Linear Programming and its Extensions

Prof. Prabha Sharma
Department of Mathematics and Statistics
Indian Institute of Technology, Kanpur

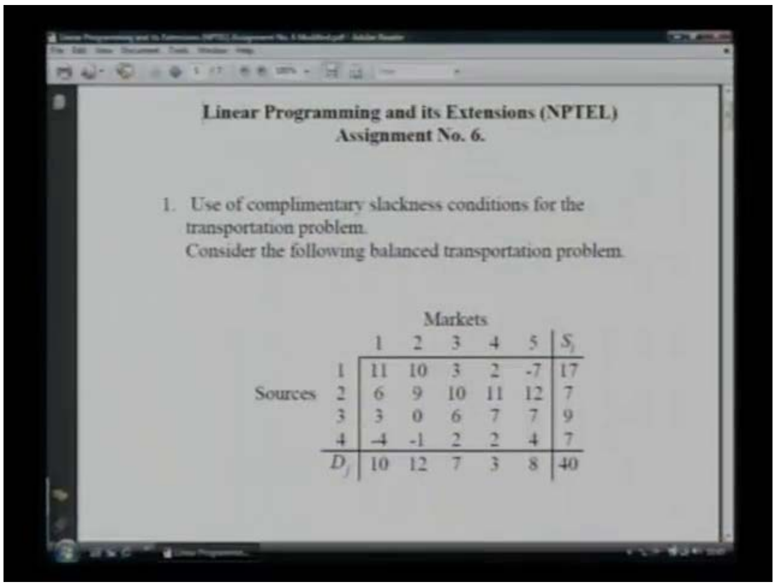
Module No. # 01

Lecture No. # 31

Assignment Six, Shortest Path Problem Shortest Path between any two Nodes Detection of Negative Cycles

Let me discuss assignment six, which is based on transportation problem with you. This assignment should have been discussed earlier, but anyway. So, let us go through these problems. In the first problem, I want to show you, how complementary slackness conditions can be used for the transportation problem.

(Refer Slide Time: 00:42)



Linear Programming and its Extensions (NPTEL)
Assignment No. 6.

1. Use of complimentary slackness conditions for the transportation problem.
Consider the following balanced transportation problem.

		Markets					
		1	2	3	4	5	S_i
Sources	1	11	10	3	2	-7	17
	2	6	9	10	11	12	7
	3	3	0	6	7	7	9
	4	-4	-1	2	2	4	7
	D_j	10	12	7	3	8	40

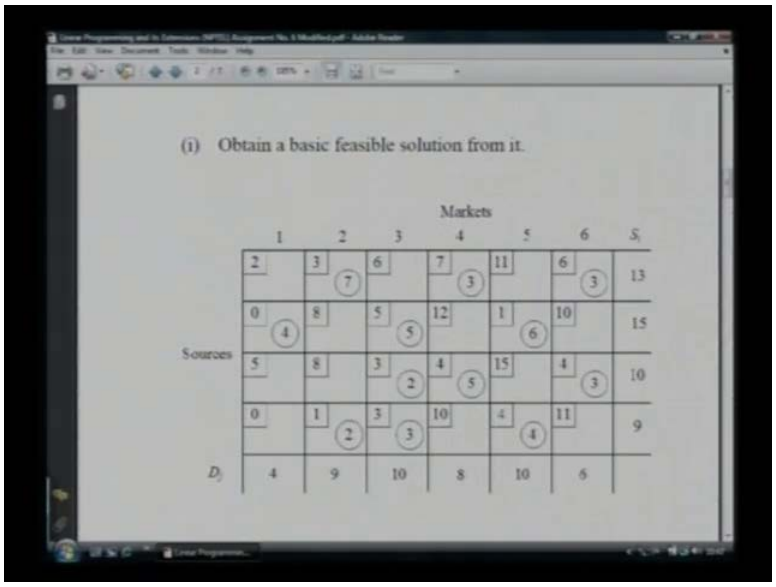
So, in the table below, balanced transportation problem is given to you. Energy as usual, the entries are the cost and the right hand side and the row below is the demands and the column on the right hand side is the supply.

Now, I have given you dual feasible solution u bar, which is equal to this, regard this infinity here, because, you have u bar corresponds to the row. So, this is minus 7 8 comma 3 comma 0 and v bar is minus 4 minus, minus 3, 2, 1, 4.

So, this is a dual feasible solution, you can verify for yourself and now using complementary slackness conditions, you see, you will only allow those x i j's to be positive for which the c i j minus u i minus v j is 0, because of the complementary slackness condition.

So, if you can find an x bar which is feasible from among the cells for which the dual constraints are satisfied is equality, and then you will have a primal dual feasible pair. And using complementary slackness conditions, again you can show that the pair x bar, u bar, v bar, which is a primal dual pair of feasible solutions will also be optimum. So, this is what you have to do in problem one, and if there are alternate optimal solutions, then also find them. That means, for if for some non basic cells, your dual constraints are satisfied this equality or the relative price is 0, then you can enter those cells into the basic feasible solution and get an alternate optimal solution right.

(Refer Slide Time: 2:16)



(i) Obtain a basic feasible solution from it.

Markets

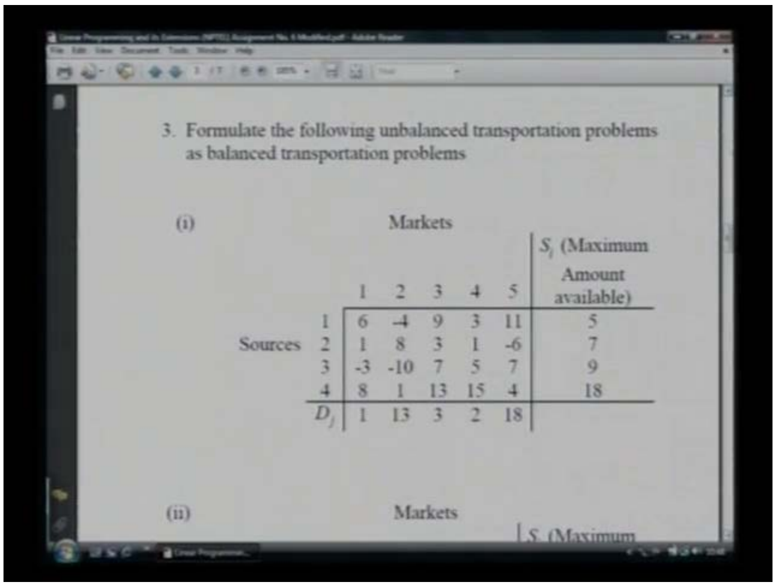
Sources	1	2	3	4	5	6	S_i
	2	3 ⑦	6	7 ③	11	6 ③	13
	0 ④	8	5 ⑤	12	1 ⑥	10	15
	5	8	3 ②	4 ⑤	15	4 ③	10
	0	1 ②	3 ③	10	4 ④	11	9
D_j	4	9	10	8	10	6	

So, this is problem 2, problem 2 transportation problem below given below, feasible solution is indicated yes and what is happening is that, you see the number of the supply points are 4, number of demand points are 6 obtained. So, you should have actually 9 basic cells; in any basic feasible solution, there should be 9 cells.

But, here we have 3 3 6 9 and 12; this is the feasible solution, but not a basic feasible solution. So, I was first want you to find out basic feasible solution from this given feasible solution and you know that because, the number of cells is more than 9 or you will have quite a few theta loops and so you have to get rid of the theta loops and then obtain a basic feasible solution.

So, do that this thing and then check if the basic feasible solution obtained is optimal. Now, let us go to problem 3, this is here, you see, I am just trying to for (( )) unbalanced transportation problems.

