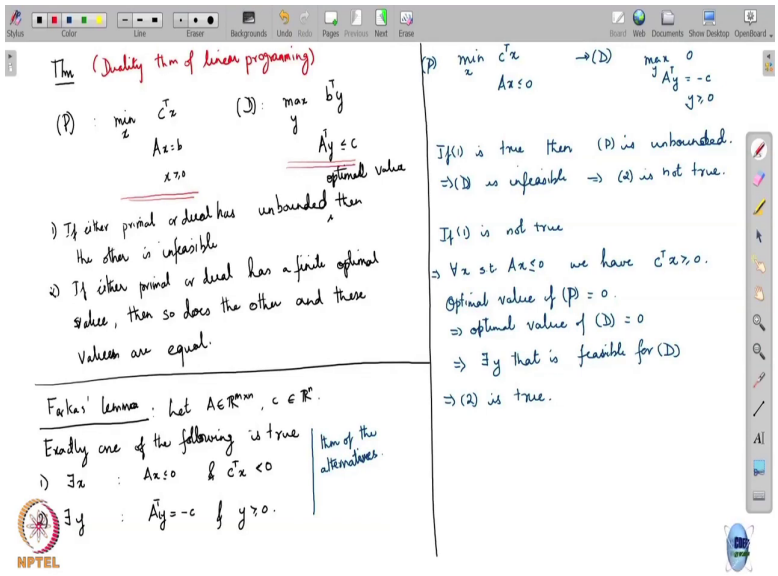
Optimization from Fundamentals
Prof. Ankur Kulkarni
Department of Systems and Control Engineering
Indian Institute of Technology, Bombay

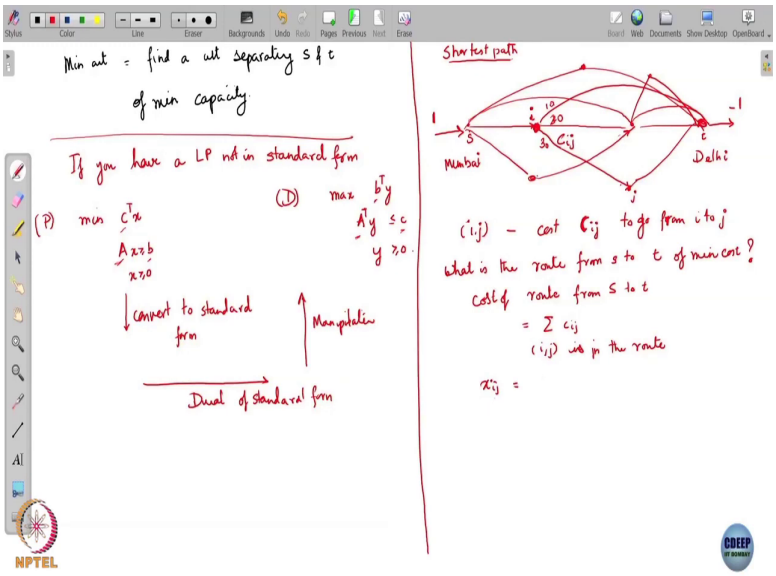
Lecture - 13C
Shortest path problem

(Refer Slide Time: 00:22)



Now, I let me also tell you a few other things related to this; I remember I mentioned. So, we wrote out the LP for, we wrote out the LP duality theorem we assuming this particular form. This particular form for the primal LP that we took it as standard form, because every problem can be reduced to this sort of standard form. And then you get it is, you can and this was the dual of that particular of the standard form LP.

(Refer Slide Time: 00:52)



Now, if you have a LP which is not in standard form ok; if you have LP not in standard form, not in standard forms, say for example, say suppose you have a LP that looks like this. Say suppose you have a LP that is say suppose this is your LP. Now, what is the dual of this one? It turns out, so the way to get to the dual. So, this LP would, this primal will also have a dual, ok.

Now, its dual will not take the same form as the standard form dual; its dual will be will take a different form. But the way to get there is that, you can convert this to standard form; then do a dual of standard form. And usually after this some manipulations are needed ok; some, because usually this will lead to a some redundant variables which you can eliminate easily and so on, do a few manipulations if possible.

And then that gives you that, would give you the dual of that, will give you a dual problem. Now, what is the dual of this one? It turns out its turns out it is maximizing b transpose y, A transpose y less than equal to c and y greater than equal to 0; this is the form of the dual of this particular LP, ok.

You can as I said consider many other different a linear programs and find their find the standard, find their duals by just converting them to this sort of by converting them to standard form and then manipulations and so on, yes.

