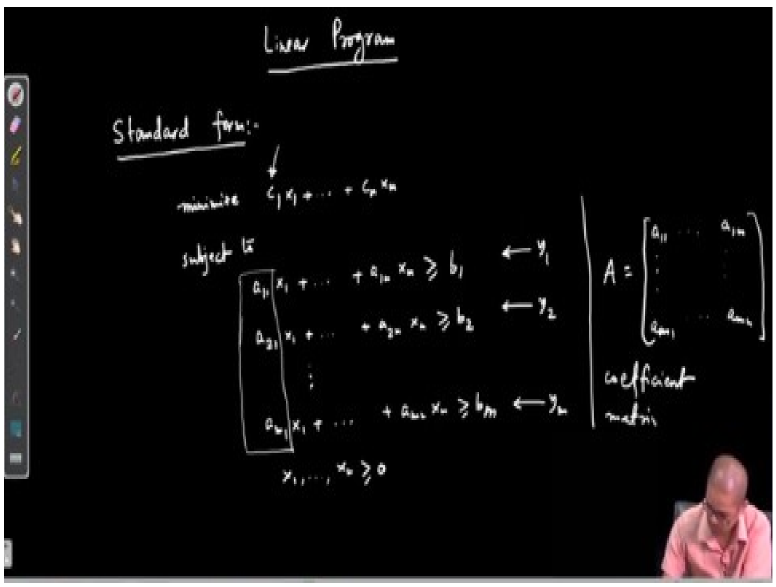
Selected Topics in Algorithm
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 48
Introduction to Linear Program (Continued)

Welcome. So, in the last class we have started studying linear program and we are doing some basic background of preliminary of linear programs. We will continue that in this class also.

(Refer Slide Time: 00:42)

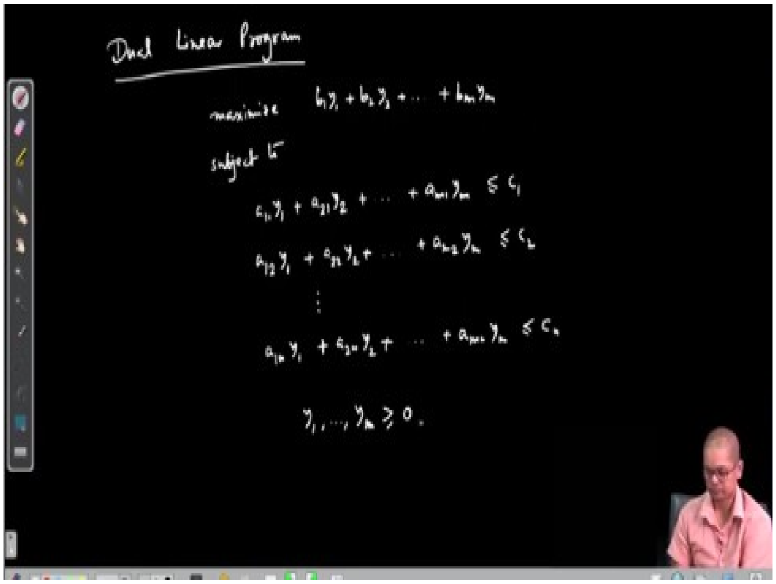


So, there is something called standard form of a linear program. It turns out that every linear program can be written as this standard form. What is this form? This minimize $b_1x_1+...+b_nx_n$ subject to linear constraints $a_{11}x_1+...+a_{1n}x_n \geq b_1$ and let us change this name $c_1,...,c_n$ $a_{21}x_1+...+a_{2n}x_n \geq b_2$. So, $a_{m1}x_1+...+a_{mn}x_n \geq b_m$ and this variables $x_1,...,x_n \geq 0$.

So, this is the standard form and it turns out that every a linear program it could be a maximization program or any other program can be converted into this sort of form with equivalent or transformation. So, for example we can define $x_1=-x_2$ and the $-y_1$ and so on so with just this sort of linear transformation we can convert or write down any linear program into this standard linear program.

And that is why this linear programming theory has been developed with respect to this standard form. With respect to the standard form what is the dual linear program?