(Refer Slide Time: 03:08)



3. Formulate the following unbalanced transportation problems as balanced transportation problems

(i)

Markets

		1	2	3	4	5	S_i (Maximum Amount available)
Sources	1	6	-4	9	3	11	5
	2	1	8	3	1	-6	7
	3	-3	-10	7	5	7	9
	4	8	1	13	15	4	18
	D_j	1	13	3	2	18	

(ii)

Markets

S_i (Maximum

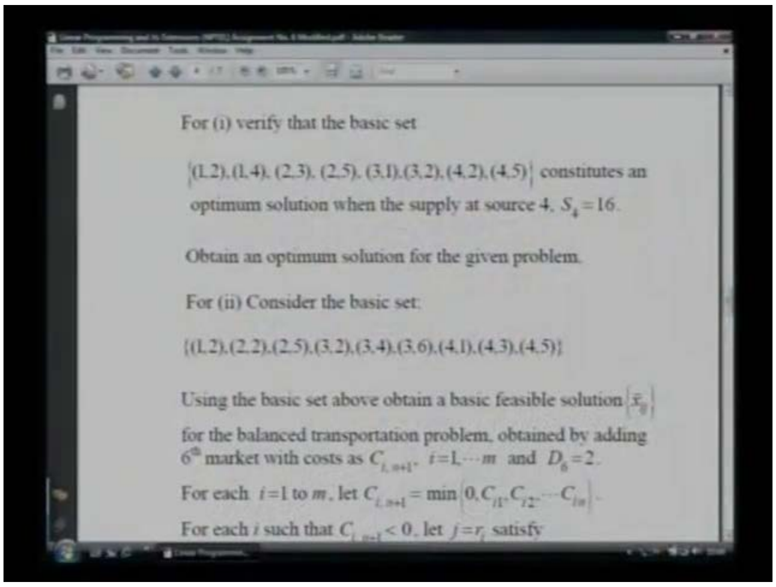
So, I have given 2 of them here, and I want you to reduce them to a balance transportation problem and we will discuss how. So, here for example, in the first problem here, the supplies the supplies add up to more than the demand, it is the demands.

So, therefore, what is happening is that, your supply constraints will be satisfied as less than or equal to because, you will not be able to use all the supplies, since the demand is less than the total supply.

So, therefore, your supply constraints will be written as inequalities and in the second case, the second problem the you have given that the maximum amount available, so that means, again the all the supplies need not be used up and the demands are not exact; that means, you minimum requirements are given.

So, you may ship more than what is required. So, here actually you have inequalities for supply points, supply constraints as well as for demand constraints. So, both kinds of inequalities you have to handle here, and we I will try to show you, how to handle these two types of unbalanced transportation problems.

(Refer Slide Time: 04:34)



For (i) verify that the basic set

$\{(1,2),(1,4),(2,3),(2,5),(3,1),(3,2),(4,2),(4,5)\}$ constitutes an optimum solution when the supply at source 4, $S_4 = 16$.

Obtain an optimum solution for the given problem.

For (ii) Consider the basic set:

$\{(1,2),(2,2),(2,5),(3,2),(3,4),(3,6),(4,1),(4,3),(4,5)\}$

Using the basic set above obtain a basic feasible solution $\{\bar{x}_{ij}\}$ for the balanced transportation problem, obtained by adding 6th market with costs as $C_{i,n+1}$, $i = 1 \cdots m$ and $D_6 = 2$.

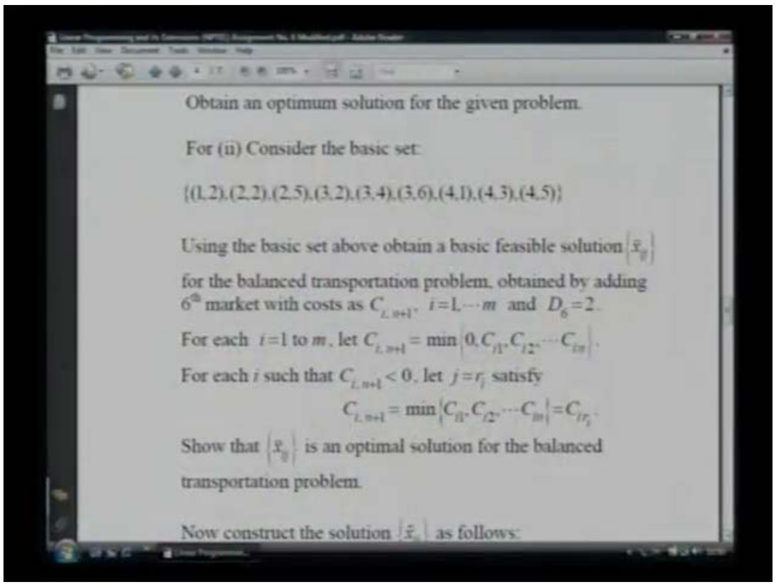
For each $i = 1$ to m, let $C_{i,n+1} = \min\{0, C_{i1}, C_{i2}, \cdots C_{in}\}$.

For each i such that $C_{i,n+1} < 0$, let $j = r_i$ satisfy

So, let us see in the. So, for the first problem, I am giving you a basic set and I am asking you to constitute an optimal solution when the supply at source 4 S 4 is 16. You see, if you if the supply if S 4 is 16, then the sum of s i's is equal to sum of d j's. So, then it becomes a balanced transportation problem and so, you can immediately check whether the given set of cells they constitute a basic feasible solution, and then verify that they are are they also satisfy the optimality criteria.

So, this is the optimal solution, now I am saying that we have to obtain optimum solution for the given problem which is the unbalanced. So, so just go ahead starting with starting with the given basic feasible solution, which is optimum for S 4 equal to 16, you have to obtain an optimal solution now. So, go ahead and do it, fine.

(Refer Slide Time: 05:30)



Obtain an optimum solution for the given problem.

For (ii) Consider the basic set:

$\{(1,2),(2,2),(2,5),(3,2),(3,4),(3,6),(4,1),(4,3),(4,5)\}$

Using the basic set above obtain a basic feasible solution $\{\bar{x}_{ij}\}$ for the balanced transportation problem, obtained by adding 6th market with costs as $C_{i,n+1}$, $i=1,\cdots m$ and $D_6=2$.

For each $i=1$ to m, let $C_{i,n+1}=\min\{0,C_{i1},C_{i2},\cdots C_{in}\}$.

For each i such that $C_{i,n+1}<0$, let $j=r_i$ satisfy

$$C_{i,n+1}=\min\{C_{i1},C_{i2},\cdots C_{in}\}=C_{ir_i}.$$

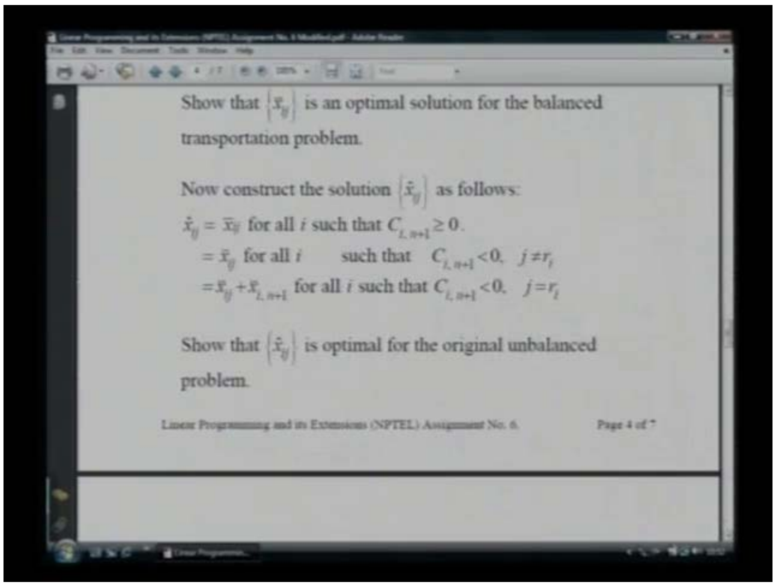
Show that $\{\bar{x}_{ij}\}$ is an optimal solution for the balanced transportation problem.

