Advanced Operations Research

Prof. G. Srinivasan

Department of Management Studies

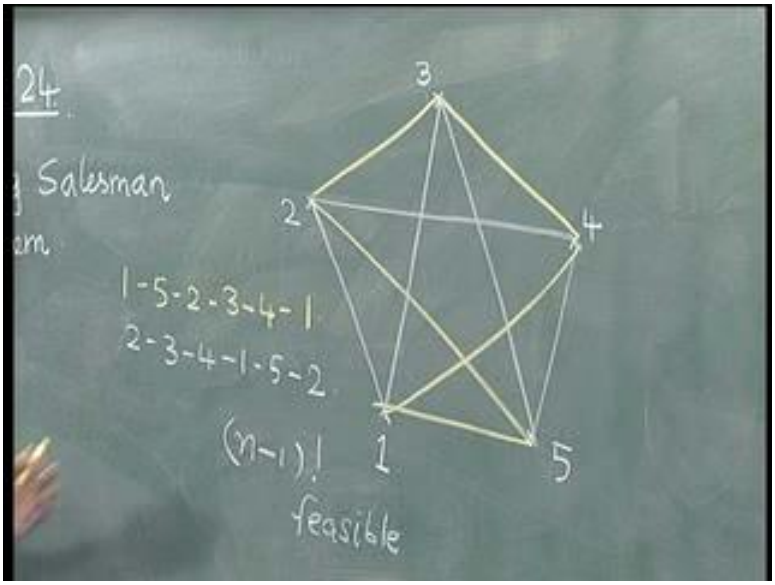
Indian Institute of Technology, Madras

Lecture 24

Travelling Salesman Problem

In this lecture, we discuss the Travelling Salesman Problem. The Travelling Salesman Problem is an extremely important problem in operations research. We first define the problem and then we look at methods or algorithms to solve the Travelling Salesman Problem. Later, we will also discuss the relevance and importance of Travelling Salesman Problem in the OR literature. What is a Travelling Salesman Problem?

(Refer Slide Time: 00:45)



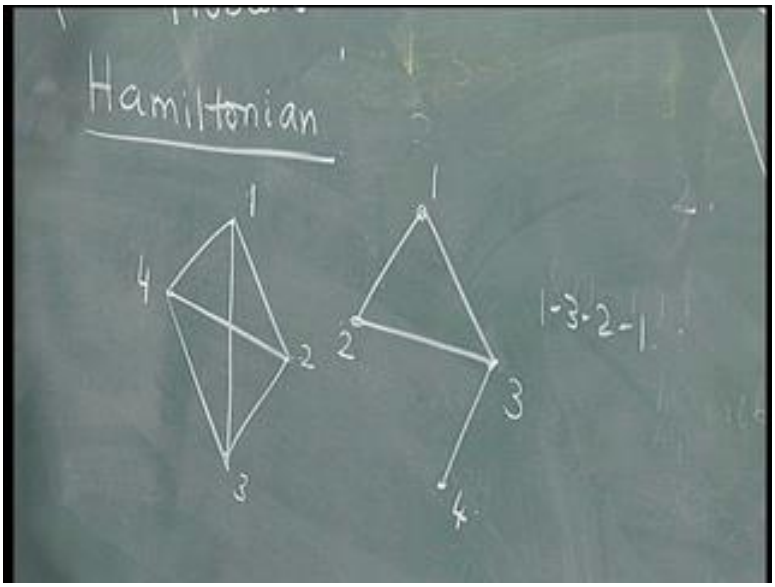
Let us look at a situation where there are five cities or five nodes. Let us say that there is a person who is right now in node 1. Now, this person has to visit each of the remaining nodes once and only once and come back to the starting point. The person may choose to start from 1 and from 1 to go to 2 and say from there to 4, to 5, then to 3 and back to 1 is one feasible solution to the Travelling Salesman Problem. Alternately, another feasible solution could be it goes through 1 to 4, 4 to 3, 3 to 2, 2 to 5 and 5 to 1. Any permutation or order in which the person starts from a particular node, in this case, the given node goes to every other node or vertex once and only once and comes back to the starting point is called the Travelling

Salesman Problem and particularly to find out that tour or circuit, which gives minimum distance travelled or minimum cost travelled.

The problem is: Given a network, given a set of points to visit every node once and only once and come back to the starting point, travelling minimum distance or incurring minimum cost. This is called the Travelling Salesman Problem as it is. In a five city Travelling Salesman Problem, if we assume that the person starts with 1, let us say, a feasible solution could be 1 to 5, 5 to 2, 2 to 3, 3 to 4 and 4 to 1. We also realize that this solution is the same as 2 to 3, 3 to 4, 4 to 1, 1 to 5 and 5 to 2. So, effectively, in a Travelling Salesman Problem, it does not matter which city or which node the travelling salesman starts. The only thing is, from any node the person can start, but the person has to visit every other node once and only once and come back to the starting node.

There are n nodes; we realize that there are n minus 1 factorial feasible solutions, because corresponding to this solution, which say the 1 5 2 3 4 1 is the same as 5 2 3 4 1 5 which is the same as to 2 3 4 1 5 2 and so on. Because each of these n factorial solutions have five solutions that repeat we have n minus 1 factorial feasible solution to the Travelling Salesman Problem. The question is to find out among these n minus 1 factorial feasible solutions, the one with the best value or the minimum distance value. This problem also comes from graph theory.

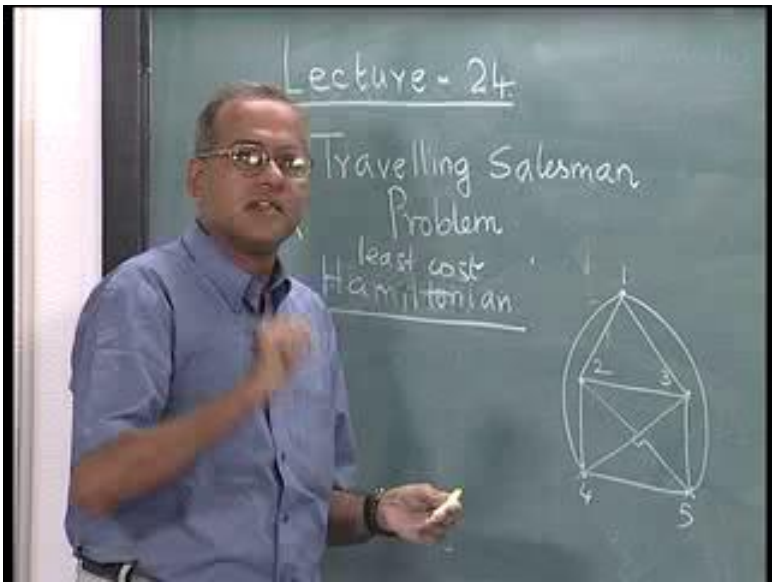
(Refer Slide Time: 04:06)



In graph theory, there is a famous problem of finding out whether a graph is Hamiltonian. If we look at a graph like this, a graph is a collection of nodes and arcs that we saw earlier as part of these. If we look at a graph like this with three nodes and three arcs, it is possible to find a sub graph which is 1 3 2 1, which means from 1, I go through every vertex once and only once and I come back. So, this is Hamiltonian. Whereas, if we have the same graph like this, then starting from 1, I can go to 3, 4 but then, I have to come back to 3 and then come to 2 and to 1. So, this graph is not Hamiltonian.

One of the interesting decision problems in graph theory is if the given graph is Hamiltonian. So, given a graph, the decision problem from graph theory is to find out whether it is Hamiltonian. When we say whether it is Hamiltonian, we say whether there is a Hamiltonian circuit which means I start from this point. If you take this particular graph, this graph is Hamiltonian 1 to 2, 2 to 3, 3 to 4 and 4 to 1. So, it has a Hamiltonian circuit. I could do 1 to 2, 2 to 3, 3 to 4 and 4 to 1 which is the same as 1 to 4, 4 to 3, 3 to 2 and 2 to 1. If I add this also into this graph, I may do 1 2 3 4 1; I may do 1 2 4 3 1 and so on. If a graph is Hamiltonian it may have more than one Hamiltonian circuit. From graph theory we also have graphs which are completely connected graphs.

