Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 10

Lecture 48

Lecture 48 : Primal-dual Algorithm for Steiner Forest Contd.

Welcome. So, in the last class we have started seeing the primal dual method-based algorithm for generalized Steiner tree or Steiner forest problem. So, let us continue that. So, let us briefly recall each edge have a cost c_e which is greater than equal to 0 for every edge. And we have k pairs of vertices $s_1, t_1, \ldots, s_k, t_k$ and I want to pick a set of edges of minimum total sum of cost such that in the graph induced by this set of edges all s_i is connected to t_i is connected for every for every $i \in [k]$.

Now, why this is a generalized Steiner tree problem? In the standard Steiner tree problem, we are given a set of terminals which we need to connect amongst themselves. In the standard Steiner tree problem, we are given a set of terminal vertices which we need to connect. Let us denote this terminal this set of terminal vertices as T. So, set T of terminal vertices which we need to connect with minimum cost So, how come this is a special case of the standard Steiner tree problem is a special case of generalized Steiner tree.

Because in the generalized Steiner tree you define for every i j so, 1 and cardinality So, suppose this set T of terminals be $\{1, 2, \ldots, l\}$. So, this is l. Now, for every pair of vertices you define $S = \{i\}$ and $T = \{j\}$. So, maybe for s_1 , t_1 then s_2 , t_2. So, you see we have l choose to pair.

So, in the generalized Steiner tree problem we have $\frac{l(l-1)}{2}$ pairs to connect. So, if we have a algorithm for generalized tiner tree problem, we can use that algorithm for the standard tiner tree problem that k will be replaced by cardinality of $\frac{t(t-1)}{2}$ ok. So, with this now let us resume our primal dual algorithm and we have observed that we need to bound the cardinality of $\delta(S)$ intersection if by some α to get an α factor approximation algorithm for all S in let us call that collection **S**. So, **S** is the set of all union of all those

s_i's ok. but it turns out that we cannot uniformly bound with some constant and here is an example.

So, consider a complete graph on $k+1$ vertices. So, let $V=\{1,\ldots,k+1\}$ ok, all the source vertices are 1. So, we have s_1 s_2 ,…, s_k all are the first vertex. On other hand $t_1=2$, $t_2=3$, $t_k=k+1$ and the cost of every edge is 1.

Cost of every edge is 1. Now, let us see where the primal dual algorithm get wrong or go wrong. So, in the beginning all vertices are isolated vertices and in the beginning the set of all connected components calc are $k+1$ and the algorithm picks one such component. So, let the algorithm picks C equal to 1 this component in the first iteration. then y_1 this dual variable will be increased to.

1 and that is all other dual variables will remain 0 throughout the algorithm. So, check that y_1 is the only nonzero dual variable at the end of the algorithm. So, let us see. So, in the first iteration this is vertex 1 and y_C is increased and all this edges. So, this is say 2 this is 3.

up to this is k all these edges becomes tight. So, the algorithm will simply keep picking all these edges in k iterations. So, the edges picked by the algorithm is 1 2 then 1 3 up to 1 k. So, these are the set of edges picked by the algorithm. Now, you see what is so, this is F.

Now, you see what is intersection of $\delta(C)$, but C for C take this one $\delta(C)$ and F this is k ok. So, we cannot uniformly bound. in any iteration we cannot say that for every set S the intersection of boundary edges of S with F this cardinality is small. It can be as high as k for some set although for other sets other singleton sets in the first iteration these are the other sets 2, 3 and so on. So, these are the candidate components in the first iteration you see for other sets the intersection is small.

So, however, for other connected components of the first iteration before any edge is picked, $\delta(C)$ other component $\delta(C)\cap F$ is only 1 ok. In particular this sum which we are actually interested to bound that $|\delta(i)\cap F|=1$ to this is $k+1$. So, let us make it $k+1$. So, for the connected component 1 this intersection is k, but for other k connected component this is exactly 1. So, this is 2 k in particular this sum is small and this is what we are interested in we are interested in not just 1 we are interested in the sum.

