O which is this is twice opt. Hence, the approximation factor of our algorithm is at least 2 ok. Let us stop here.