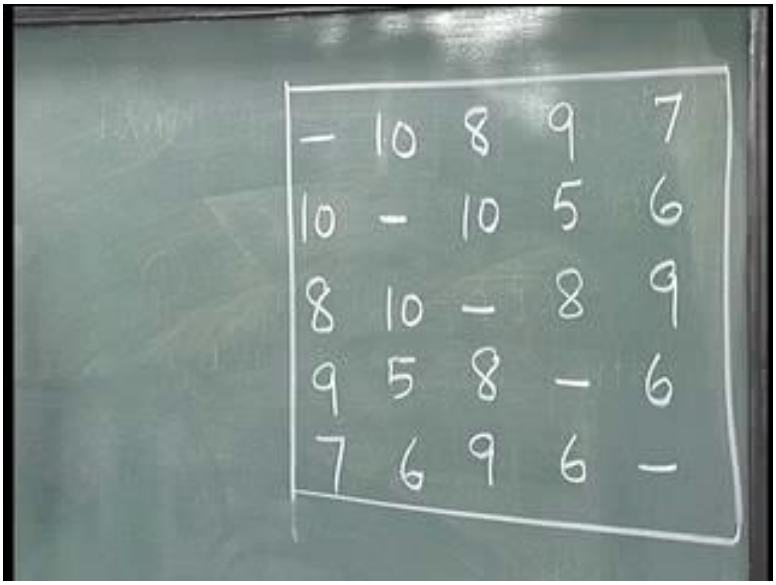
(Refer Slide Time: 06:34)



If we have a graph here where every vertex is connected to every other vertex then clearly this graph is Hamiltonian. You could do 1 2 3 4 5 1; you could do any of these n factorial possibilities. If somebody is given a graph like this and if the decision problem is posed whether this graph is Hamiltonian, then the decision problem is easy to answer because with

every vertex is connected to every other vertex, the graph has to be Hamiltonian. In such a case, the decision problems move on to an optimization problem where we try to find out, given this graph is Hamiltonian and it has more than one Hamiltonian circuit, can we find the least cost Hamiltonian circuit? Now, that leads us to a Travelling Salesman Problem wherein we find out the least cost Hamiltonian circuit in a given graph, knowing that that graph is Hamiltonian. In a TSP we normally assume that it is possible to go from every city to every other city. Usually, the distance data in a Travelling Salesman Problem will look like this.

(Refer Slide Time: 08:02)

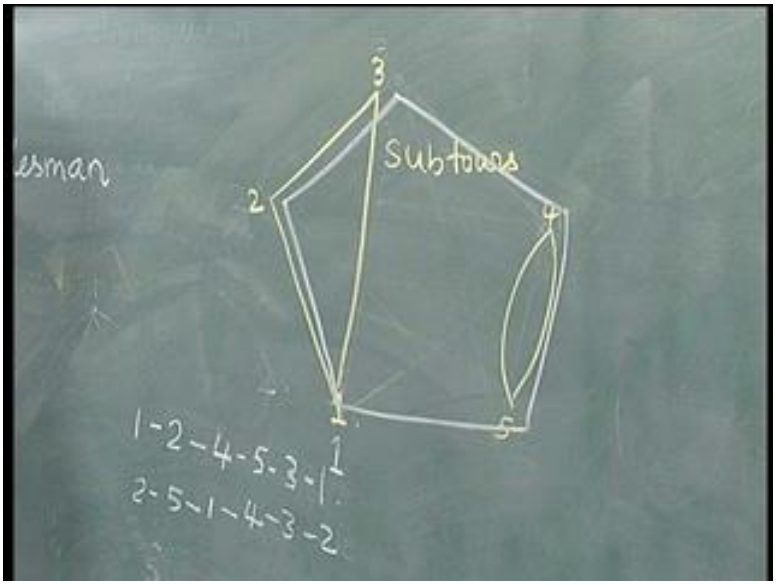


This is for a typical five city Travelling Salesman Problem where all these represent the costs or the distances. For example, to go from 1 to 3 it is 8, to go from 1 to 4 it is 9 and so on. We also observe that we do not have a situation, where we say here that I cannot go from 3 to 4. Usually in a TSP, it is assumed that you can go from every city to every other city. The only other difference is the cost or distance between a node and itself should ordinarily be 0, but we do not put a 0 here. We simply put a dash here whenever we solve a Travelling Salesman Problem where this dash represents, infinity.

Later, we will explain why we have replaced the 0 with the dash, but, otherwise, in a TSP every node is connected to every other node by arcs. This means the graph is complete, which also means, that Hamiltonian circuit exists. If every node is connected to every other node, then n minus 1 factorial Hamiltonian circuit are possible and feasible. The Travelling Salesman Problem is to find out that among the n minus 1 factorial, which has the least total cost or the least total distance. We have already said from this figure that we could have a

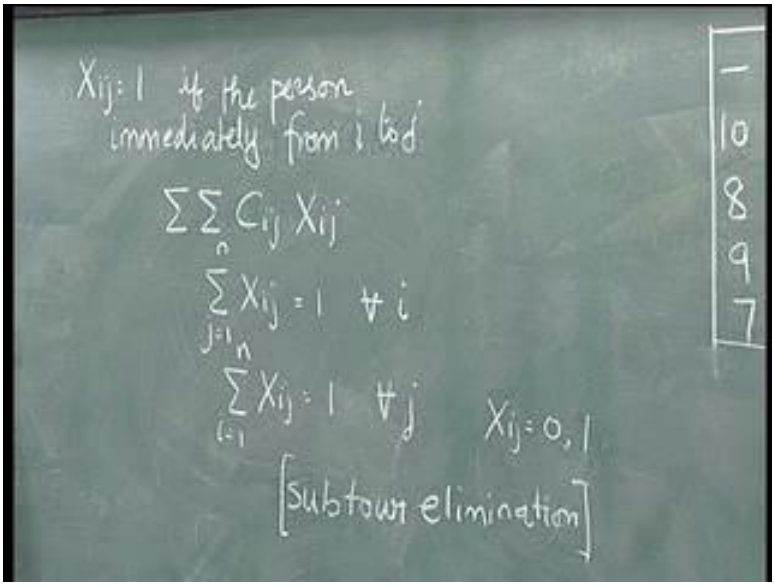
feasible solution 1 to 2, 2 to 4, 4 to 5, 5 to 3, 3 to 1 or we could do 2 to 5, 5 to 1, 1 to 4, 4 to 3, 3 to 2. These are all feasible solutions.

(Refer Slide Time: 10:24)



If we look at something like this, I go from 1 to 2, I go from 2 to 3, come back from 3 to 1; go from 4 to 5 and 5 to 4 is not feasible to the Travelling Salesman Problem. These are called subtours and we should not have subtours. We should have only a full tour where we would have 1 to 2, 2 to 3, 3 to 4, 4 to 5, 5 to 1, which means, I visit every city once and only once and I come back to the starting point. The travelling salesman feasible solution should comprise of tours and should not comprise of subtours. Before we get into solving the Travelling Salesman Problem let us first try and formulate the TSP. Very quickly, we will look at two types of formulations of a TSP.