Now construct the solution $\{\bar{x}_{ij}\}$ as follows:

For problem 2, again I am giving you a basic set of cells and I am saying that you can obtain basic feasible solution x i j bar for the given unbalanced transportation problem. Reduce it to a balanced transportation problem by adding a 6 market, and the cost will be C i and plus 1, in in our case, n plus 1 is 6 and then you have D 6 is 2, so the difference between sigma x i and sigma d j that is 2.

So, make that difference equal to 2 and so, the demand D 6 will be right and then I am telling you how to obtain the C i n plus 1 here, right and the formula I have given to you. So, you have to take the minimum of, so, if this formula you can go through. So, once you have these costs given to you, now with this cost, you have a balanced transportation problem. The number of markets is gone up to 6 and then I am asking you to first obtain x i j bar is is an. So, actually these basic sizes that I have given to you in the beginning, they constitute an optimal solution.

(Refer Slide Time: 06:41)



Show that $\{\bar{x}_{ij}\}$ is an optimal solution for the balanced transportation problem.

Now construct the solution $\{\hat{x}_{ij}\}$ as follows:

$\hat{x}_{ij} = \bar{x}_{ij}$ for all i such that $C_{i,n+1} \geq 0$.

$= \bar{x}_{ij}$ for all i such that $C_{i,n+1} < 0$, $j \neq r_i$

$= \bar{x}_{ij} + \bar{x}_{i,n+1}$ for all i such that $C_{i,n+1} < 0$, $j = r_i$

Show that $\{\hat{x}_{ij}\}$ is optimal for the original unbalanced problem.

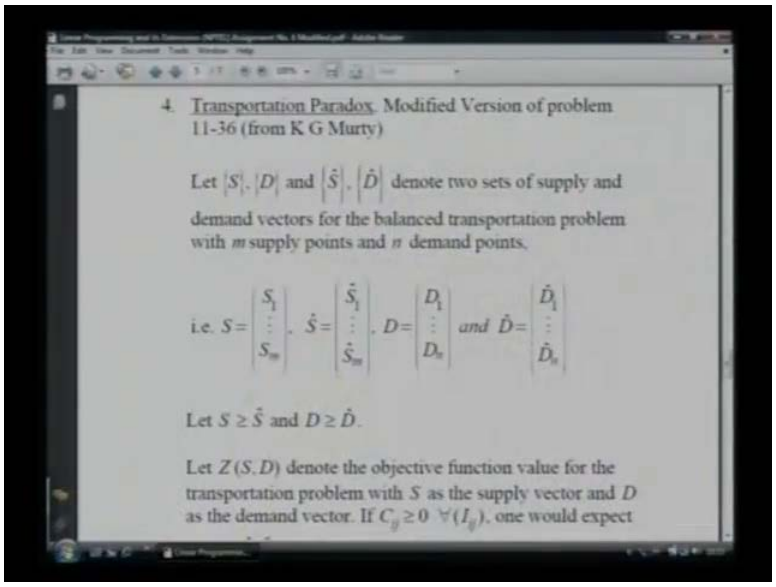
Linear Programming and its Extensions (NPTEL) Assignment No. 6. Page 4 of 7

So, verify that it is an optimal solution. So, once you have an optimum solution x i j bar for the balance transportation problem, then I am now asking a giving you a formula for constructing a solution x i j hat and the formula is given here, where, the three formula is given here. And now, you can show that x i j hat is optimum for the original unbalanced format. That means, you have now verified the optimality criteria for the balance unbalanced transportation problem. So, the optimality criteria does not change even though the constraints are now not equality constraints, but less than or equal to kind the optimality criteria will remain unchanged.

So, once I have given you this x i j hat, just check that the given solution is optimum for the unbalanced transportation problems. So, this is how I am trying to give you the treatment for unbalanced transportations problems and in fact, it will be interesting to look up the literature, for example, K G Murthy's book only and you will find many other variations of the unbalanced transportation problems.

But, since once you know your balanced transportation well, you can always reduce all these variations to a balanced transportation problem and get the optimum solution 4. I am just taken this again transportation paradox from K G Murthy's book and it is a modified version. I have modified the problem.

(Refer Slide Time: 07:55)



4. Transportation Paradox. Modified Version of problem 11-36 (from K G Murty)

Let $\{S\}, \{D\}$ and $\{\hat{S}\}, \{\hat{D}\}$ denote two sets of supply and demand vectors for the balanced transportation problem with m supply points and n demand points.

i.e. $S = \begin{pmatrix} S_1 \\ \vdots \\ S_m \end{pmatrix}, \hat{S} = \begin{pmatrix} \hat{S}_1 \\ \vdots \\ \hat{S}_m \end{pmatrix}, D = \begin{pmatrix} D_1 \\ \vdots \\ D_n \end{pmatrix}$ and $\hat{D} = \begin{pmatrix} \hat{D}_1 \\ \vdots \\ \hat{D}_n \end{pmatrix}$

Let $S \geq \hat{S}$ and $D \geq \hat{D}$.

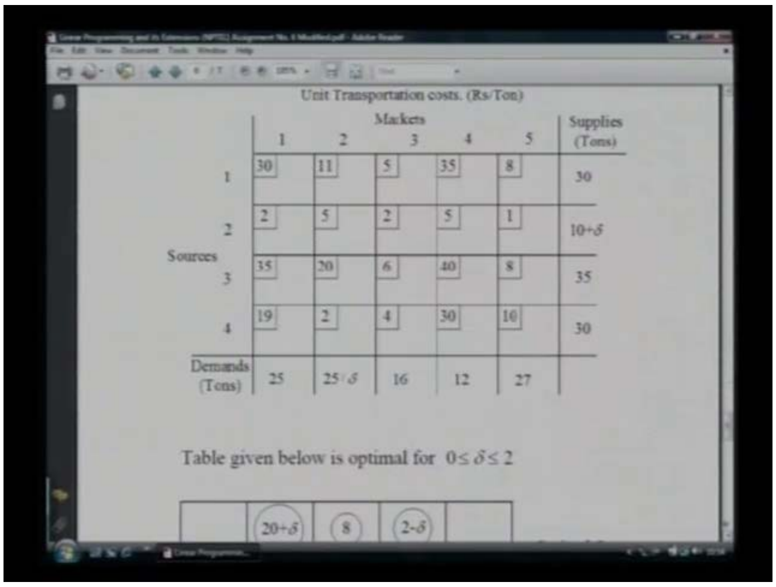
Let $Z(S, D)$ denote the objective function value for the transportation problem with S as the supply vector and D as the demand vector. If $C_{ij} \geq 0 \ \forall (I_{ij})$, one would expect

So, essentially what I am telling you is that, S and D is 1 set of supply and demand S hat and D hat is another set of supply and demand for a transportation problem and S is greater than or equal to S hat and D is greater than D hat.

This is the, these are all vectors; that means, component wise, each S i is greater than S i hat and each D j is greater than D j hat right. So, now, I am defining Z S D as the objective function value for the transportation problem. So, the cost remains the same right with S as supply vector and D as the demand vector.

If C i j's are all non negative, then, one would expect, see this, what I am saying here. So, one would, if if all C i j' s are non negative, then one would expect that, since the volume of traffic for S comma D is more than S hat D hat, you would expect that the objective function value for S, D; that means, Z S D will be greater than D S hat D hat. But, actually this is not what is happening. It is possible that, even though you are transporting more goods or whatever it is, the traffic is more, your cost may still be smaller right and I am giving you an example here.

(Refer Slide Time: 09:11)



Unit Transportation costs. (Rs/Ton)

	Markets 1	2	3	4	5	Supplies (Tons)
Sources 1	30	11	5	35	8	30
2	2	5	2	5	1	10+δ
3	35	20	6	40	8	35
4	19	2	4	30	10	30
Demands (Tons)	25	25+δ	16	12	27	

Table given below is optimal for $0 \le \delta \le 2$

	20+δ	8	2-δ	

So, in this case you see, this is a table and I have the 10 plus delta is the demand a supply at source 2 and 25 plus delta is the demand at market 2. then I am, so now, the solution shown in this tableau is feasible for delta between 0 and 2 and you can also verify that it is optimal for all values of delta between 0 and 2 right.

