Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 06

Lecture 26

Lecture 26 : A Polynomial Time Separation Oracle for Scheduling Weighted Jobs on a Single Machine

Welcome. So, in the last class we have seen a three factor approximation algorithm for scheduling jobs into one machine. to minimize the weighted completion times and for that we have used a linear programming relaxation in that linear program we had exponentially many constraints. So, today we will see how we can solve that linear programs those large linear programs in polynomial time. So, that is the topic of today's lecture. large linear programs in polynomial time.

And, the idea is there are various algorithms for solving linear programs in polynomial time and there is a particular algorithm which is called ellipsoid method. So, we use ellipsoid method. which not only solves solves means computes an optimal solution linear program in polynomial but it does not need the explicit description. of the linear program.

That is the most important point about ellipsoid method, it does like unlike typical algorithms it does not need the all the constraints and everything to be explicitly. Then what does it need? So, it only needs access to polynomial time separation oracle. Now what is it? So, again suppose let us take a general linear program. So, let us consider general linear program. like say minimize $\sum_{j=1}^{n} d_j x_j$ subject $\sum a_{ij} x_j \geq b_i$ j equal to 1 to n and we have m such constraints and x_j is greater than equal to 0 j in any case.

The running time of the ellipsoid method does not depend on m, it depends on n and the runtime of a polynomial time separation oracle. Now, what is a separation oracle? Given an assignment $\bar{x} \in R^n$ assignment of the variables, the polynomial time separation oracle oracle either outputs that $\bar{x}$ is a feasible point that means, it satisfies all constraints it outputs an inequality that $\bar{x}$ violates. in polynomial time in polynomial in n that is very important because the input to the polynomial time separation oracle is $\bar{x}$. Again it does not take all the constraints at as input it only takes $\bar{x}$ which is an n dimensional real vector

n time. in particular the runtime of the oracle is independent of the number of constraints, the number m of constraints ok.

Also in particular its input is just an n dimensional real vector, its input is an n dimensional real vector. So, what ellipsoid method does? all it needs it is a access to polynomial time separation oracle. That means, it needs it will give that oracle an assignment to the variables and it that oracle needs to tell whether it is a feasible solution or it gives a constraint which the solution which the assignment violates. and if that is the case then the ellipsoid method can solve the linear program in polynomial in n time. So, given polynomial time separation oracle.

the ellipsoid method solves an LP in polynomial in number of variables. time . In particular time does not depend on the number of constraints. The run time is independent of the run time of the ellipsoid method is independent of the number of constraints, runtime of ellipsoid method. So, next what we will show is that for the linear programming relaxation for minimizing the weighted completion time of the jobs, we will give a polynomial time separation oracle.

So, that is the next part of the lecture. Polynomial time separation oracle for the LP relaxation for minimizing weighted completion times. So, what is the job given an assignment to the variables which are c_j s those are the variables j equal to 1 to n, n was the number of jobs. We need to verify whether it satisfies all the constraints, whether it is a feasible solution and if not then we need to output a set for which it violates remember. So, let us recall the LP.

recall the LP minimize $\sum_{j=1}^{n} w_j C_j$ subject to C_j should be less than equal to C_j should be greater than equal to r_j it is the release time when it is time when it is released plus p_j this is for all jobs. So, given any assignment we can check whether it satisfies all these constraints all these n constraints in polynomial time because these are small number of constraints. The larger constraints are we have many constraints for the subsets. So, given an assignment we will assume without loss of generality that it satisfies all these first n constraints because if not then we can output simply one such constraint that it violates. Otherwise if it satisfies all constraints then we will cleverly check whether it satisfies all these $2^n - 1$ constraints.

And what are the constraints? Let us write it down this is $\sum_{j \in S} p_j C_j \geq \frac{1}{p}(S)^2$. Note that $p(S)$ is like constant because they depends on input that is not variable only C_j's are variable. So, although there is $p(S)^2$ it is like constant and it is a linear program and of

course, we have $C_j$is greater than equal to 0 for all j in n. So, these are not required because we have $C_j \geq r_j + p_j$ and p_j's are positive r_j's are greater than equal to 0 ok. So, we need to design a polynomial time separation oracle for this.

For that let $C_1, \ldots, C_n$ be an assignment to these n variables. again by renaming we can assume that $C_1 < C_2 < C_3 < \ldots < C_n$. So, by renaming the jobs by renaming the variables we assume without loss of generality that $C_1 < C_2 < C_3 < \ldots < C_n$ ok. Now, we define this sets $S_1 = \{1\}$. $S_1 = \{1,2\}$ $S_1 = \{1, \ldots, i\}$ and $S_1 = \{1, \ldots, n\}$.