(Refer Slide Time: 11:16)

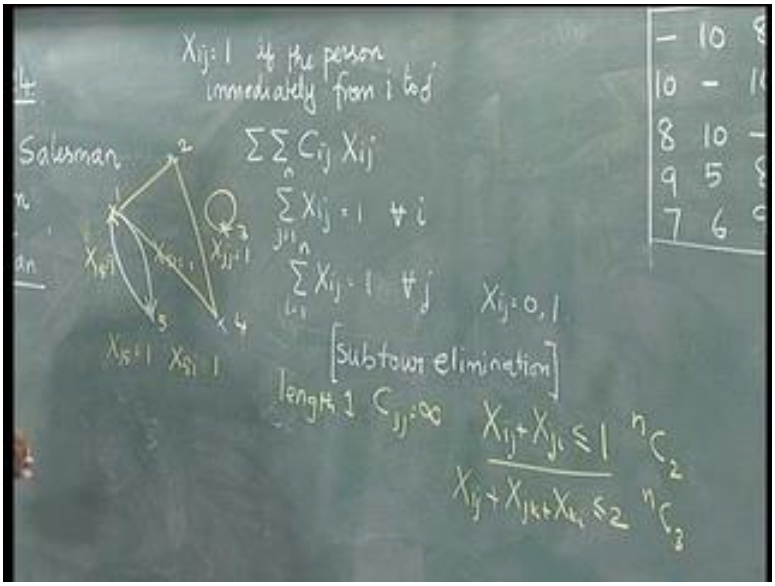


One is we define X_{ij} equal to 1, if the person goes immediately from i to j. The objective function is to minimize the total distance travelled. This will be double sigma C_{ij} X_{ij}. We have already seen that this C_{ij} could represent a cost or it would represent a distance. If we look at a 5 by 5, given by this data, the person has to leave from every city. So, we will have sigma X_{ij} for j equal to 1 to n equal to 1 summed over all i. This means, if I am in city I, I have to go to some other city j and I go to only 1 out of the remaining cities. That is taken care of by summation j equal to 1 to n, from i, it is equal to 1 for every i.

The other one is sigma X_{ij} equal to 1, i equal to 1 to n for every j. This means, if I am in a particular city at the moment, I should have come from only one of the cities into the present city that I am right now in. That is given by X_{ij} equal to 1 summed from i equal to 1 to n for every j; we also know that X_{ij} is 0 or 1. I immediately go from i to j or I do not go from i to j.

So far, the formulation of the Travelling Salesman Problem appears to be like that of the more familiar assignment problem. If you look at it carefully, the assignment problem has exactly this formulation. The only difference, of course, is because of unimodularity; we put X_{ij} greater than or equal to 0. In a TSP, we do not have that; so, we need to add some more constraints into the Travelling Salesman Problem. Those constraints are called subtour elimination constraints.

(Refer Slide Time: 13:53)



What are these subtour elimination constraints?

If I have, say, five cities; if I get into a solution like this, where I have, say this is 1 and 5, X_{15} equal to 1 and X_{51}equal to 1. Now, this is a subtour. We should not have subtours of this type. If we have a five city TSP, we could have a subtour of length 1, that is, you may have X_{jj} equal to 1, which is called subtour of length 1; this is subtour of length 2 and something like this is subtour of length 3; we could have subtour of length 4 and so on. We could have subtours of several lengths starting from 1 to n minus 1 if we are looking at an n city Travelling Salesman Problem and each of these subtours should be eliminated.

One of the ways of doing this is to eliminate subtours of length 1 by simply putting X_{jj} equal to 0, so that I do not have a solution that has X_{11} equal to 1 or X_{22} equal to 1. So, I will not have something like this, (Refer Slide Time: 15:07). That is one way of doing it, explicitly. The other way of doing is to declare the distance between the point and itself to infinity, represented by a dash, so that it does not come into the solution. That is precisely, the reason, why we have the dashes coming here instead of the 0s. Otherwise, we would always have diagonal assignments with 0s being here. We will have diagonal assignments; we do not want diagonal assignment because diagonal assignment represents subtour of length 1. We eliminate subtours of length 1, not by explicitly putting X_{jj} equal to 0, but by putting C_{jj} equal to infinity. The cost associated with the diagonals are all infinity, so that we do not have subtours of length 1. We want to eliminate subtours of length 2. We need to add a constraint like this type X_{ij} plus X_{ji} is less than or equal to 1. What does this constraint do? If we look at

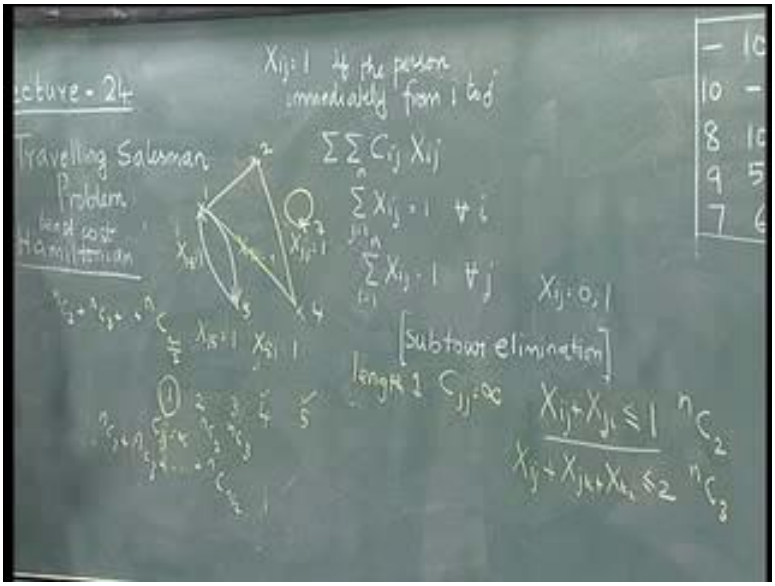
specifically this pair, this would mean, if this subtour exists then X_{15} equal to 1 and X_{51}equal to 1, which means X_{ij} plus X_{ji} should be equal to 2. The moment we add this constraint, then this will make sure that this situation does not happen.

If X_{15} is in the solution, then X_{51}cannot be in the solution. If X_{51} is in the solution then X_{15} will not be in the solution. We can have a situation where both of them are not in the solution. Therefore, this is a good way by which we eliminate subtours of length 2. Similarly, subtours of length 3 can be eliminated by X_{ij} plus X_{jk} plus X_{ki} is less than or equal to 2, which would eliminate subtours of length 3, which means, we will not have this situation. If for example, this is 1, this is 5, this is 2, this is 4 and this is 3, then if X_{12} equal to 1 and 2, 4 equal to 1, X_{41} cannot be equal to 1; otherwise, it will violate this constraint.

Any general subtour elimination constraint can be written like this: if we want to eliminate subtours of length k, then we can put X_{ij} up to k terms, is less than or equal to k minus 1. That is another way to start doing the subtour elimination constraints. If there are n nodes then we have superscript n C_2 constraints here because this is done for every pair. We have superscript n C_3 here because we are eliminating subtours of length 3 and then superscript n C_4, superscript n C_5 and so on. One of the problems in the travelling salesman formulation is that the subtour elimination constraints are large and many. If we are looking at n equal to 5, then superscript n C_2 is 10, superscript n C_3 is equal to superscript n C_2 which is also 10. Whereas, if you are looking at a twenty city problem, then superscript n C_2 will be 20 into 19 by 2, which is 190 constraints and so on.

The Travelling Salesman Problem belongs to a class of problems where we have a large number of constraints and typically exponentially increasing number of constraints, because any superscript n C_r can be treated as exponential. It has a large number of constraints associated with this particular problem. The next issue that happens is can we still reduce the number of constraints from this particular number? Let us go back and look at the same example that we have.

(Refer Slide Time: 19:10)



We have to eliminate subtours of length 1, subtours of length 2, subtours of length 3 and subtours of length 4. We are solving a five city Travelling Salesman Problem. If we were solving a ten city problem, then we have to eliminate 1, 2, 3, and 4, up to 9. We have already eliminated subtours of length 1 by putting C_{jj} equal to infinity. We should eliminate subtours of length 2 by adding this constraint, which is a superscript n C_2 constraint that we add now. How do we eliminate subtours of length 3? If there is a subtour of length 3 and there are five cities, there has to be another subtour. You cannot have a situation where there is only one subtour; if there is a subtour, then, there has to be another subtour. There will be more than one subtour if you have it.