(Refer Slide Time: 09:22)

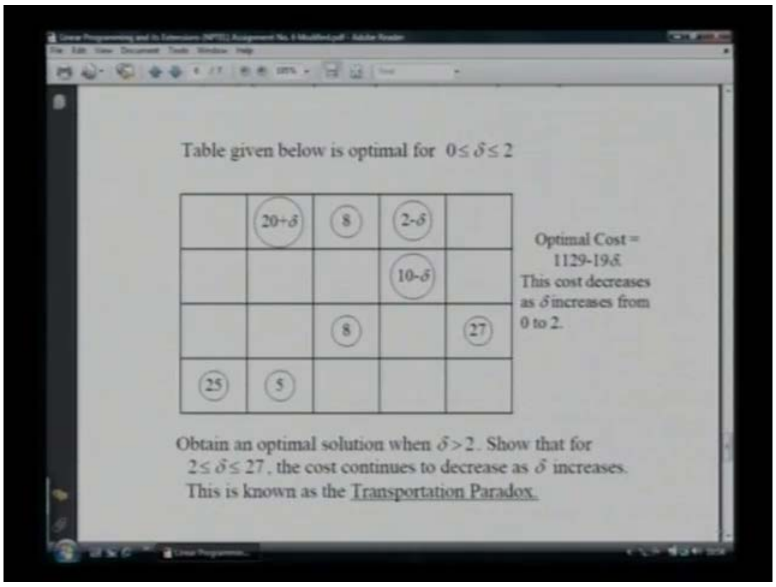


Table given below is optimal for $0 \le \delta \le 2$

	20+δ	8	2-δ	
			10-δ	
		8		27
25	5			

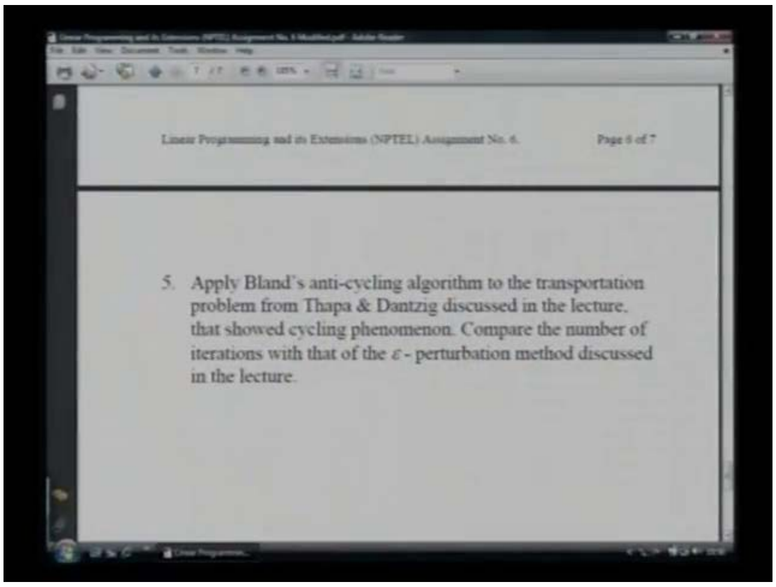
Optimal Cost = 1129-19δ. This cost decreases as δ increases from 0 to 2.

Obtain an optimal solution when $\delta > 2$. Show that for $2 \le \delta \le 27$, the cost continues to decrease as δ increases. This is known as the Transportation Paradox.

But, you see that the cost, optimal cost is 1 1 2 9 minus 19 delta; that means, as delta increases, the cost will decrease. So, this is the kind of paradox I am trying to show to you. and then, Now, I am asking you further to work out the situation when delta is greater than 2 and you will see that for delta between 2 and 27, the cost continues to decrease as delta increases, because, in the optimal cost, the coefficient of delta would be some negative number. So, the cost will continue to decrease as delta increases as long as delta is between 2 and 27.

So, this is what we mean by a transportation paradox, but surely you can try to rationalize why this is happening. Now, problem 5, I want to make a point here and this is see, I have been saying that for a a simplex algorithm to combat degeneracy, which means cycling, you apply Bland's anti-cycling rules.

(Refer Slide Time: 10:39)



Linear Programming and its Extensions (NPTEL) Assignment No. 6.   Page 6 of 7

5. Apply Bland's anti-cycling algorithm to the transportation problem from Thapa & Dantzig discussed in the lecture. that showed cycling phenomenon. Compare the number of iterations with that of the ε - perturbation method discussed in the lecture.

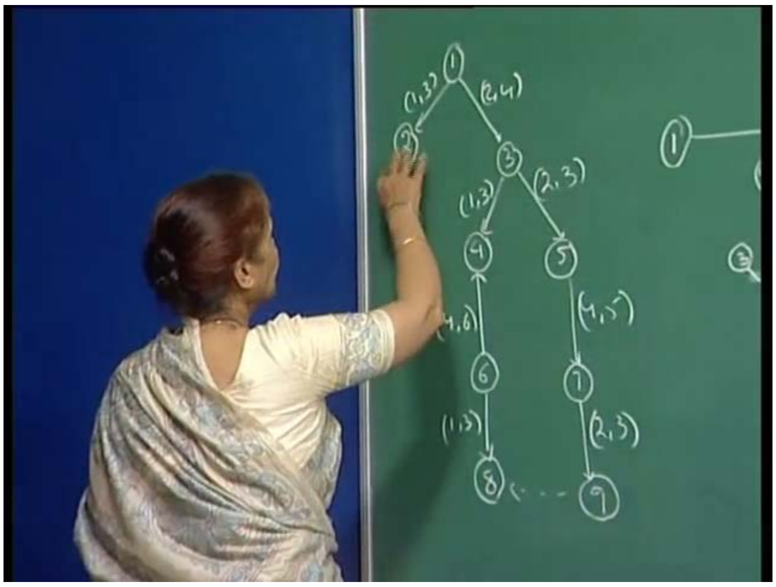
And then you get the, you can avoid stalling; that means, you can avoid cycling right.

So, the same thing we will do for the transportation problem also; now Thapa and Dantzig. I have given you the reference in the book this has been taken, it has given a transportation problem which cycles, if we use the ordinary simplex algorithm right. Now and then, I gave you an epsilon perturbation method also, which I feel and afford of course, many other practitioners feel is faster, one faster than Bland's.

So, now, here I want you to apply and Bland's anti-cycling rules to the transportation problem, for that problem which is given in Thapa and Dantzig and see, then, count the number of iterations and compare with the because, I have solve the problem with an epsilon-perturbation.

So, you should be able to compare. Of course, one problem is not enough to judge the performance of, but I just want to demonstrate to you that the Bland's anti-cycling rules become the make the algorithm slower for a specially structured problem like transportation problems, because, these problems have a special structure. So, the epsilon-perturbation method is supposed to be faster.

(Refer Slide Time: 12:22)



So, I hope you enjoy doing the cycling. So, let me get back to strongly feasible basis and I thought I should have an one or two more examples to make you more familiar, because, we do not use them so often. So, here again, I am trying to give you an example. see show you suppose this is a strongly feasible basis a tree, because, as you see, you can send positive flow from every node to the root node; this is your root node, right you can just check. See here, everything is this fine.

So, now suppose I will show you an iteration, when suppose, you with the arc 8 9 is the entering arc, I mean, sorry, what I want to say here is that, here suppose the flow is 3 3 and I have to reduce the flow on. So, to reduce flow on arc 8 9, so, that means, the orientation of the cycle is going to be this. right This is the orientation of the cycle because, I have to reduce the flow and you can see that, for example, here, the flow can get reduced by, so this will be by 1 unit right here, the flow can go up by 2 units. Then, here again, the flow can decrease by 1 unit, the orientation is like this. Here, the flow can increase by 1 unit by 1 unit and by 1 unit right.