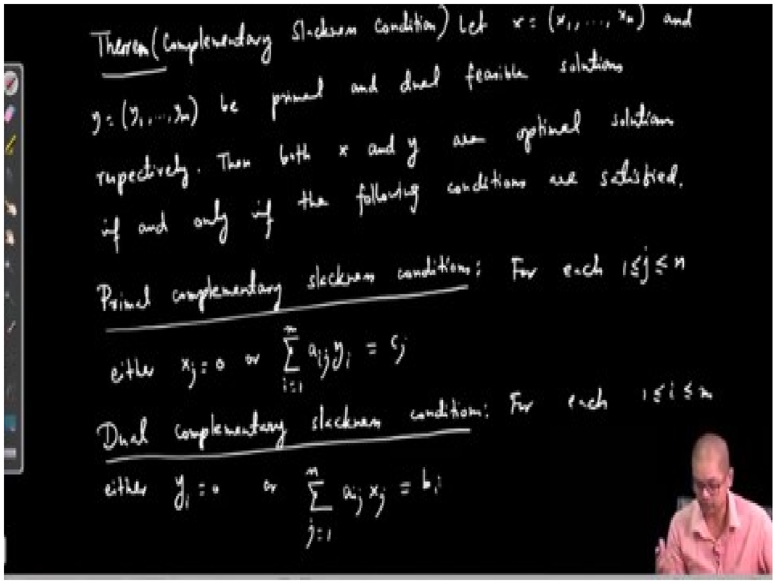
(Refer Slide Time: 03:38)



So, again let us follow the same approach. We multiply first equation with y_1, second equation with y_2 and the m-th equation with y_m. So, we maximize $b_1 y_1 + b_2 y_2 + ... + b_m y_m$ subject to, what are the conditions that when I multiply these the coefficients of x_1 multiply and sum this m equation the coefficient of x_1 should be less than equal to the coefficient of x_1 here. So, the condition is $a_{11} y_1 + a_{21} y_2 + ... + a_{m1} y_m \leq c_1$.

Similarly, $a_{12} y_1 + a_{22} y_2 + ... + a_{m2} y_m \leq c_2$ so on the last one is $a_{1n} y_1 + a_{2n} y_2 + ... + a_{mn} y_m \leq c_n$ and of course $y_1, ..., y_m$ this is greater than equal to 0.

(Refer Slide Time: 06:35)



Now we state an important result which is called complementary slackness. Complementary slackness condition it gives us a certain criteria when a solution a feasible solution is an optimal solution. So, by the way so this matrix this is called so this matrix if I write matrix A this is called the coefficient matrix. So, let x that means $(x_1, \ldots, x_n)$ and $y=(y_1, \ldots, y_m)$ be primal and dual feasible solutions respectively.

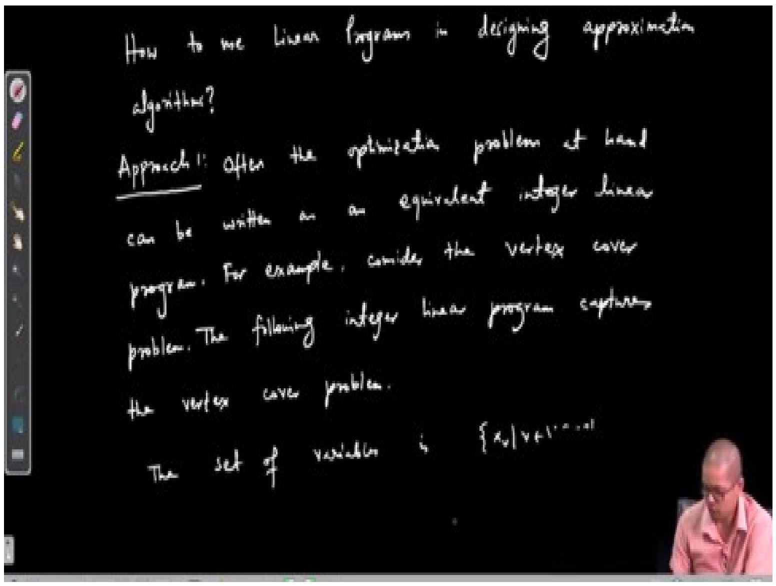
Then both x and y are optimal solutions if and only if the following conditions are satisfied. So, what are the conditions? We have two set of conditions one is primal complementary slackness conditions. So, what are primal complementary slackness conditions? It is like it says for each variable j from 1 to n primal as the variables $x_1, \ldots, x_n$ either x_j is 0 or now notice that for each variable x_j there is a constraint in the dual.

And the primal complementary slackness condition says that either the variable is 0 or if the variable is not 0 then the corresponding constraint inequality must hold with equality or summation if x_t is not equal to 0 the jth constraint corresponding to c_j this constraint should hold with equality. So, this is the primal complementary slackness conditions.