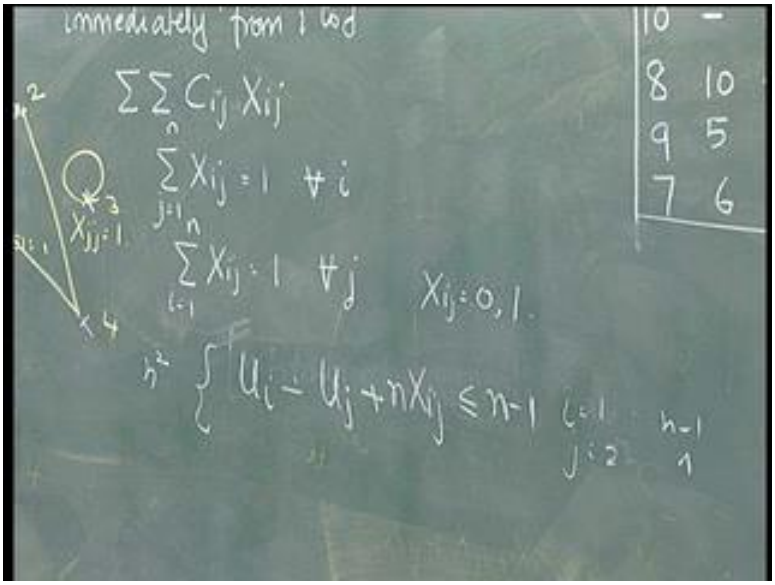
If there is a subtour of length 3, then there should be subtours of length 2 or length 1, so that the 5 is taken care of. For every subtour of length 3, there has to be another subtour, at least one more subtour, which is either a subtour of length 1 or subtour of length 2. By eliminating subtours of lengths 1 and 2, we automatically eliminate subtours of length 3. Similarly, if we have a subtour of length 4, then there has to be a subtour of length 1. So, by eliminating all subtours of length 1, we automatically eliminate subtours of length 4. For a five city problem, it is enough to eliminate subtours of length 1 and 2. Do not worry about 3 and 4. Already 1 is eliminated by this way, so we only add superscript n C_2 constraints in a five city TSP.

In a seven city TSP, we will have to eliminate subtours of length 1, 2 and 3 because 4, 5, 6 and 7, we can eliminate by carefully eliminating 1, 2 and 3. 1 is always eliminated by this, so for a seven city, we need to add superscript n C_2 plus superscript n C_3. When n is an odd

number, then the number of constraints that we will add is superscript n C_2 plus superscript n C_3 and so on up to superscript n C_n minus 1 by 2. So this many constraints we add if n is odd. If n is even then, let us say we have six cities. We should eliminate subtours of length 1, 2, 3, 4 and 5. We eliminate this by putting C_{jj} equal to infinity; this you do by superscript n C_2. You have to do this again by doing this superscript n C_3, then you can go back and say for every subtour of length 4, we should have subtours of lengths 2 and 1. So, by eliminating these two, we have eliminated this and by eliminating this, we eliminate this. So, when n is even, we end up doing superscript n C_2 plus superscript n C_3 plus etc., plus superscript n C_n by 2.

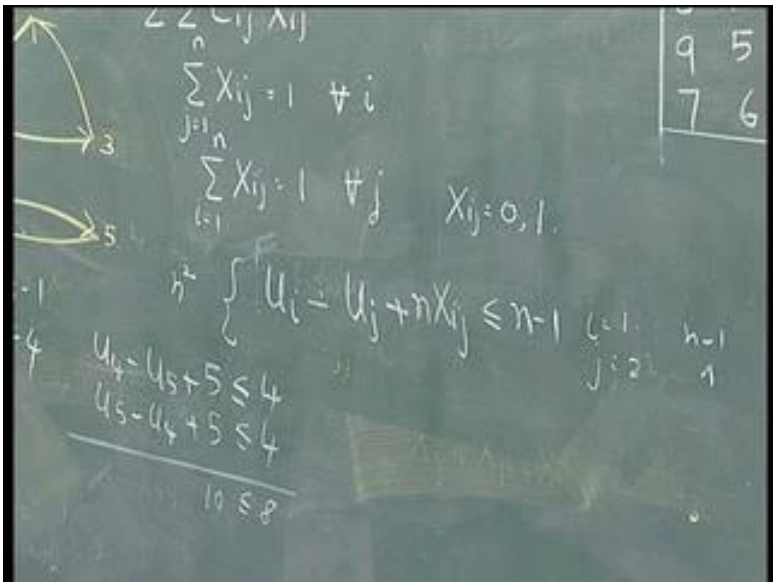
This way we actually reduce the number of constraints here. When we first formulated, we said we have to do n C_1, n C_2, n C_3 and up to n C_n minus one. We realize by carefully looking at the problem: if n is odd, it is enough to go up to superscript n C_n minus 1 by 2; if n is even, go up to superscript n C_n by 2. The number of constraints to a Travelling Salesman Problem is still very large and usually Travelling Salesman Problems are not solved directly by integer programming and by formulating it this way. This is only to understand the Travelling Salesman Problem represents a class of problems where the number of constraints in a particular type of formulation, the number of constraints is large exponential and it increases with increase in the number of nodes. Much later, people came up with a very interesting form of a subtour elimination constraint and we will look at it that way.

(Refer Slide Time: 23:45)



Instead of explicitly avoiding subtours of length 2, 3, etc., up to n minus 1, this kind of a constraint was used. We simply said U_i minus U_j plus nX_{ij} is less than or equal to n minus 1, for i is equal to 1 to n minus 1 and for j equal to 2 to n. Effectively, this has about n square constraints or n minus 1 square constraints about n square constraints, because i is equal to 1 to n minus 1, j is equal 2 to n. How do these constraints work? We have also introduced U_i and U_j, so n more variables into the formulation. Let us see how this constraint works.

(Refer Slide Time: 24:34)



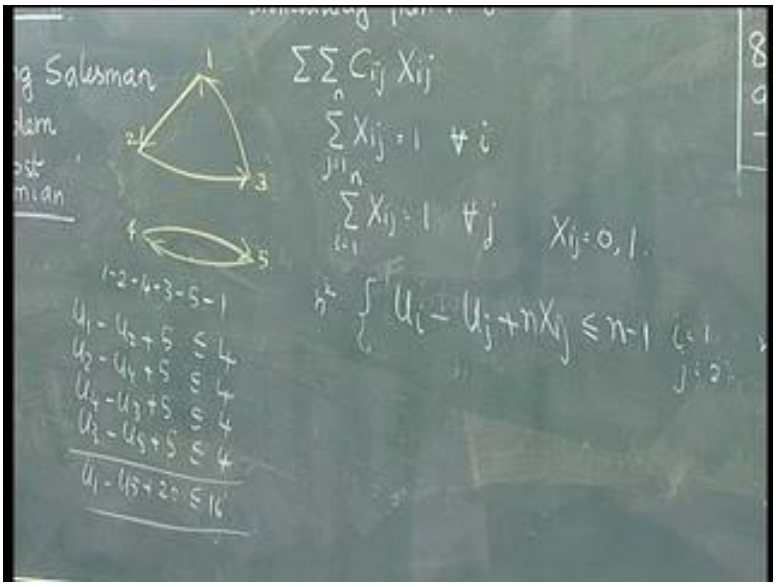
Let us take a situation where we have a subtour 1 2 3 1 and 4 5 4 for a TSP that is a five city problem. Let us say I have 1 2 3 1 and I have 4 5 1 to 2, 2 to 3, 3 to 1, 4 to 5, 5 to 4. If we have this kind of a constraint, then let us try and apply this now. U_i minus U_j minus nX_{ij} is less than or equal to n minus 1. Ordinarily, we will have U_1 minus U_2 minus $5X_{12}$ is less than or equal to 4. For 2 to 3, I will have U_2 minus U_3 plus $5X_{23}$ is less than or equal to 4. It is plus $5X_{23}$ in the first statement. Then, it is not defined for j equal to 1, so I do not have the third constraint.

I do not have a third constraint which says U_3 minus U_1 plus $5X_{31}$ less than or equal to 4. I have only two constraints because it is not defined for j equal to 1. When I add vertically, I will get U_1 minus U_3 plus 10 is less than or equal to 8. Assuming that this is equal to 1 and this is equal to 1 it is always possible to define U_1 and U_3 such that this constraint is satisfied. Therefore, when we consider a subtour involving 1, we are unable to see that this is a subtour elimination constraint. But, in fact, it is a subtour elimination constraint because for every

subtour that involves city 1, there will be a subtour that does not involve city 1. This means because this subtour involves city 1, this subtour does not involves city 1.

