Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 11

Lecture 53

Lecture 53 : Approximation Algorithm for Multicut

Welcome. So, in the last class we have seen a $\frac{3}{2}$ factor approximation algorithm for the edge multi way cut problem. Today we will see another generalization of that problem ah which is called multi cut and we will see again randomized rounding based algorithm for that problem ok. So, let us start. multi cut problem. So, what is the input? Like edge multi way cut problem, we have a graph which is undirected edge weighted graph $G=(V,E)$ and cost on edges.

And instead of given giving k vertices $s_1,\ldots,s_k$ in multi way cut, we are given k pairs of vertices s_i - t_i. s_1-t_1 to s_k-t_k and the goal is to remove a min cost edge a set of minimum cost edges. So, that each s_i gets disconnected with t_i for every $i\in[k]$. compute a set if subset of E of edges such that s_i and t_i are disconnected for every $i\in[k]$ in the graph G equal to $(V,E\setminus F)$ ok.

And the cost $\sum c_e$ is minimized ok. So, as usual let us write the ILP formulation for that problem and we will get a LP relaxation for that problem from the ILP relaxation by relaxing the integrality constraints. we have a variable x_e for every edge $e\in E$ which takes value 1 if the edge e belongs to the cut. and 0 otherwise. So, the cost of the solution is $\sum c_e x_e$ which we want to minimize.

subject to what are the constraints? For every path between s_i and t_i for all i and k, I must pick at least one edge. So, let P_i be the set of paths between s_i and t_i for $i\in[k]$. So, for all $i\in[k]$ the constraint says that for all paths for $P\in P_i$ think of path P as a set of edges in the path $\sum x_e\geq 1$. So, these are the constraints and of course, we have the integrality constraint for every edge $e\in E$ x_e is in the set $\{0,1\}$. So, this is the exact formulation of the problem.

and ILP opt equal to LP opt. So, to get LP relaxation we remove the integrality constraint and replace it with x_e is in between 0 and 1. Again we see that because it is a minimization problem all costs are non-negative and x_e appears positively in the objective function. and setting any $x_e = 1$ satisfies the constraints involving those that variable x_e, we can again safely drop this constraint that $x_e \leq 1$, we can simply have x e greater than equal to 0. So, this is the LP relaxation and as usual we have LP opt because it is a minimization problem is less than equal to ILP opt which is equal to opt ok.

So, we can see that the number of constraints is exponentially many. and hence we need to have a polynomial time separation oracle for being able to solve this LP and get an optimal solution. Since, the number of constraints in the LP could be exponential in the number of vertices we need a polynomial time separation oracle. to solve the LP. How do I compute of I how do I design a polynomial time separation oracle? Recall what is a separation oracle? Separation oracle takes an assignment to the variables and either declares that this assignment is a feasible solution that means, it satisfies all the constraints or it outputs a constraint which the assignment violates.

So, let x_e $e \in E$ be a solution which may be infeasible and we need to design a polynomial time algorithm to check whether it is feasible or not. Now, what does this constraint says? This constraint says that if we define the weight of edge e to be x_e, then the shortest or the length of or the sum of the weights of the edges in every s_i to t_i path must be at least 1 which is same as saying the shortest path between s_i and t_i must be at least 1 where the length of edge e is defined to be x_e which is greater than equal to 0. and because shortest path can be computed in polynomial time. So, either we conclude that between every pair of vertices s_i and $t_i$the length of the shortest path is at least 1 or if not if there exist an i between which the distance or the length of shortest path is less than 1, then I can find that shortest path and that path gives me a constraint which is violated by this solution. So, let us write that down define a define weight of edge e to be x_e compute the distance between s_i and t_i for all $i \in [k]$.

using x_e as the weight of edge e. If the distance between s_i and t_i is at least 1 for every $i \in [k]$, then x_e, $e \in E$ is a feasible solution. Otherwise there exists an index $i \in [k]$ and path $P \in P_i$ such that $\sum_{e \in P} x_e < 1$. Such a path P can be computed in polynomial time using standard shortest path algorithms like Dijkstra's algorithm ok. So, we have a polynomial time separation oracle, we solve the LP let $(x_e^*)_{e \in E}$ be an optimal solution.

Now, we use this axis to have a randomized rounding based algorithm and the idea is similar to the algorithm for multi way cut that we will design balls and take the edges that

cross the balls. Those are the boundary edges of balls that the same idea we will deploy here also, but to define the ball we need a notion of distance and that is slightly different here. So, we define distance or metric with respect to this x^*. ok d_{x^*} 2 vertices u and v is the distance between u and v using x_e^* as the length of the edge e.

Take it as a homework to prove that d_{x^*} defines a metric on V the set of vertices. prove that d_{x^*} is a metric or semi metric. So, we will use this terms metric and semi metric alternatively. so here semi metric on V ok. So, now, again we define balls.

So, $B_{x^*}(s_i, r)$ is the radius again we will drop x star from this notation for brevity. So, we will write it as $B(s_i, r)$ is as usual the set of vertices $v \in V$ whose distance under this metric from s_i is at most r ok. So, this is we call the ball of radius r around s_i. good. Additionally there is another analogy that is useful for this algorithm.

Additionally it is useful to view each edge $e \in E$ as a pipe of length x_e^* and cross sectional area c_e. Hence, $c_e x_e^*$ which is the contribution of this edge e in the objective function, which is the contribution of edge e. to the objective function is the volume of the pipe So, the goal of the LP is to connect this vertices with minimum volume. So, that the distance between every s_i-t_i pair is at least 1. So, LP the goal of LP is to compute a minimum volume pipe system so that distance under this metric $d_{x^*}(s_i, t_i)$is at least 1 for every $i \in [k]$ ok.

So, is nothing, but let us denote it by V^* the minimum volume which is $\sum_{e \in E} c_e x_e^*$. So, we define volume of pipe volume of all pipes in the radius of r around a vertex s_i to be $V(s_i, r)$ be the pipes of pipes in the ball of radius r around s_i plus $\frac{V^*}{k}$. We will see why it is needed it will be useful in the analysis. So, $V(s_i, r)$ is $\frac{V^*}{k}$ plus $\sum_{e=\{u,v\}: u,v \in B(s_i,r)} c_e x_e$.

So, these are the edges whose both endpoints are contained in the ball, but there could be edges. So, here is s_i and this is the ball of radius r and so, this is distance r. So, there could be edges whose both end points are contained in this ball, but there could also be an edge whose one end point is within in in the ball and the other end point is outside the ball. In those cases we will add the volume of the part of the pipe which is within the ball of radius r. So, which is edge e equal to $\{u, v\}$ such that u is in ball around s_i of radius r, v is not in ball around s_i of radius r, Now, what is this distance? So, this distance is x_e if this edge is e and we will take till r.

So, this is r minus distance of s_i to u. ok. So, this distance is $d(s_i, u)$ and we will take r more ok. So, this is we define $V(s_i, r)$. So, in the next lecture we will see why this quantities are interesting and how we can use this quantities to design and $O(\log k)$ factor approximation algorithm for this multicut problem ok.

So, let us stop here. Thank you.