Student: (Refer Time: 03:23).

Yes, this is the same A as this A, the b is the same as this b, c is the same as this c.

Student: No everything (Refer Time: 03:32).

Yes, some variables will get dropped; yes it is possible that some variables will get dropped.

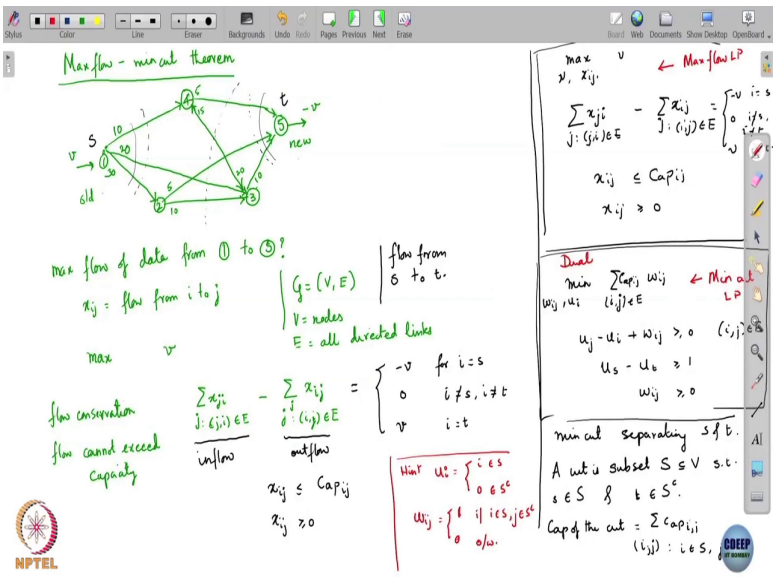
Student: (Refer Time: 03:38).

No, so it will come you can bring it to, bring it down to this form. So, this is also another sort of saddle point. See an optimization problem can have many different forms that are all giving the same optimal value. Remember at the end of the day, the duality theorem only pertains to the value of the optimization problem. There you can always introduce additional variables, you know manipulate remove variables etcetera etceteras, while not changing the optimal value, right.

So, there is no; there is no fixed form for an optimization problem that is a matter; but you all we know is that, yes this is a form for the dual, alright. So, this is how you can, if you have a LP in that is not in standard form; then you can all, this is the route by which you can get its dual, alright ok.

So, as the last topic on linear programming, let me also, let me mention to you one other thing, which is which another type of problem which comes up which can be written as a linear program, that problem is problem of shortest path.

(Refer Slide Time: 05:00)



So, the problem that I wrote here in the in this the network problem here was that, I you had an old phone and you had a new phone. You want to transfer data from an old phone to the new phone. But then the links that over which the data could flow had limitations of capacity, right.

So, you could not push data beyond a certain; beyond a certain MB per second along these links, right. So, there is a different type of problem which is of the following kind that, you have a source node; you have a destination node ok, say the source node is say Mumbai.

And the destination node is say suppose Delhi; you have, you can go from this source node to this destination node over many different possible paths. One way is that, you can go till, you can fly till somewhere here; then you can go, then you can go by, then take another flight and get to Delhi. You another way is that, you can go by you can go from s to some other intermediate node, then you fly till here or you then fly directly to Delhi from here.

Another possibility is you can go from here to here and then go there to here etcetera, etcetera, etcetera. So, there you can see there are numerous possible ways by which you can go from one node from the source node to the destination node, ok. So, now, if you have this kind of, if you have a general network like this; problem the problem which we are asked, what we have to what we want to ask here is, what is the; what is the shortest path?

So, every link here suppose has a length ok the or a length or a cost associated with it, ok. So, the length say for every i and j ok, every i and j has a cost C i j, these are all directed edges, there is a cost to go from i to j, cost C i j to from i to j. And what we want to know is, what is the; what is the route from s to t of minimum cost?

What is the route from s to t of minimum cost? Now, this is a problem that we end up solving every day when we use Google maps; to find a route from point a to point b, this is exactly what Google maps is ending up solving for you. It is assuming the cost as say the travel time from i to j.

Now, if you are in if you are within the city, may be travel time is all you care about; in another if when you are traveling from one from intercity, you are caring also about say tolls, maybe if you are driving on toll roads or your cost of a; cost of a ticket if you are changing modes of transport in between etcetera, etcetera ok.

So, you can define cost in whichever way you like the. So, long as you are you have one consistent notion of cost, this problem can be posed in this particular way, ok. Now, we want to know what is the shortest in the sense of in the sense of the total cost that you would incur of route from s to t, ok.