What are the corresponding equations here? This is U_4 minus U_5 plus 5 is less than or equal 4 and U_5 minus U_4 plus 5 less than or equal to 4. Adding these two, 10 is less than or equal to 8. For the subtour that does not involve 1, this is a subtour elimination constraint because we cannot define values for any U_4 and U_5. We will end up getting 10 less or equal to 8, which will violate this particular constraint. So, this will become a subtour elimination constraint for every subtour that does not involve 1. We also know that for every subtour that does not involve 1, there is a subtour that involves 1; therefore, that also gets eliminated. So this becomes a very valid subtour elimination constraint for every subtour, but we also have to show that this is valid for every tour.

(Refer Slide Time: 27:35)



A tour should have 1. So, if we have a tour which is 1 2 4 3 5 1, then we have U_1 minus U_2 plus 5 is less or equal to 4; U_2 minus U_4 plus 5 less than or equal to 4; U_4 minus U_3 plus 5 less than or equal to 4; U_3 minus U_5 plus 5 less than or equal to 4 and it is not defined for 1. When we add vertically, you will get U_1 minus U_5 plus 20 is less than or equal to 16. It is always possible to find U_1 and U_5 such that this is satisfied.

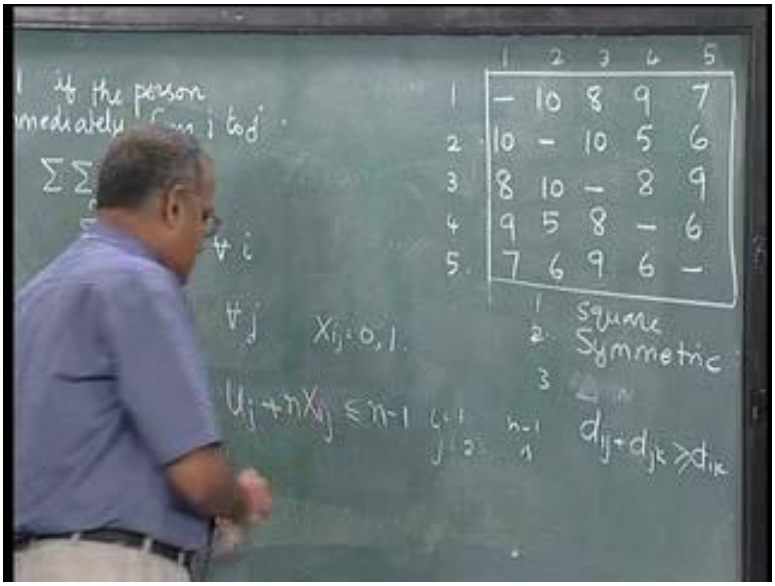
Every tour will satisfy this, so this will not eliminate a tour. This will eliminate only a subtour and it will eliminate all subtours. This becomes a very valid subtour elimination constraint for the Travelling Salesman Problem. This, with fewer than n square constraints makes it a little

easier from the formulation point of view because otherwise we were looking at superscript n C_2 plus superscript n C_3 plus up to superscript n C_n by 2, which is a much larger number than n square. This kind of a substitution has been used extensively by people who actually formulated the TSP as an integer programming problem. There are also further refinements to this from the literature, some of which have made it slightly more elegant with respect to this formulation. Nevertheless, we also as mentioned earlier we do not solve the Travelling Salesman Problem optimally by using this integer programming formulation. It becomes extremely cumbersome.

Travelling Salesman Problems are solved to exactness or to optimality normally by using branch and bound algorithms that provide exact solution to the Travelling Salesman Problem. We also know that branch and bound algorithms are all worse case exponential, in the sense, they do not guarantee polynomial running time. We could get into situations where we consume a large amount of CPU time before we terminate or we could have situations where we do not terminate at all. They are solved using branch and bound algorithms to begin with; and sometimes, we resort to heuristic algorithm which are fast, but which do not guarantee exactness or exact solutions.

We now see different versions of branch and bound algorithm to solve the Travelling Salesman Problem. Later, we also see some heuristic solutions to the Travelling Salesman Problem. So, we will first look at the branch and bound solution for this sum. Let us look at a branch and bound 1 to do this. Now, this is a 5 by 5 or a five city Travelling Salesman Problem.

(Refer Slide Time: 30:47)



We use this data, so we call this city 1 2 3 4 and 5. This is city 1 2 3 4 and 5, now let us see what we can understand from this matrix. One thing is if the person is at city 1, then the person has to leave city 1 by going to one of these four. Therefore, the person has to travel a minimum distance of 7 to leave city 1. Similarly, when the person reaches city 2, the person should travel a minimum distance of 5, a minimum distance of 8, a minimum distance of 5 and a minimum distance of 6. The sum of these minimum distances will have to necessarily be some kind of a minimum total distance that this person has to travel. That is given by the sum of the row minima which is 7 plus 5, 12 plus 8, 20 plus 5, 25 plus 6, 31. This 31 is some kind of a bare minimum that this person has to travel total distance. This 31 is a lower bound to the actual distance that the person has to travel. So row minimum gives us a good lower bound and we say that this person has to travel at least 31. The optimum solution will have to be greater than or equal to the lower bound of 31.

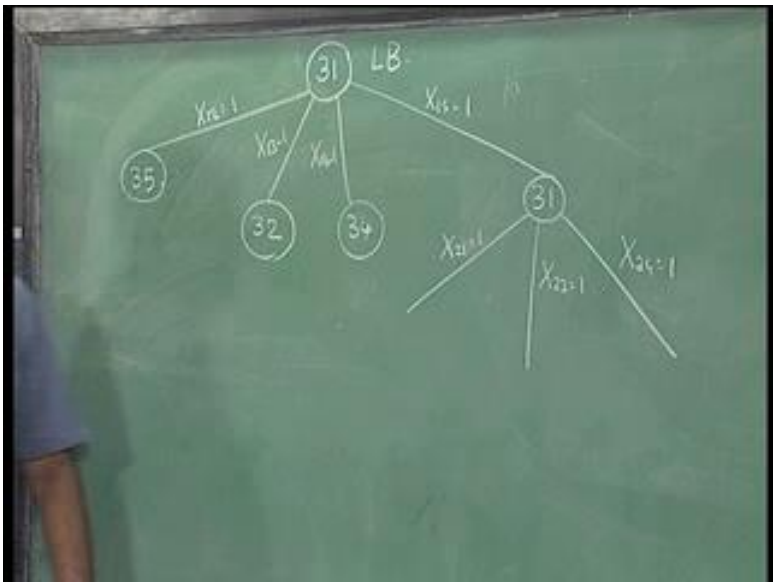
Also, realize that the column subtraction would also do something. For example, what does this column subtraction tell us? In order to reach 1, this person should have traveled a minimum of 7, a minimum of 5, a minimum of 8, a minimum of 5 and a minimum of 6, which would be the same 31 because the travelling salesman matrix given here is symmetric. Ordinarily, travelling salesman matrices or distance matrices are given as symmetric because they represent distance. Distance, usually, satisfies symmetry and triangle inequality. The Travelling Salesman Problem matrix is usually square, symmetric and satisfies triangle

inequality, which means, given any three distances, d_{ij} plus d_{jk} is greater than or equal to d_{ik}. So, it satisfies this particular inequality also.

Because of this inequality we can always go back and say that if we are looking at a TSP, the person will visit every city once and only once. If you have a situation where the person has to come back to that city already visited, then it actually violates the triangle inequality. Therefore, that will not happen. So, whenever the matrix satisfies triangle inequality, we can always show that the person will visit every city once and only once. There can be situations where we do not have a symmetric matrix, in which case, the row minima could be different from the sum of the column minima. We can consistently use either the row minima or the column minima to represent the lower bound.

In this case, we are going to use a symmetric matrix. To begin with, if we use the row minima or the column minima, we are going to get the same 31 as the lower bound. From this, what else can we do?