So, this suggests that the problem was we only increase one dual variable and that is it. So, the natural thing because this sum is small the average is small the average is is around 2 average intersection is around 2 to get the benefit of average it makes sense to

increase the dual variables of several sets in **C** simultaneously. ok. Again so, we increase all the dual variables in calc simultaneously and whenever a new edge the dual constraint corresponding to a new edge becomes tight, we pick that edge in our solution and we iterate. So, the same the remaining parts of primal dual algorithm remains same.

So, let us name the edges in the order they are added. So, let e_1 be the edge peaked in iteration 1, e_2 be the edge peaked in iteration 2, e_i be the edge picked in iteration i and so on. So, what why it is needed because it can happen that the solution when the dual when the primal dual method terminates the solution the set of edges need not be minimal. So, it may be possible to get rid of some edges and output a minimal solution and that is necessary because the edges the cost are greater than equal to 0. So, we can actually assume without loss of generality that costs are strictly greater than 0.

Because, all the edges which whose edge cost is 0, we can initially pick them and what is called the we can merge the both endpoints of it ok. So, that if that I leave it to you as an exercise. So, what we do after the algorithm the main loop of primal dual method terminates that means, if the set of edges is a Steiner forest, we see if we can remove any edge to make it minimal. Now, although the edges can be removed in any order without hampering approximation guarantee or running time, the analysis becomes much smaller much easier if we try to remove the edges from in the reverse order of their addition.

So, once the main loop of primer dual method terminates with a Steiner forest. if we go over the edges in reverse order of their inclusion or stack order. to check if any of them can be removed without disconnecting s_i and t_i ok. So, with this let us write down the pseudo code as usual we start with the dual feasible solution y equal to 0 and primal partial infeasible solution f equal to empty set and we keep track of the number of edges added. So, that that is needed in the last clean up step.

So, while not all s_i - t_i pairs are connected in (V, F) what do we do? We need to pick one edge. So, we call l+1 let **C** be the set of all connected components **C** of (V, F) such that cardinality $C \cap \{s_i, t_i\}$ this is exactly 1. Then we increase this dual variable simultaneously that is very important increase y_C for all C in **C** uniformly. you see cardinality of C is at most n number of candidate components. So, again in every iteration only at most n dual variables will be set to nonzero values and if number of iterations is at most m because in every iteration we will be adding an edge.

So, total number of dual variables which can ever be set to nonzero values is at most m times n which is polynomial meaning. So, we increase y_C for all C in **C** uniformly until for some edge $e_l \in \delta(C')\ C'$ in calc the dual constraint becomes tight. c_{e_l} cost of this edge

becomes equal to S such that $e_l \in \delta(S)$ y_S this becomes equality. And then as usual standard thing we include $e_l \in F$ the solution that we are building So, this finishes the while loop. So, at the end of the while loop F is a Steiner forest, but it may not be minimal.

So, what we do? We remove edges unnecessary edges from F in the reverse order of their addition. So, F' equal to F while or not while we will do in the reverse direction for $k = l, l-1, \ldots, 1$. If F' minus e_k is a feasible solution then we remove e_k from F' that is it and at the end we have a minimal solution. So, return F'. So, this is the primal dual algorithm.

So, here is a big theorem that we will prove the above algorithm has an approximation factor of at most 2. So, for that we need to have one lemma. So, let us write down the that in any iteration if you look at calc the set of all connected components which has exactly one of si and ti. So, for any C that means, for any iteration the corresponding C. iteration of the algorithm. If I look at the final solution and look at how many boundary edges it picks from all C in **C** and sum them. delta c intersection F' this sum is less than equal to twice cardinality C ok. So, these are lemma we need to show and assuming this lemma we will first prove this theorem and then we will prove this lemma ok. So, let us stop here. Thank you.