So, essentially, your delta comes out to be 1. So, let me or if you want to, right what I mean is that you are going this way. So, here is minus delta and this will be also minus delta this will be plus delta and so on, you can write down.

So, now let us see what are the blocking arcs, here, blocking arc, this is the blocking arc, right then, this is the blocking arc and these are all blocking arcs 3 5 5 7 7 9, because all of these limit the increase of the flow by more than decrease of flow more than delta by more than 1. right is it?

So, and this is the orientation. So, how do you decide the exiting arc? exiting arc What is the exiting arc here?. So, remember we said that we will star, and this is your apex node. right Because you have these 2 nodes, you take the path and then they meet bo[th]- path towards the root node. So, they meet at 3, this is your apex node. So, from no[de]- apex this, you start along the orientation of the cycle, because, the orientation is like this.

So, you start from here and then you go on. All these are blocking arcs you come up here, then, here, this is the last blocking arc or else this will be the last blocking arc right. So, that means, I will exit this one and you see then, the tree will hang by like this, and in that case, you see if this goes up by, this becomes 5 6, you will still be able to flow more than. Because, see now, what will happen here, the flow will be 0, but, when you exit this, the this these arcs will be hanging like this and the direction of this arc will become towards the root. Remember, for all 0 flow, arcs must be pointing towards the root node, this is the rule right and.

So, this will again. So, exiting arc will be the last blocking arc, right which is in our case, 3 4 right and the remaining tree or the new tree 1 once I update, the flow would be again strongly feasible tree. right Then, we had also mentioned the case that how do you get a starting feasible solution which is a strongly feasible tree, right and I had mentioned that when you use phase 1 and you get the artificial arcs as your a part of your feasible tree starting tree, then that will be a strongly feasible tree, because, artificial arcs do not have any.

So, no upper bounds no upper bounds on artificial arcs. right no upper bounds on artificial arcs right. So, therefore you see that if this, so this will be my root node; in this case, this root node. So, then you see this is your 2 units of flow. So, I can decrease the flow on these arcs. Along these arcs, I can decrease the flow and on all these arcs, I can increase the flow sorry straight. Again from here, I should be able to oh be this. So, for these arcs, you can increase the flow and we had said that the ah right because, there is no upper bound. So, I can increase the flow here right on arc 1 6 and 3 6, I can increase the flow.

So, there is nothing like a saturated arc and there is nothing like a 0 arc, in this case when you have. So, therefore, both the conditions do not apply, I mean in the sense that by default, the conditions apply. right.

Because there is no 0 arc here, and there will be no saturated arc, since, they are no upper bound constraints. So, a starting phase 1 starting tree will be a strongly a feasible tree. So, you can begin your iterations by that and so, it has been effectively shown that using strongly feasible basis will will will avoid cycling. But again, stalling is a new problem, is a different problem and there are methods to tackle stalling also. So, that should take care of your a min-cost flow problem.

Then, we come back to the shortest path problem and I had shown you So, the shortest path problem yes and I have shown you that this is the special case of the min-cost flow problem, where you had only having one unit available at the point from which you want the shortest paths to be found, and up to the point a node t. So, S and T we said we want to find a shortest S T path. Then, there is 1 unit available at node S and 1 unit is demanded at node T and you are computing and the C i j's are the distances or the cost of traversing the arc i j.

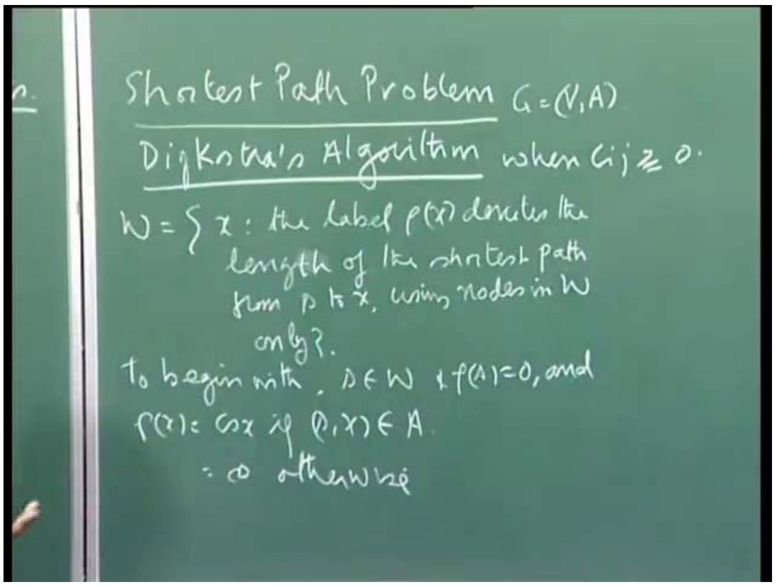
So, that is the arc right. So, and I showed you that you can you can solve it by the primal dual algorithm by repeated applications right and that turned out to be building up the path from t backwards right. So, now, if you just look at what was happening in the primal dual algorithm, from the starting node that becomes, and of course, what one can say is that, you are not actually applying the primal dual algorithm, but it is the primal dual algorithm in the sense, but your computations become very simple.

So, let me first give you the algorithm and then we will discuss the thing. So, this is known as Dijkstra's algorithm, this when C i j's are greater than or equal to 0. So, this is special case, that means, Dijkstra's algorithm will not be valid if some C i j's are negative. Hence, then later on, we will discuss the case when the C i j's can be negative. So, we will have another algorithm for computing the shortest path between any pair node. So, Dijkstra's algorithm says that you define a set W as node x such that the label rho x denotes the l e n g t h length of the shortest path from S to x using nodes in W only.

So, let me define a set w as the set in which I keep adding those nodes, for which I have short length of the shortest path from S to that node using only the nodes in w and that label, I will give it rho x right.

So, that means, I will constantly go on building my set w and the moment t belongs to w, I have found the shortest path from s to t right. So, now, how do I do it? So, to begin with, to begin with to begin with S belongs to W and rho s is 0, because, there is a starting row right fine then and let to begin with, you have this and let me say then, here, and rho x is equal to c s x, if s x belongs to belongs to arc set right now here, I write. In the beginning, I had [had/has] told you that we will be doing it for a directed arc and so, this is always understood that v is your nodes set and a is the arc sets. So, I will not write it every time, so we said that is c x, if x s belongs to a and is equal to infinity otherwise.

(Refer Slide Time: 22:51)


Shortest Path Problem $G=(V,A)$

Dijkstra's Algorithm when $c_{ij} \geq 0$.

$W = \{ x$: the label $\rho(x)$ denotes the length of the shortest path from s to x, using nodes in W only$\}$.

To begin with, $s \in W$ & $\rho(s)=0$, and

$\rho(x) = c_{sx}$ if $(s,x) \in A$

$= \infty$ otherwise

So, in the beginning, for all nodes which are connected to s by an arc, I will give the label as the actual cost for the distance of the arc x s and for all other nodes, it is infinity in the for the for the arcs which are not there right. So, we have this, then what we do is, we say that, select select x such that rho x is minimum rho minimum of rho y y not belonging to w right.

So, you have given the labels, that means, all the labels have defined here in the beginning, then I will pick up that node for which the label is the smallest for all nodes which are outside w right. So, initially, only s belongs to w, then you have labeled all the nodes and then you will look for the one which is the smallest label and you select that node and then you say that your w goes to w union x right and so, you continue and then you update. So, update the labels labels rho y as minimum the old label and rho x plus c x y see; that means, you have pictorially.

So, you update the labels as for every, for for y, this is for al x belonging to w right update the labels for y not belonging to w, right then you do it this right and continue with algorithm, right that is, go back to, I should have written here, may be, if you want we to write this is step 1, then this is step 2 and this is step 3. So, continue (( )) that is go back to step 1. So, I have to now validate the algorithm and that means, what I have to say is that, what you end end of with would be your shortest path to the node e.