(Refer Slide Time: 34:40)



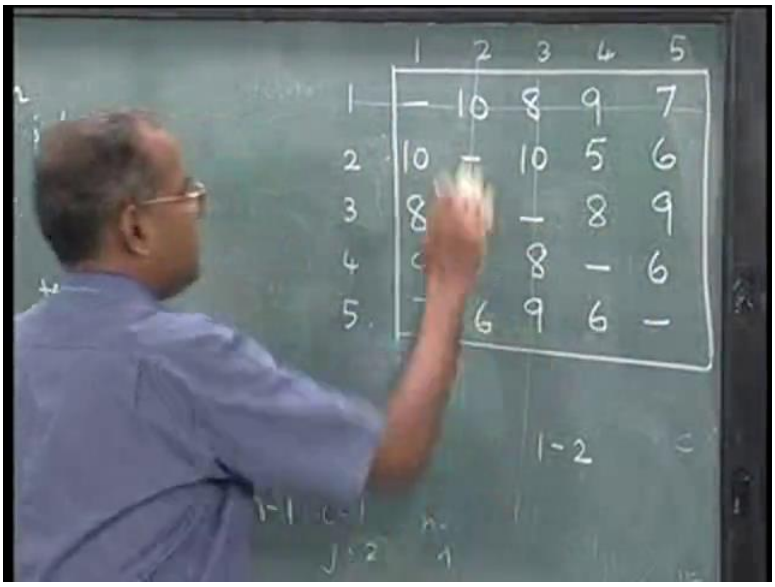
We create four branches or n minus 1 branches and say X_{12} equal to 1, X_{13} equal to 1, X_{14} equal to 1 and X_{15} equal to 1. We do not have X_{11} equal to 1 because X_{11} is a subtour of length 1; you do not have X_{11} in the solution or X_{jj} the solution. So, we make four branches or n minus 1 branches at this stage.

When we fix X_{12} to 1, it means we are assuming that this person is going to go from 1 to 2. We temporarily leave out this first row and the second column and then see the additional

minimum distance that this person has to travel. Since we know that he is going from 1 to 2, now having reached 2, the person has to leave 2. Therefore, he would have to travel this row minimum which is 5, this row minimum which is 8, now this 6 and this 6. So, the sum of the row minima, the moment we fix X_{12} equal to 1, we leave out the first row and the second column and for the reduced 4 by 4 matrix we try to add the row minima. That will become 5 plus 8, 13 plus 6, 19 plus 6, 25, 25 plus 10 we will get 35 as the lower bound.

I repeat. We fixed this 10; we leave out the first row and the second column. For the remaining 4 by 4, the row minimum is 5 plus 8 is 13, 13 plus 6 is 19. This 5 is not counted because this column has been left out. This is 13 plus 6 is 19, 19 plus 6 is 25, 25 plus 10 is 35. The motivation is that the person already goes from 1 to 2. So in order to go from 2, the person has to travel a minimum of 5. In order to go from 3, the person has to travel a minimum of 8, here a minimum of 6. This is not counted because the person has already reached 2. This should not be considered, so minimum of 6 and yet another minimum of 6, to get a bare minimum of 35, the person has to travel if you fix this 10. This would tell us that if X_{12} equal to 1, which means the person goes from 1 to 2, immediately, the minimum distance that the person travels should be 35 or more. For 1 3 equal to 1, we can do something similar.

(Refer Slide Time: 37: 40)



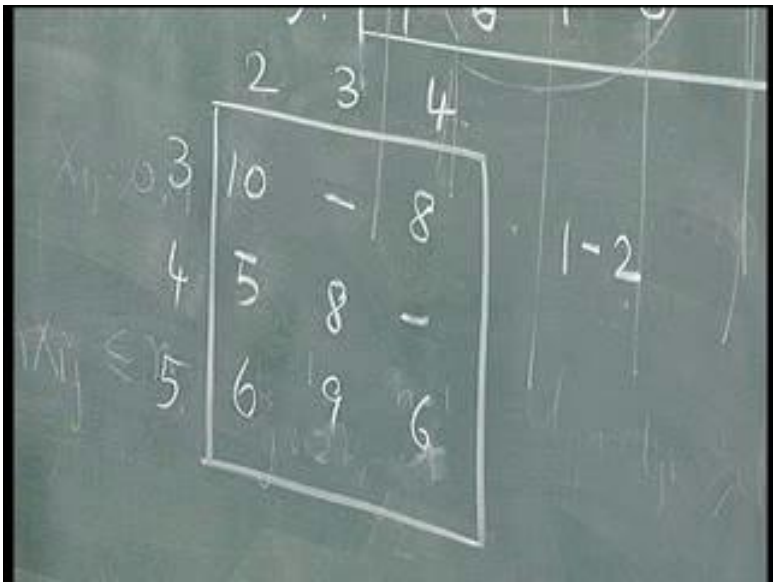
For 1 3, leave this out and leave this column. This goes and take the row minima this 8 is fixed; the remaining minimum would be 5 plus 8, 13 plus 5, 18 plus 6, 24. I repeat minimum 5, minimum here is 8, minimum here is 5, minimum here is 6, 6plus 5, 11 plus 8, 19 plus 5,

24 plus the fixed 8 would give us 32. If the person goes from 1 to 3, we would say that the minimum distance the person has to travel is 32. For every i,j that is fixed, leave out the i^{th} row and j^{th} column; take the reduced matrix and then find out the sum of the row minima and then add the fixed values to get this 32.

When we fix 1 4, we do this and this goes, so this 9 is fixed; row minimum is 6 here because this 5 goes; 6 plus 8 is 14, 14 plus 5 is 19, 19 plus 6 is 25, 25 plus 9 is 34. When we fix 1 5, this row and this column goes; this 7 is fixed, so we have 5, which is the row minimum; 5 plus 8 is 13, 13 plus 5 is 18, 18 plus 6 is 24, 24 plus 7 is 31.

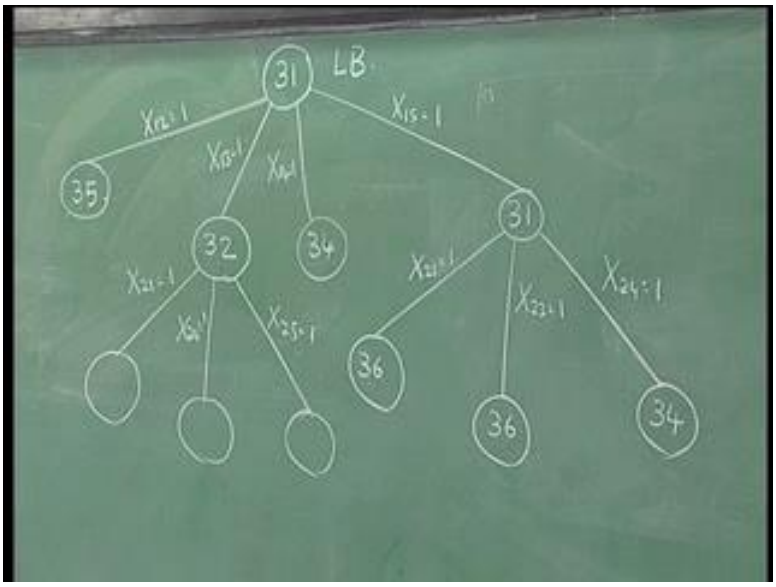
What we have done is at 1 level, we have said that if the person goes from 1 to 2. It is 35 or more, 32 or more, 34 or more, 31 or more. Then, we want to minimize the total distance travelled. So, we branch further from this because it has the minimum value of the lower bound. When we branch from here, you create three more branches. There were four branches created here, now we have to create three more branches. We now say that here the person will do X_{21}equal to 1, this will be X_{23} equal to 1 and this will be X_{24} equal to 1; we will avoid 2 2 because it is a subtour and we will not have 2 5 because we already have put 1 5. The person has already reached 5 so the person will not go from 2 to 5 again. From 2 the person can do only these three things. Now, we want to find out this. To make this better, here, I have 1 5 and 2 1. I have 1 5, so I freeze this row, this column. I also freeze this row and this column. Effectively, I have only this portion of the matrix available with me.

(Refer Slide Time: 40:47)



What I have here are 3 4 and 5 and I have 2 3 and 4. I have 10 dash 8, 5 8 dash, 6 9 6. We can actually do one more thing. This (Refer Slide Time: 40:47) would mean that 2 to 1 and 1 to 5. I have 2 to 1, 1 to 5, so, I should not have 5 to 2; otherwise it will be a subtour. I put 5 to 2 as another dash here temporarily and then I can find out the row minimum. The row minimum would give us 8 plus 5, 13 plus 6, 19 and 19 plus the 2 fixed values, that is, 19 plus 7 is 26, 26 plus 10 is 36. Let me repeat this again.