So, the cost of a route from s to t is simply the, s to t is in sum of c i j where i comma j is in the; is in the i comma. So, whenever i comma j is an edge that you use in the route, you it costs you c i j, ok. And the cost of the entire route is the sum of all these c i j's; that are where i comma j lies in the route, this is the cost of travelling from i to j.

Now, here you can see how, intuitively how would you solve a problem like this? See if you want to go from Mumbai to Delhi, you can see already this simple little graph is a extremely complicated graph, right. How would you solve a problem like this?

You would say well let me go from, let me try out this route; one route here, I go from s from. I go from my the source node to one node, then I try out and from here a let me try out this particular possibility, let me try out that particular possibility etcetera, etcetera.

I have you basically have to enumerate a huge number of possibilities and to try to get to the shortest path. Now, that seems like a, it seems like an extremely complicated optimization over so many; because even just enumerating all the possible paths itself is a would take you a lot of time, right.

And then listing out listing them all out, just simply enumerating all the paths and then try and then trying to find which is the one that is of least cost. Now, it turns out this can be solved using linear programming in a very very elegant manner, ok. So, let me explain how that is; how that is done, ok.

So, on the on the routes here, now I have say if I have a node i and a node j here; I have a cost c i j listed written for that particular link there, ok. Now, I want you to before I write out the linear program; I want you to appreciate the difference between this problem, which is the shortest path problem and the problem of maximum flow.

The problem of shortest path is simply says, I want the shortest path from go to go from s to t without; but every the when I travel on any edge in the path, any edge or any link i j on the

path, it cost me something. There is a certain toll or a certain fee that I have to pay. But there is the capacity is virtually infinite, like any amount of flow can go on, ok.

So, the problem, whereas the other problem, there is no cost associated with travelling from i to j; but there is a limitation of capacity, where you cannot send more than a certain amount of flow. So, you can think of capacity and cost as two different things; capacity is how many cars say a road can take capacity of that road. The cost of travelling on that road is the toll you have to pay for traveling on it, these are two very different things.

There are finitely many nodes yes; there are finitely many nodes that you can by which you can. And we want to know the shortest path on that particular from the source to a destination node.

Student: (Refer Time: 12:51).

No. So, in any even in a real life problem, you need to model it in such a way that it brings it you have, you know finitely many nodes and finitely many decisions to make, ok. So, even though there are, the point is even though there are just finitely many, the number is too large; if you just think of all the possible routes you can take, it is the number of combinations you can make is immense, right.

So, but what is very nice is, you can actually solve this using linear programming in a very clean in a clean way, ok. So, as I was saying there are two, these are two different problems; first problem is that of the earlier problem, which is the maximum flow problem is about sending the maximum flow over capacity constraint networks ok, that is about finding the bottleneck.

This problem is there is no bottleneck, you can set as much flow as you like, right. But the which links will you choose is the question; because the links that you will choose, based on the links you will choose, you will incur a cost, ok. So, now, so since the cost and the cost here we will; what we will do is say, let us say, ok.

Let me write this in this sort of way. Now, suppose, just imagine suppose I had a one unit of flow entering s, just one unit ok; some one unit of goods, say one truck or whatever, one unit of goods, let not one truck, one unit of goods entering node s and which has to reach a node t, ok.

So, it enters here and leaves and has to reach t. When whenever this flow, one unit of flow reaches any particular node; say it reaches this node i, it has a decision to, there is a decision. We need to make, which is that out of all the links that are going out from node i, which one should I pick.

Now, suppose at node i, I have these links which have cost say 10, 20 and 30; these are the costs of the links, not capacities, costs of the links which, ok. And suppose from I know that from here let us say for simplicity, from here, this is the; this is the cost that I would incur say. If I came if I came here and then from here if I had to go all the way till Delhi, here is suppose the cost I would incur, someone told me this number.

