CS880: Approximation and Online Algorithms **Scribe:** Kexin Li

Lecture 2: Traveling Salesman Problem(TSP), Polynomial Time Approximation Scheme(PTAS)
Date: 2019/09/06

2.1 Recap from Last Week

In general, we have many NP hard optimization problems that requires exponotial running time. Our goal of approximation is to relax the optimzation condition and find an algorithm that runs efficently (polynomial time in the size of instance) and provide a solution which satisfies the feasibility constrains and slightly sub-optimal. The degree of sub-optimality is captured with Approximation Ratio (AR). For the moment, we will consider the minimization problem. We define the optimal value as the minimum of objective function over all feasible solutions.

Definition 2.1.1 *An instand I of an optimization problem consists of a feasible set F_I and an objective function obj*

$$OPT(I) = \min_{x \in F_I} obj(x)$$

For an algorithm ALG, Approximation Ratio $= \max_{instance I} \frac{ALG(I)}{OPT(I)}$

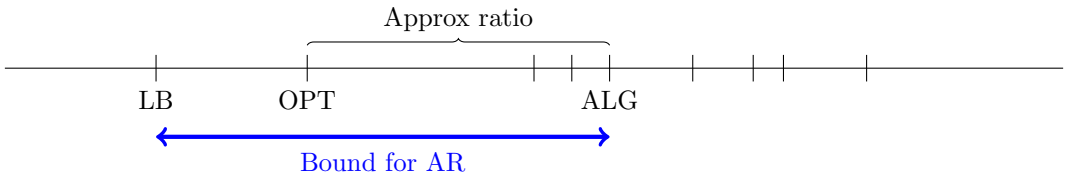


Figure 2.1.1: LB,OPT,ALG

Any given algorithm ALG could produce this objective function value over feasible set, which will be always larger than the optimal solution OPT. Therefore the approximation ratio of an algorithm is always larger than 1 for minimization problem. The closer to 1, the better the algorithm is. So we take the maximum over all the instances to give the worst-case performance of the algorithm in term of optimality.

Recall in previous lectures, we developed two approximation algorithms: one for Vertex Cover problem and one for Steiner Tree. We obtain an approximation ratio (AR) of 2 for both of them. In order to compute AR, we have to implicitly or explicitly find a lower bound (LB) for OPT, because it's often NP-hard to compute optimal value for these problems. We want to find a LB that's at least as good as OPT ($LB < OPT$). This way, we will be able to get upper bound for approximation ratio.

Algorithm 1 Approximation Algorithm for Vertex Cover

Let M be any maximal matching
Return S = Set of all end points of edges in M

The lower bound for optimal is the size of the matching $LB = |M|$. The we obtain the approximation ratio of 2.

Algorithm 2 Approximation Algorithm for Steiner Tree

Constract the metric completion of the graph G (for every pair of verteces, we add an edge with weight that equals to the shortest length in the graph)
Remove non-terminals from G and form G'
Find minimum spaning tree (MST) of G'
Replace all MST edges by shortest path in G
Remove cycles
Return resulting tree

The lower bound for optimal is half of the cost of MST of G'. Using this LB, we achieve approximation ratio of 2.

2.2 Traveling Salesmen Problem (TSP)

Today, we want to develop approximation algorithm for traveling salesmen problem (TSP) and we will see an improvement on the AR, which leads us to the polynomial time approximation scheme (PTAS).

The traveling saleman probelm is, given an undirected graph with edge lengths, to find the shortest spanning tour. Spanning means it must visit every vertex at least once in the graph and tour means it must start and end at the same vertex. This problem is generalization version of the Hamiltonian path problem, which is to visit every vertex exactly once. Whether Hamiltonian path exists in a given graph is NP-hard as well as find the shortest one even in complete graph. This imples there is no polynomial time algorithm that gives finite approximation ratio. When TSP is retricted to visit every vertex exactly once, it's NP-hard to find any finite approximation. Instead we want to consider the metric TSP. We can think of this in two ways: allowing visit vertex more than once or taking metric completion on the graph. Again, metric completion is for every pair of vertices, to add an edge with weight that equals to the shortest length in the graph. Then the resulting complete graph statisfies the triangle inequality. The triangle inequality is the edge between two vertex is the shortest than going through another other intermediate vertex. With triangle inequality we will have a shortest tour that visits every vertex exactly once because we can always shortcut out the repeting vertex. Therefore, we want to consider the version of TSP with complete graph with metric property and triangle inequality.

2.2.1 2-Approximation

Algorithm 3 Approximation Algorithm for TSP

Find the MST T
Double the edges of T to obtain a tour that traverses T

Claim 2.2.1 *cost of algorithm = 2 cost (T)*

This is because if we travese the MST T, we use every edge twice, then the cost of algorithm is twice of the sum of edge lengths in the tree.

Claim 2.2.2 *cost of MST $\leq$ cost of Opt(TSP)*

Consider the optimal tour and remove one edge. Then, we obtain a spanning tree. The cost of this spanning tree is smaller than the cost of optimal tour but at least as large as the cost of the MST. This proves the lower bound on TSP.

Together with these two claims, we can see the approximation ratio of **Algorithm 3** is 2.