(Refer Slide Time: 41:48)

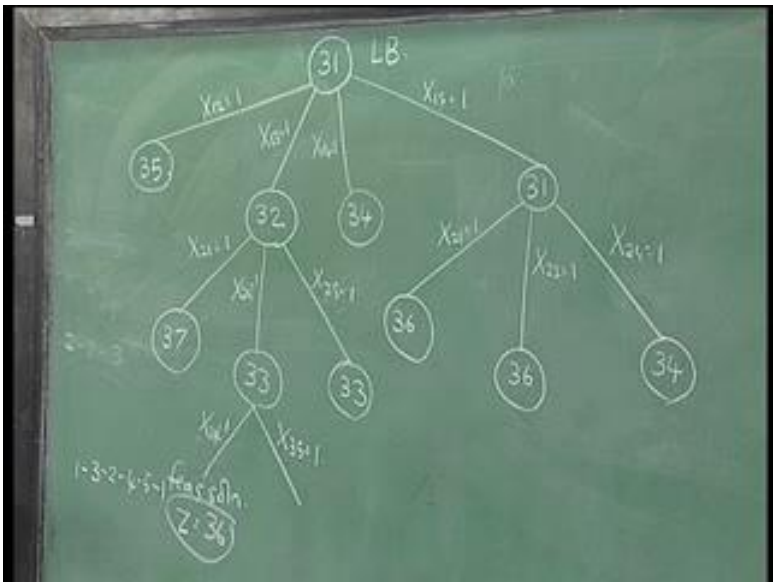


This node I fixed X_{15} to 1 and X_{21} to 1. I go back. I leave out the first row fifth column, second row first column, get the remaining 3 by 3 here. After we get that from the allocation, I realize that I do 2 to 1 and 1 to 5. So 2 to 1, 1 to 5 would mean, 5 to 2 should not be there; otherwise, it will become a subtour. I will go back and just put temporarily a dash here, so that I do not have this and then I compute the row minimum, 8 plus 5 is 13, plus 6 is 19 and 19 plus 10 is 29, plus 7 is 36. Now, to evaluate this 1 5 and 2 3, I go back, so 3 4 5 and 1 5, this one goes, 2 3 this one goes. So, I have 1 2 and 4 that is here. The values of 3 to 1 is 8 10 8, 9 5 dash, 7 6 and 6.

I have 1 to 5, now this is 1 to 5 and 2 to 3 and because I have 2 to 3, I should not have 3 to 2, so I put a dash here and get the row minimum 8 plus 5, 13 plus 6 is 19. 1 to 5 is 19 plus 7 is 26 plus 10, I get 36. So, when I do 1 to 5 and 2 to 3, I get 36 here. I do the third one which is 1 5 2 4. I eliminate this 1 5 and I do 2 4, so I have 3 4 5, 1 2 3, I get 8 10 dash, 9 5 8, 7 6 9. Since, I have 2 4, I should not have 4 2, so the row minimum would give us 8 plus 8 is 16 plus 6 is 22, 22 plus 1 to 5 is 7, 29 plus 2 to 5 is 34, so I get 34 here.

Now, I have evaluated up to this. Out of these nodes which are here, I look at that node which has the minimum value which is 32 here. I decide to branch from this, so that I may get a solution with 32 or slightly more than that; branching from here would give me a solution with 36 or more. I would like to get a smaller value as possible, so I tried to branch this node. I already have 1 3, so I create three branches here. I put X_{21} equal to 1, I will not have X_{22} because it is a subtour, I will not have X_{23} because I already have a X_{13}. I will have a X_{24} equal to 1 and I will have X_{25} equal to 1. I will go back and try to find out these three values. (Refer Slide Time: 45:30). For the first one, I put 1 3 and 2 1. So, 1 3 goes, 2 1 goes, which means, I have these remaining three. I have 3 4 5 and I have 2 4 5 here. I have used 1 3 and 2 1 and so 2 4 5, 10 8 9, 5 dash 6, 6 6 dash. I have used 1 3 and 2 1. I have 2 to 1, 1 to 3; I should not have 3 to 2, so 3 to 2 goes from here. Sum of the row minima will be 8 plus 5 is 13, plus 6 is 19, 19 plus 10, 29 plus 8 is 37.

(Refer Slide Time: 46:29)



The next one I do 1 3 and 2 4. I eliminated 1 3, I eliminated 2 4 and I have 1 2 and 5 and the values will be 3 1 is 8 10 9, 9 5 6, 7 6 dash. Again, I have used 1 3 and 2 4, because I have 2 4 here and should not have 4 2, otherwise, I will have a subtour; so, I eliminate this 4 2. I subtract the row minimum 8 plus 6 is 14, plus 6 is 20 and 20 plus 8 is 28, 28 plus 5 is 33. I go back and do 1 3 and 2 5. So (Refer Slide Time: 47:35) 1 3 is here, 2 5 is here and so 1, 2 and 4 are remaining. So, 3 1 is 8, 3 2 is 10, this is 8, 4 1 is 9, 4 2 is 5, 4 5 is dash, 5 1 is 7, 5 2 is 6, 5 4 is 6. Since I have 2 5 equal to 1, 5 2 will be not be there, which is a dash. Now, take the row minimum 8 plus 5, 13 plus 6 is 19, 19 plus 6 is 25, 25 8 is 33. Out of these values, I will

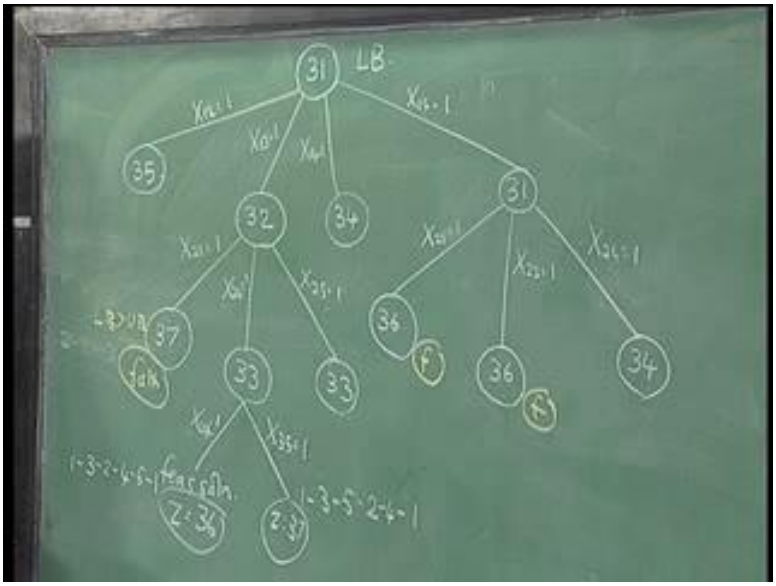
again try to find out whichever is the smallest in terms of these values and I branch from there. I could branch from either this or from this. I look at this; I create two branches, 1 3 2 4, I can do X_{31}. I will not do it. I can do X_{31}, I cannot do X_{31} because X_{13} equal to 1, I cannot do X_{31}.

I do X_{32}, I cannot do X_{33} because that is a subtour. I will not do X_{34} because I already have 2 4 here. I will do X_{35} equal to 1. I fixed three terms; I have only two more terms to go. Whenever I have two more terms to go, I do not find the lower bound. I would only find the upper bound or the feasible solution. I try to find a feasible solution here and how do I find the feasible solution? I have here 1 to 3, this would mean 1 to 3, 3 to 2, 2 to 4. Automatically, it has to be 4 to 5 and 5 to 1.

So, 1 to 3 is 8, 3 to 2 is 18, 2 to 4 is 23, 4 to 5 is 29, 5 to 1 7 36. So, I have a feasible solution with Z equal to 36 at this point. The moment I have a feasible solution with Z equal to 36, I can do a few things. I realize that I do not need to do proceed from here, because if I proceed from here down, I will get values 37 or more. One of the properties that we have here in this branch and bound tree is that, as we move down, the value can only increase and it cannot decrease. When I move down from here, I will get 37 or more.