So, if we have defined this n subsets of jobs and the and what we will prove is that if the if this variables satisfy the constraints corresponding to this n sets, then it satisfies the constraints corresponding to all subsets. And, hence it is enough to check whether any of the constraint corresponding to these sets are violated, because if the constraints corresponding to these sets are all satisfied then we will show that all for all subsets the constraints are satisfied. and if not if we found one of this n sets which for which the constraint is not satisfied then we have got a violating constraint and that the polynomial time separation oracle our oracle outputs ok. So, the only because there are n sets only there are n constraints. So, there are $2n$ total constraints that this oracle checks.

So, it runs in polynomial time. So, all we need to show is that it is enough to check this n sets. So, here is a lemma, if the let us call this constraints say 1, if the constraints in 1 are satisfied for $S_1, S_2, \ldots, S_n$ then 1 is satisfied for all subsets $S \subseteq [n]$ proof. So, as usual what do we say that when do we say that a constraint is not satisfied. So, a constraint for a set is not satisfied set S is not satisfied if.

So, what is the constraint? That . $\sum_{j \in S} p_j C_j \geq \frac{1}{p}(S)^2$ So, if it is not satisfied that means, $\sum_{j \in S} p_j C_j < \frac{1}{p}(S)^2$. If $\sum_{j \in S} p_j C_j < \frac{1}{p}(S)^2$ ok. So, if so, we will show that if there exist a set S there which does not satisfy the constraint that means, for a set S for which $\sum_{j \in S} p_j C_j < \frac{1}{p}(S)^2$, then we show that there exist a set among $S_1, \ldots, S_n$ for which violates this constraint. So, we will show that if there exists a set subset $S \subseteq [n]$ such that $\sum_{j \in S} p_j C_j < \frac{1}{p}(S)^2$ then there exist an $i \in [n]$ such that that set S_i violates this constraint ok and then that is enough ok.

So, we do it by starting with a set S assuming there exists such a set S and we iteratively either remove elements or introduce elements and ultimately get to a set S_i where it violates the constraint ok. So, if S violates the constraint, then what do we have

$\sum_{j\in S} p_j C_j - \frac{1}{p}(S)^2$ is negative, then $\sum_{j\in S} p_j C_j - \frac{1}{p}(S)^2 < 0$ Now, any change Now, if we make any change which reduce this quantity further then again then we get another set for which this condition is again violated. So, we make changes to S changes means either add element or remove element. Any any changes made to S that decreases this quantity $\sum_{j\in S} p_j C_j - \frac{1}{p}(S)^2$ results in another set results in another set is prime which also violates the constraint ok. So, let us see what is the effect of when we can remove a job j or a job k say. Suppose, we have this for a set S and if I remove job k what property it should satisfy so, that this quantity decreases. So, removing a job k, removing job k decreases $\sum_{j\in S} p_j C_j - \frac{1}{p}(S)^2$ which is already negative to begin with if S violates it. Removing job k decreases it further makes it again more negative if can it is a easy exercise take it as a homework easy math.

If $-p_k C_k + p_k p(S-k) + \frac{1}{2} p_k^2 < 0$. That is $C_k > p(S-k) + \frac{1}{2} p_k$. So, we can remove a job k from S and still the condition will get violated in the resulting set if $C_k > p(S-k) + \frac{1}{2} p_k$. Now, study when we can add a job. Adding job k decreases $\sum_{j\in S} p_j C_j - \frac{1}{p}(S)^2$ if $p_k C_k - p_k p(S) - \frac{1}{2} p_k^2 < 0$ which is same as saying $C_k < p(S) + \frac{1}{2} p_k$ ok.

So, now, we have started with a job set S for which it is violated. So, let l be the highest indexed such that this job l in such that job l is in S ok. So, we can remove l from S if this condition satisfied. So, remove l from S if $C_l > p(S-l) + \frac{1}{2} p_l$. So, we keep removing these jobs until this condition or when this condition is satisfied and suppose we stop at S' when this condition is no longer satisfied.

So, from that we get we keep removing until for the highest index job l, we have $C_l \leq p(S'-l) + \frac{1}{2} p_l$, where l is the highest index job in S'. Next what I do? So, if $S' \neq S_l = \{1, \ldots, l\}$. So, if $S' = S_l$ we do not need to do anything we have got we have proved that there exist a set S_l which violates this constraint. Otherwise if this then let us take a $k < l$ and $k \notin S$. Now notice that because we have indexed the job based on their finish time, their variable assignment C_k we have $C_k \leq C_l < p(S') < p(S') + \frac{1}{2} p_k$, . Now, this is exactly the condition I need for adding a job I can add job k to the set if

$C_k < p(S') + \frac{1}{2} p_k$ which is exactly this.

So, we can add job k and not only job k we can keep adding all the jobs in 1 to l which are not in S' because all such jobs satisfy this condition. and we keep on adding and then what we show that this S_l what we have obtained is S_l violates this constraint. Hence, we keep on adding all jobs in $S_l - S'$ and because of the argument we have shown that S' does not satisfy the constraint. So, what we have shown is that if there exist a set which violates the constraint, there exist a set among $S_1, \ldots, S_n$ which violates the constraint, which proves the correctness of polynomial type separation order. So, let us stop here. Thank you.