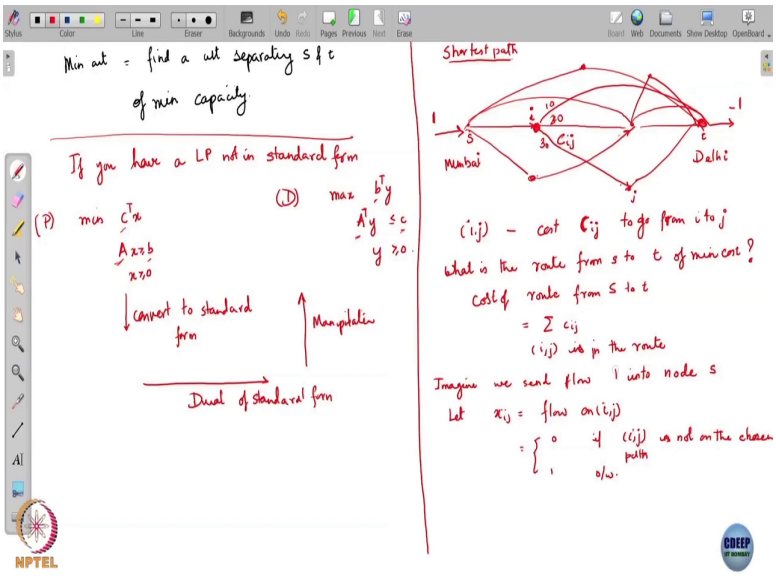
That here is the cost I would incur, if I came here I. So, if I came to node i and I chose this particular link, it would cost me 20 to get to Delhi; if I chose this particular link, it would cost me 10 to get to Delhi, the other one would cost me 30 to get to Delhi, ok.

Not just to the next node, I am taking suppose all the way till Delhi ok for simplicity. In that case, how, where would you send your flow? You would send the flow on to the one that is that has the least possible cost least cost of all of these right. Naturally the reason is because the links do not have any capacity constraint.

There is no reason for you, when the flow arrives at a particular node, there is no reason for you to split the flow across multiple edges. You would auto, you will always pick the one that the, pick the one that has the least possible cost to reach the destination from there onward, is not it. So, this simple observation is all that we actually need here, ok.

So, let us write, let us write out the some variables and I will explain to you how this works out, ok. So, suppose x i j, ok.

(Refer Slide Time: 17:02)



So, suppose I am sending. So, imagine flow 1 into node s, ok. So, you send a unit flow into node s; let x i j be the flow on i to on link i j ok, x i j be the flow on link i j. So, what is the objective in expressed in terms of x i j? Suppose I had a if I have. So, if I am sending flow x i j on link i j ok; what we will use this interpretation and it will later turn out that this is actually correct.

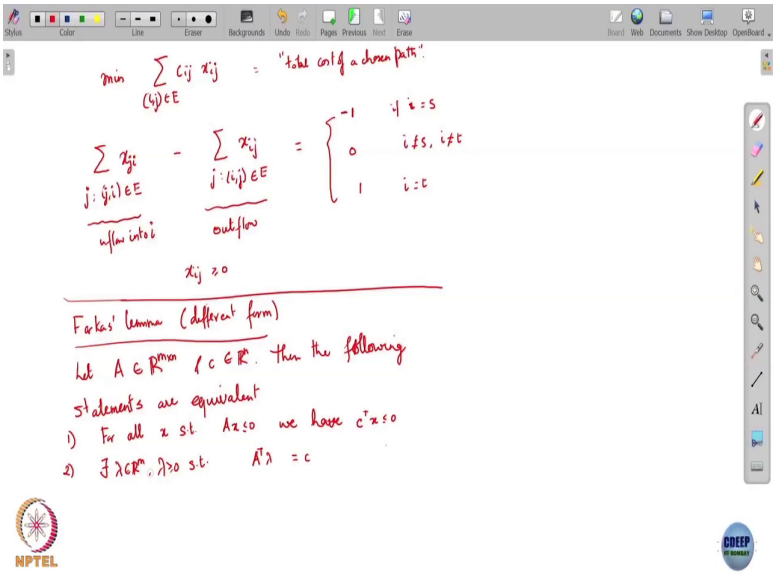
So, we will have this interpretation that, this is equal to 0, if i comma j is not on the chosen path and 1 otherwise. So, it will turn out that, if it is not on the shortest path; then i x i j will

be 0, which means that it will not send any flow on that. And if it is on the shortest path, it will have to send the full flow on it.

So, the reason for that is exactly what I just said; see whenever a full the flow arises, arrives at a particular node, if it you look at the cost that it has to incur to get to the final destination and the path that will give you the least possible cost, the entire flow will go on to that path, it has to be.

So, if at all a flow is there on a path, that path must have must be with the one that encounters the least cost; it cannot be that part of it is there here, part of it is there, is a part of it is there you know on another part, right. So, now, let x i j be the flow on link i j and it we will it will have the, we will take this interpretation that it will be 0; if i j is on the is not on the chosen path and 1 otherwise.

