Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 06

Lecture 28

Lecture 28 : 3 Factor Approximation Algorithm for Prize Collecting Steiner Tree Contd.

Welcome. So, in the last lecture we have started designing a constant factor approximation algorithm for price collecting Steiner tree problem. We have written down the linear programming relaxation for that problem and we have seen that although there are exponentially many constraints. with there is a polynomial time separation oracle and consequently we can use the ellipsoid method to compute the optimal solution of that linear program in polynomial time. So, in this lecture we will see how a linear program optimal solution for that linear program can be used as a guide to design a constant factor approximation algorithm for the price collecting standard tree problem ok.

So, three factor three approximation algorithm for price collecting stainer tree problem. So, let us recall the LP relaxation of this problem. We had a variable for each edge and a variable for each vertex. So, minimize $\sum c_e x_e + \sum \pi(i)(1 - y_i)$ subject to for every set is subset of $V \setminus \{i\}$ this is for all $i \in V \setminus \{r\}$ such that $r \in S$.

If I look at all the edges and sum them all the edges in $\delta(S)$ this should be at least 1 at least 1 edge must be picked if y_i is 1 otherwise there is no constraint. So, if y_i is 0 that you see this constraints are automatically satisfied because it says that sum of sum x is greater than equal to 1 which is obviously, which will obviously, hold because all this x is can take only non negative values ok. and then we have y_r equal to 1 and then we have for all $i \in V, 0 \leq y_i \leq 1$ for all $e \in E, 0 \leq x_e \leq 1$. So, we use ellipsoid method along with the polynomial time separation oracle to get an optimal solution for this linear program. So, the first step of our algorithm is solve the relaxed LP let (x^*, y^*) be an optimal LP solution ok.

And then a natural idea is to take some value alpha and pick all the vertices whose y values are greater than alpha and then try to connect them with as few edges as possible.

So, that is the idea. So, U is all the vertices $i \in V$ such that $y_i^* \geq \alpha$. what should be the value of α that we will see let the analysis tell what should be the value of α. And then build a mean cost using U as the set of terminals ok.

So, although this finding a mean cost trainer tree on a set of terminals is a NP complete problem, there exist approximation algorithm which we which we will use as a black box. and then return. So, this is the pseudo code of the algorithm ok. So, for this part we will use fact a lemma black box. the treaty.

So, let us number these steps this is 1, 2, 3, 4 the treaty computed at step 3 has cost the sum of the edges of the trees in a tree T c_e $e \in E[T]$ this is less than equal to $\frac{2}{\alpha} \sum c_e x_e^*$. So, this result we will use as a black box, there is a beautiful primal dual algorithm for this problem and we use that for this minimum standard tree problem and we use that as a black box and then let us proceed. So, now, we once we have that we have bounded the edge cost next word this is the edge cost of the algorithm. Now, we will bound the cost of the terminals. So, for that we have this lemma that for all vertices$v \in V \setminus V[T]$ that is the vertex set of T.

Here you see that vertex set of T is a potentially a superset of U because it is a steiner tree it will pick all the stain all the terminal vertices which is U and it can pick some Steiner vertices. So, $\sum \pi(i) \leq \frac{1}{1-\alpha} \sum \pi(i)(1-y_i^*)$ proof. Very easy. So, if i is not in the tree because this sum is the on the left hand side the sum is over the vertices not in the tree. So, if i belongs to $V \setminus V[T]$ that means, i is not in tree that means, $i \in V \setminus U$ because $U \subseteq V[T]$. That means, the y value $y_i < \alpha$ because U is the set of vertices whose y value is greater than equal to α. So, $y_i < \alpha$. So, what we have is $1 - y_i > 1 - \alpha$ which is same as saying $\frac{1-y_i}{1-\alpha} > 1$. So, now, we start with left hand side $\sum \pi(i)$ this is less than equal to we have this 1 multiplied here replacing 1 with $\frac{1-y_i}{1-\alpha}$.

So, $\frac{1}{1-\alpha} \sum_{i \in V \setminus V[T]} \pi(i)(1-y_i^*)$. And if I add some more terms because y_i s take value in between 0 and 1, then this term cannot get decrease the sum this is less than equal to $\frac{1}{(1-\alpha)} \sum_{i \in V} \pi(i)(1-y_i^*)$. So, this proves this claim. So, what is ALG then? ALG is $\sum_{e \in E[T]} c_e + \sum_{i \in V \setminus V[T]} \pi(i)$.