2.2.2 APX-hardness

This leaves us some question for ponder:

1. Is there an instance I, for which $\frac{ALG(I)}{OPT(I)} = \alpha$? No analysis of ALG will prove a better approximation ratio α.
2. Is there an instance I, for which $\frac{OPT(I)}{LB(I)} = \alpha$? No algorithm using this LB can do better than approximation ratio α
3. is there any efficient algorithm with better ratio? This is an computational hardness question

Let's go back to vertex cover example. If we have a graph with two nodes and one edge between, the optimal is 1 and approximation algorithm using maximal matching give us 2. Then the $\frac{ALG(I)}{OPT(I)} = 2$. If we have a complete graph K_n with odd n, the optimal solution is $n-1$ and the LB is $\frac{n-1}{2}$. This gives us $\frac{OPT(I)}{LB(I)} = 2$. This tells us not only our analysis is tight but also any approximation algorithm based on maximum/maximal matching cannot do better than AR of 2.
In fact, if it's NP-hard to approximate a problem better than some factor $\alpha > 1$, the problem is called APX-hard. Vertex cover is APX-hard. It turns out $2 - \epsilon$-approximation for any constant $\epsilon > 0$ is hard under a complexity assumption we won't discuss in this class. This means we don't expect to find $1.99\dot{9}$-approximation algorithm that runs in polynomial time.

2.2.3 3/2-Approximation

TSP is also APX-hard, but we haven't yet ruled out any approximation fractor better than 3/2.

Algorithm 4 Christofides Algorithm for TSP

Take the metric completion of G
Find MST T
Let S = all odd degree vertices in T
Let M = min cost perfect mathcing on S
Return $M \cup T$

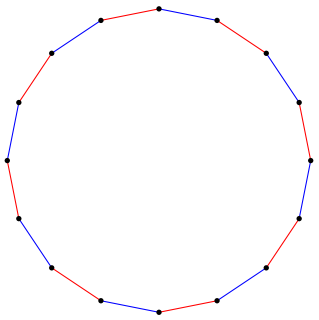
Eulerian graph is a graph where every vertex has even degree. On Eulerian graph, we can constract a Eulerian tour where every edge is visited exactly once. Christofides observed that TSP is about finding the minimum cost spanning Eulerian graph inside of the original graph. After finding the MST same as Alg. 3, we find the set of all the odd degree vertices and add matching between them to make them even degree.

Claim 2.2.3 *$M \cup T$ is Eulerian tour.*

This is easy to see because we add one edge to all the odd degree vertex in T so every vertex has even degree, which is a Eulerian graph. Thus a Eulerian tour exists and visit every edge exactly once. So ALG = cost(M) + cost(T)

Claim 2.2.4 *Cost(M) $\leq \frac{1}{2}$ OPT(TSP)*

To prove this claim, we will show the existance of a matching on S of cost at most $\frac{1}{2}$ OPT. Consider the optimal tour and remove all the vertices not in S from it. We get a shorter tour spanning all vertices in S. Now divide all edges in the tour into two alternating sets: odd edges (blue) and even edges(red) as shown in the picture below. Each of these sets is a machting over S. Therefore, one of these matching has cost at most $\frac{1}{2}$ OPT and the claim follows.



By Claim 2.2.2 $cost(T) \leq OPT(TSP)$ and Claim 2.2.4 $cost(M) \leq \frac{1}{2}OPT(TSP)$, Christofide's Algorithm is 3/2 approximation algorithm. It turns out the analysis is tight using this LB.
In fact the best known hardness of approximation result for TSP is $\frac{123}{122} \approx 1.008$. There is a gap between the best known approximation algorithm and known hardness result for TSP to the best of our knowledge.

2.3 Polynomial Time Approximation Scheme (PTAS)

Steiner tree and TSP are examples of APX-hard problems which is NP-hard to approximate upto some constant factors. Let's look at another class of problem which we can approximate arbitrary close to 1 efficently. This is called Polynomial Time Approximation Scheme (PTAS).

Definition 2.3.1 *Polynomial Time Approximation Scheme (PTAS) is an algorithm that for any constant $\epsilon > 0$ returns a $(1+\epsilon)$-approximation with running time polynomial in the size of input n for fix ϵ. For example, $n^{\frac{1}{\epsilon}}$, $n^{2^{\frac{1}{\epsilon}}}$ are some possible running times for such approximation algorithms.*

Definition 2.3.2 *Efficent PTAS (EPTAS) time is an algorithm that for any constant $\epsilon > 0$ returns a $(1+\epsilon)$-approximation with running time polynomial in input size n and arbitrary in $\frac{1}{\epsilon}$ i.e. $poly(n)f(\frac{1}{\epsilon})$. For example, $n^5 2^{\frac{1}{\epsilon}}$ is a possible running time for EPTAS*

Definition 2.3.3 *Fully PTAS (FPTAS) time is an algorithm that for any constant $\epsilon > 0$ returns a $(1+\epsilon)$-approximation with running time*

polynomial in both the problem size n and $1/\epsilon$. i.e. $poly(n, \frac{1}{\epsilon})$. For example, $n^7(\frac{1}{\epsilon})^3$ is a possible running time for FPTAS.