(Refer Slide Time: 19:50)



So, the objective is to simply, then in these terms is to minimize this. So, what is this? This is the total cost of a chosen path and now all I need to make sure is that, these flows satisfy my flow conservation constraint. So, I look at x j. So, my flow conservation constraints again my inflow is sum over j, such that i j belongs to E; this is the inflow into i, then I have a sum over j, sorry i in this not i j, this should be j i belongs to E, this is i j belongs to E, this is.

And this should be equal to this is 1, if i is the source node; 0 if i is neither the source node, nor the destination node, and minus 1 if i is the; if i is the is the destination, is this correct, sorry my mistake here, sorry this should be the other way round, this should be minus 1 here, that is correct.

So, minus this should be minus 1, if i is the source node; because, so the that positive 1 inflow into i. So, that is correct. So, it is minus 1, this would be it would be 0 at all nodes except for the source node and destination node and it would be equal to 1 at the outflow node and we have that x i j is greater than equal to 0.

So, what have we done, we have taken a problem of shortest path and of 5 which was about; which was about trying to enumerate all the possible paths in the network. And trying to find the shortest path and we have to converted it into a problem of finding of minimizing the cost of flow.

He said let us imagine a unit flow that was sent in; if a unit flow sent in and it had to follow my flow conservation equations right, what would be the path it would choose? It is almost like you know you are sending a flow of water and it is trying to find the path of least resistance or least to go from a particular source to a destination, right.

You solve this and it will, it turns out that this actually gives you the shortest path, ok. So, this is a; this is a linear programming formulation of a problem that on the face of it looks extremely complicated. Just you know if you had to enumerate all the possible paths, would be a horrendously complicated problem.

But knowing the knowing; but with this sort of way of looking at it, we are able to reduce it to a linear programming problem, ok. There are many other such problems that end up getting reduced to get end up reduced to linear programming. That is why linear programming is such a is an such an extremely powerful tool across all of engineering and engineering and science, ok alright.

So, with this we conclude our study of linear programming; what I have not touched upon at all is algorithms for linear programming, because I plan to come to that next. But what I want you to remember. The biggest takeaway I want you to remember from linear programming is linear programming duality and associated Farkas' lemma, ok.

So, Farkas' lemma is something that we will use again and in one shot you will see how that gives us also optimality conditions for any type of convex optimization problem, ok. So, that is that would be in next on the agenda, ok. The form of, there is another form of let me mention this also. So, there is another form of Farkas' lemma that which will, which we will different form, which will be very useful in the next lecture.

So, it says let A in R m cross n and c belong to R n; then the following statements are equivalent. For all x, such that A x is less than equal to 0, we have c transpose x less than equal to 0, c. For all x such that A x is less than equal to 0, we have c transpose x less than equal to 0, ok. And then there exists a lambda greater than equal to the other state.

So, the Farkas lemma is saying that, there are two statements that are equivalent. The first statement is that, for all x such that A x is less than equal to 0, we must have c transpose x less than equal to 0. The second statement is that there exists a lambda greater than equal to 0; so lambda is in R m, in R m lambda greater than equal to 0, such that A transpose lambda equal c.

So, this statement is essentially, this version of Farkas' lemma is a different form of stating Farkas' lemma; this is not stating it in the form of two statements out of which only one can

be true, rather it is saying that these two are equivalent, these are two equivalent ways of saying the same thing.

One is that for all x such that A x is less than equal to 0 we have c transpose x less than equal to 0; the other is saying that, there always is a lambda greater than equal to 0, such that A transpose lambda is equal to c, right. So, what this basically is, if you look at the second statement, what it is saying is; it says is that c is in the cone generated by the columns of A transpose, c is present in the cone generated by the columns of A transpose.

So, you can always find a conic combination of the columns of A transpose, such that the that conic combination equals c, alright. The first statement says something completely different; it says that A x is, whenever A x is less than equal to 0, it has to be that c transpose x is also less than equal to 0, ok.

So, a columns of A transpose is, consider a cone formed by the columns of A transpose that contains c that is one statement; the other statement says that if you look at the, if you look at vectors x such that A x is less than equal to 0, then they must make an obtuse or greater than equal to 90 degree angle with c, c transpose x less than equal to 0, it turns out these two are actually equivalent, ok.

So, this is another form of Farkas' lemma and we will, this again can be proved using linear programming duality; again you have to think of the right sort of linear program, look at its dual and again you will be able to argue it from there. In fact, you may even be able to prove linear programming duality itself from linear from this particular Farkas' lemma, so without you know going through separating hyper planes and so on.

So, the first principles proof of this Farkas' lemma would be as hard as LP duality itself, ok alright. So, we important thing, the reason I mention is now is that, remember this we will use this in a definitive way in the next class, ok. So, we will end it now.