Now, the first sum $\sum_{e\in E[T]} c_e \leq \frac{2}{\alpha}\sum_{e\in E[G]} c_e x_e^*$ plus and this term $\sum_{i\in V\setminus V[T]} \pi(i)$ this is less than equal to $\frac{1}{1-\alpha}\sum_{i\in V\setminus V[T]} \pi(i)(1-y_i^*)$. So, we can write this as maximum of this $\frac{2}{\alpha}$ and $\frac{1}{1-\alpha}\sum_{e\in E[G]} c_e x_e^*$. plus $\sum_{v\in V[G]} \pi(i)(1-y_i^*)$. This is LP opt. So, this is equal to $max\{\frac{2}{\alpha},\frac{1}{1-\alpha}\}$ LP opt and LP opt is less than equal to opt because it is a relaxation we have increase the search space.

So, the minimum can only dropped. So, this is less than equal to $max\{\frac{2}{\alpha},\frac{1}{1-\alpha}\}$ times opt. So, this is the approximation ratio $max\{\frac{2}{\alpha},\frac{1}{1-\alpha}\}$. So, we need to pick α so, that $max\{\frac{2}{\alpha},\frac{1}{1-\alpha}\}$ is minimum. So, pick alpha so, that $max\{\frac{2}{\alpha},\frac{1}{1-\alpha}\}$ is minimized.

That will be minimized if $\frac{2}{\alpha}=\frac{1}{1-\alpha}$ for that the intuitive reason is choosing small α increases one term and decreases other term and choosing small large α decreases the second term and increases the sorry choosing large α closer to 1 α should be in between 0 and 1 choosing large α makes the term $\frac{1}{1-\alpha}$ large than $\frac{2}{\alpha}$. So, the maximum will be ah minimized when both the terms are same for that we need $\frac{2}{\alpha}$ equal to $\frac{1}{1-\alpha}$ that is $\alpha=\frac{2}{3}$. So, if I put $\alpha=\frac{2}{3}$ you see both $\frac{2}{\alpha}$ and $\frac{1}{1-\alpha}$ becomes 3.

So, this is 3 opt choosing $\alpha=\frac{2}{3}$. So, we have designed a three factor approximation algorithm. So, what is this algorithm? This is the pseudo code of the algorithm. So, we can replace this $\alpha=\frac{2}{3}$ So, this is the complete description of the algorithm.

Here is another natural rounding technique. another natural rounding technique deterministic rounding technique is to try all possible alpha, but what do you mean by trying all possible α? α can take any value in between 0 and 1 and hence there are infinitely many possibilities. Actually if you think through it there are not because only if you look at the y_i values and plot suppose this is y_1, this is suppose y_2, suppose this is y3, there are some this way, suppose here is y_n, there are n y values and if α is in between two consecutive y values, it does not matter what is the exact value of α. it seems there are uncountably many possibilities for alpha.

trying all alpha in this set $y_i, i \in V$ this has n points. So, if I try all this n points that is enough gives all possible u. So, these again are deterministic rounding. So, what is the algorithm? You try, so new algorithm. So, you run deterministic rounding for every alpha equal to y_1^* these are stars because we we are only working with optimal LP solutions.

Try α equal to $y_1^*, y_2^*, \ldots, y_n^*$. Let $T_1, T_2, T_3, \ldots, T_n$ be the trees output by the deterministic rounding algorithm. what you do? For every tree you compute the value of the optimization function and return the tree whichever minimizes the value of the optimization function. So, for each tree $T_i, i \in [n]$ compute $\sum_{e \in E[T_i]} c_e$, this is a sum of the cost plus $\sum_{i \in V \setminus V[T_i]} \pi(i)$ this is the sum of the penalties omitted in the tree penalties of the vertices omitted in tree T_i . So, for each tree compute this quantity and output the T_i which minimizes the above sum.

What is obvious is this is also a 3 factor approximation algorithm. the output the cost of the solution is at most the cost of the solution of the deterministic rounding algorithm with $\alpha = \frac{2}{3}$. this is also a three factor approximation algorithm the new algorithm. actually it is better than we can show that it is a two factor approximation algorithm. Actually it is a two factor approximation algorithm.

But how to prove it? Unfortunately, we do not know any direct way of proving it. The proof that we know is via randomized rounding of this linear program which we will see in the next topic, next algorithm design technique when we study randomized rounding of linear programs. There we will see that a randomized rounding achieves a two factor approximation algorithm and from that proof we will get a corollary that which will say that the this deterministic rounding algorithm also has an approximation factor of at most two. So, let us stop here. Thank you.