Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 09

Lecture 45

Lecture 45 : Primal-dual Algorithm for Minimum Weighted Feedback Vertex Set Contd.

Welcome. So, in the last class we have started seeing the primal dual algorithm for minimum weight feedback vertex set problem. So, let us continue that. Minimum weight feedback vertex set. So, we discussed a primal dual based algorithm very natural algorithm.

So, let us write the pseudo code of that algorithm. So, we started with the dual feasible solution y equal to 0. So, y is a vector y equal to 0 means that all y_C's are 0 and the primal partial incomplete solution is s equal to empty set. While there exists a cycle C in $G[V \setminus S]$.

What we do? We increase the dual variable until one dual constraint involving C, y_C becomes tight. This is the natural primal dual method increase y_C until for some $l \in C$ the constraint becomes tight. So, what is the constraint? Summation all cycles c prime in calc such that $l \in C'$, $y_{C'} \leq w_l$, but this becomes tight that means, it becomes equal to w_l. that is the first step once for some vertex l it becomes tight I pick one such vertex in my solution. See that it is it may be possible that there are more than one vertex for which this constraint becomes tight in that case also I pick only one vertex in my solution.

So, $S = S \cup \{l\}$. Then remove l from G, then repeatedly remove all degree 1 vertices from G that is it and then return S. Why I say repeatedly remove? It may be possible that after removing 1 degree 1 vertex or some degree 1 vertices. there are new degree 1 vertices. For example, if you look at a path and the last vertex is degree 1, if you remove that vertex the second last vertex becomes degree 1.

So, keep doing this keep removing degree 1 vertices old and new until my graph G does not have any degree 1 vertex. So, this is the natural primal dual algorithm for minimum weighted feedback vertex set problem. So, what is ALG? ALG is summation of the weights of the vertices in S. Now, all this vertices in my solution S are tied vertices that means, for all those w_i, $w_i = \sum y_{C'}$ this. So, this I replace with $i \in S$ summation cycle $C \in \mathcal{C}$ such that i belongs to the cycle C, y_C ok this is because of the choice of i.

Then whenever we have 2 whenever we have double sum it may often make sense to exchange double sums. So, let us do that exchange. So, $C \in \mathcal{C}$ and how many times yc appears? It is the number of vertices of in the cycle C which are picked in S. So, this is $|C \cap S| y_C$. Now in the standard analysis if we can bound cardinality of C intersection S by α then we get we can write this is less than equal to $\alpha \sum_{C \in \mathcal{C}} y_C$ ok and because y_C is a dual feasible solution by weak duality we know that this is $\alpha \sum_{C \in \mathcal{C}} y_C$ is less than equal to LP-opt of primal LP this is less than equal to α times LP-opt which is less than equal to opt. So, this way we get an alpha factor approximation algorithm here if it is the case that $|C \cap S| \leq \alpha$ for all cycle C in this graph.

So, now let us see what we can hope for. So, one way to bound the number of vertices picked from a cycle is by the size of the cycle itself. So, we know that $|C \cap S| \leq |C|$. So, a natural modification of the algorithm which ensures small alpha is to pick only small cycles. So, how about the algorithm that instead of picking any cycle.

pick the smallest cycle and then iterate. So, if it happens that every times we are able to find a cycle with small length with length at most alpha, then we get an alpha factor approximation algorithm. But unfortunately this idea also fails because it may be the case that the graph is just one big cycle containing only n vertices. So, a graph may not have any small cycle. So, what to do then? But, then you observe that if a graph is just one big cycle, then the feedback vertex set is just one vertex.