So, what are we saying here is that, the moment when you select the node this in with I with this criteria, then the label, because, now w becomes part x becomes part of w and by our definition rho x must represent the length of the shortest path from s to x right.

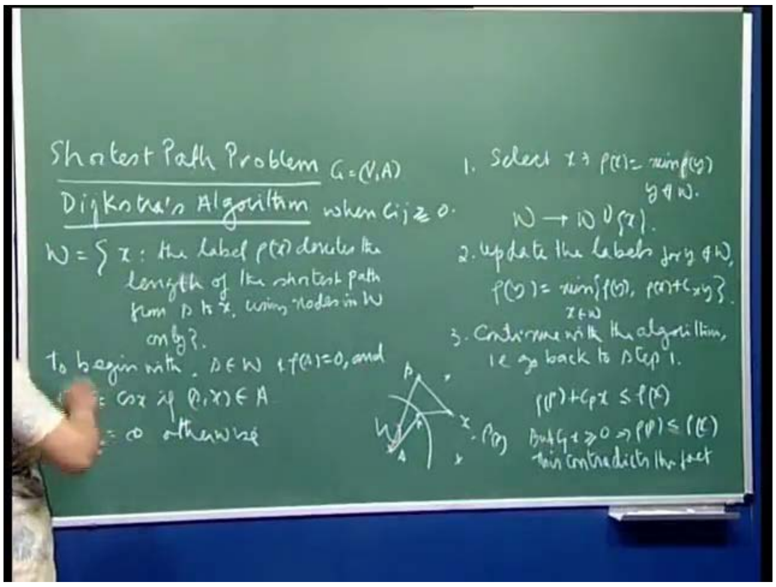
So, why do I say that? So, pictorially what was happening is that, see, I have this set w here, then I have a node x here, and I say that this is the minimum one, because you have so many other nodes outside w. So, I choose the one which has the smallest label and then I am saying that this has, that means, right now, rho x represents what rho x would represent, where I will add x to w; that means, rho x will represent the shortest.

Ah. So, s is here, right it will be the length of the shortest path from s to x by using nodes of w. only right now what can I mean suppose let say suppose it is not true that means, Suppose there is another path from s to, so, the suppose this is p. So, suppose there is another path 2 x from s going through a node p, which is outside w, right then what can what will happen?

See this path; that means, you are I am saying that rho x represents this length which is, the last node is somewhere here and then, so, therefore, what we are saying is that rho p plus c p x is less than or equal to rho x yes because, rho x I am saying, is the label for x right now and rho p is the label for p and shortest path to x is by a p. So, that means, this is the, so, this is rho p plus c, that is less than rho x, which is the contradiction of the fact that rho x was the minimum 1. So, this is not possible right and therefore, the, but but c p x better implies this right this contradicts. It is that the fact.

So, the fact that x I will have to. So, this contradicts the fact that x had the smallest label. I think I am writing the spelling of label we will verify fine. So, therefore, therefore, x must be. So, x must represent the shortest d distance from s to x [why/while] using nodes of t right therefore, therefore, once t belongs to w we have the labels representing shortest paths shortest path lengths representing.

(Refer Slide Time: 30:13)



Shortest Path Problem $G=(V,A)$

Dijkstra's Algorithm when $c_{ij} \geq 0$.

$W = \{x$: the label $\rho(x)$ denotes the length of the shortest path from s to x, using nodes in W only$\}$.

To begin with, $s \in W$ & $\rho(s)=0$, and

[illegible] $= c_{sx}$ if $(s,x) \in A$

[illegible] $= \infty$ otherwise

1. Select $x \ni \rho(x) = \min_{y \notin W} \rho(y)$.

$W \rightarrow W \cup \{x\}$.

2. Update the labels for $y \notin W$,

$\rho(y) = \min_{x \in W} \{\rho(y), \rho(x)+c_{xy}\}$.

3. Continue with the algorithm, ie go back to step 1.

$\rho(p)+c_{px} \leq \rho(x)$

But $c_{px} \geq 0 \Rightarrow \rho(p) \leq \rho(x)$

this contradicts the fact

So, we have a label representing shortest path lengths from s to each of 2, each of the other nodes, because, now, in the process of computing the shortest path from s to t, you have managed to compute shortest paths from s to every other node. right And, if you just start your dual simplex algorithm, you know, like if you can reverse the network and have t as your starting node, then, you see that the every iteration of Dijkstra's algorithm matches with your primal dual algorithm.

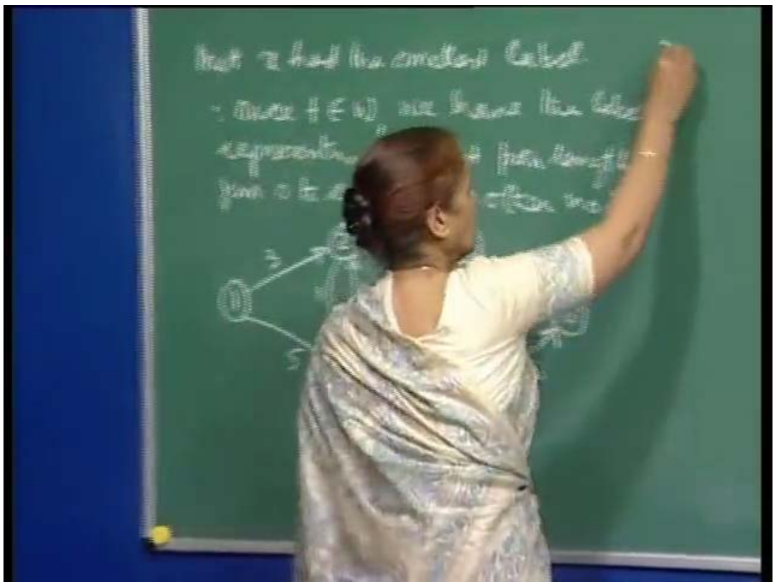
So, this. So, therefore, this is nothing different, but the thing is that without the justification of the primal dual and going through that whole process, you simply do this and the argument, simple argument, that you by defining w this way, you can where say that the labels that you are computing are the shortest path lines.

So, it is a short-cut, real, because, we are I have exploiting the structure of the algorithm network and we are using the fact that C i j's are non negative right. So, let me give you an example here.

(No audio from 30:58 to 31:48)

So, if this is the network, the distances on the arcs represent your C i j's and you want to find out the shortest path from 1 to 6.

(Refer Slide Time: 31:55)



So, the we will start with the. So, I can just start writing the labels here right. So, initial labels would be, see as I said 0 and then for arcs for nodes 2 and 3, the labels would be 3 and 5 and for all. So, it will be 3 5 and infinity. So, this will be 1 2 3 4 5 and 6, so 6 labels and these are the thing right a starting (( )) right. So, then, by then by addition, w consisted of just node 1 with label 0 right. So, now, you see that. So, I will just indicate the by star, the once which with and see the name for the nodes which get which get admitted to they said w, are called permanently labeled, because, then, their shortest path length of the label will change. Remember, we only update the labels for y, not in w. So, then, nodes which are already in w the labels do not change ok.

So, this is it. So, now, I will choose this one, right I should have put the star later on right. So, since the rho 2 is the smallest, is the smallest is the minimum one right. Therefore, your w now includes this right and then, we have to update, so, that means, you have to look at all nodes which are now connected to both 1 and 2. right See, the minima was over all nodes in w, right you are trying to find out.

So, we can just do it right on the network, here, say for example, for this one, this is connected to 2 4 is connected 2 and the label here would become 8 and the original old label was infinity. right So, that means, here I can write down 0 3 and this one would become 8, right then, for node 3, for example, the earlier node was 5 and the label was 5 from 2 you can come to 3 and the distance is 3 plus 1 4 and I do not have to do the minima, because, I know that the others are all C i j's positive. So, I cannot for example, in fact, I cannot come back to 3 from 2 by any other root.

So, these are the only possible ones. So, therefore, 3 plus 1 4, so, this 1 also gets updated to 4 right and then 2 5 2 5 you have 3 plus 1 right and you have 3 plus 1 plus 4. So, 8 8 and 3 plus 1 4, so, the minimum is 4 and that is smaller than infinity. So, this is this and this is infinity.