These conditions should be satisfied and the dual complementary slackness condition. This is because you know that dual of dual program is primal what do you mean by that you convert this dual program into standard form and then write a dual of it and it will be same as the primal program. So, here also for each the variables are y 1 to y m for each i from 1 to m either $y_i = 0$ or the ith constraint which is $\sum_{j=1}^{n} a_{ij} x_j = b_i$.

So, this is the complementary slackness condition, this will be use in our design of approximation algorithms.

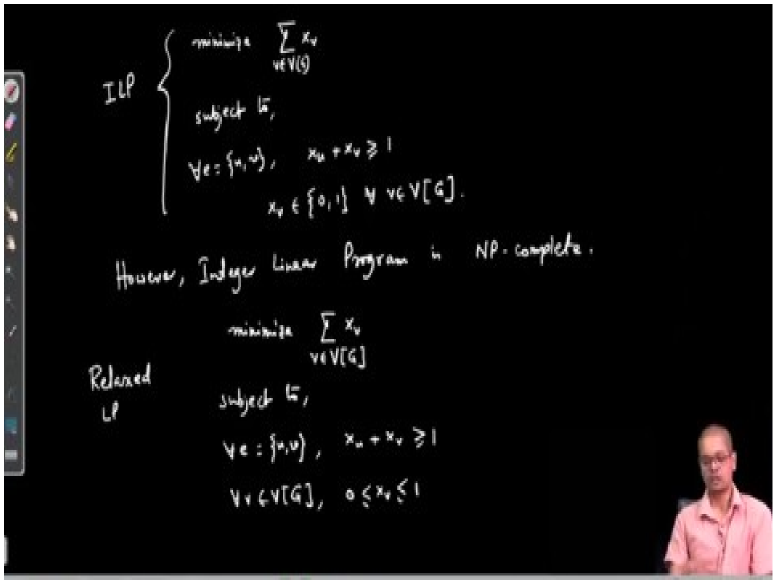
(Refer Slide Time: 13:49)



Now what is the high-level idea of how to use linear programs in designing approximation algorithms. So, there are two broad approaches, approach one often the optimization problem at hand can be written as an equivalent integer linear program. Now what do you mean by that? Integer linear program is a linear program which has a linear again like linear program it has a set of variables and we have optimization function which is a linear function over the variables and we have a linear set of constraints.

But the variables some of the variables can take only integer values. So, let us understand this with an example. So, for that let us recall the vertex cover problem. For example, consider the vertex cover problem. Recall in the vertex cover problem we are given a graph G and we are looking for a vertex cover of minimum size. So, for that so the following integer linear program captures the vertex cover problem. The set of variables is x_v where v in set of vertices.

(Refer Slide Time: 18:08)



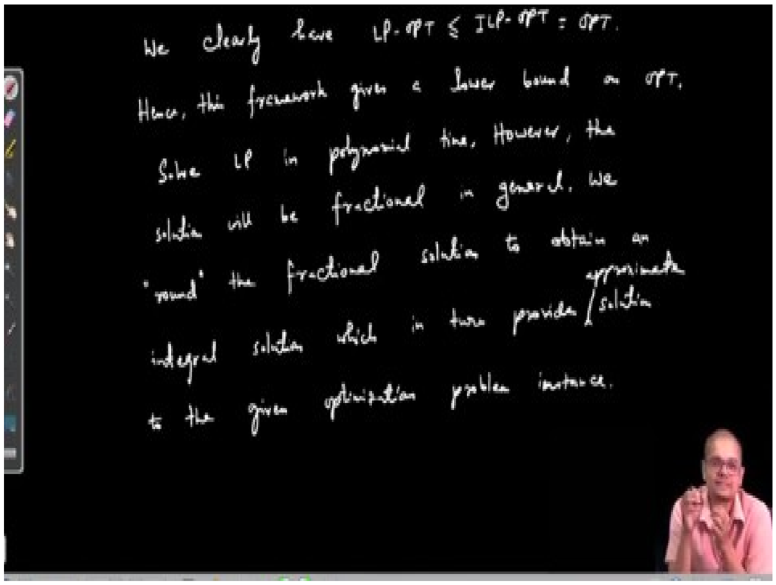
So, for each vertex in the graph G we have a variable and the idea is that variable takes only 0 1 values either 0 or 1 and if we pick it in the vertex cover then we set the variable to be one. So, to find the minimum size vertex cover we minimize summation x_v, v into G or the constraints subject to you know for each edge, one of its vertices one of its endpoints must be picked. So, that means $x_u + x_v \geq 1$.