I already have a solution with 36, so I fathom this because lower bound is greater than current upper bound. Because this 36 tells me that since I have a feasible solution with 36, my optimal solution is either 36 or less. Since the lower bound is greater than the upper bound, there is no point in doing this. Similarly, I can fathom this as well as this, because even if I proceed from here, I will get 36 or more. I already have a solution with 36. I am not interested in proceeding from this.

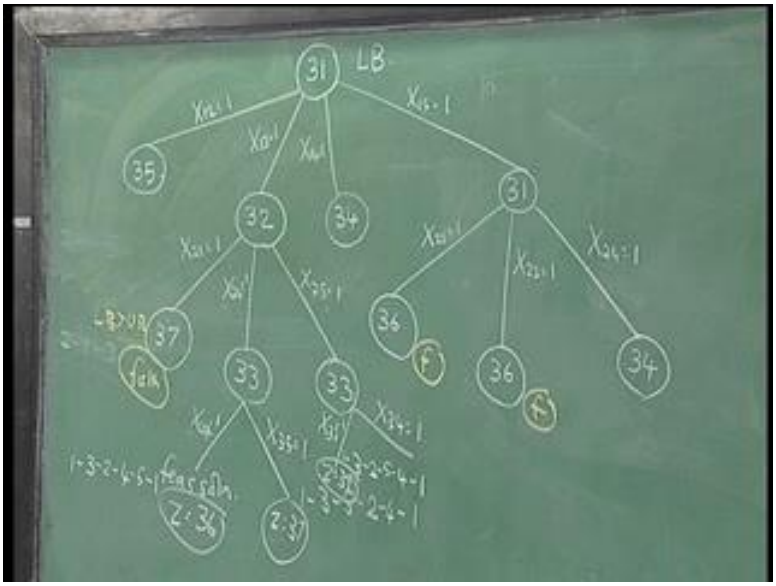
(Refer Slide Time: 51:10)



I go back to this one and see what feasible solution I will get now. This is 1 to 3, 3 to 5 and I have 2 to 4. So, this would give me 1 to 3, 2 to 4 and 3 to 5. I will go back. Therefore (Refer Slide Time: 51:33) 1 to 3, 2 to 4 and 3 to 5 would give me 4 5 and 1 2. 1 to 3, 1 to 3, 2 to 4, 3 to 5 would give me 4 5 and 1 2. I will have 9, 5, 7 6 and since I have 1 to 3, 2 to 4, 3 to 5, so 2 to 4, I will not have 4 to 2 should not be there. So, this is the only possibility, 4 to 1 and 5 to 2. 5 to 2, 2 to 4 and 4 to 1 is the only possibility that I can have here.

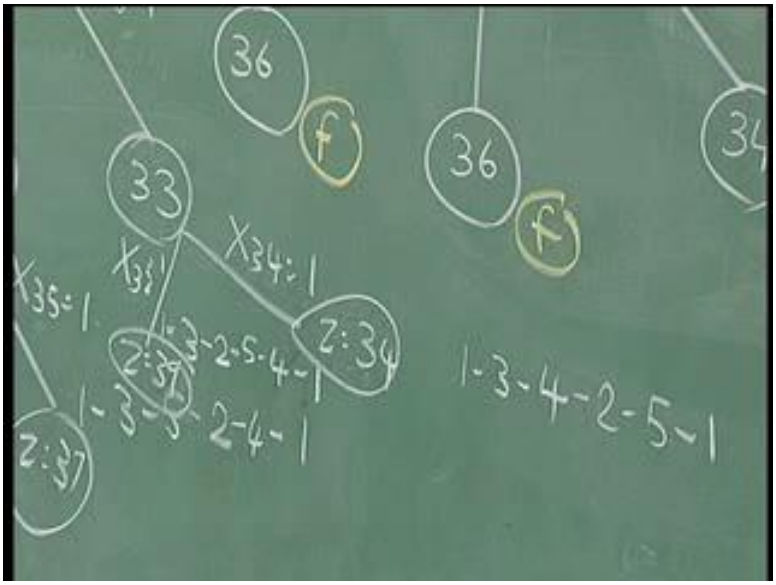
I repeat again. Since, I have 1 to 3, 3 to 5 and I also have 2 to 4, I cannot have 4 to 2. I will only have to move from 5 to 2, otherwise, I will end up coming back. Therefore, I cannot have 4 to 2. This would give me a feasible solution 1 to 3, 3 to 5, 5 to 2, 2 to 4, 4 to 1. This gives me a value, 3 to 5 is 8 plus 9 is 17, 17 plus 6 is 23, 23 plus 5 is 28, 28 plus 9 is 37. This gives me Z equal to 37, which is not being helping me in any way, because this is more than 36 and therefore, I will not be able to get any gain out of this.

(Refer Slide Time: 53:21)



I go back and see whichever is the smallest. This is the smallest, so again I branch from here. I have done 1 to 3, 2 to 5, so I cannot do 3 to 1, because I have done 1 to 3. I can do 3 to 2 equal to 1, I will not do 3 to 3, I will do 3 to 4 equal to 1. Each of these will give me a feasible solution. Here, I get 1 to 3, 3 to 2, 1 to 3, 3 to 2, 2 to 5, 5 to 4 and 4 to 1.

(Refer Slide Time: 54:29)



Now, 1 to 3 is 8, 3 to 2 is 18, 2 to 5 is 24, 5 to 4 is 30, 4 to 1 is 39 so Z is equal to 39.

Z equal to 39 is also not going to help me in anyway so I will start looking at this one. This would give me 1 to 3, 3 to 4. I have done 1 to 3, 3 to 4. I cannot do 4 to 5; I can do 4 to 2, 2 to

5 and 5 to 1. So, 1 to 3 is 8, 3 to 4 is 8, plus 8 is16, 4 to 2 is 16, plus 5 is 21, 21 plus 6 is 27 plus 7, so, Z equal to 34. The moment I have a solution with Z equal to 34, I realize I can do many things. This gives me a solution with Z equal to 34. I can fathom this because moving from here down is only going to give me 35 or more. I can fathom this also because this moving down is going to give me 34 or more. I can fathom this as well, based on the lower bound, where proceeding from here is going to give me only 34 or more. At this point, we realize that we have a feasible solution with 34, which is the best solution and right now we do not have any other node that is left for evaluation.

All the nodes are now fathomed and these nodes have been fathomed either by feasibility, feasible solution here, here, here and here, or fathomed based on the condition that lower bound is greater than upper bound. When there are no more nodes to move around, the algorithm terminates and in this case it gives us the optimal solution. The best among the feasible solutions is optimal, so this is the optimal solution: 13 4 2 5 1, where 1 to 3 is 8, 3 to 4 is 16, 4 to 2 plus 5 is 21, 2 to 5 is 21 plus 6 is 27 and plus 7 and Z equal to 34. So, the algorithm terminates with this solution, with this optimal.

Because the problem is symmetric, we would also have got the solution if we had branched this way, 1 5 2 4 3 1; it would have given us the same value of 34. This is a rudimentary branch and bound algorithm that we have seen for the Travelling Salesman Problem. This is a branch and bound algorithm. We have already seen some aspects of branch and bound algorithm earlier when we did integer programming. In integer programming we said we could fathom in three ways feasibility, infeasibility and based on the bounds. In this case, we will not have infeasibility. So we will fathom either based on feasibility or based on this condition that a lower bound at any node is greater than the current best upper bound. This is the first rudimentary branch and bound algorithm and computational experience with this branch and bound algorithm says that this is not very fast or effective. It evaluates many more nodes.

Ordinarily, the bounding strategy comes from the summation of the row minimum consistently. It also takes much more nodes than we normally expect. The branching strategy is based on the node that has the smallest value of the lower bound; the calculation of the bound comes from the minimum. Are there branch and bound methods which are better than this branch and bound? Now, we will see a couple of more branch and bound algorithms to the Travelling Salesman Problem in the next lecture.