So, these are your new labels originally, you had these permanently labeled; now you choose the minimum one.

So, you have in fact, 2 of them at the same time. So, I will choose both of them, nothing, no body stops me from not labeling more than one right. So, therefore, my w, then in case, that becomes 1 2 3 and 5.

So, therefore, now you have this label permanently, this suppose, this label permanently, this is label permanently and then this is. So, now, and the graph you can see immediately. So, to update what will I do from here, you can look at this 1 yeah from 3 we can reach 4. So, this is 5 plus 2 7 and the earlier 1 is 8 right and there is no other path to reach 4, from 2 3 is no other path right, so seven. So, that gets updated to 7 and from 5, you see that this will get updated to. So, 3 plus 4 is the old 1, so 4 plus 3 7.

So, now you have the new labels are 3 4 7 7 sorry 4 and 7, so both of them get a permanently labeled right and so, your w is now 1 2 3 4 5 and 6, so just 3 iterations and you could label the, find out the shortest path. I chose a very simple example, they can be many more roots to choose, but anyways still you see at each iteration you must.

So, what is happening is that, now let me just introduce an element of, you know, concept of complexity of an algorithm, which I have not so far talked about. But, you see here, what is happening, your size of w will induct at least 1 node. See at least 1 node in each iteration right and so, therefore, the number of iterations iterations is n, at most right order n. In fact, because, as we saw in this case, it only 3 iterations, because, I could label 2 nodes at a time. So, it could be order n right and then at each iteration how much work are you doing? At each iteration, you are just updating the will label of the once which are not in w.

So, again, this orders n because, you are simply adding and comparing the minima, so which is, before we say that, all these elementary operations, we consider them as 1 unit of work, so therefore number of iteration this and work work done at each iteration at each iteration is order n.

Therefore, you say total complexity; that means, the number of it is not right coming out very nicely. Total complexity is order n square, which is really nice because, at most, in order n square steps calculations of addition comparison and so on, we will able to find out the shortest path from in a network with having n nodes. So, let me now consider another shortest path algorithm where these C i j's are allowed to be negative, because, Dijkstra's algorithm, we saw that the proof, the validity was dependent on the fact that the C i j's were non negative right.

So, Floyd war shell algorithm in fact, finds for the shortest paths, finds shortest paths between all pairs, not just in one pair, 1 all pairs of nodes when C i j's can be less than 0 right (( )).

So, but then again, when when you have a negative C i j's, it is possible that you may have negative cycles in the network right. So, part of the cycle may be if yeah I can. So, once if this, it is a negative cycle present in the network, then, the shortest path length has no meaning, because, can go on traversing the negative cycle as number of as many number of times I want and I can go on reducing the shortest path length. In fact, so, those shortest path lengths would all be bounded, I can go I can make them go up to minus infinity. So, that is the danger and so, any algorithm which claims to compute shortest path lengths when the C i j's are negative, should also be able to to detect the presence of negative cycle and then stop, because, once the negative cycles are present, then the shortest path length has no meaning, because, you cannot define them right.

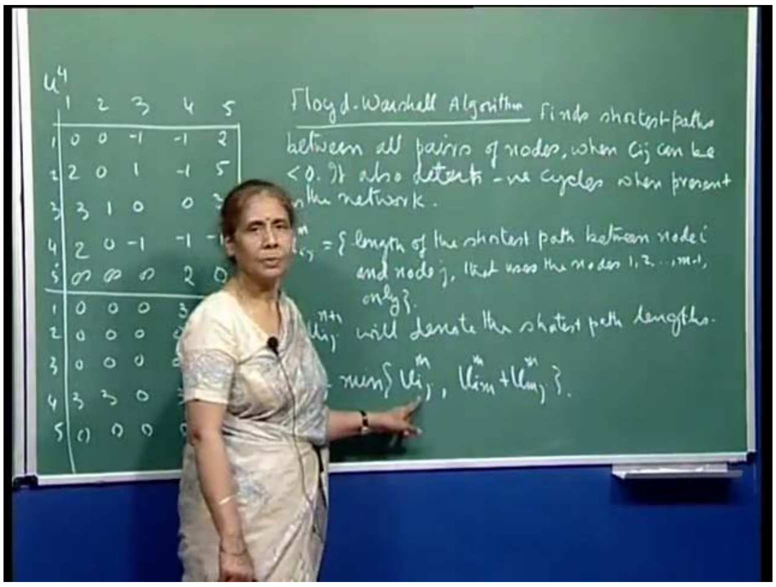
So, or you can say, as it can be negative, it also detect negative cycles when present in the when present in the network and the idea is simple and I will then give later on, its validity.

So, it detects negative cycles when present in the network right. So, definition is simple, what I do is, we define the quantities u i j m as the length of the shortest path between node I and node j that uses the nodes 1 2 2 to m minus 1 only. The idea is that, again it is a partial short shortest path that I have completing and the same that we did not Dijkstra's algorithm. Also, we limited the shortest path to the it is nodes of w, that is, I can only use the node of w and then go on building the shortest path.

So, here also we say that u i j m will denote the length of the shortest path between node i and node j when only nodes up to 1 to m minus 1 have been used. So, necessary that all of them get used, but what we are essentially saying is that, nodes m to n should not be used in the construction of this shortest path from i to j. This is what we do right.

So, and therefore, u i j n plus 1 will denote the shortest path lengths, because, by that time, I would have used all the nodes 1 to n and therefore, my claim should be, we will prove this, I will denote the shortest shortest path length, I (( )). So, once I have gone through all the nodes, that means, there will be n iterations of this algorithm again and so, we will then be able to claim and I will prove to you that the u y j and plus 1 will denote your shortest paths.

(Refer Slide Time: 44:08)



u^4

	1	2	3	4	5
1	0	0	-1	-1	2
2	2	0	1	-1	5
3	3	1	0	0	3
4	2	0	-1	-1	-1
5	∞	∞	∞	2	0
1	0	0	0	3	[illegible]
2	0	0	0	[illegible]	[illegible]
3	0	0	0	[illegible]	[illegible]
4	3	3	0	[illegible]	[illegible]
5	0	0	0	[illegible]	[illegible]

Floyd-Warshall Algorithm finds shortest paths between all pairs of nodes, when c_{ij} can be < 0. It also detects $-$ve cycles when present in the network.

u_{ij}^m = { length of the shortest path between node i and node j, that uses the nodes 1, 2, .., m-1, only }.

u_{ij}^{n+1} will denote the shortest path lengths.

$= \min \{ u_{ij}^m, u_{im}^m + u_{mj}^m \}$.

So, then how do I update? So, the idea here is that, you write u i j m plus 1 has the minimum of, that means, when you go from m to m, if you want to include the node m in the computation of the shortest path, then, you will say that this is u i j m or u i m m plus u m j n. So, this is clear if the shortest path from i to j when we are allowing the inclusion of the node math node.

So, either it goes through the node m or it does not. If it does not go through the node m, then, the earlier shortest path remains intact. So, this must denote the length of the shortest path, because, the if not using the node n, if the node m is used, then obviously, it should be made up of the shortest path length, from I to m, right because, this 1 does not include the node n right a pair minus 1.