For each edge so it I have m edges I have m number of constraints and the variables take value in 0, 1 and linear program variables take value in a real interval. But you know in integer linear program variables can take a value only in integers so this is for all V into $V[G]$. So, this is a ILP formulation of the vertex cover problem. So, it turns out that many optimization problems can be written as an integer linear program.

Now what we do? But the integral linear program solving it is NP hard. Finding an optimal value is NP hard we do not hope to have a polynomial time algorithm for integer linear programming unless P = NP. However, integer linear programs is NP complete. So, what we do is that we relax the integrality constraint. So, this is the ILP integral linear program. Now we relaxed it to an ILP. What is the relaxation?

Everything is same except the variables instead of allowing them to take value only 0 and 1, we allow them to take any real value in between 0 and 1. Minimize $\sum_{v \in V[G]} x_v$ subject to for each edge $e = \{u, v\}$, $x_u + x_v \geq 1$ and for each vertex $v \in V[G]$, x_v is in between 0 and 1.

(Refer Slide Time: 22:25)



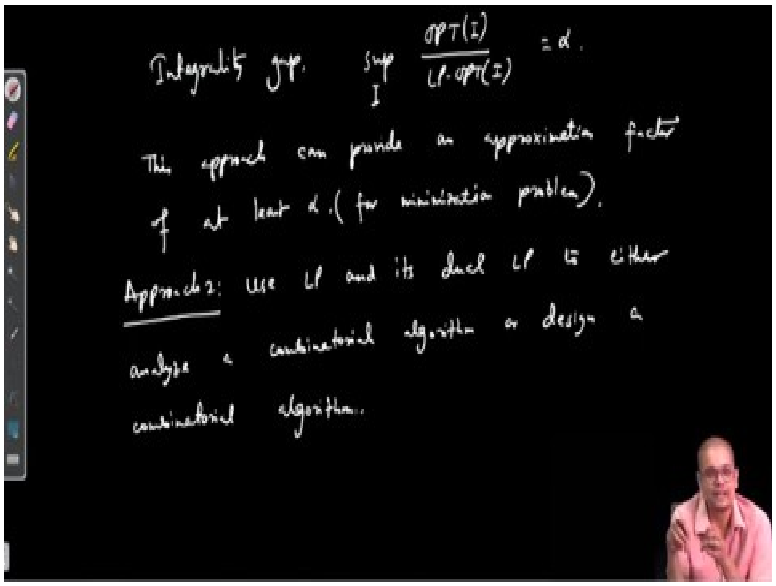
Now clearly because the search space is more the search space is a superset for integer for linear program compared to the integer program, we clearly have LP OPT is less than equal to ILP OPT. But ILP OPT is nothing but OPT because ILP is a equivalent formulation of the given optimization problem. So, using this framework we get the lower bound that we are looking for. Hence this framework gives lower bound on OPT which is often the first step in any design of approximation algorithm.

So, next what to what I have then you know we solve LP. So, idea is solve LP in polynomial time that can often be done because linear programming is NP every linear program can be solved in polynomial time that means an optimal solution can be found in polynomial time and then this solution will have a fractional solution. So, however the solution will be fractional in general. So, then what we do is that we apply in something what is called rounding technique.

From this fractional solution we create a solution of the optimization problem. We create a solution of ILP which will automatically give us a solution of the optimization problem but which may not be the optimal solution. So, the final step is we round the fractional solution to obtain an integral solution which in turn provides solution to the given optimization problem instance provides approximate solution given optimization problem instance.

So, here in this approach you know the ratio between LP OPT and ILP OPT this is the lower bound on these are bound on maximum approximation ratio possible. So, this is called integrality gap.

(Refer Slide Time: 26:59)



Integrality gap is called supremum for all instances you know ILP OPT or opt of I by LP OPT because we are using our LP OPT as the lower bound. So, this alpha this approach can provide an approximation factor of at least alpha for minimization problems. This is for minimization problems and the second approach is use this primal dual approach for designing or analysing combinatorial algorithm.

So, approach 2, use LP and its dual LP to either analyse a combinatorial algorithm. An algorithm is loosely speaking it is called combinatorial if it is does not solve a linear program to either analyse a combinatorial algorithm or design combinatorial algorithm. So, in this approach we do not directly solve linear program. So, in the next couple of lectures we will see design of approximation algorithms using both the two approaches. Thank you.