Pick any vertex from that cycle and that is a feedback vertex set, pick one minimum weight vertex that is the solution. So, that is not really the hard instance. And in particular if we have sequence of a path of degree to vertices at most one vertex can be in a solution. So, long cycles are not our problem, our problem are cycles with many vertices whose degree is greater than equal to 3. So, let us formalize this idea first make this observation for any path P of vertices of degree at most 2.

our algorithm picks at most one vertex from P. that is intersection of P and S contains at most one vertex. In the final solution where S is the feedback vertex set output by our algorithm. So, why it is the case? First of all we can assume that the in the path P the degree of every vertex is exactly 2 because if there is a degree 1 vertex in the path P then because of our pre processing rule we are repeatedly removing degree 1 vertices the entire path gets removed and no vertex from the path becomes part of solution is not picked by the solution is picked by the algorithm. So, if we have a path P of all vertices of degree 2, at most one vertex can be picked because whenever we pick one vertex and delete that from the graph G, then those path breaks into two paths each has exactly 1 or at most 1 at least 1 degree 1 vertex and if a path has a degree 1 vertex by the preprocessing rule entire path gets removed.

So, take it as a homework to write the proof formally. So, even if I have a long cycle, but it has only few degree vertices of degree more than 2, then also it is fine because in any cycle if you have focus on vertices of degree greater than 2 the paths alternate. So, let me write. any cycle contains alternating paths degree 2 vertices separated by vertices of degree 3 or more. is just a simple observation if you look at any cycle.

So, if this is a degree to path, then at this end points. there will be vertices of degree at least 3 or say the same thing or differently you have you take a cycle c and highlight the vertices of degree 3 or more. Now, the in between these regions will be paths. let us formalize this in the following lemma and now you see if I have a. So, if not if a cycle has k vertices cycle C has k vertices of degree 3 or more then it has at most k maximal paths of vertices of degree 2 ok.

Maximal paths means that those paths cannot be extended in either direction on the cycle itself. these cycles are fine because if it has a degree k it has k vertices of degree 3 or more then the number of vertices of this cycle that can be part of the solution will be at most twice k because this k high degree vertices will be part of solution and for each path at most one vertex can come in the solution. moreover C intersection S where S is the feedback vertex set output by the algorithm is at most $2k$ ok. So, I have discussed the proof you can please write the proof formally it is a good homework and a training for writing mathematical proof formally. So, what I need is cycles with large number with small number of high degree vertices.

So, does there exist a cycle with small number of high degree vertices? So, next lemma says There always exist a cycle of length cycle containing at most $2\log n$ many high degree vertices. So, here is the lemma that in any graph g that has no vertices of degree 1. this is the preprocess that is why we need that preprocessing rule of or clean up rule of removing all degree 1 vertices repeatedly so, that we can apply this lemma. $2\lceil \log_2 n\rceil$ vertices of degree at least 3. Moreover there is a cycle C.

Moreover such a cycle C can be computed in linear time in $O(|V|+|E|)$ the number of vertices plus edges. So, let us go to the proof. So, a variant of breadth first search, where we treat every maximal path of vertices degree 2 as edges ok and that is it. So, in the BFS tree in the modified BFS tree the degree of every vertex is at least 3. in particular every node has at least 2 children.

in the modified BFS tree ok. Now, if each node has 2 children the height of the BFS tree could be at most $\log n$. Hence the height of the BFS tree is at most ceiling of log of n. So, if I do not write base it should be assumed that base is 2 ok. So, now, if there indeed exist a cycle which contains at least one vertex of degree 3 or then there will be a cycle or cross edge then we will find a cycle while constructing the modified BFS tree at some level l.

So, think of this modified BFS tree I am growing it and at some level I find a cross edge at this level is l. So, whenever I find a cross edge you see that I got one cycle. The length of such a cycle is at most $2l$ ok, which is at most because l is at most filling of $\log n$ this length of such a cycle is at most of $2\log n$. Now, each node corresponds to a node of degree at least 3 in the original graph and each edge corresponds to either an edge or a path of maximal path of vertices of degree 2. So, since each edge either corresponds to an edge of G or a maximal path of vertices of degree 2 such a cycle let us give it a name C, the cycle C corresponds to a cycle in G with at most twice ceiling of $\log n$ vertices of degree 3 or more.

Now, since breadth first search can be constructed in linear time this algorithm also can be computed can be executed in linear time and hence in linear time we can compute a cycle with at most twice ceiling of $\log n$ many vertices of degree 3 or more. So, the next class we will see how using these cycles we can get a $4\log n$ approximation algorithm for minimum weight feedback vertex set problem. Thank you.