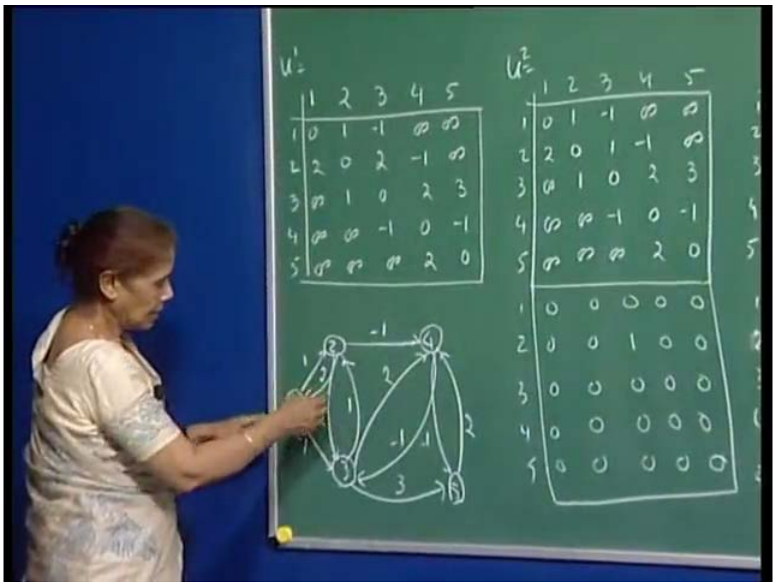
So, you are including node m here, so this plus u m j m, so, from I to m m to j. So, this is the only two possible things and therefore, if you choose the minimum with the 2 as u i j m plus 1, then you have updated the u i j values and you are the saying that this will be the ah shortest path, when you allow nodes up to one to m to be included in the shortest path right. So, this is the thing and so, once you say that this is this we can you can say that u i j n plus 1, actually you can now write as u i j right and you see that this will, because of this, this will always satisfy u i k plus u k j. That means, if you take the triplet i k j, because, you are choosing your updating your distances shortest path lengths by this formula. Once you have done it for all the nodes, then of course, there will be nodes because these things get fixed up and so, u i j will be less than or equal to u i k plus u k. this is sufficient condition for right. So, this is your sufficient condition for. So, it is a sufficient condition for optimality, that means, if the set of numbers u i j indicate it satisfy all these, how many will be there and yes.

So, let us see, we said that we will have n iterations for each pair, because, you are computing from 1 to n plus 1 n iterations and then you are doing or maybe after I show you the computations, you can have a better feeling and you doing it for n square pairs.

So, your computation should not be more than order n-cube, right I mean. So, the if you if you just in a in a very rough way count the basic number of computations that you have to do, that will not in more than or an n square.

Now, we have a various way of representing the computation for these quantities through the matrix and. So, here let us write out, suppose you have given this network.

(Refer Slide Time: 46:50)



$$u^1 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & -1 & \infty & \infty \\ 2 & 2 & 0 & 2 & -1 & \infty \\ 3 & \infty & 1 & 0 & 2 & 3 \\ 4 & \infty & \infty & -1 & 0 & -1 \\ 5 & \infty & \infty & \infty & 2 & 0 \end{array}$$

$$u^2 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & -1 & \infty & \infty \\ 2 & 2 & 0 & 1 & -1 & \infty \\ 3 & \infty & 1 & 0 & 2 & 3 \\ 4 & \infty & \infty & -1 & 0 & -1 \\ 5 & \infty & \infty & \infty & 2 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Again, the numbers are the c i j’s. So, you can see that, quite a few c i j’s are negative and so, let us show the calculations for the Floyd war shell algorithm on this matrix.

So, I have this the numbers node numbers are here and then, so, u i i’s, we will always take to be 0. Right now, right the starting thing, you can say that the c i c i i’s c 1 1 c 2 2 c 3 3 c 4 4 c 5 5 are all 0 and this gives you 1 to 2 1 to 2 1 to 3.

So, these are the show you the distance arcs a distance matrix right. So, then you want to compute, for example, u i j. So, the that this 1 is u i j 1. So, I am calling the matrix capital u on1 and the components are u i j 1 right.

So, this are u i j 1. So, let us use the matrix representation and I will show try to show you the recursions for the Floyd war shell algorithm. So, for example, if you want to compute u 2 1 2, let us say u 2 1 2 right then, this will be, according to a formula minimum of u 1 2 1 and then, it'll be u 1 1 1 plus u 1 2 1.

So, you see here and so, the thing to remember is that, these 2 indices must match right. So, u1 1 plus and this is 0. So, this is u 1 2. So, there is no difference, so, in fact, want to show you that, when you are doing when you are doing the computations for u 2, that means, you are including the node 1.

So, interpretation for this is that, you want the shortest path between 1 and 2, where node 1 is included. So, either you include the node 1 or you do not fine. So, this would be the computation.

But, you see, it will not make a difference, since, we are taking the diagonal elements to be 0. So, this is this right and here the trick is. So, for all distances here for the corresponding nodes and for this, for the pair of nodes and this, they will be no change. It is only for the other the this thing set, the change can take place and then I want to show you.

Say, for example, consider 2 3. So, 2 3 would be yes here u 2 3 u 2 3 1 would be minimum of u 2 sorry this is 2 2 3 1 and then it'll be u 2 1 1 plus u 1 3 1 right and let us quickly see that. So, you wanting the distance between 2 and 3 and so, if you include the node 1, then it is 2 1 and 1 3.

So, this is 2 minus 1, which is 1 and so, there is a change 1 3 u 2 this is 1 ah sorry this is u 2. So, 2 3 is 1, right initially it was 2. So, the moment this change takes place, then I will record the last node visited that means, the change came because of node 1.

So, then you can immediately trace the path, because, the path from 2 to 3 has to go via 1. So, therefore, your shortest path right now, when you are allowing [thu/to]- add node 1 is 2 1 3, that is the path.

So, this matrix will record the changes that occur and the last node that you visit, it while finding out the shortest path. So, this is it. So, now, when you can check that no other change takes place and this is it right and so, similarly, you will compute, this will be your u 3 and then this will be u 4 and now, here we just want to.

So, for example, when you want to compute u 3, then we will underline the this thing here, second row and second column and in this 1. So, now, again there will be no change for the pair of nodes in this row and in this column.

But, there will be, you can consider for example, now I can just maybe show you that this has change, because there is a recorded node here. So, that is 1 4, so 1 4. So, the idea is that, you look at this and then you just go up to the row for which you are considering, for example, we are allowing to include nodes 1 and 2 in the shortest path right.

So, this will be this and this will be this. So, you will compare this entry, I am looking at this recursion formula for you. So, this will be this entry and compare it with the sum of these 2. So, 1 minus 1 is 0.

So, obviously, 0 is smaller than infinity, so that changes and so, one 4 the distance becomes 0 and you visited and that came through the node 2 right.

So, when you want to trace the path for 1 to 4 say for example, then it tells you to go to 2; so, from 1 to 2 and 2 to 4, because there is no change here for 1 4. So, the path you immediately have 1 to 4 and the length of the path, the shortest path.

When you allow nodes1 and 2 to be included in the path is 0. So, this is the idea. So, when you are computing u 4 then, you will circle the third column and the third row and again make the computation, same computation, right any entry you take here, then you all all you have to do is to go this way and this way, then compare this with the sum of these 2 and whenever you find a smaller number, you record that.

So, for example, you can just verify that these are the 3 entries where the change took place and then, when you allowed nodes up to 3, then you will have this computation. You can please check all the computations here, they might be some error.

But, anyway, the change takes place because of node 3 and you see this is what we want to point point out that, u 4 4 u 4 4 is negative, that means, the there is a negative cycle and you can immediately check the cycle because this is 4 to 4.

So, to do that, the change came place from node 3, right so, that means, you go to 4 3 4 3 and then you suppose to then a 4 3, you can look at the this 1 here, there is no change 4 3.

So, your path should be 4 3 and then, it 3, the change took place; yes here the change took place via 2, so 3 2, so you have to visit 2 and then you go to 4. So, if you look at 4 3 3 2 2 1, yes 4 3 is minus 1 3 2 is 1 2 4 is this. So, therefore, the total length of the path is minus 1 and hence, there is a negative cycle and so.

We stop here, because, now the problem is in the sense infeasible, because, the shortest path lengths is not defined, since, I can go on reducing the shortest path lengths by traversing the cycle again and again. So, you can see everything right on the graph and through this matrix method, we can compute these distances and the complexity is order n cube. It also tells you the presence of negative cycles, if there are any, and you can stop.